VARCHART JGantt

Version 3.2

Reference Guide

NETRONIC Software GmbH
Pascalstr. 15
D-52076 Aachen
Phone: +49-(0)2408-141-0
Fax: +49-(0)2408-141-33
E-Mail: sales@netronic.com
www.netronic.com

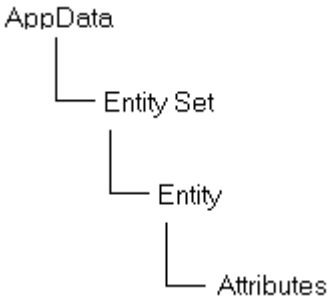
Last Revision: 22. August 2016

Table of Contents

1 AppData

This component offers classes and interfaces to handle the application data.

To do this, this component lets you specify and administer a hierarchy of objects that define the data structure of your application. In analogy of a data set the main element in the VARCHART JGantt data structure is an entity. It is comparable to a data set that holds all data of an activity. An entity on one hand consists of attributes, comparable to the fields in a data set that hold the details. On the other hand, several entities form an entity set, similar to a data table. The total of all entity sets represents the application data (abbreviated by appData), similar to a data base. In contrast to a data base that holds simple data, the appData contains objects.



Besides, there are properties and methods to handle the XML file that the data structure and the data of the appData can be stored to.

The AppData-Component consists of the below classes:

NeAppDataAdapter	This class is an adapter class for the interface NeAppDataEventListener.
NeAppDataEvent	This class lets you handle events of entities and entity sets.
NeAppDataEventListener	This is the listener interface for receiving AppData events.
NeIAppData	This interface lets you handle entity sets.
NeIEntity	This interface lets you handle entity attributes.
NeIEntitySet	This interface lets you handle entities.
NeIXML	This interface lets you handle the XML file.

1.1 NeAppDataAdapter

Belongs to AppData

Package name	**de.netronic.common.event**
Extends	**java.lang.Object**
Implements	**de.netronic.common.event.NeAppDataEventListener**

This class is an adapter class for the interface NeAppDataEventListener. Its methods are empty. This class exists as convenience for creating listener objects.

1.2 NeAppDataEvent

Belongs to AppData

Package name	**de.netronic.common.event**
Extends	**java.util.EventObject**

This class lets you handle events of entities and entity sets.

Properties to Handle Events that Affect Entities or Entity Sets

AttributNameOfChange	Name of the attribute changed
Entity	The entity affected by the event
EntitySet	The entity set affected by the event

Constructors of the Class

NeAppDataEvent

Constructor of NeAppDataEvent

This constructor lets you generate an event that holds information on the modified entities of the application data.

Declaration
NeAppDataEvent (java.lang.Object source, int index, java.lang.Object obj)

Parameter	Data Type	Description
source	java.lang.Object	Object that represents the source of the event.
index	int	Index of the entity that was changed.
obj	java.lang.Object	Entity that was changed.

NeAppDataEvent

Constructor of NeAppDataEvent

This constructor lets you generate an event that holds information on the modified entities of the application data and lets you specify the attribute affected.

Declaration
NeAppDataEvent (java.lang.Object source, int index, java.lang.Object obj, java.lang.String attrName)

Parameter	Data Type	Description
source	java.lang.Object	Object that represents the source of the event.
index	int	Index of the entity that was changed.
obj	java.lang.Object	Entity that was changed.
attrName	java.lang.String	Name of the attribute affected.

Properties of the Class

AttributNameOfChange

Read Only Property of NeAppDataEvent

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the name of the attribute modified.

Accessing Methods
java.lang.String getAttributNameOfChange()

Entity

Read Only Property of NeAppDataEvent

Typ	**de.netronic.common.intface.NeIEntity**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the entity affected by an event, that is, the entity that was changed, added or removed.

Accessing Methods
de.netronic.common.intface.NeIEntity getEntity()

EntitySet

Read Only Property of NeAppDataEvent

Typ	**de.netronic.common.intface.NeIEntitySet**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the entity set affected by an event, that is, the entity set that was changed, added or removed or the entity set of which an entity was changed, added or removed.

Accessing Methods
de.netronic.common.intface.NeIEntitySet getEntitySet()

1.3 NeAppDataEventListener

Belongs to AppData

Package name	**de.netronic.common.event**
Extends	**java.util.EventListener**

This is the listener interface for receiving AppData events. The class that is interested in processing an AppData event either implements this interface, (including the methods it contains), or extends the AppDataAdapter class (overriding only the methods of interest). The listener object is then registered with the AppData object using the AppData's addListener method. An AppData event is generated on creating, modifying or deleting entities or entity sets.

Methods to React to an Entity Changement

entityChanged(...)	This method is invoked after an entity of appData was changed.
entityCreated(...)	This method is invoked after an entity of application data object was created.
entityDeleted(...)	This method is invoked after an entity of application data object was deleted.

Methods to React to Changement of an Entity Set

entitySetChanged(...)	This method is invoked after an entity set in the application data was changed.
entitySetCreated(...)	This method is invoked after an entity set in the application data was created.
entitySetDeleted(...)	This method is invoked after an entity set in the application data was deleted.

Methods of the Interface

entityChanged

Method of NeAppDataEventListener

This method is invoked after an entity of appData was changed.

Declaration
void entityChanged (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	This parameter points to the object NeAppDatEvent. By its method getEntity you can retrieve the entity changed.
Return Value	void	

Also see entityCreated
entityDeleted

entityCreated

Method of NeAppDataEventListener

This method is invoked after an entity of the application data object was created.

Declaration
void entityCreated (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	This parameter points to the object NeAppDataEvent. By its method getEntity you can retrieve the entity created.
Return Value	void	

Also see entityChanged
entityDeleted

entityDeleted

Method of NeAppDataEventListener

This method is invoked after an entity of application data object was deleted.

Declaration
void entityDeleted (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	This parameter points to the object NeAppDataEvent. By its method getEntity you can retrieve the entity deleted.
Return Value	void	

Also see entityChanged
entityCreated

entitySetChanged

Method of NeAppDataEventListener

This method is invoked after an entity set in the application data was changed.

Declaration
void entitySetChanged (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	This parameter points to the object NeAppDataEvent. By its method getEntitySet you can retrieve the entity changed.
Return Value	void	

Also see entitySetCreated
entitySetDeleted

entitySetCreated

Method of NeAppDataEventListener

This method is invoked after an entity set in the application data was created.

Declaration
void entitySetCreated (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	This parameter points to the object NeAppDataEvent. By its method getEntitySet you can retrieve the entity created.
Return Value	void	

Also see entitySetChanged
entitySetDeleted

entitySetDeleted

Method of NeAppDataEventListener

This method is invoked after an entity set in the application data was deleted.

Declaration
void entitySetDeleted (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	This parameter points to the object NeAppDataEvent.
Return Value	void	

Also see entitySetChanged
entitySetCreated

1.4 NeIAppData

Belongs to AppData

Package name **de.netronic.common.intface**

This interface lets you handle entity sets.

Properties to Handle Entity Sets

EntitySetCount	Number of existing entity sets of the application data object
TransactionHandlerInterface	Retrieves the transaction handler interface
XMLInterface	XML interface

Methods to Handle Entity Sets

addAppDataEventListener(...)	Adds a listener for change events
clear()	Clears all data of the application data object

createEntitySet(...)	Creates an entity set
deleteAllEntitySets()	Deletes all entity sets of the application data object
deleteEntitySet(...)	Deletes an entity set
getEntitySet(...)	Retrieves an entity set by its name
getEntitySetAtIndex(...)	Retrieves an entity set by its index
removeAppDataEventListener(...)	Removes a listener for change events
setLastEntityChangeInfo(...)	Assigns an information to the entity most recently modified on the modification

Properties of the Interface

EntitySetCount

Read Only Property of NeIAppData

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the number of entity sets existing in the application data object.

Accessing Methods
int getEntitySetCount()

TransactionHandlerInterface

Read Only Property of NeIAppData

Typ	**de.netronic.common.intface.NeITransactionHandler**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the transaction handler interface which lets you use transactions for working with the appData

Accessing Methods
de.netronic.common.intface.NeITransactionHandler getTransactionHandlerInterface()

XMLInterface

Read Only Property of NeIAppData

Typ	**de.netronic.common.intface.NeIXML**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the XML interface. Its methods let you access and handle the application data in XML format.

Accessing Methods
de.netronic.common.intface.NeIXML getXMLInterface()

Methods of the Interface

addAppDataEventListener

Method of NeIAppData

This method lets you add a listener for change events to the entity set specified. If you apply this method to an entity set that is not empty, you can select whether or not events are to be triggered on the existing entities.

Declaration
void addAppDataEventListener (de.netronic.common.event.NeAppDataEventListener l, java.lang.String entitySetName, boolean addEvents)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeAppDataEventListener	Listener object to be added.
entitySetName	java.lang.String	Name of the entity set that the listener is to be set up for. If you pass an empty name string, the listener will be registered for all entity sets.
addEvents	boolean	If you set this parameter to **true**, events will also be triggered on existing entities. If you pass **false**, events will not be triggered on existing entities. Please note: The value **true** will only take effect if the listener was registered for a specified entity set, that is, if the entitySetName parameter is not an empty string.
Return Value	void	

Also see removeAppDataEventListener

clear

Method of NeIAppData

This method lets you clear all data of the application data object. Both this method and the method **deleteAllEntitySets()** throw the event **entitySetDeleted**. In contrast to **deleteAllEntitySets()**, this method in addition launches the event **entityDeleted** for each entity deleted. So if you wish to react to each single entity that was deleted, you should set this method; if it is not important to do that, using **deleteAllEntitySets()** will be sufficient.

Declaration
void clear ()

	Data Type	Description
Return Value	void	

Also see deleteAllEntitySets

createEntitySet

Method of NeIAppData

This method lets you create an entity set and assigns it a name. The name cannot be changed after its creation.

Declaration
NeIEntitySet createEntitySet (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the entity set
Return Value	NeIEntitySet	Entity created

Also see deleteAllEntitySets
deleteEntitySet

deleteAllEntitySets

Method of NeIAppData

This method lets you delete all entity sets of the application data object. This method as well as the method **clear()** throw the event **entitySetDeleted**. In contrast to **clear()**, this method in addition does not trigger the event **entityDeleted** for each entity deleted. So if you wish to react to each single entity that was deleted, you should set **clear()**; if it is not important to do that, using **deleteAllEntitySets()** will be sufficient.

Declaration
int deleteAllEntitySets ()

	Data Type	Description
Return Value	int	Presently meaningless

Also see clear
createEntitySet
deleteEntitySet

deleteEntitySet

Method of NeIAppData

This method lets you delete an entity set. It does not trigger the event **entityDeleted**.

Declaration
int deleteEntitySet (de.netronic.common.intface.NeIEntitySet entitySet)

	Data Type	Description
Parameter		
entitySet	de.netronic.common.intface.-NeIEntitySet	Entity set to be deleted
Return Value	int	Presently meaningless

Also see createEntitySet
deleteAllEntitySets

getEntitySet

Method of NeIAppData

This method lets you retrieve an entity set by its name.

Declaration
NeIEntitySet getEntitySet (java.lang.String entitySetName)

	Data Type	Description
Parameter		
entitySetName	java.lang.String	Name of the entity set.
Return Value	NeIEntitySet	Entity set returned

Also see getEntitySetAtIndex

getEntitySetAtIndex

Method of NeIAppData

This method lets you retrieve an entity set by its index. The number of available entity sets can be obtained by getEntitySetCount().

Declaration
NeIEntitySet getEntitySetAtIndex (int index)

	Data Type	Description
Parameter		
index	int	Index of the entity set
Return Value	NeIEntitySet	Entity set returned

Also see getEntitySet

removeAppDataEventListener

Method of NeIAppData

This method lets you remove a listener for change events from the entity set specified.

Declaration
void removeAppDataEventListener (de.netronic.common.event.NeAppDataEventListener l, java.lang.String entitySetName)

	Data Type	Description
Parameter		
l	de.netronic.common.event.NeAppDataEventListener	Listener to be removed
entitySetName	java.lang.String	Name of the entity set that the listener is to be removed from.
Return Value	void	

Also see addAppDataEventListener

setLastEntityChangeInfo

Method of NeIAppData

This method lets you assign an information on the modification of the entity most recently modified.

Declaration
void setLastEntityChangeInfo (int id, java.lang.String entitySetName, java.lang.String xml)

	Data Type	Description
Parameter		
id	int	Identitfication tag of the entity generated by the system.
entitySetName	java.lang.String	String containing the name of the entity set.
xml	java.lang.String	Stirng containing the information to be delivered.
Return Value	void	

1.5 NeIEntity

Belongs to AppData

Package name **de.netronic.common.intface**

This interface lets you handle entity attributes.

Properties to Handle Attributes

AttributeCount	Number of attributes of the entity

Properties to Handle Entities

EntitySet	Entity set that the entity belongs to
ID	String that identifies the entity
SystemID	Identification of the entity generated by the system
UserID	The identification of the entity generated by the user

Methods to Handle Attributes

contains(...)	Retrieves, whether or not the entity contains an attribute named as passed by the parameter

copyAttributeValuesFrom(...)	Copies the values from a different entity
getAttributeAtIndex(...)	Retrieves an attribute by its index
getValue (...)	Retrieves a value as an object from an attribute of the entity
getValueAsBoolean(...)	Retrieves a value as a Boolean value from an attribute of the entity
getValueAsDate(...)	Retrieves a value as an object of the class Date
getValueAsDouble(...)	Retrieves a value as a "double" value from an attribute of the entity
getValueAsFloat(...)	Retrieves a value as a float value from an attribute of the entity
getValueAsInt(...)	Retrieves a value as a integer value from an attribute of the entity
getValueAsLong(...)	Retrieves a value as "long" from an attribute of the entity
getValueAsString(...)	Retrieves a value as a character string from an attribute of the entity.
setValue(...)	Assigns a "double" value to an attribute of the entity
setValue(...)	Assigns an integer value to an attribute of the entity
setValue(...)	Assigns a boolean value to an attribute of the entity
setValue(...)	Assigns a "long" value to an attribute of the entity
setValue(...)	Assigns an object to an attribute of the entity
setValue(...)	Assigns a float value to an attribute of the entity
setValues(...)	Sets attribute values via a map.

Methods to Handle Entities

clear()	Deletes all data from an entity.
setUserID(...)	Sets the identification tag of the entity
toXML()	Turns the entity into a string of characters
toXMLDeleteInfo()	Turns the entity that is to be deleted into a string of characters
toXMLLastChangeInfo()	Turns the alteration of the entity into a string of characters

Properties of the Interface

AttributeCount

Read Only Property of NeIEntity

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the number of attributes of the entity.

Accessing Methods
int getAttributeCount()

EntitySet

Read Only Property of NeIEntity

Typ	**de.netronic.common.intface.NeIEntitySet**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the entity set that the entity belongs to.

Accessing Methods
de.netronic.common.intface.NeIEntitySet getEntitySet()

ID

Property of NeIEntity

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve a string that identifies the entity.

Accessing Methods
void setID (java.lang.String newValue)
java.lang.String getID ()

SystemID

Read Only Property of NeIEntity

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the identification number of the entity that was generated by the system.

Accessing Methods
int getSystemID()

UserID

Read Only Property of NeIEntity

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the identification number of the entity that was assigned by the user.

Accessing Methods
java.lang.String getUserID()

Methods of the Interface

clear

Method of NeIEntity

This method lets you delete the data of an entity.

Declaration
void clear ()

	Data Type	Description
Return Value	void	

contains

Method of NeIEntity

This method retrieves, whether or not the entity contains an attribute named as passed by the parameter.

Declaration
boolean contains (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute looked for.
Return Value	boolean	**True**: The entity contains the attribute, **false**: the entity does not contain the attribute.

copyAttributeValuesFrom

Method of NeIEntity

This method copies the values from the entity passed by the parameter into this entity.

Declaration
void copyAttributeValuesFrom (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity the attribute values of which are to be copied from.
Return Value	void	

getAttributeAtIndex

Method of NeIEntity

This property lets you retrieve an attribute by its index.

Declaration
java.lang.String getAttributeAtIndex (int index)

	Data Type	Description
Parameter		
index	int	Index of the attribute to be retrieved.
Return Value	java.lang.String	Name of the attribute returned.

getOrCreateID

Method of NeIEntity

This method lets you retrieve the string that identifies the entity. If the identification string does not exist, a unique one is created.

Declaration
java.lang.String getOrCreateID ()

	Data Type	Description
Return Value	java.lang.String	String that identifies the entity.

getValue

Method of NeIEntity

This method retrieves a value as an object from an attribute of the entity. The attribute is identified by the name passed.

Declaration
java.lang.Object getValue (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	java.lang.Object	Object returned.

getValueAsBoolean

Method of NeIEntity

This method retrieves a value as a Boolean value from an attribute of the entity. The attribute is identified by the name passed.

Declaration
boolean getValueAsBoolean (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	boolean	Boolean value returned.

Also see setValue

getValueAsDate

Method of NeIEntity

This method retrieves a value as an object of the class **Date** from an attribute of the entity. The attribute is identified by its name.

Declaration
java.util.Date getValueAsDate (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	java.util.Date	Object of the class Date returned.

getValueAsDouble

Method of NeIEntity

This method retrieves a value as a "double" from an attribute of the entity. The attribute is identified by the name passed.

Declaration
double getValueAsDouble (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	double	"Double" value returned.

Also see setValue

getValueAsFloat

Method of NeIEntity

This method sets a value as a float value from an attribute of the entity. The attribute is identified by the name passed.

Declaration
float getValueAsFloat (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	float	"Float" value returned.

Also see setValue

getValueAsInt

Method of NeIEntity

This method retrieves a value as an integer from an attribute of the entity. The attribute is identified by the name passed.

Declaration
int getValueAsInt (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	int	Integer value returned.

Also see setValue

getValueAsLong

Method of NeIEntity

This method retrieves a value as "long" from an attribute of the entity. The attribute is identified by its name.

Declaration
long getValueAsLong (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	long	Object returned.

Also see setValue

getValueAsString

Method of NeIEntity

This method retrieves a value as a character string from an attribute of the entity. The attribute is identified by its name.

Declaration
java.lang.String getValueAsString (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be retrieved.
Return Value	java.lang.String	Character string returned.

setUserID

Method of NeIEntity

This method sets the identification tag of the entity. The tag can be set only for once, either via the method createEntity() or, if it wasn't set by the latter method, alternatively by setUserID(). An identification tag once set cannot be modified. It must not consist of blanks only.

The identification tag set here is used by links to reference the node. So their existence is a prerequisite for introducing links.

Declaration
boolean setUserID (java.lang.String userID)

	Data Type	Description
Parameter		
userID	java.lang.String	Identification tag to be set.
Return Value	boolean	**True**: The identification tag of the entity was set successfully, **false**: the identification tag of the entity was not set successfully.

setValue

Method of NeIEntity

This method assigns an integer value to an attribute of the entity. The attribute is identified by the name passed.

Declaration
void setValue (java.lang.String name, int value)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be set.
value	int	Value to be set.
Return Value	void	

Also see getValueAsInt

setValue

Method of NeIEntity

This method assigns a "double" value to an attribute of the entity. The attribute is identified by the name passed.

Declaration
void setValue (java.lang.String name, double value)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be set.
value	double	Value to be set.
Return Value	void	

Also see getValueAsDouble

setValue

Method of NeIEntity

This method assigns a boolean value to an attribute of the entity. The attribute is identified by the name passed.

Declaration
void setValue (java.lang.String name, boolean value)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be set.
value	boolean	Value to be set.
Return Value	void	

Also see getValueAsBoolean

setValue

Method of NeIEntity

This method assigns a "long" value to an attribute of the entity. The attribute is identified by the name passed.

Declaration
void setValue (java.lang.String name, long value)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be set.
value	long	Value to be set.
Return Value	void	

Also see getValueAsLong

setValue

Method of NeIEntity

This method assigns an object to an attribute of the entity. The attribute is identified by the name passed.

Declaration
void setValue (java.lang.String name, java.lang.Object value)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be set.
value	java.lang.Object	Value to be set.
Return Value	void	

Also see getValue
getValueAsDate

setValue

Method of NeIEntity

This method assigns a float value to an attribute of the entity. The attribute is identified by the name passed.

Declaration
void setValue (java.lang.String name, float value)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute, the value of which is to be set.
value	float	Value to be set.
Return Value	void	

Also see getValueAsFloat

setValues

Method of NeIEntity

This method lets you set several attribute values via a map. In the map, the attributes are identified via their names.

Declaration
void setValues (java.util.Map values)

	Data Type	Description
Parameter		
values	java.util.Map	
Return Value	void	

toXML

Method of NeIEntity

This method turns the entity into a string of characters. This method is used to exchange alterations in websites.

Declaration
java.lang.String toXML ()

	Data Type	Description
Return Value	java.lang.String	String of characters containing the data of the entity.

Also see toXMLDeleteInfo
toXMLLastChangeInfo

toXMLDeleteInfo

Method of NeIEntity

This method turns the entity, that is to be deleted, into a string of characters. This method is used to exchange alterations in websites.

Declaration
java.lang.String toXMLDeleteInfo ()

	Data Type	Description
Return Value	java.lang.String	String of characters containing the data of the entity to be deleted.

Also see toXML
toXMLLastChangeInfo

toXMLLastChangeInfo

Method of NeIEntity

This method turns the alterations of the entity into a string of characters. This method is used to exchange alterations in websites.

Declaration
java.lang.String toXMLLastChangeInfo ()

	Data Type	Description
Return Value	java.lang.String	String of characters containing the data of the alterations.

Also see toXML
toXMLDeleteInfo

1.6 NeIEntitySet

Belongs to AppData

Package name **de.netronic.common.intface**

This interface lets you handle entities.

Properties to Handle Entities

AllEntityCount	Number of entities present in all entity sets
EntityAttributeCount	Number of attributes present in an entity
EntityCount	Number of entities present in the entity set
TemplateEntity	The template entity currently valid

Properties to Handle the Entity-Set

AppData	The application data object that holds the entity set
Name	Retrieves the name of the entity set

Methods to Handle Entities

createEntity(...)	Creates an entity according to a template passed
createEntity(...)	Creates an entity
deleteEntity(...)	Deletes an entity
getEntityAtIndex(...)	Retrieves an entity by its index
getEntityViaSystemID(...)	Retrieves an entity via its system id
getEntityViaUserID(...)	Retrieves an entity via its user id

Methods to Handle Entity-Sets

deleteAllEntitySetEntities()	Deletes all entities of the entity set

Methods to Handle Attributes

addEntityAttribute(...)	Adds an attribute to an entity
getEntityAttributeAtIndex(...)	Retrieves the name of an entity attribute via its index
getEntityAttributeClass(...)	Retrieves the class of an attribute
getEntityAttributeDisplayName(...)	Retrieves the display name of an attribute
hasEntityAttribute(...)	Retrieves, whether or not the attribute passed exists
removeEntityAttribute(...)	Removes an attribute to from entity
setEntityAttributeDisplayName(...)	Sets the display name of the attribute passed

Properties of the Interface

AllEntityCount

Read Only Property of NeIEntitySet

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the number of entities present in all entity sets.

Accessing Methods
int getAllEntityCount()

Also see EntityCount

AppData

Read Only Property of NeIEntitySet

Typ	**de.netronic.common.intface.NeIAppData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the application data object that holds the entity set.

Accessing Methods
de.netronic.common.intface.NeIAppData getAppData()

EntityAttributeCount

Read Only Property of NeIEntitySet

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the number of attributes present in an entity.

Accessing Methods
int getEntityAttributeCount()

EntityCount

Read Only Property of NeIEntitySet

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the number of entities present in the entity set.

Accessing Methods
int getEntityCount()

Also see AllEntityCount

Name

Read Only Property of NeIEntitySet

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the name of the entity set.

Accessing Methods
java.lang.String getName()

TemplateEntity

Read Only Property of NeIEntitySet

Typ	**de.netronic.common.intface.NeIEntity**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the existing template entity. The template entity cannot be set, just the existing one can be modified. The default template entity is used when you create an entity by **createEntity (java.lang.String userID)** , while your own template entity is used when you create an entity by **createEntity (java.lang.String userID, NeIEntity entity)**. We recommend to generate entities by template entities, if a larger series of similar entities is to be created. It will contribute to a good performance rather than setting single attributes to empty entities.

Accessing Methods
de.netronic.common.intface.NeIEntity getTemplateEntity()

Methods of the Interface

addEntityAttribute

Method of NeIEntitySet

This method lets you add an entity to an attribute.

Declaration
void addEntityAttribute (java.lang.String attributeName, java.lang.Object classType, java.lang.String descriptiveName)

	Data Type	Description
Parameter		
attributeName	java.lang.String	Name to be given to the attribute
classType	java.lang.Object	Class type. Avsaailable class types: String.class, Long.class, Integer.class, Double.class, Date.class
descriptiveName	java.lang.String	Description of the attribute. The contents of this field can be displayed as in the table.
Return Value	void	

Also see removeEntityAttribute

createEntity

Method of NeIEntitySet

This method lets you create an entity according to the template passed and assigns it a user identification. We recommend to generate entities by template entities, if a larger series of similar entities is to be created. It will contribute to a good performance rather than setting single attributes to empty entities.

Declaration
de.netronic.common.intface.NeIEntity createEntity (java.lang.String UserID, de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
UserID	java.lang.String	Identification tag of the entity generated by the user
entity	de.netronic.common.intface.NeIEntity	Template entity
Return Value	de.netronic.common.intface.NeIEntity	Entity created or null, in case a user ID was passed that already exists.

Also see createEntity
deleteAllEntitySetEntities
deleteEntity

createEntity

Method of NeIEntitySet

This method lets you create an entity an assign it an identification tag. When creating the entity, the internal default entity will be used as a template. If you wish to use a different template, you can define one and pass it in a second paramter. Please see createEntity (java.lang.String userID, NeIEntity entity).

Declaration
de.netronic.common.intface.NeIEntity createEntity (java.lang.String UserID)

	Data Type	Description
Parameter		
UserID	java.lang.String	Identification tag of the entity generated by the user
Return Value	de.netronic.common.intface.NeIEntity	Entity created or null, in case a user ID was passed that already exists.

Also see createEntity
deleteAllEntitySetEntities
deleteEntity

deleteAllEntitySetEntities

Method of NeIEntitySet

This method lets you delete all entities of the entity set.

Declaration
void deleteAllEntitySetEntities ()

	Data Type	Description
Return Value	void	

Also see createEntity
createEntity
deleteEntity

deleteEntity

Method of NeIEntitySet

This method lets you delete the entity passed by the parameter.

Declaration
boolean deleteEntity (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity to be deleted
Return Value	boolean	Returns **true** if the entity could be deleted.

Also see createEntity
createEntity
deleteAllEntitySetEntities

getEntityAtIndex

Method of NeIEntitySet

This property lets you retrieve an entity by its index.

Declaration
de.netronic.common.intface.NeIEntity getEntityAtIndex (int index)

	Data Type	Description
Parameter		
index	int	Index of the entity
Return Value	de.netronic.common.intface.NeIEntity	Entity returned

Also see getEntityAttributeAtIndex

getEntityAttributeAtIndex

Method of NeIEntitySet

This property lets you retrieve the name of an entity attribute via its index.

Declaration
java.lang.String getEntityAttributeAtIndex (int index)

	Data Type	Description
Parameter		
index	int	Index of the attribute
Return Value	java.lang.String	Name of the entity returned

Also see getEntityAtIndex

getEntityAttributeClass

Method of NeIEntitySet

This property lets you retrieve the class of an attribute.

Declaration
java.lang.Object getEntityAttributeClass (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute
Return Value	java.lang.Object	Class object returned

getEntityAttributeDisplayName

Method of NeIEntitySet

This property lets you retrieve the display name of an entity attribute via its name.

Declaration
java.lang.String getEntityAttributeDisplayName (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute
Return Value	java.lang.String	Display name of the attribute

Also see setEntityAttributeDisplayName

getEntityViaSystemID

Method of NeIEntitySet

This property lets you retrieve an entity via the identification tag generated by the system.

Declaration
de.netronic.common.intface.NeIEntity getEntityViaSystemID (int systemID)

	Data Type	Description
Parameter		
systemID	int	Identification tag of the entity generated by the system
Return Value	de.netronic.common.intface.NeIEntity	Entity returned

Also see getEntityViaUserID

getEntityViaUserID

Method of NeIEntitySet

This method lets you retrieve an entity via the identification tag generated by the user.

Declaration
de.netronic.common.intface.NeIEntity getEntityViaUserID (java.lang.String userID)

	Data Type	Description
Parameter		
userID	java.lang.String	Identification tag of the entity generated by the user
Return Value	de.netronic.common.intface.NeIEntity	Entity returned

Also see getEntityViaSystemID

hasEntityAttribute

Method of NeIEntitySet

This method retrieves, whether or not the attribute passed exists in the entities of the entity set.

Declaration
boolean hasEntityAttribute (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute looked for
Return Value	boolean	**True**, if the attribute exists, **false**, if it doesn't.

removeEntityAttribute

Method of NeIEntitySet

This method lets you remove an attribute from an entity.

Declaration
void removeEntityAttribute (java.lang.String attributeName)

	Data Type	Description
Parameter		
attributeName	java.lang.String	Name of the attribute to be removed
Return Value	void	

Also see addEntityAttribute

setEntityAttributeDisplayName

Method of NeIEntitySet

This method lets you assign the display name of the attribute passed.

Declaration
void setEntityAttributeDisplayName (java.lang.String attributeName, java.lang.String value)

	Data Type	Description
Parameter		
attributeName	java.lang.String	Name of the attribute, the display name of which is to be set.
value	java.lang.String	Display name to be assigned
Return Value	void	

Also see getEntityAttributeDisplayName

1.7 NeIXML

Belongs to AppData

Package name **de.netronic.common.intface**

This interface lets you handle the XML file.

Properties to Handle the XML-Data

XMLLogging	The logging state of changes in entities
XMLLogString	The log string

Methods to Handle XML Data

appendToXMLLogString(...)	Manually appends data to the log string.
loadXML(...)	Loads the application data from an XML file.
loadXML(...)	Loads the application data from an XML file.
loadXML(...)	Loads the application data from an XML file.
saveXML(...)	Stores the application data to an XML file.
saveXML(...)	Stores the application data to an XML file (optionally plus its data structure and compressed format).
saveXML(...)	Stores the application data to an XML file (optionally plus its data structure and compressed format).
saveXML(...)	Stores the application data to an XML file.
sendXMLData(...)	Sends an XML string to a file or to a URL.

Properties of the Interface

XMLLogging

Property of NeIXML

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**NeIXML.NE_NO_LOGGING**

This property describes the logging state of changes in entities.

Possible Values	Description
NeIXML.NE_COMPLETE_LOGGING	The complete entity is stored for logging. Attributes that do not have a value are included.
NeIXML.NE_FULL_LOGGING	The complete entity is stored for logging; attributes that do not have a value are omitted.
NeIXML.NE_NO_LOGGING	Logging is switched off.
NeIXML.NE_PARTIAL_LOGGING	Only the parts of the entity that were changed are stored for logging.

Accessing Methods
void setXMLLogging (int newValue)
int getXMLLogging ()

Also see XMLLogString

XMLLogString

Property of NeIXML

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property contains the log string.

Accessing Methods
void setXMLLogString (java.lang.String newValue)
java.lang.String getXMLLogString ()

Also see XMLLogging

Methods of the Interface

appendToXMLLogString

Method of NeIXML

Continued changes in the application data are automatically appended to the log string. This method allows to manually append data to the log string that is written in XML.

Declaration
void appendToXMLLogString (java.lang.String xml)

	Data Type	Description
Parameter		
xml	java.lang.String	String to be appended
Return Value	void	

loadXML

Method of NeIXML

This method loads the application data that are written in XML. The source may be a true file, with the address formatted as given in the example c:\\directory1\\directory2\\file.ext. Alternatively, it may be a URL such as http://www.mywebsite.com.

It makes sense to use this call for the persistency of calendars, where entity sets are required to store profiles, time spans and their relations.

Declaration
void loadXML (java.lang.String uri, java.lang.String[] entitySetNames)

	Data Type	Description
Parameter		
uri	java.lang.String	String containig either a URL or a file name including its path.
entitySetNames	java.lang.String[]	Array of strings that contain the names of entity sets to be loaded.
Return Value	void	

loadXML

Method of NeIXML

This method loads the application data, that are written in XML. The source may be a true file, with the address formatted as given in the example c:\\directory1\\directory2\\file.ext. Alternatively, it may be a URL such as http://www.mywebsite.com.

Declaration
void loadXML (java.io.InputStream in)

	Data Type	Description
Parameter		
in	java.io.InputStream	Input stream containing either a file name including its path or a URL.
Return Value	void	

loadXML

Method of NeIXML

This method loads the application data that are written in XML. The source may be a true file, with the address formatted as given in the example c:\\directory1\\directory2\\file.ext. Alternatively, it may be a URL such as http://www.mywebsite.com.

Declaration
void loadXML (java.lang.String uri)

	Data Type	Description
Parameter		
uri	java.lang.String	String containig either a file name including its path or a URL.
Return Value	void	

Also see saveXML
saveXML

saveXML

Method of NeIXML

This method stores the application data to a file written in XML. The target may be a true file, with the address formatted as given in the example c:\\directory1\\directory2\\file.ext. Alternatively, it may be a URL such as http://www.mywebsite.com.

It makes sense to use this call for the persistency of calendars, where entity sets are required to store profiles, time spans and their relations.

Declaration
void saveXML (java.lang.String uri, java.lang.String[] entitySetNames, boolean saveScheme)

	Data Type	Description
Parameter		
uri	java.lang.String	String containig either a URL or a file name including its path.
entitySetNames	java.lang.String[]	Array of strings that contain the names of entity sets to be loaded.
saveScheme	boolean	**True**: the data structure is to be stored, **false**: the data structure is not to be stored.
Return Value	void	

saveXML

Method of NeIXML

This method stores the application data to file written in XML. The target may be a true file, with the address formatted as given in the example c:\\directory1\\directory2\\file.ext. Alternatively, it may be a URL such as http://www.mywebsite.com.

Beside, this method lets you specify whether or not a data structure and the data are to be saved and whether or not the file is to be stored as a zip file.

Declaration
void saveXML (java.io.OutputStream out, boolean saveScheme, boolean saveData)

	Data Type	Description
Parameter		
out	java.io.OutputStream	Input stream containig either a file name including its path or a URL.
saveScheme	boolean	**True**: the data structure is to be stored, **false**: the data structure is not to be stored.
saveData	boolean	**True**: the data is to be stored, **false**: the data is not to be stored
Return Value	void	

saveXML

Method of NeIXML

This method stores the application data to file written in XML. The target may be a true file, with the address formatted as given in the example c:\\directory1\\directory2\\file.ext. Alternatively, it may be a URL such as http://www.mywebsite.com.

Beside, this method lets you specify whether or not a data structure and the data are to be saved and whether or not the file is to be stored as a zip file.

Declaration
void saveXML (java.lang.String uri, boolean saveScheme, boolean saveData, boolean compressed)

	Data Type	Description
Parameter		
uri	java.lang.String	String containig either a file name including its path or a URL.
saveScheme	boolean	**True**: the data structure is to be stored, **false**: the data structure is not to be stored.
saveData	boolean	**True**: the data is to be stored, **false**: the data is not to be stored
compressed	boolean	**True**: the file is to be stored as a compressed file, **false**: the data is not to be stored as a compressed file
Return Value	void	

Also see loadXML
saveXML

saveXML

Method of NeIXML

This method stores the application data to a file written in XML. The target may be a true file, with the address formatted as given in the example c:\\directory1\\directory2\\file.ext. Alternatively, it may be a URL such as http://www.mywebsite.com.

Declaration
void saveXML (java.lang.String uri)

	Data Type	Description
Parameter		
uri	java.lang.String	String containig either a URL or file name including its path.
Return Value	void	

Also see loadXML
saveXML

sendXMLData

Method of NeIXML

Sends an xml string to a file or to a URL. The target may be a true file with the address formatted as given in the example c:\\directory1\\directory2\\file.ext, alternatively, it may be a URL such as http://www.mywebsite.com.

Declaration
java.lang.String sendXMLData (java.lang.String uri, java.lang.String xml)

	Data Type	Description
Parameter		
uri	java.lang.String	String containing either a file name including its path or a URL.
xml	java.lang.String	XML string to be send.
Return Value	java.lang.String	String containing the error code of the HTTP protocol.

2 Calendar

This component contains interfaces to compose a VARCHART JGantt calendar.

A calendar is the logic object that frames the other objects needed to display time patterns. A calendar is automatically generated when a JGantt object is created, therefore you will also find calendar properties in the JGantt component. One calendar is allocated to each one JGantt object.

A JGantt object can change its calendar. In case, for example, you are working with several JGantt objects and you wish them to display the same time patterns, it is useful to assign the same calendar to all of them.

In a calendar, different units of time patterns may exist. Those units are called "profiles".

A profile needs to be filled with time spans. They define the time pattern to be displayed by the time scale and Gantt graph of the Gantt diagram.

Time spans may be of different qualities. On one hand, they may be a working or a non-working time span, on the other hand, they may be unique, such as a workfree day, or they may occur periodically (recurrent time spans), such as weekends.

The Calendar-Component consists of the below classes:

NeCalendarEvent	This class lets you handle events of calendars.
NeCalendarListener	This is the listener interface for receiving profile change events.
NeICalendar	This interface lets you compose calendars from profiles and time spans and handle the calendar.
NeIProfile	This interface lets you compose a profile from time spans and perform calculations on time spans.
NeITimeSpan	This interface lets you compose and handle time spans.

2.1 NeCalendarEvent

Belongs to Calendar

Package name	**de.netronic.common.event**
Extends	**java.util.EventObject**

This class lets you handle events of calendars.

Properties to Handle Profiles

Profile	Retrieves the profile affected by the event.

Constructors of the Class

NeCalendarEvent

Constructor of NeCalendarEvent

This class lets you generate an event that holds information on the source of the event and the profile affected.

Declaration
NeCalendarEvent (java.lang.Object source, de.netronic.common.intface.NeIProfile profile)

Parameter	Data Type	Description
source	java.lang.Object	Object that represents the source of the event.
profile	de.netronic.common.intface.NeIProfile	Profile that was changed.

Properties of the Class

Profile

Read Only Property of NeCalendarEvent

Typ	**de.netronic.common.intface.NeIProfile**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the profile affected by the event.

Accessing Methods
de.netronic.common.intface.NeIProfile getProfile()

2.2 NeCalendarListener

Belongs to Calendar

Package name **de.netronic.common.event**
Extends **java.util.EventListener**

This is the listener interface for receiving profile change events. The class that is interested in processing a calendar change event will implement this interface, including the methods it contains.The listener object will then be registered with the object using the object's addListener method. A calendar change event is generated after creating, modifying or deleting a calendar or profile.

Event Methods for Profiles

profileAdded(...)	Event method on adding of a profile
profileChanged(...)	Event method on change of a profile
profileDeleted(...)	Event method on deleting of a profile
rowDefinitionChanged(...)	Event method on change of a row definition
rowDefinitionDeleted(...)	Event method on deleting a row definition

Methods of the Interface

profileAdded

Method of NeCalendarListener

This method is invoked on adding of a profile.

Declaration
void profileAdded (de.netronic.common.event.NeCalendarEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeCalendarEvent	Event triggered
Return Value	void	

Also see profileDeleted

profileChanged

Method of NeCalendarListener

This method is invoked on change of a profile.

Declaration
void profileChanged (de.netronic.common.event.NeCalendarEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeCalendarEvent	Event triggered
Return Value	void	

profileDeleted

Method of NeCalendarListener

This method is invoked on deleting of a profile.

Declaration
void profileDeleted (de.netronic.common.event.NeCalendarEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeCalendarEvent	Event triggered
Return Value	void	

Also see profileAdded

2.3 NeICalendar

Belongs to Calendar

Package name **de.netronic.common.intface**

This interface lets you compose calendars from profiles and time spans and handle the calendar. Beside, you will find methods for time calculations.

Properties to Handle Calendar Elements

AppData	AppData Object that the calendar refers to
MinimalDaysInFirstWeek	This property describes how many days the first week of a year should contain at minimum .
ProfileEntitySetName	Name of the profile entity set of a calendar
ProfileTimeSpanRelationEntitySetName	Name of the entity set for the relations of profiles and time spans of a calendar
TimeSpanEntitySetName	Name of the time span entity set of a calendar
ValidRange	Time span that represents the valid time range of the calendar

Methods of Time Calculation

convertToRelativeTime(...)	Converts the time amount passed into the equivalent number of seconds.
getDayPart(...)	Returns the number of full days present in the time value passed.
getHourPart(...)	Returns the number of full hours present in the time value passed.
getMinutePart(...)	Returns the number of full minutes present in the time value passed.
getSecondPart(...)	Returns the number of remaining seconds present in the time value passed.

Methods to Handle Calendar Elements

addCalendarListener(...)	Adds a listener to the calendar.
clear()	Deletes all objects in calendar.
createProfile()	Creates an empty profile object.
createTimeSpan(...)	Creates an initialised time span.
createTimeSpan()	Creates an empty time span.
createTmpTimeSpan()	For internal use only.
deleteProfile(...)	Deletes a profile.

deleteTimeSpan(...)	Deletes a time span.
fetchEntitySetNames()	Retrieves the names of profiles required for the persistency of the calendar
getProfile(...)	Retrieves a profile by its name.
removeCalendarListener(...)	Removes a listener object.
setValidRange(...)	Time span that represents the valid time range of the calendar

Properties of the Interface

AppData

Property of NeICalendar

Typ	**NeIAppData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Persistency of a calendar requires to combine it with the appData object that it refers to. By this property you can assign the appData to the calendar.

Accessing Methods
void setAppData (NeIAppData newValue)
NeIAppData getAppData ()

MinimalDaysInFirstWeek

Property of NeICalendar

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes how many days the first week of a year should contain at minimum. If the first week is designed to contain the first day of the first month of a year, please set this property to 1. If you want it to be the first full week, please set it to 7.
This property will affect the time scale only.
Please note that this property will become effective only if set during the initialization of the JGantt object.

Accessing Methods
void setMinimalDaysInFirstWeek (int newValue)
int getMinimalDaysInFirstWeek ()

Example Code

```
jGantt1.getCalendar().setMinimalDaysInFirstWeek(7);
```

ProfileEntitySetName

Property of NeICalendar

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

For the persistency of a calendar, entity sets of profiles, time spans and the relations between them are required. This property sets/retrieves the name of entity set for profiles of the calendar.

Accessing Methods
void setProfileEntitySetName (java.lang.String newValue)
java.lang.String getProfileEntitySetName ()

Also see ProfileTimeSpanRelationEntitySetName
TimeSpanEntitySetName

ProfileTimeSpanRelationEntitySetName

Property of NeICalendar

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

For the persistency of a calendar, entity sets of profiles, time spans and the relations between them are required. This property sets/retrieves the name of entity set for the relations between profiles and timespans in the calendar.

Accessing Methods
void setProfileTimeSpanRelationEntitySetName (java.lang.String newValue)
java.lang.String getProfileTimeSpanRelationEntitySetName ()

Also see ProfileEntitySetName
TimeSpanEntitySetName

TimeSpanEntitySetName

Property of NeICalendar

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

For the persistency of a calendar, entity sets of profiles, time spans and the relations between them are required. This property sets/retrieves the name of entity set for time spans of the calendar.

Accessing Methods
void setTimeSpanEntitySetName (java.lang.String newValue)
java.lang.String getTimeSpanEntitySetName ()

Also see ProfileEntitySetName
ProfileTimeSpanRelationEntitySetName

ValidRange

Property of NeICalendar

Typ	**de.netronic.common.intface.NeITimeSpan**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the time span that represents the valid time range of the calendar.

Accessing Methods
void setValidRange (de.netronic.common.intface.NeITimeSpan newValue)
de.netronic.common.intface.NeITimeSpan getValidRange ()

Methods of the Interface

addCalendarListener

Method of NeICalendar

This method lets you add a listener to the calendar.

Declaration
void addCalendarListener (de.netronic.common.event.NeCalendarListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeCalendarListener	Listener object to be added.
Return Value	void	

clear

Method of NeICalendar

This method lets you delete all objects in a calendar.

Declaration
void clear ()

	Data Type	Description
Return Value	void	

convertToRelativeTime

Method of NeICalendar

This method lets you convert the time amount passed into the equivalent number of seconds.

Declaration
int convertToRelativeTime (int day, int hour, int minute, int seconds)

	Data Type	Description
Parameter		
day	int	Number of days to be converted and added to the other parameters.
hour	int	Number of hours to be converted and added to the other parameters.
minute	int	Number of minutes to be converted and added to the other parameters.
seconds	int	Number of seconds present in the time amount that is to be added to the other parameters.
Return Value	int	Number of seconds returned

createProfile

Method of NeICalendar

This method lets you generate an empty profile object. A profile serves to create a pattern of working and non-working times. It can be assigned to other objects such as the time scale to visualize the pattern.

Declaration
NeIProfile createProfile ()

	Data Type	Description
Return Value	NeIProfile	Profile created

Also see deleteProfile

createTimeSpan

Method of NeICalendar

This method creates an empty time span. A time span is characterized by a start date and end date that are to be set by the method setValidRange. A time span may serve for example to fill a profile with a pattern of working and non-working times.

Declaration
de.netronic.common.intface.NeITimeSpan createTimeSpan ()

	Data Type	Description
Return Value	de.netronic.common.intface.-NeITimeSpan	Time span created

Also see createTimeSpan
deleteTimeSpan

createTimeSpan

Method of NeICalendar

This method lets you create a time span and initialize some of its parameters. You can assign values to the start date and to the end date, to an efficiency parameter and to an identification number.

Declaration
de.netronic.common.intface.NeITimeSpan createTimeSpan (java.lang.String span, int activity, int id)

	Data Type	Description
Parameter		

span	java.lang.String	String containing the dates of the beginning and of the end of the time span. In addition, it contains whether the time span will be of unique or repeating occurrence. If the sequence returned for example is 22.5.2003 7:30-25.5.2003 17:00, the time span is of unique occurrence. If the sequence returned for example is 22.5.2003-25.5.2003 7:30-17:00, the time span is of repeating occurrence. The date of the day can be replaced by defined strings: • Monday ..Sunday • First Monday .. First Sunday • Last Monday ..Last Sunday • Last day • * Examples: "First Wednesday.3.2004" corresponds to the third of March 2004, "Last Day.2.2005" corresponds to 28.2.2005. "Friday.3.2005" defines all Fridays in the month of March in 2005. Using the asterisk "*.4.05" comprises the period of 1.4.2005-31.4.2005.The asterisk can also replace the month or the year. For example *.*.* *:* covers the complete range of the calendar, as defined by the NeICalendar method setValidRange. Beside, the date of a day can be replaced by defined names of floating holidays: • Ash Wednesday • Good Friday • Easter Sunday • Easter Monday • Feast of Corpus Christ • Ascension of Christ • Whit Sunday • Whit Monday • Day of Prayer • Advent The names listed are self-explaining except for "Advent", which describes the first Sunday of Advent. The Sundays of Advent following can easily be derived by adding 7 or a multiple of 7 to Advent, such as "Advent+14.04", resulting in the third Sunday of Advent on 12.12.2004.

activity	int	The activity or efficiency parameter may have a value between 0 and 1000, 0 turning the time span into a non-working time, whereas the values 1...1000 represent the range of efficiency in order to, for example, indicate the amount of man power.
id	int	User id, for example to assign different colors to different time spans
Return Value	de.netronic.common.intface.-NeITimeSpan	Time span created

Also see createTimeSpan
deleteTimeSpan

createTmpTimeSpan

Method of NeICalendar

For internal use only.

Declaration
de.netronic.common.intface.NeITimeSpan createTmpTimeSpan ()

	Data Type	Description
Return Value	de.netronic.common.intface.-NeITimeSpan	

deleteProfile

Method of NeICalendar

This method lets you delete a profile.

Declaration
void deleteProfile (de.netronic.common.intface.NeIProfile profile)

	Data Type	Description
Parameter		
profile	de.netronic.common.intface.NeIProfile	Profile object to be deleted.
Return Value	void	

Also see createProfile

deleteTimeSpan

Method of NeICalendar

This method lets you delete a time span.

Declaration
void deleteTimeSpan (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span object to be deleted.
Return Value	void	

Also see createTimeSpan
createTimeSpan

fetchEntitySetNames

Method of NeICalendar

This method retrieves the names of profiles required for the persitency of the calendar

Declaration
java.lang.String[] fetchEntitySetNames ()

	Data Type	Description
Return Value	java.lang.String[]	Array of names of the entity sets that contain profiles, time spans and the relations between them.

getDayPart

Method of NeICalendar

This method returns the number of full days that are present in the number of seconds passed by the parameter.

Declaration
int getDayPart (int timeVal)

	Data Type	Description
Parameter		
timeVal	int	Number of seconds to be turned to full days.
Return Value	int	Number of full days

Also see getHourPart
getMinutePart
getSecondPart

getHourPart

Method of NeICalendar

This method returns the number of full hours that are present in the number of seconds passed by the parameter.

Declaration
int getHourPart (int timeVal)

	Data Type	Description
Parameter		
timeVal	int	Number of seconds to be turned to full hours.
Return Value	int	Number of full hours

Also see getDayPart
getMinutePart
getSecondPart

getMinutePart

Method of NeICalendar

This method returns the number of full minutes that are present in the number of seconds passed by the parameter.

Declaration
int getMinutePart (int timeVal)

	Data Type	Description
Parameter		
timeVal	int	Number of seconds to be turned to full minutes.
Return Value	int	Number of full minutes

Also see getDayPart
getHourPart
getSecondPart

getProfile

Method of NeICalendar

This method retrieves a profile by its name.

Declaration
de.netronic.common.intface.NeIProfile getProfile (java.lang.String profileName)

	Data Type	Description
Parameter		
profileName	java.lang.String	Name of the profile to be retrieved.
Return Value	de.netronic.common.intface.NeIProfile	Profile returned

getSecondPart

Method of NeICalendar

This method returns the number of remaining seconds that are left after all possible seconds of the time value passed were turned into full days, full hours and full minutes.

Declaration
int getSecondPart (int timeVal)

	Data Type	Description
Parameter		
timeVal	int	Total number of seconds that the remaining seconds are to be calculated from.
Return Value	int	Number of remaining seconds

Also see getDayPart
getHourPart
getMinutePart

removeCalendarListener

Method of NeICalendar

Listener object to be added.

Declaration
void removeCalendarListener (de.netronic.common.event.NeCalendarListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.NeCalendarListener	Listener object to be removed.
Return Value	void	

setValidRange

Method of NeICalendar

Using this method a time span that represents the valid time range of the calendar can be set by a string parameter.

Declaration
void setValidRange (java.lang.String timeSpanString)

	Data Type	Description
Parameter		

timeSpanString	java.lang.String	String containing the dates of the beginning and of the end of the time span. In addition, it contains whether the time span will be of unique or repeating occurrence. If the sequence returned for example is 22.5.2003 7:30-25.5.2003 17:00, the time span is of unique occurrence. If the sequence returned for example is 22.5.2003-25.5.2003 7:30-17:00, the time span is of repeating occurrence. The date of the day can be replaced by defined strings: • Monday ..Sunday • First Monday .. First Sunday • Last Monday ..Last Sunday • Last day • * Examples: "First Wednesday.3.2004" corresponds to the third of March 2004, "Last Day.2.2005" corresponds to 28.2.2005. "Friday.3.2005" defines all Fridays in the month of March in 2005. Using the asterisk "*.4.05" comprises the period of 1.4.2005-31.4.2005.The asterisk can also replace the month or the year. For example *.*.* *:* covers the complete range of the calendar, as defined by the NeICalendar method setValidRange. Beside, the date of a day can be replaced by defined names of floating holidays: • Ash Wednesday • Good Friday • Easter Sunday • Easter Monday • Feast of Corpus Christ • Ascension of Christ • Whit Sunday • Whit Monday • Day of Prayer • Advent The names listed are self-explaining except for "Advent", which describes the first Sunday of Advent. The Sundays of Advent following can easily be derived by adding 7 or a multiple of 7 to Advent, such as "Advent+14.04", resulting in the third Sunday of Advent on 12.12.2004.
Return Value	void	

2.4 NeIProfile

Belongs to Calendar

Package name **de.netronic.common.intface**

This interface lets you compose a profile from time spans and perform calculations on time spans.

Properties to Handle the Profile

Calendar	Calendar object that the profile is allocated to.
Name	Name of the profile

Methods of Time Calculation

calcEndTime(...)	Adds time to the start of a time span and calculates the new final date.
calcSpanTime(...)	Calculates the total working time (in seconds) of a section of a profile.
calcSpanTimeInMilli(...)	Calculates the total working time (in milliseconds) within a given time span.
calcStartTime(...)	Subtracts time from the end of a time span and calculates the new start date.

Methods to Handle Time Spans

addTimeSpan(...)	Adds a time span object to a profile.
getCurrentTimeSpan(...)	Retrieves the time span that contains the start time of the time span passed.
getNextTimeSpan(...)	Retrieves the time span following the time span passed.
getPrevTimeSpan(...)	Retrieves the time span previous to the time span passed.

iterateTimeSpans(...)	Retrieves the time span that contains the start time of the time span passed.
removeTimeSpan(...)	Removes a time span from the profile.

Properties of the Interface

Calendar

Read Only Property of NeIProfile

Typ	**de.netronic.common.intface.NeICalendar**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the calendar object that the profile is allocated to.

Accessing Methods
de.netronic.common.intface.NeICalendar getCalendar()

Name

Property of NeIProfile

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the profile.

Accessing Methods
void setName (java.lang.String newValue)
java.lang.String getName ()

Methods of the Interface

addTimeSpan

Method of NeIProfile

This method lets you add a time span object to a profile.

Declaration
void addTimeSpan (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span to be added.
Return Value	void	

Also see removeTimeSpan

calcEndTime

Method of NeIProfile

This method lets you add a duration to the start of a time span and calculate the resulting final date. A duration (in milliseconds) representing for example 2 days is added to the start date of the time span passed. Whether or not non-working times are considered, depends on the timeSpanType.

Declaration
void calcEndTime (de.netronic.common.intface.NeITimeSpan timeSpan, long durationVal, int timeSpanType)

	Data Type	**Description**
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span object passed, then refreshed by the time span returned.
durationVal	long	Duration (in milliseconds) to be added to the final date of the time span.
timeSpanType	int	Type of period that durationVal is to be added to. If the timeSpanType is set to NE_INACTIVE_TIME, the time will be added to non-working days only. A time span for example starting on Monday 1st and ending on Saturday 6th, will be extended to Saturday 13th by adding two work-free days (Sunday 7th and Saturday 13th). If timeSpanType is set to NE_ANY_TIME, the time will be added regardless of working or non-working times. In our example, the time span would end on Monday, 8th (adding Sunday, 7th and Monday, 8th). If timeSpanType is set to NE_INACTIVE_TIME, the time will be added to working days only. In our example, the time span would end on Tuesday, 9th (adding Monday, 8th and Tuesday, 9th).
	Possible Values:	
	NE_ACTIVE_TIME	Time span representing a working time.
	NE_ANY_TIME	Time span not representing a working or a non-working time.
	NE_INACTIVE_TIME	Time span representing a non-working time.
Return Value	void	

Also see calcStartTime

calcSpanTime

Method of NeIProfile

This method lets you calculate the total working time of a section of a profile. Of the working and non-working time spans in the profile section only the working times (i.e. time spans owning an activity parameter > 0) will be considered. The profile section affected is to be passed as a time span. Its own activity will remain unconsidered. The unit of the value returned is seconds.

Declaration
int calcSpanTime (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Section of the profile, the working time of which is to be calculated.
Return Value	int	Number of seconds representing the total working time of the time span passed.

Also see calcSpanTimeInMilli

calcSpanTimeInMilli

Method of NeIProfile

This method lets you calculate the total working time of a section of a profile. Of the working and non-working time spans in the profile section only the working times (i.e. time spans owning an activity parameter > 0) will be considered. The profile section affected is to be passed as a time span. Its own activity will remain unconsidered. The unit of the value returned is milliseconds.

Declaration
long calcSpanTimeInMilli (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Section of the profile, the working time of which is to be calculated.
Return Value	long	Number of milliseconds representing the total working time of the time span passed.

Also see calcSpanTime

calcStartTime

Method of NeIProfile

This method lets you subtract a duration from the end of a time span and calculate the resulting start date. A duration (in milliseconds) representing for example 2 days is subtracted from the end date of the time span passed. Whether or not non-working times are considered, depends on the timeSpanType.

Declaration
void calcStartTime (de.netronic.common.intface.NeITimeSpan timeSpan, long durationVal, int timeSpanType)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span object passed, then refreshed by the time span returned.
durationVal	long	Duration (in milliseconds) to be added to the start date of the time span.
timeSpanType	int	Type of period that durationVal is to be added to. If timeSpanType is set to NE_INACTIVE_TIME, the time will be added to non-working days only. A time span for example ending on Friday 28th and starting on Sunday 24th, will be extended to Sunday 17th by adding two work-free days (Saturday 23rd and Sunday 17th). If timeSpanType is set to NE_ANY_TIME, the time will be added regardless of working or non-working times. In our example, the time span would start on Friday, 22nd (adding Saturday, 23rd and Friday, 22nd). If timeSpanType is set to NE_ACTIVE_TIME, the time will be added to working days only. In our example, the time span would start on Thursday, 21st (adding Friday, 22nd and Thursday, 21th).
Return Value	void	

Also see calcEndTime

getCurrentTimeSpan

Method of NeIProfile

This method lets you retrieve the time span that contains the start time of the time span passed.

Declaration
boolean getCurrentTimeSpan (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span containing the start date of the time span searched for.
Return Value	boolean	Returns, whether (**true**) or not (**false**) a current time span exists.

Also see getNextTimeSpan
getPrevTimeSpan

getNextTimeSpan

Method of NeIProfile

This method can be used in an iterative loop. It retrieves the time span following the one passed in the parameter.

Declaration
boolean getNextTimeSpan (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span, of which the following one is to be searched for.
Return Value	boolean	Returns, whether (**true**) or not (**false**) a following time span exists.

Also see getCurrentTimeSpan
getPrevTimeSpan

getPrevTimeSpan

Method of NeIProfile

This method can be used in an iterative loop. It retrieves the time span previous to the one passed in the parameter.

Declaration
boolean getPrevTimeSpan (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span, of which the previous one is to be searched for.
Return Value	boolean	Returns, whether (**true**) or not (**false**) a previous time span exists.

Also see getCurrentTimeSpan
getNextTimeSpan

iterateTimeSpans

Method of NeIProfile

This method lets you retrieve the time span that contains the start time of the time span passed.

Declaration
java.util.Iterator iterateTimeSpans (long start, long end)

	Data Type	Description
Parameter		
start	long	Start of the time section, of which the time spans shall be iterated.
end	long	End of the time section, of which the time spans shall be iterated.
Return Value	java.util.Iterator	

removeTimeSpan

Method of NeIProfile

This method lets you remove the specified time span from a profile.

Declaration
void removeTimeSpan (de.netronic.common.intface.NeITimeSpan timeSpan)

	Data Type	Description
Parameter		
timeSpan	de.netronic.common.intface.-NeITimeSpan	Time span to be removed.
Return Value	void	

Also see addTimeSpan

2.5 NeITimeSpan

Belongs to Calendar

Package name **de.netronic.common.intface**

This interface lets you compose and handle time spans.

Properties to Handle the Time Span

Activity	Degree of activity of a time span
Calendar	Calendar object that the time span is allocated to.
DailyRepetition	The time span as a unique or a recurrent interval
End	The finish date (milliseconds since 1.1.1970) of the time span.
EndDay	Day (1..31) of the finish date of the time span
EndMonth	Month (1..12) of the finish date of the time span.
EndRelative	Number of units by which the finish date of the time span is separated from the calendar start.
EndYear	Year of the finish date of the time span
SpanID	Identifier tag of the time span

Start	Start date (milliseconds since 1.1.1970) of the time span.
StartDay	Day (1..31) of the start date of the time span
StartMonth	Month (1..12) of the start date of the time span
StartRelative	Number of units by which the start date of the time span is separated from the calendar start.
StartYear	Year of the start date of the time span.

Methods for Internal Use only

startTimeToInt(...)	For internal use only.

Methods to Handle the Time Span

endTimeToInt(...)	For internal use only.
getSpan(...)	Beginning and the end of the time span
setEndDate(...)	Sets the finish date of a time span.
setSpan(...)	Beginning and the end of the time span
setStartDate(...)	Sets the start date of a time span.

Properties of the Interface

Activity

Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the degree of activity of a time span. The values of the degree of activity may range between 0 and 1,000. 0 will turn a time span into a non-working time, values from 1..999 represent partial working-times, whereas 1,000 will turn a time span into a full working-time.

Accessing Methods
void setActivity (int newValue)
int getActivity ()

Calendar

Read Only Property of NeITimeSpan

Typ	**de.netronic.common.intface.NeICalendar**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the calendar that the time span is allocated to.

Accessing Methods
de.netronic.common.intface.NeICalendar getCalendar()

DailyRepetition

Property of NeITimeSpan

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property defines whether or not the time span is a unique or a recurrent interval.

The value set here can be overridden by setting the span property (setSpan).

Accessing Methods
void setDailyRepetition (boolean newValue)
boolean getDailyRepetition ()

End

Property of NeITimeSpan

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the finish date (milliseconds since 1.1.1970) of the time span.

Accessing Methods
void setEnd (long newValue)
long getEnd ()

Also see EndMonth
Start

EndDay

Read Only Property of NeITimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This this property defines the day (1..31) of the finish date of the time span.

Accessing Methods
int getEndDay()

Also see EndRelative
EndYear

EndMonth

Read Only Property of NeITimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the month (1..12) of the finish date of the time span.

Accessing Methods
int getEndMonth()

Also see End

EndRelative

Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the number of units by which the finsih date of the time span is separated from the start date of the calendar.

Accessing Methods
void setEndRelative (int newValue)
int getEndRelative ()

Also see StartRelative

EndYear

Read Only Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the year of the finish date of the time span.

Accessing Methods
int getEndYear()

Also see EndDay
EndRelative

SpanID

Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property you describes the identifier tag of the time span.

Accessing Methods
void setSpanID (int newValue)
int getSpanID ()

Start

Property of NelTimeSpan

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the start date (milliseconds since 1.1.1970) of the time span.

Accessing Methods
void setStart (long newValue)
long getStart ()

Also see End
StartRelative

StartDay

Read Only Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the day (1..31) of the start date of the time span.

Accessing Methods
int getStartDay()

Also see StartMonth

StartYear

StartMonth

Read Only Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the month (1..12) of the start date of the time span.

Accessing Methods
int getStartMonth()

Also see StartDay
StartYear

StartRelative

Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the number of units by which the start date of the time span is separated from the start date of the calendar.

Accessing Methods
void setStartRelative (int newValue)
int getStartRelative ()

Also see EndRelative

StartYear

Read Only Property of NelTimeSpan

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes the year of the start date of the time span.

Accessing Methods
int getStartYear()

Also see StartDay
StartMonth

Methods of the Interface

endTimeToInt

Method of NelTimeSpan

For internal use only.

Declaration
int endTimeToInt (int endTime)

	Data Type	Description
Parameter		
endTime	int	For internal use only
Return Value	int	

getSpan

Method of NelTimeSpan

By this method you can retrieve the beginning and the end of the time span and, in addition, whether the time span is of unique or repeating occurrence.
If the sequence for example is 22.5.2010 7:30-25.5.2010 17:00, the time span is of unique occurrence.
If the sequence for example is 22.5.2010-25.5.2010 7:30-17:00, the time span is of repeating occurrence.

The date of the day may appear to have been replaced by defined strings:

- Monday ..Sunday
- First Monday .. First Sunday
- Last Monday ..Last Sunday
- Last day
- *

Examples: "First Wednesday.3.2010" corresponds to the third of March 2010, "Last Day.2.2011" corresponds to 28.2.2011. "Friday.3.2012" defines all Fridays in the month of March in 2012. Using the asterisk "*.4.12" comprises the period from 1.4.2012-30.4.2012.The asterisk can also replace the month or the year. For example *.*.* *:* covers the complete range of the calendar, as defined by the NeICalendar method setValidRange.
Beside, the date of a day may appear to have been replaced by defined names of floating holidays:

- Ash Wednesday
- Good Friday
- Easter Sunday
- Easter Monday
- Feast of Corpus Christ
- Ascension of Christ
- Whit Sunday
- Whit Monday
- Day of Prayer
- Advent

The names listed are self-explaining except for "Advent", which describes the first Sunday of Advent. The Sundays of Advent following can easily be derived by adding 7 or a multiple of 7 to Advent, such as "Advent+14.11", resulting in the third Sunday of Advent on 12.12.2011.

Declaration
int getSpan (java.lang.String newValue)

	Data Type	Description
Parameter		
newValue	java.lang.String	Character string containing the start and the end date of the time span.
Return Value	int	If the return value equals 0, the method was performed successfully.

Also see setSpan

setEndDate

Method of NelTimeSpan

Sets the finish date of a time span. In contrast to the method **setSpan**, this method allows to set the finish date separately. Correspondingly, the start date can be set by the method **setStartDate**.

Declaration
int setEndDate (int endYear, int endMonth, int endDay, int dayOffset)

	Data Type	Description
Parameter		
endYear	int	Year of the finish date
endMonth	int	Month of the finish date
endDay	int	Day of the finish date
dayOffset	int	Number of days for an additional offset, for example 2.5.2003 + 3 (days). This parameter is useful for floating holidays.
Return Value	int	Always 0.

Also see setStartDate

setSpan

Method of NelTimeSpan

By this method you can set the beginning and the end of the time span and, in addition, whether the time span is of unique or repeating occurrence.
If the sequence for example is 22.5.2010 7:30-25.5.2010 17:00, the time span is of unique occurrence.
If the sequence for example is 22.5.2010-25.5.2010 7:30-17:00, the time span is of repeating occurrence.
The date of the day may appear to have been replaced by defined strings:

- Monday ..Sunday
- First Monday .. First Sunday
- Last Monday ..Last Sunday
- Last day
- *

Examples: "First Wednesday.3.2010" corresponds to the third of March 2010, "Last Day.2.2011" corresponds to 28.2.2011. "Friday.3.2012" defines all Fridays in the month of March in 2012. Using the asterisk "*.4.12" comprises the period from 1.4.2012-30.4.2012.The asterisk can also replace the month or the year. For example *.*.* *:* covers the complete range of the calendar, as defined by the NeICalendar method setValidRange.
Beside, the date of a day may appear to have been replaced by defined names of floating holidays:

- Ash Wednesday
- Good Friday
- Easter Sunday
- Easter Monday
- Feast of Corpus Christ
- Ascension of Christ
- Whit Sunday
- Whit Monday
- Day of Prayer
- Advent

The names listed are self-explaining except for "Advent", which describes the first Sunday of Advent. The Sundays of Advent following can easily be derived by adding 7 or a multiple of 7 to Advent, such as "Advent+14.11", resulting in the third Sunday of Advent on 12.12.2011.

Declaration
int setSpan (java.lang.String newValue)

	Data Type	**Description**
Parameter		
newValue	java.lang.String	Character string containing the start and the end date of the time span.
Return Value	int	If the return value equals 0, the method was performed successfully.

Also see getSpan

setStartDate

Method of NelTimeSpan

Sets the start date of a time span. In contrast to the method setSpan, this method allows to set the start date separately. Correspondingly, the finish date and the repetition flag can also be set separately by the corresponding methods.

Declaration
int setStartDate (int startYear, int startMonth, int startDay, int dayOffset)

	Data Type	Description
Parameter		
startYear	int	Year of the start date
startMonth	int	Month of the start date
startDay	int	Day of the start date
dayOffset	int	Number of days for an additional offset, for example 2.5.2006 + 3 (days). This parameter is useful for floating holidays.
Return Value	int	Always 0.

Also see setEndDate

startTimeToInt

Method of NelTimeSpan

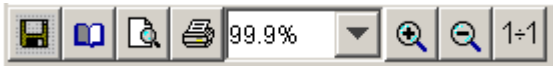
For internal use only.

Declaration
int startTimeToInt (int startTime)

	Data Type	Description
Parameter		
startTime	int	For internal use only
Return Value	int	

3 DiagramControlPanel

This component contains a single class called NeDiagramControlPanel to handle the control panel of the diagram.



The DiagramControlPanel-Component consists of the below classes:

NeDiagramControlPanel	This class lets you handle the control panel of the diagram.

3.1 NeDiagramControlPanel

Belongs to DiagramControlPanel

Package name	**de.netronic.bean.diagramcontrolpanel**
Extends	**javax.swing.JComponent**
Implements	**java.beans.PropertyChangeListener**
I	**java.io.FilenameFilter**

This class lets you handle the control panel of the diagram.

Properties to Handle the Push Buttons

ButtonFont	Font for the user defined buttons.
ButtonSize	The size in pixels of the buttons in the diagram control bar
CurrentDirectory	Absolute path of the current directory
InfoText1	Information string in the void part of the control panel
InfoText2	Information string in the void section of the control panel

Methods for Internal Use only

accept(...)	For internal use only

Methods to Handle the Push Buttons

getButtonEnabled(...)	Retrieves, whether a button is enabled.

getButtonEvent(...)	'Faked' property change event of a button
getButtonIcon(...)	Retrieves the icon of a button
getButtonPressed(...)	Retrieves the state of a button
getButtonPressedIcon(...)	Retrieves the icon to indicate the "pressed" state of a button.
getButtonText(...)	Retrieves the text of a button.
getButtonToolTipText(...)	Retrieves the tool tip text of a button.
getButtonVisible(...)	Retrieves whether a button is visible
openFileDialog()	Invokes the **file open** dialog
propertyChange(...)	Change of a property in the diagram control panel
saveFileDialog()	Invokes the **save file** dialog
setButtonEnabled(...)	Enables/disables buttons
setButtonEvent(...)	'Faked' property change event
setButtonIcon(...)	Sets the icon of a button
setButtonPressed(...)	Sets a button to pressed/unpressed
setButtonPressedIcon(...)	Assigns the icon to indicate the "pressed" state of a button.
setButtonText(...)	Sets the text of a button.
setButtonToolTipText(...)	Sets the tool tip text of a button.
setButtonVisible(...)	Sets the button visible/unvisible
updateScaleFactorEntry(...)	Updates the scale factor in the combo box.

Constructors of the Class

NeDiagramControlPanel

Constructor of NeDiagramControlPanel

This constructor lets you generate a diagram control panel. By default, it will show the buttons

- NE_FILE_SAVE_BUTTON
- NE_PAGE_FORMAT_BUTTON
- NE_PREVIEW_BUTTON
- NE_PRINT_BUTTON
- NE_SCALE_FACTOR_COMBO_BOX

- NE_ZOOM_IN_BUTTON, NE_ZOOM_OUT_BUTTON
- NE_1TO1_BUTTON

Other buttons need to be added by the methods offered by the class.

Declaration
NeDiagramControlPanel ()

Properties of the Class

ButtonFont

Property of NeDiagramControlPanel

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can specify a font for the texts of your userdefined buttons.

Accessing Methods
void setButtonFont (java.awt.Font newValue)
java.awt.Font getButtonFont ()

Example Code

```
myControlPanel.setButtonFont (new Font ("Times New Roman", Font.PLAIN, 20));
```

ButtonSize

Property of NeDiagramControlPanel

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**27**

By this property you can specify the size of the buttons in the diagram control bar.

Accessing Methods
void setButtonSize (int newValue)
int getButtonSize ()

Example Code

```
((NeDiagramControlPanel)jGantt1.getDiagramControlBar()).setButtonSize(40);
```

CurrentDirectory

Property of NeDiagramControlPanel

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Absolute path of the current directory for the methods **openFileDialog** and **saveFileDialog**.

Accessing Methods
void setCurrentDirectory (java.lang.String newValue)
java.lang.String getCurrentDirectory ()

InfoText1

Property of NeDiagramControlPanel

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Information string (the left one out of two possible strings) in the void part of the control panel.

Accessing Methods
void setInfoText1 (java.lang.String newValue)
java.lang.String getInfoText1 ()

Also see InfoText2

InfoText2

Property of NeDiagramControlPanel

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Information string (the right one out of two possible strings) in the void section of the control panel.

Accessing Methods
void setInfoText2 (java.lang.String newValue)
java.lang.String getInfoText2 ()

Also see InfoText1

Methods of the Class

accept

Method of NeDiagramControlPanel

For internal use only

Declaration
boolean accept (java.io.File dir, java.lang.String name)

	Data Type	Description
Parameter		
dir	java.io.File	For internal use only
name	java.lang.String	For internal use only
Return Value	boolean	

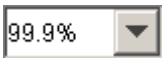
getButtonEnabled

Method of NeDiagramControlPanel

This method lets you retrieve, whether a button is enabled (**true**) or disabled (**false**).

Declaration
boolean getButtonEnabled (int button)

	Data Type	Description
Parameter		
button	int	Push button the state of which is to be retrieved.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_- RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	boolean	**True**: the button is enabled, **false**: the button is disabled.

Also see getButtonVisible
setButtonEnabled

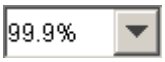
getButtonEvent

Method of NeDiagramControlPanel

Retrieves the name of a 'faked' property change event that is generated by the diagram control panel when a button was pressed.

Declaration
java.lang.String getButtonEvent (int button)

	Data Type	Description
Parameter		
button	int	Button pressed
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	java.lang.String	

Also see setButtonEvent

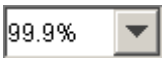
getButtonIcon

Method of NeDiagramControlPanel

This method lets you retrieve the icon of a button in the diagram control panel.

Declaration
javax.swing.ImageIcon getButtonIcon (int button)

	Data Type	Description
Parameter		
button	int	Push button of which the icon is to be retrieved.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	javax.swing.ImageIcon	Icon retrieved

Also see getButtonPressedIcon
setButtonIcon

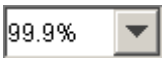
getButtonPressed

Method of NeDiagramControlPanel

This method lets you retrieve the state of a button of the diagram control panel.

Declaration
boolean getButtonPressed (int button)

	Data Type	Description
Parameter		
button	int	Push button the state of which is to be retrieved.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_- RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	boolean	**True**: the button appears pressed, **false**: the button appears not to be pressed.

Also see setButtonPressed

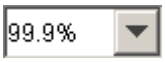
getButtonPressedIcon

Method of NeDiagramControlPanel

This method lets you retrieve the icon to indicate the "pressed" state of a button in the diagram control panel.

Declaration
javax.swing.ImageIcon getButtonPressedIcon (int button)

	Data Type	Description
Parameter		
button	int	Button of which the icon is to be retrieved.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	javax.swing.ImageIcon	Icon retrieved

Also see getButtonPressed
setButtonPressedIcon

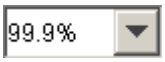
getButtonText

Method of NeDiagramControlPanel

This method lets you retrieve the text of a button of the diagram control panel.

Declaration
void getButtonText (int button)

	Data Type	Description
Parameter		
button	int	Push button of which the text is to be retrieved.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_- RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	void	Text retrieved

Also see getButtonToolTipText
setButtonText
setButtonToolTipText

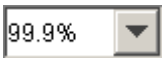
getButtonToolTipText

Method of NeDiagramControlPanel

This method lets you retrieve the tool tip text of a button of the diagram control panel.

Declaration
java.lang.String getButtonToolTipText (int button)

	Data Type	Description
Parameter		
button	int	Push button of which the tool tip text is to be retrieved.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	java.lang.String	Tool tip text retrieved

Also see setButtonToolTipText

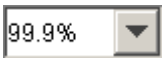
getButtonVisible

Method of NeDiagramControlPanel

This method lets you retrieve whether or not a button of the diagram control panel is visible.

Declaration
boolean getButtonVisible (int button)

	Data Type	Description
Parameter		
button	int	Push button the visibility of which is to be retrieved.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_- RECT_BUTTON	 Button to zoom an area of the chart.
Return Value	boolean	**True**: the button is visible, **false**: the button is invisible.

Also see setButtonVisible

openFileDialog

Method of NeDiagramControlPanel

This method lets you invoke the **file open** dialog.

Declaration
void openFileDialog ()

	Data Type	Description
Return Value	void	

Also see saveFileDialog

propertyChange

Method of NeDiagramControlPanel

By this method you can pass the event that evoked the change of a property in the diagram control panel object.

Declaration
void propertyChange (java.beans.PropertyChangeEvent evt)

	Data Type	Description
Parameter		
evt	java.beans.PropertyChangeEvent	Event that evoked the change of the property.
Return Value	void	

saveFileDialog

Method of NeDiagramControlPanel

This method lets you invoke the **save file** dialog.

Declaration
void saveFileDialog ()

	Data Type	Description
Return Value	void	

Also see openFileDialog

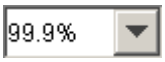
setButtonEnabled

Method of NeDiagramControlPanel

This method lets you enable or disable a button of the diagram control panel.

Declaration
void setButtonEnabled (int button, boolean newState)

	Data Type	Description
Parameter		
button	int	Button to be set to **enabled** or to **disabled**.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_- RECT_BUTTON	 Button to zoom an area of the chart.
newState	boolean	**True**: the button will be enabled, **false**: the button will be disabled.
Return Value	void	

Also see getButtonEnabled
setButtonVisible

Example Code

```
// Get instance of ControlPanel from JGantt
myControlPanel = (NeDiagramControlPanel) jGantt.getDiagramControlBar();

// Enable some buttons
myControlPanel.setButtonEnabled(myControlPanel.NE_TITLE_BUTTON, true);
myControlPanel.setButtonEnabled(myControlPanel.NE_LEFT_ROW_HEADER_BUTTON, true);

// Restoring ControlPanel
myControlPanel.init();
```

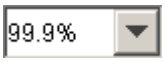
setButtonEvent

Method of NeDiagramControlPanel

Sets the name of a 'faked' property change event that is generated by the diagram control panel when a button is pressed.

Declaration
void setButtonEvent (int button, java.lang.String newValue)

	Data Type	Description
Parameter		
button	int	Button pressed
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
newValue	java.lang.String	New value generated by pressing the button.
Return Value	void	

Also see getButtonEvent

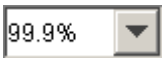
setButtonIcon

Method of NeDiagramControlPanel

This method lets you set the icon of a button in the diagram control panel.

Declaration
void setButtonIcon (int button, javax.swing.ImageIcon icon)

	Data Type	Description
Parameter		
button	int	Button of which the icon is to be set.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
icon	javax.swing.ImageIcon	Icon to be assigned
Return Value	void	

Also see getButtonIcon
setButtonPressedIcon

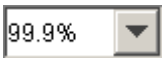
setButtonPressed

Method of NeDiagramControlPanel

This method lets you initially set a button of the panel to pressed or to unpressed at the start of the program.

Declaration
void setButtonPressed (int button, boolean newState)

	Data Type	Description
Parameter		
button	int	Push button to be set to pressed or unpressed.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
newState	boolean	**True**: the button appears pressed, **false**: the button appears not to be pressed.
Return Value	void	

Also see getButtonPressed

Example Code

```
// Get instance of ControlPanel from JGantt
```

```
myControlPanel = (NeDiagramControlPanel) jGantt.getDiagramControlBar();

// Set some buttons to pressed mode
myControlPanel.setButtonPressed(myControlPanel.NE_TITLE_BUTTON, true);
myControlPanel.setButtonPressed(myControlPanel.NE_LEFT_ROW_HEADER_BUTTON, true);

// Restoring ControlPanel
myControlPanel.init();
```

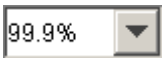
setButtonPressedIcon

Method of NeDiagramControlPanel

This method lets you assign the icon to indicate the "pressed" state of a button in the diagram control panel.

Declaration
void setButtonPressedIcon (int button, javax.swing.ImageIcon icon)

	Data Type	Description
Parameter		
button	int	Push button of which the icon is to be set.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
icon	javax.swing.ImageIcon	Icon to be assigned
Return Value	void	

Also see getButtonPressedIcon
setButtonIcon

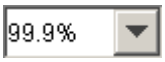
setButtonText

Method of NeDiagramControlPanel

This method lets you set the text of a button (which is to be specified) of the diagram control panel.

Declaration
void setButtonText (int button, java.lang.String text)

	Data Type	Description
Parameter		
button	int	Push button of which the text is to be set.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
text	java.lang.String	Text to be set
Return Value	void	

Also see getButtonText
getButtonToolTipText
setButtonToolTipText

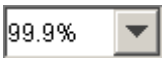
setButtonToolTipText

Method of NeDiagramControlPanel

This method lets you set the tool tip text of a button (which is to be specified) of the diagram control panel.

Declaration
java.lang.String setButtonToolTipText (int button, java.lang.String toolTipText)

	Data Type	Description
Parameter		
button	int	Push button of which the tool tip text is to be set.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_-RECT_BUTTON	 Button to zoom an area of the chart.
toolTipText	java.lang.String	Tool tip text to be set
Return Value	java.lang.String	

Also see getButtonToolTipText

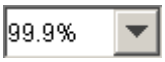
setButtonVisible

Method of NeDiagramControlPanel

This method lets you make a button of the diagram control panel visible or unvisible..

Declaration
void setButtonVisible (int button, boolean newState)

	Data Type	Description
Parameter		
button	int	Push button to be set to visible or invisible.
	Possible Values:	
	NE_1TO1_BUTTON	 Button to display the chart in its original size.
	NE_BACKWARD_BUTTON	 Button to recall a preceding interaction.
	NE_DIAGRAM_BUTTON	 Button to display the complete view of the diagram.
	NE_EXPLORER_BUTTON	 Button to indicate the directory structure.
	NE_FILE_OPEN_BUTTON	 Button to open a file.
	NE_FILE_SAVE_BUTTON	 Button to save a file.
	NE_FORWARD_BUTTON	 Button to recall a succeeding interaction.
	NE_HELP_BUTTON	 Button to invoke the online-help.
	NE_HISTOGRAM_BUTTON	 Button to toggle the histogram.
	NE_HOME_BUTTON	 Button to re-display the first page.
	NE_LEFT_ROW_HEADER_BUTTON	 Button to toggle the row titles of the table on the left.
	NE_LEFT_TABLE_BUTTON	 Button to toggle the table on the left.

NE_LEGEND_BUTTON	 Button to toggle the legend.
NE_LOWER_TIMESCALE_BUTTON	 Button to toggle the bottom time scale.
NE_NEW_BUTTON	 Button to create a new diagram.
NE_OPTIONS_BUTTON	 Button to invoke more options or tools.
NE_PAGE_FORMAT_BUTTON	 Button to invoke the **page format** dialog.
NE_PANES_BUTTON	 Button to display the panes of the chart.
NE_PREVIEW_BUTTON	 Button to invoke the page setup.
NE_PRINT_BUTTON	 Button to print the file.
NE_RIGHT_ROW_HEADER_-BUTTON	 Button to toggle the row titles of the table on the right.
NE_RIGHT_TABLE_BUTTON	 Button to toggle the table on the right.
NE_SCALE_FACTOR_COMBO_BOX	 Input and output field for the scaling factor (in percent).
NE_TITLE_BUTTON	 Button to toggle the diagram title.
NE_UPPER_TIMESCALE_BUTTON	 Button to toggle the top time scale.
NE_USER_BUTTON0	User-defined button No. 0 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON1	User-defined button No. 1 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).

NE_USER_BUTTON2	User-defined button No. 2 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON3	User-defined button No. 3 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON4	User-defined button No. 4 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON5	User-defined button No. 5 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON6	User-defined button No. 6 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON7	User-defined button No. 7 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON8	User-defined button No. 8 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_BUTTON9	User-defined button No. 9 of the control panel. A user-defined button serves to trigger an action (button does not remain in the pressed position).
NE_USER_TOGGLE_BUTTON0	User-defined toggle button No. 0 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON1	User-defined toggle button No. 1 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON2	User-defined toggle button No. 2 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
NE_USER_TOGGLE_BUTTON3	User-defined toggle button No. 3 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).

	NE_USER_TOGGLE_BUTTON4	User-defined toggle button No. 4 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON5	User-defined toggle button No. 5 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON6	User-defined toggle button No. 6 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON7	User-defined toggle button No. 7 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON8	User-defined toggle button No. 8 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_USER_TOGGLE_BUTTON9	User-defined toggle button No. 9 of the control panel. A user-defined toggle button serves to toggle between two states (button remains in the pressed position).
	NE_ZOOM_IN_BUTTON	 Button to enlarge the chart.
	NE_ZOOM_OUT_BUTTON	 Button to downsize the chart.
	NE_ZOOM_RUBBER_- RECT_BUTTON	 Button to zoom an area of the chart.
newState	boolean	**True**: the button is visible, **false**: the button is invisible.
Return Value	void	

Also see setButtonEnabled
getButtonVisible

updateScaleFactorEntry

Method of NeDiagramControlPanel

This method lets you update scaling factor in the combo box NE_SCALE_-FACTOR_COMBO_BOX to indicate the proper value, for example after zooming.

Declaration
void updateScaleFactorEntry (double scaleFactor)

	Data Type	Description
Parameter		
scaleFactor	double	Scaling factor, by which the field is to be updated
Return Value	void	

4 GanttGraph

This component contains interfaces that you need to set up the Gantt Graph.

The GanttGraph-Component consists of the below classes:

JGIGanttGraph	This interface lets you configure the Gantt graph.
NeIHorLineGrid	This interface provides methods and properties to handle a single horizontal line grid.
NeIHorLineGrids	This interface provides methods and properties to handle horizontal line grids.
NeILayerDefinition	This interface lets you to define a layer.
NeILayouterGroup	This interface lets you handle groups.
NeILinkDefinition	This interface lets you define a link definition.
NeMouseObserverEvent	This class offers a constructor to handle events of the mouse.
NeMouseObserverListener	This is the listener interface for receiving mouse events.

4.1 JGIGanttGraph

Belongs to GanttGraph

Package name **de.netronic.jgantt**

This interface lets you configure the Gantt graph.

Properties to Handle Info Windows

InfoWindowDateFormat	Replaces the default date format in the information window of the Gantt graph.
InfoWindowDecimalFormat	Replaces the default decimal format in the information window of the Gantt graph.
InfoWindowEndDateFormat	Replaces the default date format for the enddate in the information window of the Gantt graph.
InfoWindowShowGroupCode	The group codes of the active row are displayed.
InfoWindowShowStartEndAndDuration	Start, duration and end of a layer or node are displayed.

InfoWindowTexts	Replaces the default texts of the information window.

Properties to Handle Interactions

AntialiasSymbols	Antialiasing on symbols
AntialiasText	Antialiasing on texts
AutoScrollMargin	Auto-scroll area
MenuActions	Actions for the context menu
MousePosAtLastPopUpTrigger	Mouse position of the pop up menu most recently invoked
NodeModifyInteractionsOnLabelsEnabled	Interactions with nodes via symbols and annotations

Properties to Handle Links

DirectLinksEnabled	Straight links between "single" and "single-optimized" nodes
LineStyle	Line color of the links
LinkArrowSize	Setting the size when shaping a link tip as an arrow
LinkArrowType	Shaping a link end as an arrow tip
LinkBendStyle	Style for the direction changes of link lines
LinkLineStyle	Line color of the links
LinkNodeEndReference	ValueReference object for the end of a link
LinkNodeStartReference	ValueReference object for the beginning of a link
LinkOffset	Minimum distance between collateral links
LinksForNodesNotInSeparateRowsEnabled	Links between "single" and "single-optimized" nodes
LinksInFrontOfNodes	Links between "single" and "single-optimized" nodes remain visible
MarkedLineStyle	Line color of the marked links
MarkedLinkLineStyle	Line color of the marked links

Properties to Handle Markings

NodeMarkAreaStyle	Marks a node by an area
NodeMarkLineStyle	Color of the node marking
NodeMarkStyle	Mark style of nodes
PickMarkSize	Size of the pickmarks

RectangleMarkOffset	Offset of the marking rectangle

Properties to Handle Objects in the Gantt Graph

DateLinesInFrontOfNodesAndLinks	DateLines are positioned in front of nodes and links or they are hidden by them
GlassProfileInFrontOfNodesAndLinks	The workfree times of the glass profile are positioned in front of nodes and links or they are partly hidden by them
NumberOfLinks	Number of links in the Gantt graph
NumberOfNodes	Number of nodes in the Gantt graph
VerticalNodeAlignmentInsideRow	Vertical allignment of the nodes in a row of the Gantt graph, in case the row height is larger than height of the nodes

Properties to Handle Phantoms

CreatePhantomAreaStyle	Area color of the phantom for creating objects
CreatePhantomLineStyle	Line color of the phantom for creating objects
LinkPhantomLineStyle	Line color of the phantom for moving links
NodeMovePhantomAreaStyle	Area color of the phantom for node moving
NodeMovePhantomLineStyle	Line color of the phantom for node moving
NodeResizePhantomAreaStyle	Area color of the phantom for node resizing
NodeResizePhantomLineStyle	Line color of the phantom for node resizing
SelectPhantomAreaStyle	Area color of the phantom for selecting objects
SelectPhantomLineStyle	Line color of the phantom for selecting objects

Properties to Handle Rows in the Gantt Graph

IgnoredByOptimizationNodeFilter	The **optimized** arrangement will be ignored by the nodes matching this filter.
NodeDrawingPriorityComparator	Specifies a comparator that defines the drawing sequence of nodes in a group with RowLayout GROUP_ROWLAYOUT_SINGLE.
SubRowBy	Node arrangement based on JGantt.GROUP_ROWLAYOUT_SINGLE_OPTIMIZED
SubRowComparator	Specifies a comparator that defines the sequence of the subrows, according to which they should be filled by nodes.

SubRowMargin	Offset between the row border and a node in a sub row (when optimized group layout)
SubRowMinimalHeight	Minimum height of a sub row (when optimized group layout)

Methods to Handle Interactions

addMouseObserverListener(...)	Adds a listener for events of mouse movements.
createValueReference(...)	Generates a value reference object.
getAction(...)	Retrieves an action from the action map.
putAction(...)	Adds an action
removeAction(...)	Removes an action
removeMouseObserverListener(...)	Removes an existing listener for events of mouse movements.
scrollToEntity(...)	Scrolls the view for a node or a link to become visible completely.

Methods to Handle Objects in the Gantt Graph

createLayerDefinition(...)	Generates a layer definition.
getHorLineGrids()	Retrieves the line grid of the type NeIHorLineGrids that belongs to the Gantt graph.
getLayerDefinition(...)	Retrieves a layer definition by its ID.
getLinkDefinition(...)	Retrieves a link definition by its ID.
isCompletelyOnScreen(...)	Retrieves, whether or not the object represented by the entity is completely displayed on the screen.
isOnScreen(...)	Retrieves, whether or not the object represented by the entity is displayed on the screen.
iterateLayerDefinitions()	Returns an iterator object on existing layer definitions.
iterateLinkDefinitions()	Returns an iterator object on existing linkdefinitions.
iterateLinkEntities(...)	Returns an iterator object on existing link entities
iterateNodeEntities(...)	Returns an iterator object on existing node entities
removeLayerDefinition(...)	Removes a layer definition.
removeLinkDefinition(...)	Removes a linkdefinition.

Properties of the Interface

AntialiasSymbols

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Antialiasing on symbols can be switched on or off.

Accessing Methods
void setAntialiasSymbols (boolean newValue)
boolean isAntialiasSymbols ()

Also see AntialiasText

AntialiasText

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Antialiasing on text strings can be switched on or off.

Accessing Methods
void setAntialiasText (boolean newValue)
boolean isAntialiasText ()

Also see AntialiasSymbols

AutoScrollMargin

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**25**

This property lets you define an area along the border of the Gantt graph (view), in which auto-scrolling is triggered by mouse motion. Unit: pixels. Auto-scrolling starts automatically as soon as the mouse approaches the border as close as - for example - 25 pixels (default value). Positive values define an inner area; the value set could also be zero or negative. In the latter case auto-scrolling starts when the mouse leaves the Gantt graph.

Accessing Methods
void setAutoScrollMargin (int newValue)
int getAutoScrollMargin ()

CreatePhantomAreaStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Area color of the phantom for creating objects. When using NeAreaStyle instead of the Color class as a color data type, the fill patterns of that class are available.

Accessing Methods
void setCreatePhantomAreaStyle (java.awt.Color newValue)
java.awt.Color getCreatePhantomAreaStyle ()

CreatePhantomLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the phantom for creating objects. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported).

Accessing Methods
void setCreatePhantomLineStyle (java.awt.Color newValue)
java.awt.Color getCreatePhantomLineStyle ()

DateLinesInFrontOfNodesAndLinks

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

If the value of this property is set to **true**, date lines will be drawn in front of nodes and links. This way, all parts of a date line remain visible at any time. If the value in contrast is set to **false**, nodes and links will hide date lines. This way, only parts of a date line may remain visible.

Accessing Methods
void setDateLinesInFrontOfNodesAndLinks (boolean newValue)
boolean hasDateLinesInFrontOfNodesAndLinks ()

DirectLinksEnabled

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you display straight links from or to activities that were positioned by the options GROUP_ROWLAYOUT_SINGLE oder GROUP_ROWLAYOUT_SINGLE_OPTIMIZED. In most cases, links are routed with their horizontal section paralleling the top line border. By setting this property, links of the type **end-start** can be displayed straight from the end of the predecessor node to the start of the successor node in the same row.

Accessing Methods
void setDirectLinksEnabled (boolean newValue)
boolean isDirectLinksEnabled ()

Also see LinksForNodesNotInSeparateRowsEnabled
LinksInFrontOfNodes

GlassProfileInFrontOfNodesAndLinks

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

The workfree times of the glass profile are positioned in front of nodes and links or they are partly hidden by them

Accessing Methods
void setGlassProfileInFrontOfNodesAndLinks (boolean newValue)
boolean hasGlassProfileInFrontOfNodesAndLinks ()

Example Code

```
jGantt1.getGanttGraph().setGlassProfileInFrontOfNodesAndLinks(true);
```

IgnoredByOptimizationNodeFilter

Property of JGIGanttGraph

Typ	**de.netronic.common.intface.NelFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Nodes that fulfill the conditions of the filter passed are put into the first sub-row. They are not subject to layout, so they may hide each other. Even if many nodes eclipse each other, no sub-row will be generated. Other nodes that do not match

the filter conditions, may be positioned and may also hide the nodes in the first sub-row.

Accessing Methods
void setIgnoredByOptimizationNodeFilter (de.netronic.common.intface.NeIFilter newValue)
de.netronic.common.intface.NeIFilter getIgnoredByOptimizationNodeFilter ()

Also see SubRowBy
SubRowComparator

InfoWindowDateFormat

Property of JGIGanttGraph

Typ	**java.text.DateFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you replace the default date format in the information window of the Gantt graph. The information window is invoked on interactions of the user. The default date format is pre-defined by the locale settings of Java.

Accessing Methods
void setInfoWindowDateFormat (java.text.DateFormat newValue)
java.text.DateFormat getInfoWindowDateFormat ()

InfoWindowDecimalFormat

Property of JGIGanttGraph

Typ	**java.text.DecimalFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you replace the default decimal format in the information window of the Gantt graph. The information window is invoked on interactions of the user. The default decimal format is #.##.

Accessing Methods
void setInfoWindowDecimalFormat (java.text.DecimalFormat newValue)
java.text.DecimalFormat getInfoWindowDecimalFormat ()

InfoWindowDurationUnit

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**NE_INFO_WINDOW_DAYS**

In case you set the JGIGanttGraph property **InfoWindowShowStartEndAndDuration**, this property allows to assign the unit of the duration displayed by the InfoWindow.

Possible Values	Description
NE_INFO_WINDOW_DAYS	The InfoWindow displays the duration of the layer in the unit of days.
NE_INFO_WINDOW_HOURS	The InfoWindow displays the duration of the layer in the unit of hours.
NE_INFO_WINDOW_MINUTES	The InfoWindow displays the duration of the layer in the unit of minutes.
NE_INFO_WINDOW_SECONDS	The InfoWindow displays the duration of the layer in the unit of seconds.

Accessing Methods
void setInfoWindowDurationUnit (int newValue)
int getInfoWindowDurationUnit ()

Example Code

```
jGantt1.getGanttGraph().setInfoWindowDurationUnit(JGIGanttGraph.NE_INFO_WINDOW_HOU
RS);
```

InfoWindowEndDateFormat

Property of JGIGanttGraph

Typ	**java.text.DateFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property lets you replace the default date format for the end date in the information window of the Gantt graph. The information window is invoked on interactions of the user. The default date format is pre-defined by the locale settings of Java.

Accessing Methods
void setInfoWindowEndDateFormat (java.text.DateFormat newValue)
java.text.DateFormat getInfoWindowEndDateFormat ()

Example Code

```
jGantt1.getGanttGraph().setInfoWindowEndDateFormat(new LbDateFormat ());
```

InfoWindowShowGroupCode

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

The group codes of the active row are displayed in an additional row of the InfoWindow.

Accessing Methods
void setInfoWindowShowGroupCode (boolean newValue)
boolean hasInfoWindowShowGroupCode ()

Example Code

```
jGantt1.getGanttGraph().setInfoWindowShowGroupCode(true);
```

InfoWindowShowStartEndAndDuration

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

If you set this property to **True**, the infoWindow will display the start, the end and the duration of the corresponding layer or node.

Accessing Methods
void setInfoWindowShowStartEndAndDuration (boolean newValue)
boolean hasInfoWindowShowStartEndAndDuration ()

Also see InfoWindowDurationUnit

Example Code

```
jGantt1.getGanttGraph().setInfoWindowShowStartEndAndDuration(true);
```

InfoWindowTexts

Property of JGIGanttGraph

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you replace the default texts of the information window that occurs on interactions in the Gantt Graph. Replacing the texts is necessary for example if your customer speaks a different language.

Accessing Methods

```
void  setInfoWindowTexts (integer index, java.lang.String newValues)
void  setInfoWindowTexts (java.lang.String[] newValue)
java.lang.String  getInfoWindowTexts (integer index)
java.lang.String[]  getInfoWindowTexts ()
```

Example Code

```
Sample 1:
// Replaces the default string "Duration" by the string "Number of hours":
JGIGanttGraph.setInfoWindowTexts (INFO_WINDOW_DURATION, "Number of hours");

Sample 2:
/* Sets a string to invisible (to make it visible again, it needs to be
set anew):*/
JGIGanttGraph.setInfoWindowTexts (INFO_WINDOW_MOVE_LAYER,
INFO_WINDOW_LINE_DISABLED);
```

LinkArrowSize

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**100**

This property lets you set the size of the arrow type set to the ends of links by the property set/getLinkArrowType. The unit is 1/100 mm and describes the side length of the quadrat that encloses the arrow tip.

Accessing Methods
void setLinkArrowSize (int newValue)
int getLinkArrowSize ()

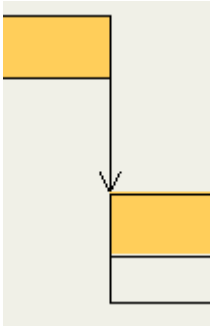
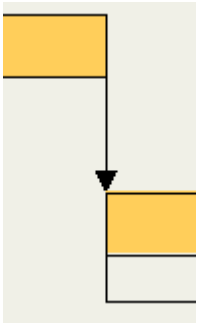
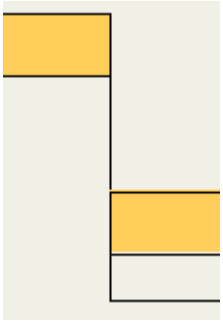
Also see LinkArrowType

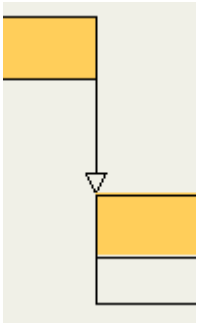
LinkArrowType

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**LINK_ARROW_TYPE_ANGLE**

This property lets you assign an arrow tip to the ends of links.

Possible Values	Description
LINK_ARROW_TYPE_ANGLE	The tips of the links receive an arrow that is shaped like an angle. 
LINK_ARROW_TYPE_FILLED_TRIANGLE	The tips of the links receive an arrow that is shaped like a filled triangle. 
LINK_ARROW_TYPE_NONE	The tips of the links do not receive an arrow shape. 
LINK_ARROW_TYPE_TRIANGLE	The tips of the links receive an arrow that is shaped like an empty triangle.



Accessing Methods
void setLinkArrowType (int newValue)
int getLinkArrowType ()

Also see LinkArrowSize

LinkBendStyle

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**LINK_BEND_STYLE_RECT**

When a link line changes its direction, this can be depicted by a orthoganal corner, a slant or a quadrant.

Possible Values	Description
LINK_BEND_STYLE_RECT	When the link line changes its direction, the change is done by orthogonal lines.
LINK_BEND_STYLE_ROUND	When the link line changes its direction, the change is done by a quadrant.
LINK_BEND_STYLE_SLANT	When the link line changes its direction, the change is done by a slant.

Accessing Methods
void setLinkBendStyle (int newValue)
int getLinkBendStyle ()

LinkLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the linkss. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported).

Accessing Methods
void setLinkLineStyle (java.awt.Color newValue)
java.awt.Color getLinkLineStyle ()

LinkNodeEndReference

Property of JGIGanttGraph

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set a ValueReference object for the end of a link. Alternatively, you can use the JGantt property **LinkNodeDateIndexes** combined with **NodeDates** to set an end date to the link. We recommend to use the valueReference object, if all nodes are defined by layer definitions and if the JGantt property **NodeDates** is not used at all. You should not use **LinkNodeDateIndexes** and this property simultaneously.

Accessing Methods
void setLinkNodeEndReference (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getLinkNodeEndReference ()

LinkNodeStartReference

Property of JGIGanttGraph

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set a ValueReference object for the beginning of a link. Alternatively, you can use the JGantt property **LinkNodeDateIndexes** combined with **NodeDates** to set a start date to the link. We recommend to use the valueReference object, if all nodes are defined by layer definitions and if the JGantt property **NodeDates** is not used at all. You should not use **LinkNodeDateIndexes** and this property simultaneously.

Accessing Methods
void setLinkNodeStartReference (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getLinkNodeStartReference ()

LinkOffset

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Minimum distance between collateral links or between a link and a collateral layer. Unit: 1/100 mm.

Accessing Methods
void setLinkOffset (int newValue)
int getLinkOffset ()

LinkPhantomLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the phantom for moving links

Accessing Methods
void setLinkPhantomLineStyle (java.awt.Color newValue)
java.awt.Color getLinkPhantomLineStyle ()

LinksForNodesNotInSeparateRowsEnabled

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you display links from or to activities, that were positioned by the options GROUP_ROWLAYOUT_SINGLE oder GROUP_ROWLAYOUT_SINGLE_OPTIMIZED. In most cases, links are routed with their horizontal section paralleling the top line border. By setting the property **DirectLinksEnabled**, links of the type **end-start** can be displayed straight from the end of the predecessor node to the start of the successor node in the same row. Nodes positioned between the successor and the predecessor nodes are cut by the link. By setting the property **LinksInFrontOfNodes** you can make the links appear in front, so they remain visible.

Accessing Methods
void setLinksForNodesNotInSeparateRowsEnabled (boolean newValue)
boolean isLinksForNodesNotInSeparateRowsEnabled ()

Also see DirectLinksEnabled
LinksInFrontOfNodes

LinksInFrontOfNodes

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you display links from or to activities, that were positioned by the options GROUP_ROWLAYOUT_SINGLE oder GROUP_ROWLAYOUT_SINGLE_OPTIMIZED. In most cases, links are routed with their horizontal section paralleling the top line border. By setting the property DirectLinksEnabled, links of the type end-start can be displayed straight from the end of the predecessor node to the start of the successor node in the same row.

Nodes positioned between the successor and the predecessor nodes are cut through by the link. By setting this property to **true** you can make the links appear in front, so they remain visible.

Accessing Methods
void setLinksInFrontOfNodes (boolean newValue)
boolean hasLinksInFrontOfNodes ()

Also see DirectLinksEnabled
LinksForNodesNotInSeparateRowsEnabled

MarkedLinkLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the marked links. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported).

Accessing Methods
void setMarkedLinkLineStyle (java.awt.Color newValue)
java.awt.Color getMarkedLinkLineStyle ()

MenuActions

Property of JGIGanttGraph

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Actions of the action map for the pop up menu of the gantt graph. A dash "-" represents the separating line between two menu items.

Accessing Methods
void setMenuActions (integer index, java.lang.String newValues)
void setMenuActions (java.lang.String[] newValue)
java.lang.String getMenuActions (integer index)
java.lang.String[] getMenuActions ()

MousePosAtLastPopUpTrigger

Read Only Property of JGIGanttGraph

Typ	**java.awt.Point**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the mouse position of the pop up menu most recently invoked.

Accessing Methods
java.awt.Point getMousePosAtLastPopUpTrigger()

NodeDrawingPriorityComparator

Property of JGIGanttGraph

Typ	**java.util.Comparator**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you specify a comparator that defines the drawing sequence of the nodes in a group with RowLayout GROUP_ROWLAYOUT_SINGLE. The comparator needs to be existent, so you'll probably need to write it before you can invoke it. It decides, whether node 1 is to be drawn after or before node2.

Accessing Methods
void setNodeDrawingPriorityComparator (java.util.Comparator newValue)
java.util.Comparator getNodeDrawingPriorityComparator ()

Example Code

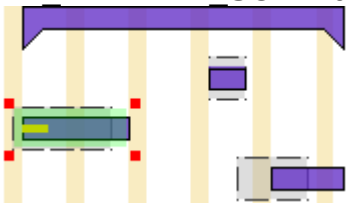
```
jGantt1.getGanttGraph().setNodeDrawingPriorityComparator(new
ComparatorForNodeDrawingPriority ());
```

NodeMarkAreaStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Marks a node by placing an area on and around it. Marking a node by an area only makes sense if transparent colors are used. When using NeAreaStyle instead of the Color class as a color data type, the fill patterns FILL_PATTERN_SOLID und FILL_PATTERN_NONE are supported only.



Transparent area for marking a node; pickmarks in addition.

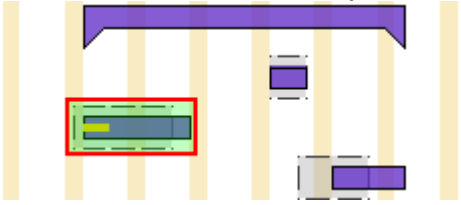
Accessing Methods
void setNodeMarkAreaStyle (java.awt.Color newValue)
java.awt.Color getNodeMarkAreaStyle ()

NodeMarkLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Color of the node marking, that is, of the pick marks or of the marking line. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported)



Transparent area, surrounded by a colored line for marking a node.

Accessing Methods
void setNodeMarkLineStyle (java.awt.Color newValue)
java.awt.Color getNodeMarkLineStyle ()

NodeMarkStyle

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Mark style of nodes. The below constants are available:

Possible Values	Description
MARKSTYLE_NOMARKS	The object is not marked.
MARKSTYLE_PICKMARKS	The object is marked by four pickmarks that are placed around the object.
MARKSTYLE_PICKMARKS_INSIDE	The object is marked by four pickmarks that are placed inside the object.
MARKSTYLE_RECTANGLE	The object is marked by a frame that is placed around the object.
MARKSTYLE_RECTANGLE_INSIDE	The object is marked by a frame that is placed inside the object.

Accessing Methods
void setNodeMarkStyle (int newValue)
int getNodeMarkStyle ()

NodeModifyInteractionsOnLabelsEnabled

Property of JGIGanttGraph

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

By this property you can set or retrieve, whether or not interactions with the node can be performed via their symbols and annotations. On **true** a user can, for example, move a node by picking its symbol or its annotation.

Accessing Methods
void setNodeModifyInteractionsOnLabelsEnabled (boolean newValue)
boolean hasNodeModifyInteractionsOnLabelsEnabled ()

NodeMovePhantomAreaStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Area color of the phantom for node moving. When using NeAreaStyle instead of the Color class as a color data type, the fill patterns of that class are available.

Accessing Methods
void setNodeMovePhantomAreaStyle (java.awt.Color newValue)
java.awt.Color getNodeMovePhantomAreaStyle ()

NodeMovePhantomLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the phantom for node movement. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported)

Accessing Methods
void setNodeMovePhantomLineStyle (java.awt.Color newValue)
java.awt.Color getNodeMovePhantomLineStyle ()

NodeResizePhantomAreaStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Area color of the phantom for node resizing. When using NeAreaStyle instead of the Color class as a color data type, the fill patterns of that class are available.

Accessing Methods
void setNodeResizePhantomAreaStyle (java.awt.Color newValue)
java.awt.Color getNodeResizePhantomAreaStyle ()

NodeResizePhantomLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the phantom for node resizing. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported).

Accessing Methods
void setNodeResizePhantomLineStyle (java.awt.Color newValue)
java.awt.Color getNodeResizePhantomLineStyle ()

NumberOfLinks

Read Only Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the number of links in the Gantt graph.

Accessing Methods
int getNumberOfLinks ()

Also see NumberOfNodes

NumberOfNodes

Read Only Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the number of nodes in the Gantt graph.

Accessing Methods
int getNumberOfNodes ()

Also see NumberOfLinks

PickMarkSize

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the size of the pickmarks. Unit: pixels.

Accessing Methods
void setPickMarkSize (int newValue)
int getPickMarkSize ()

RectangleMarkOffset

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the extent of the marking frame. The value defines the offset between the element to be marked and the marking frame. Unit: pixels.

Accessing Methods
void setRectangleMarkOffset (int newValue)
int getRectangleMarkOffset ()

SelectPhantomAreaStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Area color of the phantom for selecting objects. When using NeAreaStyle instead of the Color class as a color data type, the fill patterns of that class are available.

Accessing Methods
void setSelectPhantomAreaStyle (java.awt.Color newValue)
java.awt.Color getSelectPhantomAreaStyle ()

SelectPhantomLineStyle

Property of JGIGanttGraph

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the phantom for selecting objects. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported)

Accessing Methods
void setSelectPhantomLineStyle (java.awt.Color newValue)
java.awt.Color getSelectPhantomLineStyle ()

SubRowBy

Property of JGIGanttGraph

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**""**

This property lets you arrange nodes, the layout of which was optimized by JGantt.GROUP_ROWLAYOUT_SINGLE_OPTIMIZED. This property sets the name of an attribute of an entity. If the value of the attribute equals 0 or if no attribute was set, a sub-row for the node will be chosen automatically.

If the attribute holds a value > 0, the node will be placed in the corresponding sub-row. It is not important whether the node overlaps other nodes in the same preset sub-row. When the nodes are positioned, nodes that have a number will be considered prior to nodes that do not have a row number. If nodes overlap, the nodes that have no row number will evade to different rows.

If a value < 0 is set, the number will be transformed into an absolute number and the node will be placed into a corresponding sub-row. In contrast to nodes of positive values, these nodes will not be positioned with priority - so they may overlap other nodes.

As an alternative to specifying the sub-row, you can have a comparator do the job.

Accessing Methods
void setSubRowBy (java.lang.String newValue)
java.lang.String getSubRowBy ()

Also see IgnoredByOptimizationNodeFilter
SubRowComparator

SubRowComparator

Property of JGIGanttGraph

Typ	**java.util.Comparator**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you specify a comparator that defines the sequence of the sub-rows to be filled by nodes. The comparator needs to be existent, so you'll probably need to write it before you can invoke it. It decides, whether node 1 is to be displayed above or below node2, or in the same row.

As an alternative, you can select the sub-row manually by setting the property **SubRowBy**

Accessing Methods
void setSubRowComparator (java.util.Comparator newValue)
java.util.Comparator getSubRowComparator ()

Also see IgnoredByOptimizationNodeFilter
SubRowBy

SubRowMargin

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**50**

This property defines the minimum offset between the row border and a node in a subrow (when optimized GroupLayout). The unit is millimeters.

Accessing Methods
void setSubRowMargin (int newValue)
int getSubRowMargin ()

SubRowMinimalHeight

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0**

This property defines the minimum height of a subrow (when optimized GroupLayout). The unit is millimeters.

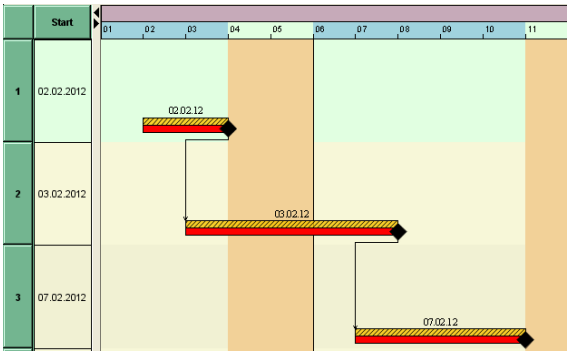
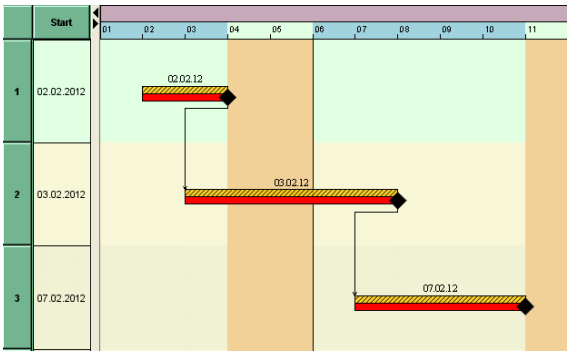
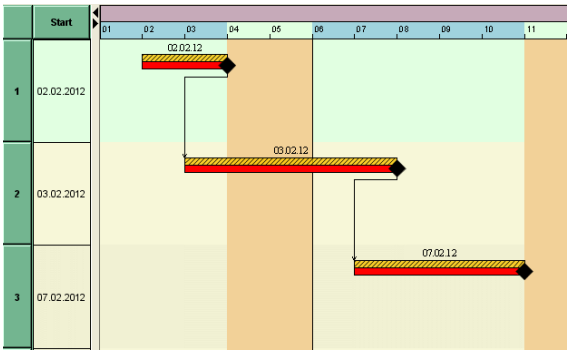
Accessing Methods
void setSubRowMinimalHeight (int newValue)
int getSubRowMinimalHeight ()

VerticalNodeAlignmentInsideRow

Property of JGIGanttGraph

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**VERTICAL_NODE_ALIGNMENT_CENTER_OF_ROW**

This property lets you define in which way the nodes should be vertically aligned in a row of the Gantt graph, in case the row height is larger than the height of the nodes. You can place the nodes at the top, in the center or at the bottom of the row.

Possible Values	Description
VERTICAL_NODE_ALIGNMENT_BOTTOM_OF_ROW	The nodes are positioned at the bottom margin of the row.
	
VERTICAL_NODE_ALIGNMENT_CENTER_OF_ROW	The nodes are positioned at the center of the row.
	
VERTICAL_NODE_ALIGNMENT_TOP_OF_ROW	The nodes are positioned at the top margin of the row.
	

Accessing Methods
void setVerticalNodeAlignmentInsideRow (int newValue)
int getVerticalNodeAlignmentInsideRow ()

Methods of the Interface

addMouseObserverListener

Method of JGIGanttGraph

This method lets you add a listener for events of mouse movements. The listener will be informed each time the mouse position has changed within the Gantt graph.

Declaration
void addMouseObserverListener (de.netronic.common.event.NeMouseObserverListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeMouseObserverListener	Mouse listener to be added.
Return Value	void	

createLayerDefinition

Method of JGIGanttGraph

This method lets you generate a layer definition. It serves to describe the position and appearance of a layer.

Declaration
de.netronic.common.intface.NeILayerDefinition createLayerDefinition (java.lang.String id)

	Data Type	Description
Parameter		
id	java.lang.String	Identification string of the layer definition.
Return Value	de.netronic.common.intface.-NeILayerDefinition	Returns the layer definition created.

createValueReference

Method of JGIGanttGraph

This method lets you generate a value reference object. It serves to transmit values (dates, in the first place) of a layer or a node correctly and comfortably during interactions.

Declaration
de.netronic.common.intface.NeIValueReference createValueReference (java.lang.String definitionString)

	Data Type	Description
Parameter		
definitionString	java.lang.String	Character string that defines what values of a node or layer are passed. You can specify the name of an attribute, such as, for example, "start", or use terms of relative dates, for example "start+duration(d)".
Return Value	de.netronic.common.intface.-NeIValueReference	Returns the value reference object created.

getAction

Method of JGIGanttGraph

This method lets you retrieve an action from the action map via its name.

Declaration
javax.swing.Action getAction (java.lang.Object name)

	Data Type	Description
Parameter		
name	java.lang.Object	Name of the action to be added
Return Value	javax.swing.Action	

getHorLineGrids

Method of JGIGanttGraph

Retrieves the line grid of the type NeIHorLineGrids that belongs to the Gantt graph. The line grid returned allows to set horizontal grouping lines.

Declaration
de.netronic.common.intface.NeIHorLineGrids getHorLineGrids ()

	Data Type	Description
Return Value	de.netronic.common.intface.NeIHorLine Grids	The line grid of the type NeIHorLineGrids that belongs to the Gantt graph.

getLayerDefinition

Method of JGIGanttGraph

This method lets you retrieve a layer definition by its ID.

Declaration
de.netronic.common.intface.NeILayerDefinition getLayerDefinition (java.lang.String id)

	Data Type	Description
Parameter		
id	java.lang.String	Identification string of the layer definition.
Return Value	de.netronic.common.intface.- NeILayerDefinition	Returns the layer definition found.

getLinkDefinition

Method of JGIGanttGraph

This method lets you retrieve a link definition by its ID.

Declaration
de.netronic.common.intface.NeILinkDefinition getLinkDefinition (java.lang.String id)

	Data Type	Description
Parameter		
id	java.lang.String	Identification string of the link definition.
Return Value	de.netronic.common.intface.-NeILinkDefinition	Returns the link definition found.

isCompletelyOnScreen

Method of JGIGanttGraph

Retrieves, whether or not the object represented by the entity is completely displayed on the screen.

Declaration
boolean isCompletelyOnScreen (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity representing the graphical object (node or link) in Gantt graph.
Return Value	boolean	**True**: the node or link is completely displayed on the screen, **false**: the node or link is not completely displayed on the screen.

isOnScreen

Method of JGIGanttGraph

Retrieves, whether or not the object represented by the entity is displayed on the screen.

Declaration
boolean isOnScreen (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter entity	de.netronic.common.intface.NeIEntity	Entity representing the graphical object (node or link) in Gantt graph.
Return Value	boolean	**True**: the node or link is displayed on the screen, **false**: the node or link is not displayed on the screen.

iterateLayerDefinitions

Method of JGIGanttGraph

This method returns an iterator object on existing layer definitions.

Declaration
java.util.Iterator iterateLayerDefinitions ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned.

iterateLinkDefinitions

Method of JGIGanttGraph

This method returns an iterator object on existing linkdefinitions.

Declaration
java.util.Iterator iterateLinkDefinitions ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned.

iterateLinkEntities

Method of JGIGanttGraph

This method returns an iterator object on existing link entities. Only those entites are selected that match the filter passed. If you remove entities by the iterator, the corresponding links in the Gantt graph will be removed.

Declaration
java.util.Iterator iterateLinkEntities (de.netronic.common.intface.NeIFilter filter)

	Data Type	Description
Parameter		
filter	de.netronic.common.intface.NeIFilter	Filter to select for links that show the properties defined.
Return Value	java.util.Iterator	Iterator object returned.

iterateNodeEntities

Method of JGIGanttGraph

This method returns an iterator object on existing node entities. Only those entites are selected, that match the filter passed. If you remove entities via the iterator, the corresponding nodes in the Gantt graph will be removed.

Declaration
void iterateNodeEntities (de.netronic.common.intface.NeIFilter filter)

	Data Type	Description
Parameter		
filter	de.netronic.common.intface.NeIFilter	Filter to select for nodes that show the properties defined.
Return Value	void	Iterator object returned.

putAction

Method of JGIGanttGraph

This method lets you add an action to the action map.

Declaration
void putAction (java.lang.Object name, javax.swing.Action action)

	Data Type	Description
Parameter		
name	java.lang.Object	Name of the action to be added
action	javax.swing.Action	Action to be added.
Return Value	void	

removeAction

Method of JGIGanttGraph

This method lets you remove an action from the action map via its name.

Declaration
void removeAction (java.lang.Object name)

	Data Type	Description
Parameter		
name	java.lang.Object	Name of the action to be removed
Return Value	void	

removeLayerDefinition

Method of JGIGanttGraph

This method lets you remove a layer definition.

Declaration
void removeLayerDefinition (java.lang.String def)

	Data Type	Description
Parameter		
def	java.lang.String	Layer definition to be deleted.
Return Value	void	

removeLinkDefinition

Method of JGIGanttGraph

This method lets you remove a link definition.

Declaration
void removeLinkDefinition (java.lang.String def)

	Data Type	Description
Parameter		
def	java.lang.String	Link definition to be deleted.
Return Value	void	

removeMouseObserverListener

Method of JGIGanttGraph

This method lets you remove an existing listener for events of mouse movements.

Declaration
void removeMouseObserverListener (de.netronic.common.event.NeMouseObserverListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeMouseObserverListener	Mouse listener to be removed.
Return Value	void	

scrollToEntity

Method of JGIGanttGraph

This method lets you scroll the view for a node or a link to become visible completely. The view may or may not be scrolled exactly to the center of the object.

Declaration
void scrollToEntity (de.netronic.common.intface.NeIEntity entity, boolean toCenter)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of a node or a link to be scrolled to.
toCenter	boolean	Defines, whether (**true**) or not (**false**) the node or link should be placed in the center of the view.
Return Value	void	

4.2 NeIHorLineGrid

Belongs to GanttGraph

Package name **de.netronic.common.intface**
Extends **de.netronic.common.intface.NeILineAttributes**

This interface provides methods and properties to handle a single horizontal line grid.

Properties to handle a line grid

NumberOfElementsBetweenLines	Sets after how many elements a line of the line grid is to be drawn.

Properties of the Interface

NumberOfElementsBetweenLines

Property of NeIHorLineGrid

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**1**

This property allows to set after how many elements a line of the line grid is to be drawn.

Accessing Methods
void setNumberOfElementsBetweenLines (int newValue)
int getNumberOfElementsBetweenLines ()

Example Code

```
NeIHorLineGrid horGroupGridLevel0 = horLineGrids.getGridForGroups(0);
horGroupGridLevel0.setNoOfElementsBetweenLines(1);
```

4.3 NeIHorLineGrids

Belongs to GanttGraph

Package name **de.netronic.common.intface**

This interface provides methods and properties to handle horizontal line grids.

Properties to handle the grouping levels

ShowFirstSeparationLines	In front of the first group of a grouping level a horizontal splitter line can be drawn.

Methods to handle the grouping levels

getGridForGroups(...)	Returns the horizontal line grid of the gouping level specified.
getGridForNodes(...)	Retrieves the horizontal line grid that appears between the node lines.

Properties of the Interface

ShowFirstSeparationLines

Property of NeIHorLineGrids

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property lets you set, whether or not in front of the first group of a grouping level a horizontal splitter line is to be drawn.

Accessing Methods
void setShowFirstSeparationLines (boolean newValue)
boolean isShowFirstSeparationLines ()

Example Code

```
NeIHorLineGrids horLineGrids = jGantt1.getGanttGraph().getHorLineGrids();
horLineGrids.setShowFirstSeparationLines(true);
```

Methods of the Interface

getGridForGroups

Method of NeIHorLineGrids

This method retrieves the horizontal line grid allocated to the grouping level.

Declaration
de.netronic.common.NeIHorLineGrid getGridForGroups (int level)

	Data Type	Description
Parameter		
level	int	Grouping level of which the horizontal line grid is to be retrieved.
Return Value	de.netronic.common.NeIHorLineGrid	Horizontal line grid of the grouping level

Also see getGridForNodes

Example Code

```
NeIHorLineGrid horGroupGridLevel0 = horLineGrids.getGridForGroups(0);
```

getGridForNodes

Method of NeIHorLineGrids

If the JGantt property **GroupMode** was set to **GROUP_BY_VALUE** and the property **GroupLayout** was set to **GROUP_ROWLAYOUT_MULTIPLE**, this method will return the horizontal line grid that is displayed between the node lines of the entity sets on the gouping level specified.
If the JGantt property **GroupMode** was set to **GROUP_BY_HIERARCHY**, this method will return the horizontal line grid that is displayed between the node lines at the hierarchy level specified.

Declaration
de.netronic.common.NeIHorLineGrid getGridForNodes (int level)

	Data Type	Description
Parameter		
level	int	Grouping level of which the horizontal line grid is to be retrieved.
Return Value	de.netronic.common.NeIHorLineGrid	Horizontal line grid of the grouping level

Also see getGridForGroups

Example Code

```
NeIHorLineGrid horGroupGridLevel0 = horLineGrids.getGridForNodes(0);
```

4.4 NeILayerDefinition

Belongs to GanttGraph

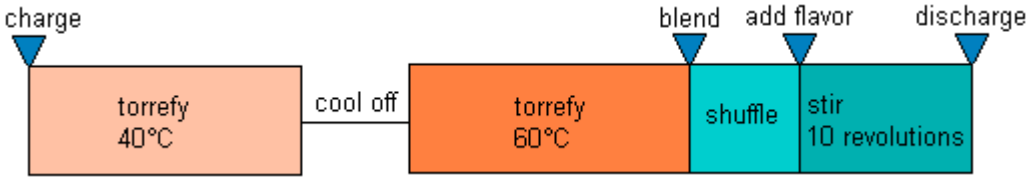
Package name **de.netronic.common.intface**

This interface lets you to define a layer. A layer is one out of two basic graphical elements that compose a node. A node may contain several layers:

A layer may have the shape of a rectangle, a line or other shapes, depending on the pre-defined types available.

A layer represents a duration that has a start date and an end date. If the dates are identical, the layer does not have an extent. The dates are defined by the properties StartValueReference und EndValueReference.

The other graphical element that a node is composed of are the decorations. They may consist of symbols or annotations:



For annotations and symbols, please see classes NeAnnotation and NeSymbol.

Properties to Define the Identity of the Layer

ID	Identification of the layer definition
LayerType	Type of the layer

Properties to Handle Graphical Elements

3DMode	Three-dimensional display of the layer
AreaStyle	Appearance of the inner area of an object
BottomPicture	Places a decoration object as a picture beneath a layer.
CenterPicture	Places a decoration object as a picture in a layer.
DrawingPriority	Drawing priority
EndValueReference	Value reference for the end of the layer
ExcludedFromLegend	Excluding this layer definition from being displayed in the legend.

ExcludedFromLegend	This property defines whether or not the link definition apperas in the legend.
Filter	Filter that contains conditions on the display of the layer
Height	Height of the layer
LineStyle	Appearance of the lines of an object
LineWidth	Line width
ReferencesSetByInteractiveCreation	Dates are stored to the layer when created
StartValueReference	Value reference for the start of the layer
TopPicture	Places a decoration object as a picture above a layer.
VerticalOffset	Vertical offset from the center of the node
ZeroLengthLabel	Defines the symbol for labels of no extent
ZeroLengthLayerVisible	Defines, whether or not layers of no extent are displayed

Properties to Handle Interactions

ExcludedFromMarking	Taking layers in account when marking
Fixed	Layer fixed during interactions on the node
LeftResizeable	Can the layer be resized at its left end?
Master	Is the node is the target of an interaction with the layer?
Moveable	Moveability of the layer
RightResizeable	Can the layer be resized at its right end?

Methods to Handle Graphical Elements

attachLabel(...)	Attaches a decoration object as a label to a layer.
copyPropertiesTo(...)	Copies all properties set to a different NeILayerDefinition object.
iterateLabels()	Returns an iterator object on existing label attachments.
removeLabel(...)	Removes a label from a layer.

Properties of the Interface

3DMode

Property of NeILayerDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**MODE_3D_NONE**

This property defines whether or not the layer is to be displayed three-dimensionally and if so, whether or not the surface appears to be pressed or raised.

Possible Values	Description
MODE_3D_LOWERED	The three-dimensional display is switched on with surfaces appearing lowered.
MODE_3D_NONE	The three-dimensional display is switched off.
MODE_3D_RAISED	The three-dimensional display is switched on with surfaces appearing raised.

Accessing Methods
void set3DMode (int newValue)
int get3DMode ()

AreaStyle

Property of NeILayerDefinition

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color and pattern of the inner area of an object. Being a parameter of the type **Color**, also an object of the class **NeAreaStyle** can be passed.

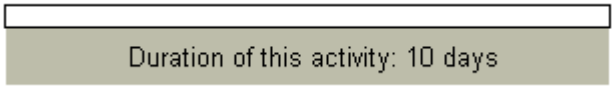
Accessing Methods
void setAreaStyle (java.awt.Color newValue)
java.awt.Color getAreaStyle ()

BottomPicture

Property of NeILayerDefinition

Typ	**de.netronic.common.intface.NeIPicture**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property places a decoration object as a picture beneath a layer.



Annotation added as a picture object beneath the layer.

Accessing Methods
void setBottomPicture (de.netronic.common.intface.NeIPicture newValue)
de.netronic.common.intface.NeIPicture getBottomPicture ()

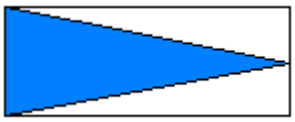
Also see CenterPicture
TopPicture

CenterPicture

Property of NeILayerDefinition

Typ	**de.netronic.common.intface.NeIPicture**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property places a decoration object as a picture in a layer.



Symbol added as a picture to center of the layer. In this example the symbol is stretched to the full extent of the layer.

Accessing Methods
void setCenterPicture (de.netronic.common.intface.NeIPicture newValue)
de.netronic.common.intface.NeIPicture getCenterPicture ()

Also see BottomPicture

TopPicture

DrawingPriority

Property of NeILayerDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0**

This property sets the priority of display to a layer. Layers that have high values are drawn on top of layers that have low values.

Accessing Methods
void setDrawingPriority (int newValue)
int getDrawingPriority ()

EndValueReference

Property of NeILayerDefinition

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the value reference for the end of the layer.

Accessing Methods
void setEndValueReference (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getEndValueReference ()

Also see StartValueReference

ExcludedFromLegend

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not the layer definition appears in the legend.

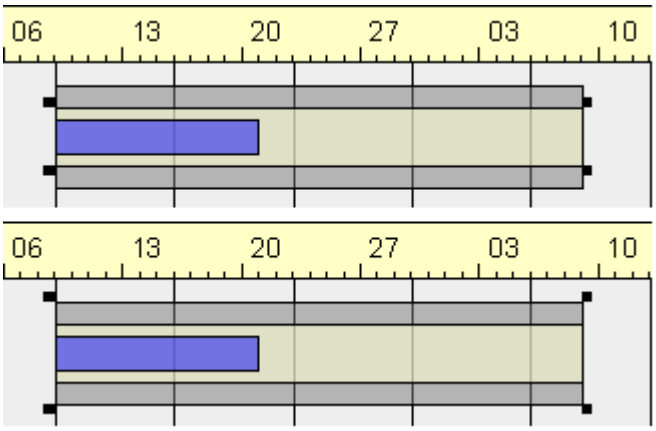
Accessing Methods
void setExcludedFromLegend (boolean newValue)
boolean isExcludedFromLegend ()

ExcludedFromMarking

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you exclude a layer from marking when the node is marked. This may be useful e.g. for fixed layers that are excluded from node interactions.



Above: The gray layer is excluded from node marking. Below: The gray layer is included in node marking.

Accessing Methods
void setExcludedFromMarking (boolean newValue)
boolean isExcludedFromMarking ()

Also see Fixed

Filter

Property of NeILayerDefinition

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets or retrieves a filter. A filter contains conditions, that decide whether or not the layer is to be drawn for a specific filter.

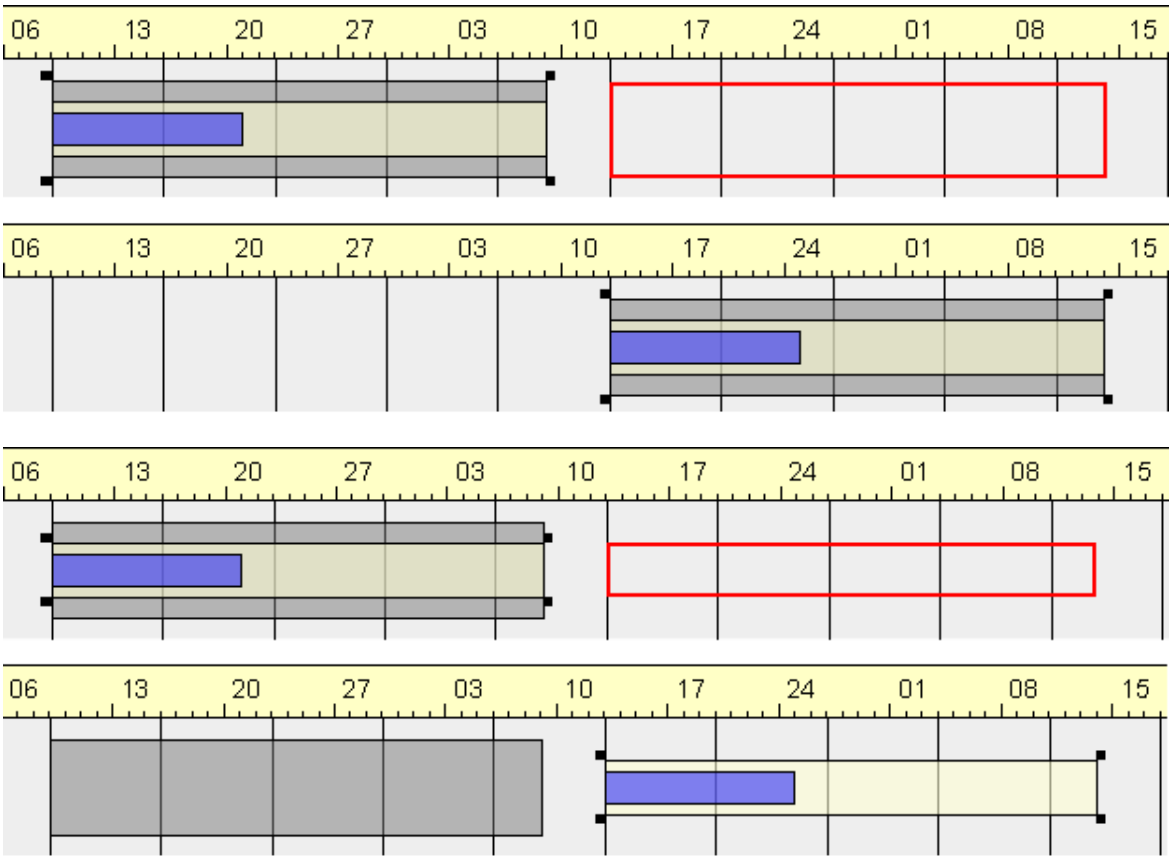
Accessing Methods
void setFilter (de.netronic.common.intface.NeIFilter newValue)
de.netronic.common.intface.NeIFilter getFilter ()

Fixed

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

By this property a layer can be made resistable against the interactions **Move** and **Resize** of nodes. While the layer properties **Moveable**, **LeftResizable** and **RightResizeable** prohibit interactions referring directly to the layer, this property also prevents interactions on the nodes to modify the layer. Exception: if a fixed layer uses the same dates as a layer that is not fixed, both layers will be moved if the non-fixed layer is moved.



Above: The large gray layer is not fixed and therefore is moved when the node is moved. Below. The large gray layer is fixed and remains in its place when the node is moved.

Accessing Methods
void setFixed (boolean newValue)
boolean isFixed ()

Also see ExcludedFromMarking
LeftResizeable
Moveable
RightResizeable

Height

Property of NeILayerDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets or retrieves the height of a layer. Unit: 1/100 mm.

Accessing Methods
void setHeight (int newValue)
int getHeight ()

ID

Read Only Property of NeILayerDefinition

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the identification that was assigned to the layer definition when it was created.

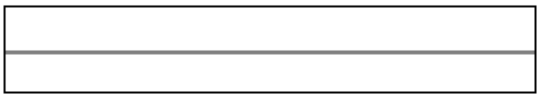
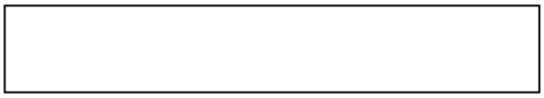
Accessing Methods
java.lang.String getID()

LayerType

Property of NeILayerDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**LAYERTYPE_RECTANGLE**

This property defines the layer type. Basically, a layer is a rectangle that is filled in a certain way. This property defines, in which way the rectangle is filled.

Possible Values	Description
LAYERTYPE_INVISIBLE	This layer type is invisible. At the same time, attached decorations (NeIPicture objects) are displayed. The decorations control the size of the layer, which, for example, becomes apparent when marked.
LAYERTYPE_LINE	This layer type places a line into the rectangle. 
LAYERTYPE_RECTANGLE	This layer type fills the rectangle completely. 
LAYERTYPE_SUMMARY	This layer type fills the rectangle completely and adds two framing triangels to the bottom of the layer. They have the same height as the layer: 
LAYERTYPE_SYMBOL_ONLY	This layer abstains from filling the rectangle at all. Therefore the node later on will be represented by symbols only (or more general: by decorations only, since annotations are allowed, too), if they were assigned. 

Accessing Methods
void setLayerType (int newValue)
int getLayerType ()

LeftResizeable

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property permits whether or not the layer can be resized on its left end.

Accessing Methods
void setLeftResizeable (boolean newValue)
boolean isLeftResizeable ()

Also see RightResizeable

Legendtext

Property of NeILayerDefinition

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

By this property you can specify a legend text for a layer definition. The default value for the legend text is the name of the layer definiton.

Accessing Methods
void setLegendtext (java.lang.String newValue)
java.lang.String getLegendtext ()

LineStyle

Property of NeILayerDefinition

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color and pattern of the lines of an object. To define the properties of the line style object please see class NeLineStyle.

Accessing Methods
void setLineStyle (java.awt.Color newValue)
java.awt.Color getLineStyle ()

LineWidth

Property of NeILayerDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1**

This property defines the line width of an object. Unit: 1/100 mm

Accessing Methods
void setLineWidth (int newValue)
int getLineWidth ()

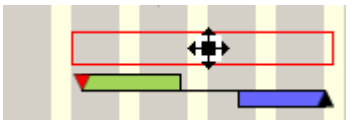
Master

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not an interaction, such as moving, affects the complete node or just the layer.

False: Only the layer reacts. In addition, the cursor shape indicates that the layer can be moved horizontally only.



True: The complete node reacts. In addition, the cursor shape indicates that the node can be moved horizontally and vertically.

Accessing Methods
void setMaster (boolean newValue)
boolean isMaster ()

Moveable

Property of NelLayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property permits whether (**true**) or not (**false**) the layer can be moved. Only horizontal dislocation of layers is possible.

Accessing Methods
void setMoveable (boolean newValue)
boolean isMoveable ()

ReferencesSetByInteractiveCreation

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

By this property you can decide whether or not dates will be automatically set to the layer when it is created interactively. The dates are stored to the entity via the value reference object.

Accessing Methods
void setReferencesSetByInteractiveCreation (boolean newValue)
boolean hasReferencesSetByInteractiveCreation ()

RightResizeable

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property permits whether or not the layer can be resized on its right end.

Accessing Methods
void setRightResizeable (boolean newValue)
boolean isRightResizeable ()

Also see LeftResizeable

StartValueReference

Property of NeILayerDefinition

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the value reference for the start of the layer.

Accessing Methods
void setStartValueReference (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getStartValueReference ()

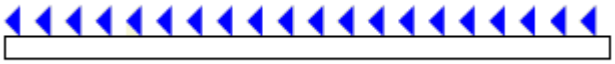
Also see EndValueReference

TopPicture

Property of NeILayerDefinition

Typ	**de.netronic.common.intface.NeIPicture**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property places a decoration object as a picture above a layer.



Picture object above a layer. In this example the display of the symbol is repeated until it completely fills the space of the picture.

Accessing Methods
void setTopPicture (de.netronic.common.intface.NeIPicture newValue)
de.netronic.common.intface.NeIPicture getTopPicture ()

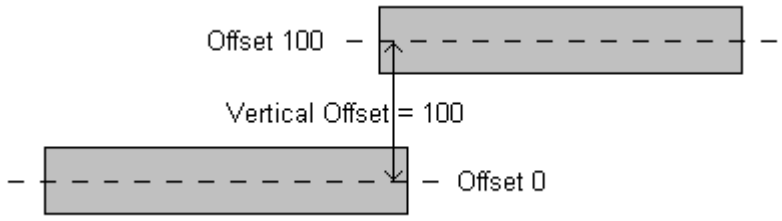
Also see BottomPicture
CenterPicture

VerticalOffset

Property of NeILayerDefinition

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets or retrieves the vertical offset of the layer from the center of the node. Layers of which the vertical offset=0 are situated in the center of a node. Unit: 1/100 mm.



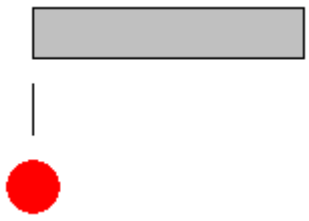
Accessing Methods
void setVerticalOffset (long newValue)
long getVerticalOffset ()

ZeroLengthLabel

Property of NeILayerDefinition

Typ	**de.netronic.common.intface.NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines a symbol that is displayed in the place of a layer that has no extent. To make the symbol visible, the property **setZeroLengthLabelVisible** needs to be set to **true**.



Top: Layer that has an extent.

Center: Layer that has no extent. The start date and end date are identical and are represented by a line.
Bottom: Layer that has no extent and is represented by a symbol

Accessing Methods
void setZeroLengthLabel (de.netronic.common.intface.NeILabel newValue)
de.netronic.common.intface.NeILabel getZeroLengthLabel ()

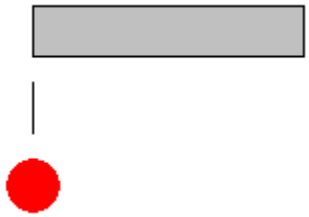
Also see ZeroLengthLayerVisible

ZeroLengthLayerVisible

Property of NeILayerDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines, whether or not layers, that have no length, i.e. where the start date equals the end date, will be displayed on the screen. If you set this propery to **true**, the layer will be displayed by a short vertical line, which you can replace by a symbol via the property **setZeroLengthLabel**.



Top: Layer that has an extent.
Center: Layer that has no extent. The start date and end date are identical and are represented by a line.
Bottom: Layer that has no extent, represented by a symbol.

Accessing Methods
void setZeroLengthLayerVisible (boolean newValue)
boolean isZeroLengthLayerVisible ()

Also see ZeroLengthLabel

Methods of the Interface

attachLabel

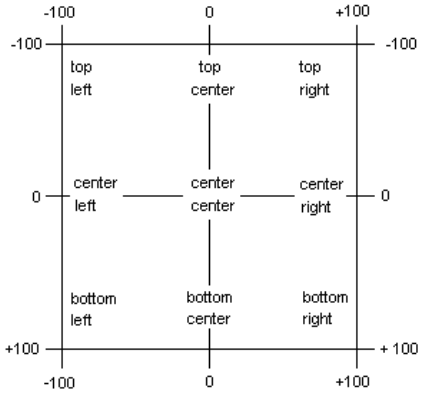
Method of NeILayerDefinition

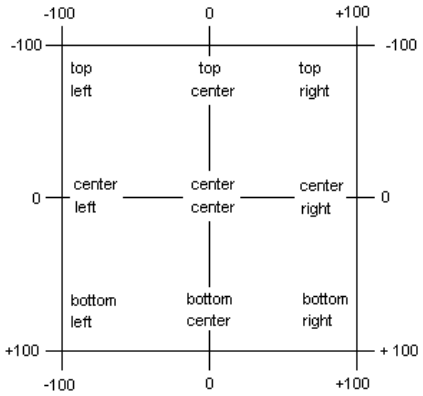
This method lets you add a label to the layer. To position the label, two coordinate systems are used:

1. the coordinate system of the layer
2. the coordinate system of the symbol

In both, you need to define a position via the parameters **position** bzw. **refPosition**. By this method, the postions are put on top of each other and allow for a very preceise positioning of the symbol on the layer.

Declaration
de.netronic.common.intface.NeILabelAttachment attachLabel
(de.netronic.common.intface.NeILabel label, java.awt.Point position, java.awt.Point refPosition)

	Data Type	**Description**
Parameter		
label	de.netronic.common.intface.NeILabel	Symbol to be attached
position	java.awt.Point	Position in coordinate system of the layer. You can select one of the below constants, to place the symbol at defined places within a positioning square. Alternatively, you may enter coordinates from -100 to +100 to place the symbol inbetween the pre-defined positions. In addition, you can enter coordinates beyond +100 and -100 to place the symbol outside of the positioning square. 

refPosition	java.awt.Point	Position in coordinate system of the symbol. You can select one of the below constants, to place the symbol at defined places within a positioning square. Alternatively, you may enter coordinates from -100 to +100 to place the symbol inbetween the pre-defined positions. In addition, you can enter coordinates beyond +100 and -100. 
Return Value	de.netronic.common.intface.NeILabelAttachment	

Also see iterateLabels
removeLabel

copyPropertiesTo

Method of NeILayerDefinition

This method lets you copy the properties set to a different NeILayerDefinition object.

Declaration
void copyPropertiesTo (de.netronic.common.intface.NeILayerDefinition drain)

	Data Type	Description
Parameter		
drain	de.netronic.common.intface.-NeILayerDefinition	Layer definifinition object, to wich the properties are to be copied.
Return Value	void	

iterateLabels

Method of NeILayerDefinition

This method returns an iterator object on the node entites that exist in a row. During the loop deleting of label attachments may only be performed via the method iterator.remove(); the method removeLabel() will lead to a concurrentModificationException.

Declaration
java.util.Iterator iterateLabels ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned.

Also see attachLabel
removeLabel

removeLabel

Method of NeILayerDefinition

This method lets you remove a label from a layer.

Declaration
void removeLabel (de.netronic.common.intface.NeILabelAttachment attachment)

	Data Type	Description
Parameter		
attachment	de.netronic.common.intface.-NeILabelAttachment	Label type attachment to be removed from the layer.
Return Value	void	

Also see attachLabel
iterateLabels

4.5 NeILayouterGroup

Belongs to GanttGraph

Package name **de.netronic.common.intface**

This interface lets you handle groups.

Properties to Handle the Layouter Group Object

Collapseable	Collapsability of the group
Collapsed	Collapse state of the group
Empty	Retrieves, whether or not the group is empty.
EndRow	Index of the last row of the group
Entity	Retrieves an entity
Expandable	Expandability the group
GroupComparator	Sorting of sub-groups
GroupingLevel	Grouping level of the group
NodesArrangedOptimized	Value of the group layout
NodesPositionedInSeparateRows	Nodes of a group in separate rows
NoofChildGroups	Number of child groups.
NoofNodes	Number of nodes.
NoofRows	Number of rows occupied bya group.
ParentGroup	Parent group of this group
RowForNodesNotPositionedInSeparateRows	Row for nodes that are NOT positioned in separate rows
RowLayout	Row layout for individual groups
StartRow	Index of the start row of this group

Methods to Handle the LayouterGroup Object

getGroupViaGroupCodes(...)	Retrieves a group subordinated to the reference layouter group

getGroupViaRowIndex(...)	Determines the group that the specified row belongs to.
getLayoutables(...)	Retrieves layoutable entities of a group.
iterateChildGroups()	Iterates over child groups
iterateEntities()	Iterates over entities

Properties of the Interface

Code

Read Only Property of NeILayouterGroup

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this method you can retrieve the complete grouping code of an entity. The code according to which groups are formed on different levels may compose from several attributes, for example from the attributes named "Alpha", "Name" and "Color". The string will deliver for example the code "A.Miller.red".

Accessing Methods
java.lang.String getCode()

Collapseable

Read Only Property of NeILayouterGroup

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the group can be collapsed.

Accessing Methods
boolean isCollapseable()

Collapsed

Property of NelLayouterGroup

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can assign to a group to be collapsed (**true**) or to be expanded (**false**), or you can retrieve the state assigned.

Accessing Methods
void setCollapsed (boolean newValue)
boolean isCollapsed ()

Empty

Read Only Property of NelLayouterGroup

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves, whether the group is empty. If the value **true** is returned, neither subgroups nor entities exist in the goup, if **false** is returned, at least one of the objects exists.

Accessing Methods
boolean isEmpty()

EndRow

Read Only Property of NelLayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the index of the last row (bottom) of the group.

Accessing Methods
int getEndRow()

Entity

Read Only Property of NeILayouterGroup

Typ	**de.netronic.common.intface.NeIEntity**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the entity passed.

Accessing Methods
de.netronic.common.intface.NeIEntity getEntity()

Expandable

Read Only Property of NeILayouterGroup

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the group can be expanded.

Accessing Methods
boolean isExpandable()

GroupComparator

Property of NeILayouterGroup

Typ	**java.util.Comparator**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Sub-groups in groups can be sorted. The sequence of the sub-groups can be defined by a comparator object. At runtime, two objects of the type NeILayoutGroup need to be passed to the comparator for comparison.
If the sorting is intended to be dynamical, that is, if on the modification of data the goups should be sorted anew, the comparator passed in addition needs to implement the interface NeIGroupComparator. The class de.netronic.bean.layouter.NeGroupComparator provides a pre-defined default implementation of a comparator.

This property defines the sorting of individual groups. It overwrites the values set by the JGantt property "GroupsSortedBy", which sets the sorting for all groups of the Gantt chart. Setting the JGantt "GroupsSortedBy" property again will overwrite the settings of the NeILayouterGroup property "GroupComparator".

Accessing Methods
void setGroupComparator (java.util.Comparator newValue)
java.util.Comparator getGroupComparator ()

GroupingLevel

Read Only Property of NeILayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the grouping level of the group.

Accessing Methods
int getGroupingLevel()

NodesArrangedOptimized

Read Only Property of NeILayouterGroup

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the group layout has been set to **optimized**.

Accessing Methods
boolean isNodesArrangedOptimized()

NodesPositionedInSeparateRows

Read Only Property of NelLayouterGroup

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) each node of a group is displayed in a sepaerate row.

Accessing Methods
boolean isNodesPositionedInSeparateRows()

NoofChildGroups

Read Only Property of NelLayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the number of child groups subordinated to this group.

Accessing Methods
int getNoofChildGroups()

NoofNodes

Read Only Property of NelLayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the number of nodes subordinated to this group.

Accessing Methods
int getNoofNodes()

NoofRows

Read Only Property of NeILayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the number of rows occupied by a group.

Accessing Methods
int getNoofRows()

ParentGroup

Read Only Property of NeILayouterGroup

Typ	**NeILayouterGroup**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the parent group of this group.

Accessing Methods
NeILayouterGroup getParentGroup()

RowForNodesNotPositionedInSeparateRows

Read Only Property of NeILayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the index of the row for nodes that are NOT positioned in separate rows. It will equal -1 if there is no subtitle or if the nodes are **positionedInSeparateRows** or if the group is invisible.

Accessing Methods
int getRowForNodesNotPositionedInSeparateRows()

RowLayout

Property of NelLayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the layout for individual groups. It overwrites the values set by the JGantt property "GroupLayout", which sets the layout for all groups of the Gantt chart. Another setting of the JGantt "GroupLayout" property will overwrite the settings of the GanttGraph property "RowLayout".

Possible Values	Description
GROUP_ROWLAYOUT_MULTIPLE	Each activity of a group is placed in a row of its own.
GROUP_ROWLAYOUT_SINGLE	All activities are placed in the same row, allowing them to overlap.
GROUP_ROWLAYOUT_SINGLE_OPTIMIZED	All activities are placed in the same row, except for overlapping ones that are shifted to another row.

Accessing Methods
void setRowLayout (int newValue)
int getRowLayout ()

StartRow

Read Only Property of NelLayouterGroup

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the index of the start row (top) of this group.

Accessing Methods
int getStartRow()

Methods of the Interface

getGroupViaGroupCodes

Method of NelLayouterGroup

Starting from a layouter group, this method lets you retrieve a subordinated group on any lower level via the group code. Assuming a three level grouping comprising the below groups:

Group	Code Level 1	Code Level 2	Code Level 3
g1	A		
g2	A	a	
g3	A	a	1
g4	A	a	2
g5	A	b	
g6	B		

To retrieve for example group g3, you can retrieve the sub-group of code "1" from g2:

```
g3 = g2.getGroupViaGroupCodes (new String {"1"});
```

.. or you can retrieve sub-group of code "a" and "1" from g1:

```
g3 = g1.getGroupViaGroupCodes (new String {"a", "1"});
```

As the starting point, you can certainly also use the (invisible) root node, to search through the complete number of group levels for a certain group.

```
g3 = jGantt.getRootGroup().getGroupViaGroupCodes (new String {"A", "a", "1"});
```

Declaration
void getGroupViaGroupCodes (java.lang.String codes [])

	Data Type	Description
Parameter		
codes []	java.lang.String	Group codes to be seached.
Return Value	void	

getGroupViaRowIndex

Method of NeILayouterGroup

By this method you can retrieve the child group by the index of a row.

Declaration
de.netronic.common.intface.NeILayouterGroup getGroupViaRowIndex (int rowIndex)

	Data Type	Description
Parameter		
rowIndex	int	Index of the row.
Return Value	de.netronic.common.intface.-NeILayouterGroup	Child group at the index of a row.

getLayoutables

Method of NeILayouterGroup

This method fills a list with the layoutable entities (nodes) of this group. If child groups are found, the method will recurse into the child groups and the boolean parameter will return **true**.

Declaration
void getLayoutables (java.util.Collection theList, boolean withChildgroups)

	Data Type	Description
Parameter		
theList	java.util.Collection	List of entities (nodes) of a group that allow for a layout.
withChildgroups	boolean	Returns, whether (**true**) or not (**false**) child groups exist.
Return Value	void	

iterateChildGroups

Method of NeILayouterGroup

This method lets you iterate over the child groups of a group.

Declaration
java.util.Iterator iterateChildGroups ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned

iterateEntities

Method of NeILayouterGroup

This method lets you iterate over the child groups of a group.

Declaration
java.util.Iterator iterateEntities ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned

4.6 NeILinkDefinition

Belongs to GanttGraph

Package name **de.netronic.common.intface**

This interface lets you define a link definition. There may be link definitions as many as you like at a link.

Link definitions specify decorations for the horizontal line of a link. They may consist of symbols, pictures or annotations:

For annotations, pictures and symbols, please see classes NeAnnotation, NePicture and NeSymbol.

Properties to Define the Identity of the Link Definition

ID	Identification of the link definition

Properties to Handle Graphical Elements

DrawingPriority	Drawing priority
Filter	Filter that contains conditions on the display of the link definition

Methods to Handle Graphical Elements

attachLabel(...)	Attaches a decoration object as a label to a link.
attachLabel(...)	Attaches a decoration object as a label to a link.
iterateLabels()	Returns an iterator object on existing label attachments.
removeLabel(...)	Removes a label from a link.

Properties of the Interface

DrawingPriority

Property of NeILinkDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0**

This property sets the priority of display to a link definition. Link definitions that have high values are drawn on top of link definitions that have low values.

Accessing Methods
void setDrawingPriority (int newValue)
int getDrawingPriority ()

ExcludedFromLegend

Property of NeILinkDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not the link definition apperas in the legend.

Accessing Methods
void setExcludedFromLegend (boolean newValue)
boolean isExcludedFromLegend ()

Filter

Property of NeILinkDefinition

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets or retrieves a filter. A filter contains conditions that decide whether or not the link definition is to be drawn for a specific filter.

Accessing Methods
void setFilter (de.netronic.common.intface.NeIFilter newValue)
de.netronic.common.intface.NeIFilter getFilter ()

ID

Read Only Property of NeILinkDefinition

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the identification that was assigned to the link definition when it was created.

Accessing Methods
java.lang.String getID()

Legendtext

Property of NeILinkDefinition

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

By this property you can specify a legend text for a link definition. The default value for the legend text is the name of the link definiton.

Accessing Methods
void setLegendtext (java.lang.String newValue)
java.lang.String getLegendtext ()

LineStyle

Property of NeILinkDefinition

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the links of this link definition. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported).

Accessing Methods
void setLineStyle (java.awt.Color newValue)
java.awt.Color getLineStyle ()

MarkedLineStyle

Property of NeILinkDefinition

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line color of the marked links of this link definition. When using NeLineStyle instead of the Color class as a color data type, the line thickness can be varied (different line types are not supported).

Accessing Methods
void setMarkedLineStyle (java.awt.Color newValue)
java.awt.Color getMarkedLineStyle ()

Methods of the Interface

attachLabel

Method of NeILinkDefinition

This method lets you add a label to horizontal section of the link. To allocate the label merely the horizontal the position of the section is to be used.
By using the parameter **position**, you need to define a point on the horizontal section of the link, to which the label is placed in the center.

Declaration
de.netronic.common.intface.NeILinkLabelAttachment attachLabel (de.netronic.common.intface.NeILabel label, int position)

	Data Type	Description
Parameter		
label	de.netronic.common.intface.NeILabel	Symbol to be attached
position	int	
Return Value	de.netronic.common.intface.NeILinkLabelAttachment	Generated NeILinkLabelAttachment.

Also see attachLabel
iterateLabels
removeLabel

attachLabel

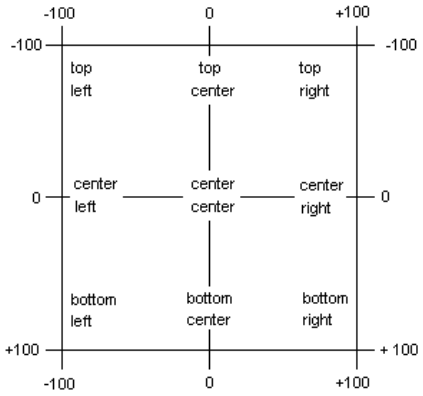
Method of NeILinkDefinition

This method lets you add a label to the link. To position the label, two coordinate systems are used:

1. the coordinate system of the link (position at the horizontal line of the link)
2. the coordinate system of the symbol

In both, you need to define a position using the parameters **position** or **refPosition**. By this method, the postions are put on top of each other and allow for a very precise positioning of the symbol on the link.

Declaration
de.netronic.common.intface.NeILinkLabelAttachment attachLabel
(de.netronic.common.intface.NeILabel label, int position, java.awt.Point refPosition)

	Data Type	Description
Parameter		
label	de.netronic.common.intface.NeILabel	Symbol to be attached
position	int	
refPosition	java.awt.Point	Position in coordinate system of the symbol. You can select one of the below constants, to place the symbol at defined places within a positioning square. Alternatively, you may enter coordinates from -100 to +100 to place the symbol inbetween the pre-defined positions. In addition, you can enter coordinates beyond +100 and -100. 
Return Value	de.netronic.common.intface.NeILinkLabelAttachment	Generated NeILinkLabelAttachment.

Also see attachLabel
iterateLabels
removeLabel

iterateLabels

Method of NeILinkDefinition

This method returns an iterator object on the node entites that exist in a row. During the loop deleting of label attachments may only be performed via the method iterator.remove(); the method removeLabel() will lead to a concurrentModificationException.

Declaration
void iterateLabels ()

	Data Type	Description
Return Value	void	Iterator object returned.

removeLabel

Method of NeILinkDefinition

This method lets you remove a label from a link.

Declaration
void removeLabel (de.netronic.common.intface.NeILinkLabelAttachment attachment)

	Data Type	Description
Parameter		
attachment	de.netronic.common.intface.NeILinkLabelAttachment	Label type attachment to be removed from the link.
Return Value	void	

4.7 NeMouseObserverEvent

Belongs to GanttGraph

Package name **de.netronic.common.event**
Extends **java.util.EventObject**

This class offers a constructor to handle events of the mouse.

Constructors of the Class

NeMouseObserverEvent

Constructor of NeMouseObserverEvent

This class lets you generate a mouse event that holds information on the

source of the event, the component and the coordinates where the mouse is located.

Declaration
NeMouseObserverEvent (java.lang.Object source, java.awt.Point pwd, java.awt.Component component)

Parameter	Data Type	Description
source	java.lang.Object	Object that represents the source of the event.
pwd	java.awt.Point	Coordinate where the mouse is located
component	java.awt.Component	Component where the mouse is located

4.8 NeMouseObserverListener

Belongs to GanttGraph

Package name **de.netronic.common.event**
Extends **java.util.EventListener**

This is the listener interface for receiving mouse events. The class that is interested in processing a position change event will implement this interface, including the methods it contains.The listener object will then be registered with the object using the object's add...Listener method. A mouse event is generated on changing the postion or the motion of the mouse.

Event Methods for the Mouse

mouseClickedLeft(...)	Event method on a click on the left mouse button
mouseClickedRight(...)	Event method on a click on the right mouse button
mouseExited(...)	Event method of mouse motion on leaving the component
mouseMoved(...)	Event method on motion of the mouse
mouseStopped(...)	Event method on stopping the mouse motion

Methods of the Interface

mouseClickedLeft

Method of NeMouseObserverListener

This method is invoked on clicking the left mouse button.

Declaration
void mouseClickedLeft (de.netronic.common.event.NeMouseObserverEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeMouseObserverEvent	Event triggered
Return Value	void	

Also see mouseClickedRight

mouseClickedRight

Method of NeMouseObserverListener

This method is invoked on clicking the right mouse button.

Declaration
void mouseClickedRight (de.netronic.common.event.NeMouseObserverEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeMouseObserverEvent	Event triggered
Return Value	void	

Also see mouseClickedLeft

mouseExited

Method of NeMouseObserverListener

This method is invoked when the mouse exits the component.

Declaration
void mouseExited (de.netronic.common.event.NeMouseObserverEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeMouseObserverEvent	Event triggered
Return Value	void	

mouseMoved

Method of NeMouseObserverListener

This method is invoked on motion of the mouse.

Declaration
void mouseMoved (de.netronic.common.event.NeMouseObserverEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeMouseObserverEvent	Event triggered
Return Value	void	

Also see mouseStopped

mouseStopped

Method of NeMouseObserverListener

This method is invoked on stopping the mouse motion.

Declaration
void mouseStopped (de.netronic.common.event.NeMouseObserverEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeMouseObserverEvent	Event triggered
Return Value	void	

Also see mouseMoved

5 Histogram

This component contains some of the classes and interfaces that you need to develop a histogram.

The Histogram-Component consists of the below classes:

JGIHistogram	This interface represents the histogram in a diagram.
NeArrayCurveData	This class lets you generate curve data objects.
NeCalculatedCurveData	This class provides a constructor to calculate curve data.
NeCurveData	This class lets you define, aggregate and cumulate the data of a curve in a histogram.
NeCurveStyle	This class allows to define the appearance of curves and of the areas between them.
NeICurve	This interface represents a curve in a histogram.
NeICurveData	This interface represents curve data and offers properties and methods to handle them.
NeICurveStyle	This interface represents a curve style object and lets you define the appearance of a curve.
NeLineCurve	This class lets you generate line curves.
NeStackedCurveData	This class lets you stack (add up) curve data.
NeStepCurve	This class lets you generate step curves.

5.1 JGIHistogram

Belongs to Histogram

Package name **de.netronic.jgantt**

This interface represents the histogram in a diagram. You can set the AppData object, which provides the data of the histogram, you can add or remove curves or add settings of the numeric scale.

Properties to Handle the Histogram Graph

AppData	AppData object of the histogram
Curves	Curve objects of the histogram
EndYValue	End value of the numeric scale

HistoDisplayProfile	Name of the profile that is used to display the calendar grid in the Gantt graph
StartYValue	Start value of the numeric scale

Properties to Handle the Numeric Scale

NumScaleBackgroundColor	Background color of the numeric scale
NumScaleCaption	Caption of the numeric scale
NumScaleCaptionPosition	Position of numeric scale caption
NumScaleFont	Font of the numeric scale
NumScaleResolution	Resolution of the numeric scale
NumScaleSubdivisions	Minor ticks of the numeric scale
NumScaleTextColor	Color of text annotations in the numeric scale
NumScaleUnitsBetweenMajorTicks	Annotated major ticks of the numeric scale

Methods to Handle the Histogram Graph

addCurve(...)	Adds a curve at the last index
addMouseListener(...)	By this method you can add a MouseListener to your histogram.
addMouseMotionListener(...)	By this method you can add a MouseMotionListener to your histogram.

Properties of the Interface

AdjustNumScaleResolution

Property of JGIHistogram

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you set or retrieve whether the NumScaleResolution should be adjusted automatically in such a way that no vertical scroll bar is needed for the histogram. Also after having moved the divider between the histogram and the Gantt graph the resolution of the numeric scale will be adjusted automatically.

Accessing Methods
void setAdjustNumScaleResolution (boolean newValue)
boolean isAdjustNumScaleResolution ()

AppData

Property of JGIHistogram

Typ	**de.netronic.common.intface.NeIAppData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the AppData object of the histogram. It provides the data for the curves.

Accessing Methods
void setAppData (de.netronic.common.intface.NeIAppData newValue)
de.netronic.common.intface.NeIAppData getAppData ()

Curves

Property of JGIHistogram

Typ	**de.netronic.common.intface.NeICurve[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set an array of curves to the histogram or retrieve the array of curves present in the histogram.

Accessing Methods
void setCurves (integer index, de.netronic.common.intface.NeICurve newValues)
void setCurves (de.netronic.common.intface.NeICurve[] newValue)
de.netronic.common.intface.NeICurve getCurves (integer index)
de.netronic.common.intface.NeICurve[] getCurves ()

EndYValue

Property of JGIHistogram

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the end value of the numeric scale.

Accessing Methods
void setEndYValue (double newValue)
double getEndYValue ()

HistoColorScheme

Property of JGIHistogram

Typ	**de.netronic.jgantt.JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the color scheme of the histogram graph. The shadedColor is used for the background of the histogram graph. The main color dyes the date line while the alternateColor and the shadedAlternateColor are used for the calendar grid (ganttCalendarGrid). The lineColor applies to the lines of the line grid (ganttGrid).

Accessing Methods
void setHistoColorScheme (de.netronic.jgantt.JGColorScheme newValue)
de.netronic.jgantt.JGColorScheme getHistoColorScheme ()

HistoDisplayProfile

Property of JGIHistogram

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property handles the name of a profile that is used to display the calendar grid in the Gantt graph. The calendar grid is displayed if the property de.netronic.JGantt.setGanttCalendarGrid was set to **true**.

If no HistoDisplayProfile has been set when retrieving the name, **null** will be returned.
The colors of the calendar grid are controlled by the HistoColorScheme property. TimeSpans of the spanID 0 will adopt the shadedAlternateColor, all other timeSpans will be displayed in the alternateColor. Varying colors you can set by using NeMappedColor.

Accessing Methods
void setHistoDisplayProfile (java.lang.String newValue)
java.lang.String getHistoDisplayProfile ()

NumScaleBackgroundColor

Property of JGIHistogram

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the background color of the numeric scale.

Accessing Methods
void setNumScaleBackgroundColor (java.awt.Color newValue)
java.awt.Color getNumScaleBackgroundColor ()

NumScaleCaption

Property of JGIHistogram

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the caption of the numeric scale.

Accessing Methods
void setNumScaleCaption (java.lang.String newValue)
java.lang.String getNumScaleCaption ()

Also see NumScaleCaptionPosition

NumScaleCaptionPosition

Property of JGIHistogram

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**CAPTION_TOP**

This property lets you set or retrieve the position of the numeric scale caption.

Possible Values	Description
CAPTION_BOTTOM	Caption at bottom
CAPTION_CENTER	Caption centered
CAPTION_TOP	Caption at top

Accessing Methods
void setNumScaleCaptionPosition (int newValue)
int getNumScaleCaptionPosition ()

Also see NumScaleCaption

NumScaleFont

Property of JGIHistogram

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the font of the numeric scale.

Accessing Methods
void setNumScaleFont (java.awt.Font newValue)
java.awt.Font getNumScaleFont ()

NumScaleResolution

Property of JGIHistogram

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1.0**

This property lets you set or retrieve the resolution of the numeric scale. The resolution is the number of basic units per millimeter.

Accessing Methods
void setNumScaleResolution (double newValue)
double getNumScaleResolution ()

NumScaleSubdivisions

Property of JGIHistogram

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1**

This property lets you set or retrieve minor ticks (that are not annotated) of the numeric scale. For this please specify how many subunits are contained in a main unit.

Accessing Methods
void setNumScaleSubdivisions (int newValue)
int getNumScaleSubdivisions ()

Also see NumScaleUnitsBetweenMajorTicks

NumScaleTextColor

Property of JGIHistogram

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the color of text annotations in the numeric scale.

Accessing Methods
void setNumScaleTextColor (java.awt.Color newValue)
java.awt.Color getNumScaleTextColor ()

NumScaleUnitsBetweenMajorTicks

Property of JGIHistogram

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**10.0**

This property lets you set or retrieve annotated major ticks of the numeric scale. For this please specify after how many base units an annotated main tick should be set.

Accessing Methods
void setNumScaleUnitsBetweenMajorTicks (double newValue)
double getNumScaleUnitsBetweenMajorTicks ()

Also see NumScaleSubdivisions

StartYValue

Property of JGIHistogram

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the start value of the numeric scale. Unit: milliseconds since 1.1.1970.

Accessing Methods
void setStartYValue (double newValue)
double getStartYValue ()

Methods of the Interface

addCurve

Method of JGIHistogram

This method lets you assign a curve to the histogram at the last index.

Declaration
void addCurve (de.netronic.common.intface.NeICurve theCurve)

	Data Type	Description
Parameter		
theCurve	de.netronic.common.intface.NeICurve	Curve to be assigned to the histogram.
Return Value	void	

addMouseListener

Method of JGIHistogram

By this method you can add a MouseListener to your histogram.

Declaration
void addMouseListener (MouseListener mouseListener)

	Data Type	Description
Parameter		
mouseListener	MouseListener	MouseListener to be added to the histogram
Return Value	void	

Also see addMouseMotionListener

Example Code

```
final JGIHistogram jGHisto = jGantt1.getHistogram();
jGHisto.addMouseListener (new MouseAdapter() {

    public void mouseClicked(MouseEvent evt)
      {
      // Replace this line by your mouse event handling
      }
```

```
}
```

addMouseMotionListener

Method of JGIHistogram

By this method you can add a MouseMotionListener to your histogram.

Declaration
void addMouseMotionListener (MouseMotionListener mouseMotionListener)

	Data Type	Description
Parameter		
mouseMotionListener	MouseMotionListener	MouseMotionListener to be added
Return Value	void	

Also see addMouseListener

Example Code

```
final JGIHistogram jGHisto = jGantt1.getHistogram();
jGHisto.addMouseMotionListener (new MouseMotionAdapter() {

    public void mouseMoved(MouseEvent evt)
      {
      // Replace this line by your mouse motion event handling
      }
}
```

5.2 NeArrayCurveData

Belongs to Histogram

Package name **de.netronic.bean.histogram**
Implements **de.netronic.common.intface.NeICurveData**

This class lets you generate curve data objects. There are two arrays holding the curve data: one for the x-values and the other one for the y-values of the curve points. Both arrays have the same size.

Properties to Handle CurveData

XValue	Array to hold the x-values of the curve points.
YValue	Array to hold the y values of the curve points.

Methods to Handle CurveData

addValue(...)	Adds a curve point.

Constructors of the Class

The constructors of this class let you generate curve data objects.

NeArrayCurveData

Constructor of NeArrayCurveData

This constructor lets you generate curve data object the arrays of which are to be defined in the parameters.

Declaration
NeArrayCurveData (long XValue [], float YValue [])

Parameter	Data Type	Description
XValue []	long	Array to hold the x-values of the curve point data. The data type **long** accepts the number of milliseconds since 1.1.1970.
YValue []	float	Array to hold the y-values of the curve point data.

NeArrayCurveData

Constructor of NeArrayCurveData

This constructor lets you generate curve data object the array size of which is to be defined in the parameter.

Declaration
NeArrayCurveData (int initialSize)

Parameter	Data Type	Description
initialSize	int	Size of the arrays to hold the curve point data.

NeArrayCurveData

Constructor of NeArrayCurveData

This constructor lets you generate a curve data object that is automaticlly initialized with an array for 1,000 points.

Declaration
NeArrayCurveData ()

Properties of the Class

XValue

Property of NeArrayCurveData

Typ	**long[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Array to hold the x-values of the curve points. The data type **long** accepts the number of milliseconds since 1.1.1970.

Accessing Methods
void setXValue (integer index, long newValues)
void setXValue (long[] newValue)
long getXValue (integer index)
long[] getXValue ()

YValue

Property of NeArrayCurveData

Typ	**float[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Array to hold the y values of the curve points.

Accessing Methods
void setYValue (integer index, float newValues)
void setYValue (float[] newValue)
float getYValue (integer index)
float[] getYValue ()

Methods of the Class

addValue

Method of NeArrayCurveData

This method lets you add the x- and y-values of a curve point to the curve.

Declaration
void addValue (long XValue, float YValue)

	Data Type	Description
Parameter		
XValue	long	X-value of the curve point. The data type long accepts the number of milliseconds since 1.1.1970.
YValue	float	Y-value of the curve point.
Return Value	void	

5.3 NeCalculatedCurveData

Belongs to Histogram

Package name **de.netronic.bean.histogram**
Implements **de.netronic.common.intface.NeICurveData**

This class provides a constructor to calculate curve data.

Properties to Calculate Curve Data

CurveData	Source data of the curve for the calculation
Factor	Factor by which the source data will be multiplied.
Summand	Summand that is added to the result of the multiplication.

Constructors of the Class

NeCalculatedCurveData

Constructor of NeCalculatedCurveData

This constructor lets you generate a curve data object of which the data are to be calculated.The calculation consists of a multiplication and an addition: inputCurveData*factor+summand.

Declaration
NeCalculatedCurveData (de.netronic.common.intface.NeICurveData inputCurveData, float factor, float summand)

Parameter	Data Type	Description
inputCurveData	de.netronic.common.intface.-NeICurveData	Object that holds the source data of the curve for the calculation.
factor	float	Factor by which the source data will be multiplied.
summand	float	Summand that is added to the result of the multiplication.

Properties of the Class

CurveData

Property of NeCalculatedCurveData

Typ	**de.netronic.common.intface.NeICurveData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Source data of the curve for the calculation

Accessing Methods
void setCurveData (de.netronic.common.intface.NeICurveData newValue)
de.netronic.common.intface.NeICurveData getCurveData ()

Factor

Property of NeCalculatedCurveData

Typ	**float**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Factor by which the source data will be multiplied.

Accessing Methods
void setFactor (float newValue)
float getFactor ()

Summand

Property of NeCalculatedCurveData

Typ	**float**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Summand that is added to the result of the multiplication.

Accessing Methods
void setSummand (float newValue)
float getSummand ()

5.4 NeCurveData

Belongs to Histogram

Package name	**de.netronic.bean.histogram**
Implements	**de.netronic.common.intface.NeICurveData**

This class lets you define, aggregate and cumulate the data of a curve in a histogram.

Properties to Handle Curve Data

AggregatePeriod	Time period during which values are aggregated (not yet implemented).
AggregateReferencePoint	Location within the period, where the curve point is to be placed (not yet implemented).

AppData	Application data, from which the curve is to be generated.
Calendar	Calendar to be visualized by the curve data
CalendarProfileBy	Profile for nodes when using a calendar in the histogram
Cumulated	Cumulation of curve values (not yet implemented)
DataElementFilter	Filter admitting defined entities only.
DefaultCalendarProfile	Default profile for nodes when using a calendar in the histogram
EntitySetName	Name of the entity set, from which the curve is to be generated.
XEndValueBy	Final date of the curve
XStartValueBy	Start date of the curve
YValueBy	Y-value of the curve

Methods for Internal Use only

entityChanged(...)	For internal use only
entityCreated(...)	For internal use only
entityDeleted(...)	For internal use only
entitySetChanged(...)	For internal use only
entitySetCreated(...)	For internal use only
entitySetDeleted(...)	For internal use only

Constructors of the Class

NeCurveData

Constructor of NeCurveData

This constructor lets you generate an object composed of curve data. The data is given by an entity set which has to be specified yet.
To simplify matters the constructor with all necessary parameters can be used.

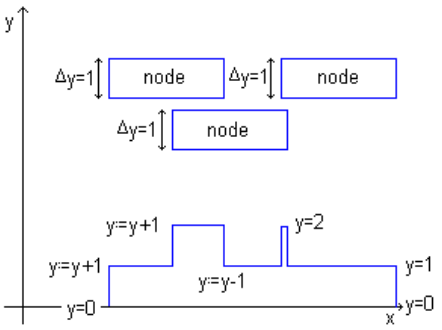
Declaration
NeCurveData ()

NeCurveData

Constructor of NeCurveData

This constructor lets you generate a curve object composed of data that were extracted from an entity set.

Declaration
NeCurveData (de.netronic.common.intface.NeIAppData appData, java.lang.String entitySetName, de.netronic.common.intface.NeIFilter filter, de.netronic.common.intface.NeIValueReference xStartValueBy, de.netronic.common.intface.NeIValueReference xEndValueBy, de.netronic.common.intface.NeIValueReference yValueBy)

Parameter	Data Type	Description
appData	de.netronic.common.intface.-NeIAppData	Application data object that the curve data are taken from.
entitySetName	java.lang.String	Name of the entity set that the curve data are generated from.
filter	de.netronic.common.intface.NeIFilter	Filter admitting defined entities only.
xStartValueBy	de.netronic.common.intface.-NeIValueReference	Start date of the curve. If xEndValueBy equals null, xStartValueBy and xEndValueBy form the curve points.
xEndValueBy	de.netronic.common.intface.-NeIValueReference	Final date of the curve. If xEndValueBy equals null, xStartValueBy and xEndValueBy form the curve points. If this value does not equal null, xStartValueBy and xEndValueBy will form the start and end date of a layer. yValueBy represents the height of the layer. A capacity curve will be derived from the values. 
yValueBy	de.netronic.common.intface.-NeIValueReference	Y values of the curve that can be aggregated

Properties of the Class

AggregatePeriod

Property of NeCurveData

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AGGREGATE_NO**

Time period during which the y-values of the curve are aggregated (not yet implemented).

Possible Values	Description
AGGREGATE_QUARTERS	The aggregation period for the y-values of the curve is a quarter of a year.
AGGREGATE_DAYS	The aggregation period for the y-values of the curve is a day.
AGGREGATE_NO	No aggregation will take place
AGGREGATE_WEEKS	The aggregation period for the y-values of the curve is a week.
AGGREGATE_YEARS	The aggregation period for the y-values of the curve is a year.

Accessing Methods
void setAggregatePeriod (int newValue)
int getAggregatePeriod ()

AggregateReferencePoint

Property of NeCurveData

Typ	**float**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0.0**

Location within the time period where the curve point is to be placed. Range of values allowed: 0.0...1.0 (not yet implemented).

Accessing Methods
void setAggregateReferencePoint (float newValue)
float getAggregateReferencePoint ()

AppData

Property of NeCurveData

Typ	**de.netronic.common.intface.NeIAppData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

AppData object, from which the curve is to be generated.

Accessing Methods
void setAppData (de.netronic.common.intface.NeIAppData newValue)
de.netronic.common.intface.NeIAppData getAppData ()

Calendar

Property of NeCurveData

Typ	**de.netronic.common.intface.NeICalendar**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you assign a calendar, of which the defined non-working and working periods are displayed in the histogram. The calendar is activated by setting a profile via the property CalendarProfileBy.

Accessing Methods
void setCalendar (de.netronic.common.intface.NeICalendar newValue)
de.netronic.common.intface.NeICalendar isCalendar ()

CalendarProfileBy

Property of NeCurveData

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you assign a profile to nodes. The string parameter needs to receive the name of an entity attribute which contains the profile.

Accessing Methods
void setCalendarProfileBy (java.lang.String newValue)
java.lang.String isCalendarProfileBy ()

Cumulated

Property of NeCurveData

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you switch the cumulation of the curve values on or off (not yet implemented).

Accessing Methods
void setCumulated (boolean newValue)
boolean isCumulated ()

DataElementFilter

Property of NeCurveData

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Filter that admits only defined entities of the entity set.

Accessing Methods
void setDataElementFilter (de.netronic.common.intface.NeIFilter newValue)
de.netronic.common.intface.NeIFilter getDataElementFilter ()

DefaultCalendarProfile

Property of NeCurveData

Typ	**de.netronic.common.intface.NeIProfile**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you assign a profile to nodes that did not receive a profile of their own. It will be used as the default profile that applies if no other profiles apply.

Accessing Methods
void setDefaultCalendarProfile (de.netronic.common.intface.NeIProfile newValue)
de.netronic.common.intface.NeIProfile isDefaultCalendarProfile ()

EntitySetName

Property of NeCurveData

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the entity set, from which the curve is to be generated.

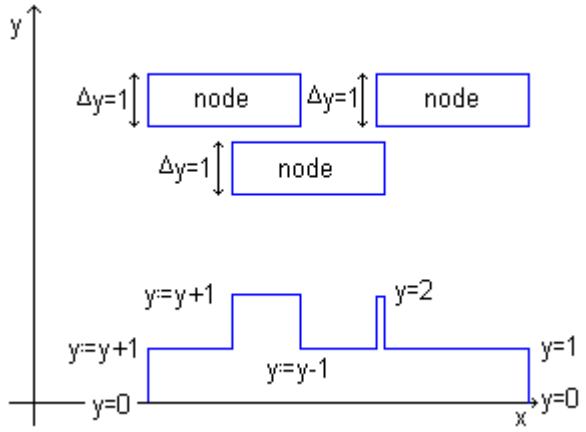
Accessing Methods
void setEntitySetName (java.lang.String newValue)
java.lang.String getEntitySetName ()

XEndValueBy

Property of NeCurveData

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Final date of the curve. If xEndValueBy equals null, xStartValueBy and xEndValueBy form the curve points.If this value does not equal null, xStartValueBy and xEndValueBythe will form the start and end date of a layer. yValueBy represents the height of the layer. A capacity curve will be derived from the values.



Accessing Methods
void setXEndValueBy (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getXEndValueBy ()

XStartValueBy

Property of NeCurveData

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Start date of the curve. If xEndValueBy equals null, xStartValueBy and xEndValueBy form the curve points.

Accessing Methods
void setXStartValueBy (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getXStartValueBy ()

YValueBy

Property of NeCurveData

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Y value of a curve point.

Accessing Methods
void setYValueBy (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getYValueBy ()

Methods of the Class

entityChanged

Method of NeCurveData

For internal use only

Declaration
void entityChanged (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	For internal use only
Return Value	void	

entityCreated

Method of NeCurveData

For internal use only

Declaration
void entityCreated (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	For internal use only
Return Value	void	

entityDeleted

Method of NeCurveData

For internal use only

Declaration
void entityDeleted (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	For internal use only
Return Value	void	

entitySetChanged

Method of NeCurveData

For internal use only

Declaration
void entitySetChanged (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	For internal use only
Return Value	void	

entitySetCreated

Method of NeCurveData

For internal use only

Declaration
void entitySetCreated (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	For internal use only
Return Value	void	

entitySetDeleted

Method of NeCurveData

For internal use only

Declaration
void entitySetDeleted (de.netronic.common.event.NeAppDataEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeAppDataEvent	For internal use only
Return Value	void	

5.5 NeCurveStyle

Belongs to Histogram

Package name **de.netronic.bean.histogram**
Implements **de.netronic.common.intface.NeICurveStyle**

This class allows to define the appearance of curves and of the areas between them.

Properties to Handle the curveStyle-Object

AreaStyle	AreaStyle object to give an appearance to the area between the curve and the reference curve.
Condition	Condition for the curve style to apply

Label	Label object that can be placed on the points of the curve.
LineStyle	LineStyle object to give an appearance to the curve.
Profile	Profile object to define workfree periods for the curveStyle object to apply (not yet implemented).
RefCurve	Reference curve

Constructors of the Class

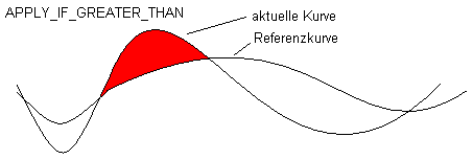
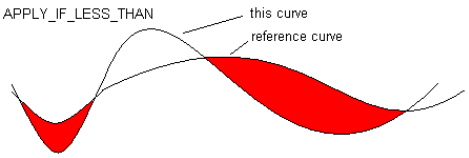
The constructors of this class let you generate curve style objects that are either empty or contain line style objects, a reference curve, the appearance of the area between the curves, a label object and a condition of which displaying the curve style object depends.

NeCurveStyle

Constructor of NeCurveStyle

This constructor lets you generate a curve style object with a line style object defined, with a reference curve, with the appearance of the area between the curves and with a condition for the appearance to apply to the area. In addition, you can generate a label to mark the points of the curve, for example, by little triangles.

Declaration
NeCurveStyle (int condition, de.netronic.common.intface.NeICurve refCurve, java.awt.Color areaStyle, java.awt.Color lineStyle, de.netronic.common.intface.NeILabel label)

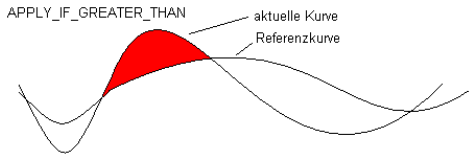
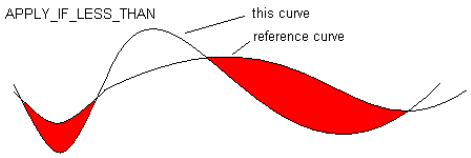
Parameter	Data Type	Description
condition	int	Condition for the curve style object to apply.
	Possible Values:	
	APPLY_ALWAYS	The object will always be applied, so this is no condition.
	APPLY_IF_GREATER_THAN	The object will be applied, where the y-values of the present curve are greater than the y-values of the reference curve: 
	APPLY_IF_LESS_THAN	The object will be applied, where the y-values of the present curve are samller than the y-values of the reference curve: 
refCurve	de.netronic.common.intface.-NeICurve	Reference curve of the curve
areaStyle	java.awt.Color	Area style object which defines the appearance of the area limited by the reference curve.
lineStyle	java.awt.Color	Line style object that determines the appearance of the curve line.
label	de.netronic.common.intface.-NeILabel	Label object, by which the curve points are represented, for example a little symbol in the shape of a triangle.

NeCurveStyle

Constructor of NeCurveStyle

This constructor lets you generate a curve style object with a line style object defined, with a reference curve, with the appearance of the area between the curves and with a condition for the appearance to apply to the area.

Declaration
NeCurveStyle (int condition, de.netronic.common.intface.NeICurve refCurve, java.awt.Color areaStyle, java.awt.Color lineStyle)

Parameter	Data Type	Description
condition	int	Condition for the curve style object to apply.
	Possible Values:	
	APPLY_ALWAYS	The object will always be applied, so this is no condition.
	APPLY_IF_GREATER_THAN	The object will be applied, where the y-values of the present curve are greater than the y-values of the reference curve: 
	APPLY_IF_LESS_THAN	The object will be applied, where the y-values of the present curve are samller than the y-values of the reference curve: 
refCurve	de.netronic.common.intface.-NeICurve	Reference curve of the curve
areaStyle	java.awt.Color	Area style object which defines the appearance of the area limited by the reference curve.
lineStyle	java.awt.Color	Line style object that determines the appearance of the curve line.

NeCurveStyle

Constructor of NeCurveStyle

This constructor lets you generate a curve style object that has a line style object.

Declaration
NeCurveStyle (java.awt.Color lineStyle)

Parameter	Data Type	Description
lineStyle	java.awt.Color	Line style object that determines the appearance of the curve line.

NeCurveStyle

Constructor of NeCurveStyle

This constructor lets you generate an empty curve style object.

Declaration
NeCurveStyle ()

Properties of the Class

AreaStyle

Property of NeCurveStyle

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Area style object that gives an appearance to the area limited by this curve and the reference curve. In the most simple case, the appearance consists of a background color. Alternatively, you can set a hatching pattern. Since an NeAreaStyle object also is derived from java.awt.Color, you can set one to specify a hatching pattern.

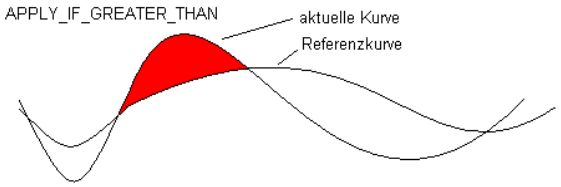
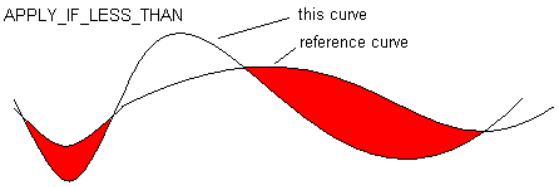
Accessing Methods
void setAreaStyle (java.awt.Color newValue)
java.awt.Color getAreaStyle ()

Condition

Property of NeCurveStyle

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**APPLY_ALWAYS**

Condition to decide whether or not the curveStyle object should apply.

Possible Values	Description
APPLY_ALWAYS	The object will always be applied, so this is no condition.
APPLY_IF_GREATER_THAN	The object will be applied, where the y-values of the present curve are greater than the y-values of the reference curve: 
APPLY_IF_LESS_THAN	The object will be applied, where the y-values of the present curve are samller than the y-values of the reference curve: 

Accessing Methods
void setCondition (int newValue)
int getCondition ()

Label

Property of NeCurveStyle

Typ	**de.netronic.common.intface.NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Label object that can be placed on the points of the curve.

Accessing Methods
void setLabel (de.netronic.common.intface.NeILabel newValue)
de.netronic.common.intface.NeILabel getLabel ()

LineStyle

Property of NeCurveStyle

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Line style object, that gives an appearance to the curve. In the most simple case, the appearance consists of a line color. Alternatively, you can set a line type. Since an NeLineStyle object also is derived from java.awt.Color, you can set one to specify a line style.

Accessing Methods
void setLineStyle (java.awt.Color newValue)
java.awt.Color getLineStyle ()

Profile

Property of NeCurveStyle

Typ	**de.netronic.common.intface.NeIProfile**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Profile object to define workfree periods for the curveStyle object to apply (not yet implemented).

Accessing Methods
void setProfile (de.netronic.common.intface.NeIProfile newValue)
de.netronic.common.intface.NeIProfile getProfile ()

RefCurve

Property of NeCurveStyle

Typ	**de.netronic.common.intface.NeICurve**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

While this curve limits an area on this side, the reference curve limits the area on the opposite side.

Accessing Methods
void setRefCurve (de.netronic.common.intface.NeICurve newValue)
de.netronic.common.intface.NeICurve getRefCurve ()

5.6 NeICurve

Belongs to Histogram

Package name **de.netronic.common.intface**

This interface represents a curve in a histogram.

Properties to Handle the Appearance of the Curves

CurveData	Retrieves the curve data
CurvePoints	Retrieves the curve data
CurveStyles	Appearance of the curve

Methods to Handle the Appearance of the Curves

getCurvePointsInInterval(...)	Curve points of an interval
updateDrawingElements()	For internal use only

Properties of the Interface

CurveData

Read Only Property of NeICurve

Typ	**de.netronic.common.intface.NeICurveData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the curve data. They define the shortes line to connect the data.

Accessing Methods
de.netronic.common.intface.NeICurveData getCurveData()

Also see CurvePoints

CurvePoints

Read Only Property of NeICurve

Typ	**de.netronic.common.intface.NeICurveData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the curve data. They contain the data of the actual line (not just of the shortest line to connect the data), for example all data of the steps in a step curve.

Accessing Methods
de.netronic.common.intface.NeICurveData getCurvePoints()

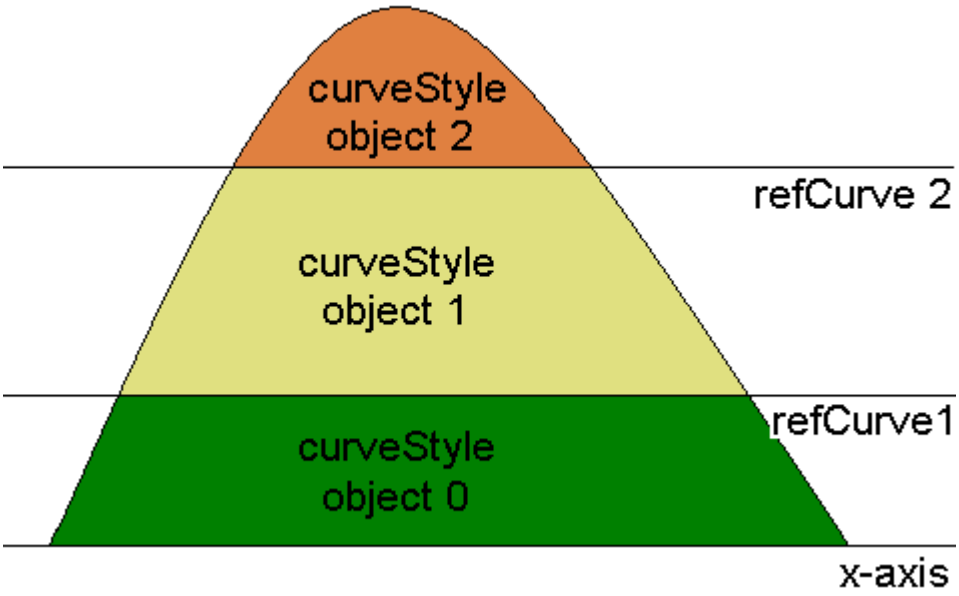
Also see CurveData

CurveStyles

Property of NeICurve

Typ	**NeICurveStyle[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

The CurveStyle object to determines the appearance of the curve. A curve may have several curveStyle objects that are applied according to the priority of their index. As shown in the picture below, first curveStyle object[0] is drawn, then curveStyle object[1] and finally curveStyle object[2]:



Accessing Methods
void setCurveStyles (integer index, NeICurveStyle newValues)
void setCurveStyles (NeICurveStyle[] newValue)
NeICurveStyle getCurveStyles (integer index)
NeICurveStyle[] getCurveStyles ()

Methods of the Interface

getCurvePointsInInterval

Method of NeICurve

Curve points of an interval

Declaration
void getCurvePointsInInterval (long xEnd, long xStart)

	Data Type	Description
Parameter		
xEnd	long	Ende of the interval, of which the curve points are to be retrieved.
xStart	long	Start of the interval, of which the curve points are to be retrieved.
Return Value	void	

updateDrawingElements

Method of NeICurve

For internal use only

Declaration
de.netronic.common.drawingelement.NeDEList updateDrawingElements ()

	Data Type	Description
Return Value	de.netronic.common.drawingelement.-NeDEList	

5.7 NeICurveData

Belongs to Histogram

Package name **de.netronic.common.intface**

This interface represents curve data and offers properties and methods to handle them.

Properties to Handle the Curve Data

NumberOfPoints	Retrieves the number of curve points.

Methods to Handle the Curve Data

addCurveDataListener(...)	Adds an NeCurveDataListener
getIndexOfClosestPoint(...)	Index of the closest a curve point
getIndexOfNextPoint(...)	Index of the curve point following
getIndexOfPreviousPoint(...)	Index of the curve point following
getXValueAsDouble(...)	X-value of a curve point as "double" data type
getXValueAsFloat(...)	X-value of a curve point as "float" data type.
getXValueAsLong(...)	X-value of a curve point as "long" data type
getYValueAsDouble(...)	Y-value of a curve point as "double" data type
getYValueAsFloat(...)	Y-value of a curve point as "float" data type.

getYValueAsLong(...)	Y-value of a curve point as "long" data type

Properties of the Interface

NumberOfPoints

Read Only Property of NeICurveData

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Number of curve points.

Accessing Methods
int getNumberOfPoints()

Methods of the Interface

addCurveDataListener

Method of NeICurveData

This method lets you add a NeCurveDataListener to the listener list.

Declaration
void addCurveDataListener (de.netronic.common.event.NeCurveDataListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeCurveDataListener	Listener to be added
Return Value	void	

getIndexOfClosestPoint

Method of NeICurveData

Index of the point located closest to the point specified in the parameter.

Declaration
int getIndexOfClosestPoint (java.awt.Point point)

	Data Type	Description
Parameter		
point	java.awt.Point	Curve point the closest point of which is to be retrieved.
Return Value	int	Index of the curve point returned

getIndexOfNextPoint

Method of NeICurveData

This method can be used in an iterative loop. It retrieves the index of the curve point following the point specified in the parameter.

Declaration
int getIndexOfNextPoint (java.awt.Point point)

	Data Type	Description
Parameter		
point	java.awt.Point	Curve point the succeeding point of which is to be retrieved.
Return Value	int	Index of the curve point returned

getIndexOfPreviousPoint

Method of NeICurveData

This method can be used in an iterative loop. It retrieves the index of the curve point preceding the point specified in the parameter.

Declaration
int getIndexOfPreviousPoint (java.awt.Point point)

	Data Type	Description
Parameter		
point	java.awt.Point	Curve point the preceding point of which is to be retrieved.
Return Value	int	Index of the curve point returned

getXValueAsDouble

Method of NelCurveData

This method lets you retrieve the x-value of a curve point as "double" data type.

Declaration
double getXValueAsDouble (int index)

	Data Type	Description
Parameter		
index	int	Index of the curve point, the x-value of which is to be retrieved.
Return Value	double	X-value of the curve point returned

getXValueAsFloat

Method of NelCurveData

This method lets you retrieve the x-value of a curve point as "float" data type.

Declaration
float getXValueAsFloat (int index)

	Data Type	Description
Parameter		
index	int	Index of the curve point, the x-value of which is to be retrieved.
Return Value	float	X-value of the curve point returned

getXValueAsLong

Method of NeICurveData

This method lets you retrieve the x-value of a curve point as a "long" data type.

Declaration
long getXValueAsLong (int index)

	Data Type	Description
Parameter		
index	int	Index of the curve point, the x-value of which is to be retrieved.
Return Value	long	X-value of the curve point returned

getYValueAsDouble

Method of NeICurveData

This method lets you retrieve the y-value of a curve point as "double" data type.

Declaration
double getYValueAsDouble (int index)

	Data Type	Description
Parameter		
index	int	Index of the curve point, the y-value of which is to be retrieved.
Return Value	double	Y-value of the curve point returned

getYValueAsFloat

Method of NeICurveData

This method lets you retrieve the y-value of a curve point as "float" data type.

Declaration
float getYValueAsFloat (int index)

	Data Type	Description
Parameter		
index	int	Index of the curve point, the y-value of which is to be retrieved.
Return Value	float	Y-value of the curve point returned

getYValueAsLong

Method of NelCurveData

This method lets you retrieve the y-value of a curve point as a "long" data type.

Declaration
long getYValueAsLong (int index)

	Data Type	Description
Parameter		
index	int	Index of the curve point, the y-value of which is to be retrieved.
Return Value	long	Y-value of the curve point returned

5.8 NelCurveStyle

Belongs to Histogram

Package name **de.netronic.common.intface**

This interface represents a curve style object and lets you define the appearance of a curve.

Properties to Handle the curveStyle object

AreaStyle	AreaStyle object to give an appearance to the area between the curve and the reference curve.
Condition	Condition for the curve style to apply
Label	Label object to be placed on the points of the curve.

LineStyle	LineStyle object to give an appearance to the curve line.
Profile	Profile object of the curve style object.
RefCurve	Reference curve

Properties of the Interface

AreaStyle

Property of NeICurveStyle

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Area style object, that gives an appearance to the area limited by this curve and the reference curve. For the first, the object represents a color. But then you can also pass an NeAreaStyle object derived from the class java.awt.Color. In the latter case, you could also pass a colored hatching pattern instead of a mere background color.

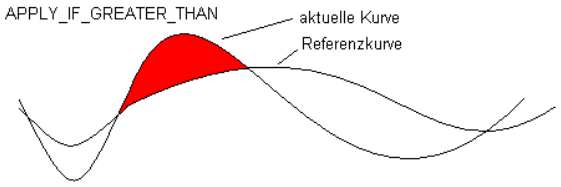
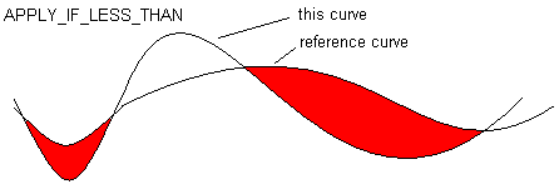
Accessing Methods
void setAreaStyle (java.awt.Color newValue)
java.awt.Color getAreaStyle ()

Condition

Property of NeICurveStyle

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**APPLY_ALWAYS**

Condition to decide whether or not the curveStyle object should apply

Possible Values	Description
APPLY_ALWAYS	The object will always be applied, so this is no condition.
APPLY_IF_GREATER_THAN	The object will be applied, where the y-values of the present curve are greater than the y-values of the reference curve: 
APPLY_IF_LESS_THAN	The object will be applied, where the y-values of the present curve are samller than the y-values of the reference curve: 

Accessing Methods
void setCondition (int newValue)
int getCondition ()

Label

Property of NeICurveStyle

Typ	**de.netronic.common.intface.NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Label object that is to be placed on the points of the curve.

Accessing Methods
void setLabel (de.netronic.common.intface.NeILabel newValue)
de.netronic.common.intface.NeILabel getLabel ()

LineStyle

Property of NeICurveStyle

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve an line style object, that gives an appearance to the curve line. Alternatively, you can pass an NeLineStyle object derived from java.awt.Color, which matches in terms of the type. It allows to also pass a line type an a line thickness. In case a curve has several curve styles, only the line style of no. zero will apply.

Accessing Methods
void setLineStyle (java.awt.Color newValue)
java.awt.Color getLineStyle ()

Profile

Property of NeICurveStyle

Typ	**de.netronic.common.intface.NeICurveStyle**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Profile object that can be assigned to the curve style object or that can be retrieved.

Accessing Methods
void setProfile (de.netronic.common.intface.NeICurveStyle newValue)
de.netronic.common.intface.NeICurveStyle getProfile ()

RefCurve

Property of NeICurveStyle

Typ	**de.netronic.common.intface.NeICurve**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference curve, that limits an area on the opposite side.

Accessing Methods
void setRefCurve (de.netronic.common.intface.NeICurve newValue)
de.netronic.common.intface.NeICurve getRefCurve ()

5.9 NeLineCurve

Belongs to Histogram

Package name	**de.netronic.bean.histogram**
Implements	**de.netronic.common.intface.NeICurve**

This class lets you generate line curves. Line curves are characterized by direct lines between points.

Properties to Handle the Line Curve

Methods for Internal Use only

updateDrawingElements(...)	For internal use only
updateDrawingElements(...)	For internal use only

Methods to Handle the Line Curve

getCurvePointsInInterval(...)	Retrieves all curve points within an interval.

Constructors of the Class

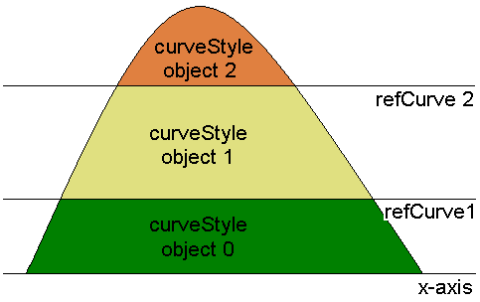
The constructors of this class let you generate empty line curve objects or objects that have an appearance and curve data.

NeLineCurve

Constructor of NeLineCurve

This constructor lets you generate a line curve object with dates and a curveStyle object set.

Declaration
NeLineCurve (de.netronic.common.intface.NeICurveData curveData, de.netronic.common.intface.NeICurveStyle[] curveStyles, boolean visible, boolean doubleXValues)

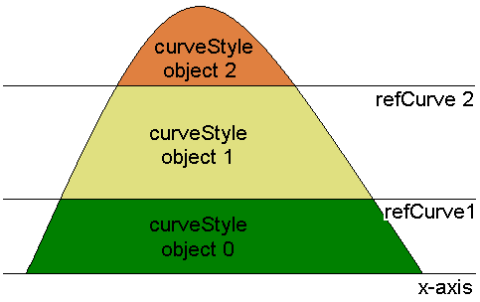
Parameter	Data Type	Description
curveData	de.netronic.common.intface.-NeICurveData	Curve data passed to generate the curve.
curveStyles	de.netronic.common.intface.-NeICurveStyle[]	CurveStyle objects to set the appearance of the curve. A curve may have several curveStyle objects that are applied according to the priority of their index. As shown in the picture below, first curveStyle object[0] is drawn, then curveStyle object[1] and finally curveStyle object[2]: 
visible	boolean	Defines whether the curve is visible (true) or invisible (false).
doubleXValues	boolean	For future versions. Defines, whether (true) or not (false) the X value is to be stored twice.

NeLineCurve

Constructor of NeLineCurve

This constructor lets you generate a line curve object with dates and a curveStyle object set.

Declaration
NeLineCurve (de.netronic.common.intface.NeICurveData curveData, de.netronic.common.intface.NeICurveStyle[] curveStyles)

Parameter	Data Type	Description
curveData	de.netronic.common.intface.-NeICurveData	Curve data passed to generate the curve.
curveStyles	de.netronic.common.intface.-NeICurveStyle[]	CurveStyle objects to set the appearance of the curve. A curve may have several curveStyle objects that are applied according to the priority of their index. As shown in the picture below, first curveStyle object[0] is drawn, then curveStyle object[1] and finally curveStyle object[2]: 

NeLineCurve

Constructor of NeLineCurve

This constructor lets you generate an empty line curve object.

Declaration
NeLineCurve ()

Methods of the Class

getCurvePointsInInterval

Method of NeLineCurve

Retrieves all curve points within a defined interval.

Declaration
de.netronic.common.intface.NeICurveData getCurvePointsInInterval (long XStart, long XEnd)

	Data Type	Description
Parameter		
XStart	long	Start of the interval. Unit: 1/1000 seconds since January 1st, 1970, 00:00 h
XEnd	long	End of the interval. Unit: Unit: 1/1000 seconds since January 1st, 1970, 00:00 h
Return Value	de.netronic.common.intface.-NeICurveData	

updateDrawingElements

Method of NeLineCurve

For internal use only

Declaration
void updateDrawingElements (de.netronic.bean.histogram.NeXYGraph histo)

	Data Type	Description
Parameter		
histo	de.netronic.bean.histogram.NeXYGraph	For internal use only
Return Value	void	

updateDrawingElements

Method of NeLineCurve

For internal use only

Declaration
void updateDrawingElements (char toDo, de.netronic.bean.histogram.NeXYGraph histo)

	Data Type	Description
Parameter		
toDo	char	For internal use only
histo	de.netronic.bean.histogram.NeXYGraph	For internal use only
Return Value	void	

5.10 NeStackedCurveData

Belongs to Histogram

Package name **de.netronic.bean.histogram**

This class lets you stack (add up) curve data.

Properties to Handle Stacked Curve Data

AdditionType	The way of calculation for curves to be added
CurveData	Curve data to be stacked
StackedCurveData	Resulting curve of the stacked data

Methods to Handle Stacked Curve Data

addCurveData(...)	Adds curve data to existing curve data.

Constructors of the Class

The constructor of this class lets you generate a curve data object of stacked (added) curve data.

NeStackedCurveData

Constructor of NeStackedCurveData

This constructor lets you generate a curve data object of stacked (added) curve data. The curves to be stacked and the type of addition are passed via parameters.

Declaration
NeStackedCurveData (de.netronic.common.intface.NeICurveData [] summands, int addtionType)

Parameter	Data Type	Description
summands	de.netronic.common.intface.-NeICurveData []	Array of curve objects to be stacked
addtionType	int	Type of addition, according to which stacked curves are to be summed up.
	Possible Values:	
	CAPACITY	Addition type for capacity curves

NeStackedCurveData

Constructor of NeStackedCurveData

This constructor lets you generate a curve data object of stacked (added) curve data. You can already set the number of curves to be added, as well as the addition type, while the curve data itself will be set later on via the properties.

Declaration
NeStackedCurveData (int maximumNumberOfData, int addtionType)

Parameter	Data Type	Description
maximumNumberOf Data	int	Maximum number of curve data to be stacked
addtionType	int	Type of addition, according to which stacked curves are to be summed up.
	Possible Values:	
	CAPACITY	Addition type for capacity curves

Properties of the Class

AdditionType

Property of NeStackedCurveData

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**CAPACITY**

This property describes the way of calculations for the curves to be added.

Possible Values	Description
CAPACITY	Addition type for capacity curves

Accessing Methods
void setAdditionType (int newValue)
int getAdditionType ()

CurveData

Property of NeStackedCurveData

Typ	**de.netronic.common.intface.NeICurveData[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Curve data to be stacked.

Accessing Methods
void setCurveData (integer index, de.netronic.common.intface.NeICurveData newValues)
void setCurveData (de.netronic.common.intface.NeICurveData[] newValue)
de.netronic.common.intface.NeICurveData getCurveData (integer index)
de.netronic.common.intface.NeICurveData[] getCurveData ()

StackedCurveData

Read Only Property of NeStackedCurveData

Typ	**de.netronic.common.intface.NeICurveData[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the resulting curve, the data of which were stacked.

Accessing Methods
de.netronic.common.intface.NeICurveData StackedCurveData (integer index)
de.netronic.common.intface.NeICurveData[] StackedCurveData()

Methods of the Class

addCurveData

Method of NeStackedCurveData

Adds curve data to existing curve data.

Declaration
void addCurveData (de.netronic.common.intface.NeICurveData summand)

	Data Type	Description
Parameter		
summand	de.netronic.common.intface.-NeICurveData	Curve data to be added to existing curve data.
Return Value	void	

5.11 NeStepCurve

Belongs to Histogram

Package name **de.netronic.bean.histogram**
Implements **de.netronic.common.intface.NeICurve**

This class lets you generate step curves. Step curves are characterised by direct lines between points.

Properties to Handle the Step Curve

CurvePoints	Retrieves the points of the curve.
ReferencePoint	Situation of the step

Methods to Handle the Step Curve

getCurvePointsInInterval(...)	Retrieves the curve points within the interval specified.

Constructors of the Class

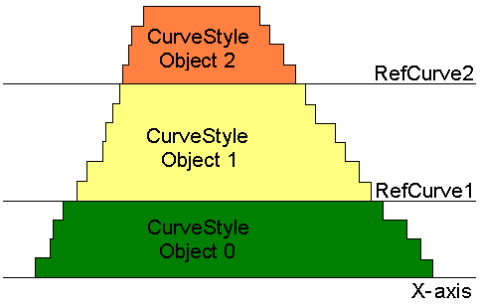
The constructors of this class let you generate empty step curve objects or objects that have an appearance and curve data.

NeStepCurve

Constructor of NeStepCurve

This constructor lets you generate a step curve object with dates and a curveStyle object set.

Declaration
NeStepCurve (de.netronic.common.intface.NeICurveData curveData, de.netronic.common.intface.NeICurveStyle[] curveStyles, float referencePoint, boolean visible, boolean doubleXValues)

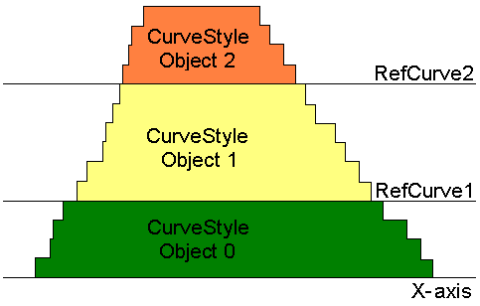
Parameter	Data Type	Description
curveData	de.netronic.common.intface.-NeICurveData	Curve data passed to that the curve is generated from.
curveStyles	de.netronic.common.intface.-NeICurveStyle[]	CurveStyle objects to set the appearance of the curve. A curve may have several curveStyle objects that are applied according to the priority of their index. As shown in the picture below, first curveStyle object[0] is drawn, then curveStyle object[1] and finally curveStyle object[2]: 
referencePoint	float	Situation of a step between two points of a step curve. The value 1.0 will place it onto a point (which makes the curve a capacity curve), the value 0.5 will place it exactly in the center between two points. Any value between 0.0 and 1.0 is allowed.
visible	boolean	Defines whether the curve is visible (true) or invisible (false).
doubleXValues	boolean	For future versions. Defines, whether (true) or not (false) the X value is to be stored twice.

NeStepCurve

Constructor of NeStepCurve

This constructor lets you generate a step curve object with dates and a curveStyle object set.

Declaration
NeStepCurve (de.netronic.common.intface.NeICurveData curveData, de.netronic.common.intface.NeICurveStyle[] curveStyles)

Parameter	Data Type	Description
curveData	de.netronic.common.intface.-NeICurveData	Curve data passed to that the curve is generated from.
curveStyles	de.netronic.common.intface.-NeICurveStyle[]	CurveStyle objects to set the appearance of the curve. A curve may have several curveStyle objects that are applied according to the priority of their index. As shown in the picture below, first curveStyle object[0] is drawn, then curveStyle object[1] and finally curveStyle object[2]: 

NeStepCurve

Constructor of NeStepCurve

This constructor lets you generate an empty step curve object.

Declaration
NeStepCurve ()

Properties of the Class

CurvePoints

Read Only Property of NeStepCurve

Typ	**de.netronic.common.intface.NeICurveData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the points of a curve.

Accessing Methods
de.netronic.common.intface.NeICurveData getCurvePoints()

ReferencePoint

Property of NeStepCurve

Typ	**float**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1.0**

Situation of a step between two points of a step curve. The value 1.0 will place it onto a point (which makes the curve a capacity curve), the value 0.5 will place it exactly in the center between two points. Any value between 0.0 and 1.0 is allowed.

Accessing Methods
void setReferencePoint (float newValue)
float getReferencePoint ()

Methods of the Class

getCurvePointsInInterval

Method of NeStepCurve

Retrieves the curve points within the interval specified.

Declaration
void getCurvePointsInInterval (long xStart, long xEnd)

	Data Type	Description
Parameter		
xStart	long	Start of the Interval
xEnd	long	End of the Interval
Return Value	void	

6 JGantt

This component contains some of the classes and interfaces that you need to develop the visible parts of a JGantt diagram.

The JGantt-Component consists of the below classes:

JGantt	This class lets you handle the visible elements of a JGantt diagram.
JGanttSynchronizerPanel	The JGanttSynchronizerPanel class places two instances of VARCHART JGantt on top of each other and synchronizes their time-related sections and their tables.
JGColorScheme	This class lets you handle a color scheme.
JGDiagramAnnotation	This class contains properties and methods to design headers and footers of the diagram.
JGEntitySetFilter	This class lets you filter entities which are contained in the given entity set.
JGIPersistenceManager	This interface is the access to the persistency concept of VARCHART JGantt.
JGIPrintManager	This interface contains properties to handle the printing.
JGIsHierarchyFilter	This class allows to check whether or not **grouping by hierarchy** was set to VARCHART JGantt.
JGLayouterHelper	This class holds the methods of the layout of grouping and hierarchy structures in a Gantt chart.
JGLegend	This class handles the legend for the Gantt diagram.
JGLevelFilter	This class lets you filter entities which have a specific hierarchy or grouping level.
JGNodeDesign	This class lets you handle a node design.
JGSymbol	This class lets you handle node symbols.
JGVertLineGrid	This class describes a line grid in a section of the Gantt graph and the histogram.
JGVertLineGrids	This class offers methods to handle a collection of the vertical line grids of a time scale section.
JPEIJGanttPropertyEditor	

6.1 JGantt

Belongs to JGantt

Package name	**de.netronic.jgantt**
Extends	**javax.swing.JComponent**
Implements	**java.awt.event.MouseMotionListener**
I	**java.awt.event.MouseListener**
I	**java.beans.PropertyChangeListener**
I	**java.io.Externalizable**

This class lets you handle the visible elements of a JGantt diagram.

Properties to Handle Filters

Properties to Handle Groups of Nodes

GroupBy	Name of the attribute that holds the grouping value
GroupHierarchyBy	Name of the group node attribute to hold the hierarchy code
GroupLayout	Type of group layout
GroupMode	Grouping mode
GroupNodeAnnotations	Attributes that hold annotations of group nodes
GroupNodeColorScheme	Color scheme of group nodes
GroupNodeDateFormat	Format of dates in group node annotations
GroupNodeDates	Attributes for dates of group nodes
GroupNodeDesign	Design object of group nodes
GroupNodeFont	Font of group node annotations
GroupNodeProfile	Default profile for group nodes
GroupNodeProfileBy	Name of the group node attribute holding a calendar profile
GroupNodePropertiesEnabled	Defines, whether or not the other properties of group nodes are to apply
GroupNodeSetName	Name of the group node entity set
GroupNodeSymbols	Symbols for group nodes
GroupNodeZeroLengthSymbol	Symbol to display group nodes of no extent
GroupNodeZeroLengthVisible	Defines, whether or not group nodes of no extent are displayed
GroupsInitiallyCollapsed	Defines, whether or not group nodes are collapsed when initially loaded

GroupsSortedBy	Sorting of all groups in the Gantt chart
RootGroup	Retrieves the root group in multiple grouping.

Properties to Handle Links and Link Entities

LinkNodeDateIndexes	Names of attributes that hold the node dates for links to join
LinkSetName	Name of the link entity set
LinkSourceNodeBy	Name of the attribute holding the user-generated identification tag of the source node
LinkTargetNodeBy	Name of the attribute holding the user-generated identification tag of the target node
LinkTypeBy	Name of the attribute holding the link type

Properties to Handle Nodes and Node Entities

CollapseFilter	Filter for collapsed nodes
EntityEditorDialog	Dialog to edit entities.
GroupSetFilter	Filter for entities present in the group entity set
NodeAnnotations	Names of attributes to hold annotations of nodes
NodeColorScheme	Color scheme for nodes
NodeDateFormat	Format of dates in node annotations
NodeDates	Names of attributes for dates of nodes
NodeDesign	Design object for the nodes
NodeFont	Font of node annotations
NodeHierarchyBy	Attribute that holds the hierarchy code
NodePhantomPositionLine	Place of the positioning line in node phantoms
NodeProfile	Default profile for the nodes
NodeProfileBy	Name of a node attribute to hold the name of a profile individually for a node.
NodeSetName	Name of the node entity set to hold node entities
NodesSortedBy	Comparator object to compare nodes
NodeSymbols	Symbols for the graphical expression of the nodes
NodeZeroLengthSymbol	Symbol to display nodes of no extent
NodeZeroLengthVisible	Defines whether or not nodes of zero length are visible

Properties to Handle the Application Data

AppData	The application data object of the JGantt
CanonicalHierarchyCode	When using **KeepIncompleteHierarchy** you can set here the entity atttribute for the internally generated hierarchy codes.
KeepIncompleteHierarchy	An incomplete hierarchy can be kept the way it is, as far as possible.

Properties to Handle the Customizer

DataDefinitionName	Name of the XML file used by the customizer dialog

Properties to Handle the Date Line

GanttDateLine	Date line in the Gantt graph

Properties to Handle the Gantt Diagram

ActiveNodeFilter	Filter on nodes that are visible in the Gantt chart
BuildNumber	Retrieves the build number of the JGantt object
DiagramAnnotationsOnScreen	Defines, whether or not the diagram title is displayed on the screen
DiagramControlBar	Control panel for user interactions
DiagramControlBarVisible	Is the control panel for the ScreenManager functions visible?
DiagramEmptyRow	Defines, whether or not an empty row is available at the bottom of the diagram
DiagramHistogramHeightPercent	Portion of the total diagram height occupied by the histogram
DiagramPageBreakMode	Page Breaking Mode for Printing
DiagramRowMargin	Offset between the row border and a node
DiagramRowMinimumHeight	Minimum height of rows in the Gantt diagram
DiagramTableWidthPercent	Relative width of the table
DiagramTitleColorScheme	Color scheme of the title of the Gantt diagram
DiagramTitleFont	Font of the diagram title
DiagramTitleOnScreen	Defines, whether or not the diagram title is displayed on the screen
DiagramTitleText	String of the title of the Gantt diagram

DiagramZoomFactor	Zoom factor of the Gantt diagram
GanttFixedTopRow	Fixes the first row at the top of the gantt chart.
InteractionAutoScrollEnabled	Defines, whether or not auto-scrolling of the diagram is switched on
InteractionMoveHorizontalWithChildren	Child nodes of non-leaf nodes in hierarchies also move when their parents are moved.
NumberOfRows	Number of rows
PrintManager	Retrieves the print manager object.
SectionVertLineGridsEx	Vertical line grids of a time scale section
Version	Retrieves the version number of the JGantt object
VersionBuildNumber	Retrieves the version and build number of the JGantt object
VertLineGridsEx	Vertical line grids of the first time scale section

Properties to Handle the Gantt Graph

Calendar	The calendar of the JGantt object
GanttCalendarGrid	Defines, whether or not a calendar grid is displayed in the Gantt graph
GanttColorScheme	Color scheme of the Gantt graph
GanttDisplayProfile	Name of the profile that is used to display the calendar grid in the Gantt graph
GanttGlassProfile	Name of a profile that is used to additionally mark certain time spans in the ganttgraph.
GanttGraph	Instance of the Gantt graph
GanttGrid	Line grid of the Gantt graph
GanttGridLineStyle	Appearance of the lines of the line grid in the gantt chart.
GanttSectionGridLineStyles	Line grids of the Gantt graph in a specific time scale section
GanttSectionGrids	Line grids of the Gantt graph in a specific time scale section
HistoColorScheme	Color scheme of the histogram graph
Id	Identification of the JGantt object

Properties to Handle the Histogram

Histogram	The histogram of JGantt
HistogramVisible	Should the histogram be visible in the diagram?

Properties to Handle the Scheduler

Scheduler	Retrieves the instance of the scheduling module

Properties to Handle the Table

Table	The table of JGantt
Table3D	Defines, whether or not the table is displayed three-dimensionally
TableAutoWrap	Different automatic break modes for head lines in the default table
TableColorScheme	Color scheme of the table
TableColumns	Names of node attributes for table columns
TableColumnTitleFont	Font of the column titles of the table
TableColumnTitlesSource	Source that the table column titles are read from
TableColumnWidths	Widths of table columns
TableCornerText	Text string for the top left corner of the table
TableDateFormat	Format of table dates
TableEditable	Can table be edited?
TableFont	Font for the node rows in the table
TableGroupFont	Font for the group rows in the table
TableHierarchyColumn	Table column to display the node hierarchy
TableHierarchyIndentWidth	Width of the indent of the string in the table hierarchy column
TableRowTitleFont	Font of the row titles of the table
TableRowTitlesVisible	Visibility of leftmost column (row titles)
TableSashOneTouchExpandable	If you set this property to false, the arrows at the table sash (the separator between table and gantt graph), which enable collapsing or expanding of the table, will disappear.
TableVisible	Defines whether the table is visible

Properties to Handle the Time Scale

ResolutionUserModifiable	Sets the **resolution modify** interaction.
TimeScale	Instance of the time scale
TimeScale3D	Defines, whether or not the time scale is displayed three-dimensionally

TimeScaleAbsoluteResolution	The absolute resolution of the time scale
TimeScaleAntialiasText	Antialiasing on texts
TimeScaleAtBottomVisible	Defines, whether the bottom time scale is visible
TimeScaleAtTopVisible	Defines, whether the top time is scale visible
TimeScaleCollapseDisplayMode	Defines, in which way collapsed non-working times are visualized.
TimeScaleCollapseFactor	Factor, by which the time scale is downsized by collapsing.
TimeScaleCollapseProfile	Collapse profile for the time scale
TimeScaleColorScheme	Color scheme of the time scale
TimeScaleDisplayProfile	The name of the calendar profile of the time scale.
TimeScaleDynamic	Automatic adaption of time scale resolution
TimeScaleEnd	Final date of the time scale
TimeScaleFont	Font of the time scale
TimeScaleResolution	The resolution of the time scale
TimeScaleResolutionUserModifiable	Sets the **resolution modify** interaction.
TimeScaleSectionAbsoluteResolutions	The absolute resolutions of the time scale sections
TimeScaleSectionColorSchemes	The JGColorSchemes of the time scale sections
TimeScaleSectionFonts	The Fonts of the time scale sections
TimeScaleSectionResolutions	The resolution of the time scale sections
TimeScaleSectionTypes	The types of time scale sections
TimeScaleStart	The start date of the time scale
TimeScaleType	The type of time scale
TimeScaleViewEnd	End date of the view that takes the time scale and the Gantt graph to the screen
TimeScaleViewStart	Start date of the view that displays the time scale and the Gantt graph
ViewEnd	End date of the view that takes the time scale and the Gantt graph to the screen
ViewStart	Start date of the view that displays the time scale and the Gantt graph
VisibleAtBottom	Defines, whether the bottom time scale is visible
VisibleAtTop	Defines, whether the top time is scale visible

Properties to Handle User Interactions

AdvancedMouseInteractions	Hereby you can specify an alternative reaction to mouse interactions in the timecale.
InteractionDataEditingEnabled	Allows to interactively edit data, if the data-editing license is available.
InteractionDragEnabled	Defines, whether or not the JGantt object acts as a drag source
InteractionDropEnabled	Defines, whether or not the JGantt object acts as a drop target
InteractionEditLinksEnabled	Dialog to modify existing link data
InteractionEditNewLinkEnabled	Entry dialog to modify link data
InteractionEditNewNodeEnabled	A dialog is invoked to edit the data of a new node
InteractionEditNodesEnabled	Dialog to modify existing node data
InteractionGroupNodeMoveMode	Defines, whether or not group nodes or parts of group nodes can be moved
InteractionGroupNodeResizeMode	Defines, whether or not group nodes or parts of group nodes can be resized
InteractionInfoWindowEnabled	Defines, whether or not an information window is displayed when an interaction is performed
InteractionLinksAtLeafNodesOnly	Defines, whether or not links can be drawn interactively between leaf nodes only
InteractionLinkSelectionRange	Selection area of a link
InteractionMode	Interaction mode of the JGantt object
InteractionMoveMultipleNodesEnabled	Moving several nodes in one go
InteractionNodeMoveMode	Defines, whether or not nodes or parts of nodes can be moved
InteractionNodeResizeMode	Defines, whether or not nodes or parts of nodes can be resized
InteractionPopupEnabled	Defines, whether or not the right mouse button menu is enabled to pop up
InteractionResizeMultipleNodesEnabled	Re-sizing several nodes in one go
InteractionSelectionSynchronized	Controls synchronous selection in table and Gantt graph.
InteractionSelectTopNodeOrLinkOnly	After clicking a group of overlaying nodes only the node at the top will be selected.
InteractionTableMouseWheelScrolling	Reaction of the table to a mouse wheel event
InteractionTimeQuantum	Size of leaps in the horizontal motion of nodes
PopupEnabled	Defines, whether or not the right mouse button menu is enabled to pop up on the time scale

TimeScaleAdvancedMouseInteractions	Hereby you can specify an alternative reaction to mouse interactions in the timecale.
TimeScalePopupEnabled	Defines, whether or not the right mouse button menu is enabled to pop up on the time scale

Methods for Internal Use Only

doLayout()	For internal use only
generateMouseClicked(...)	For internal use only
makeImage(...)	For internal use only
makeImage(...)	For internal use only
mouseClicked(...)	For internal use only.
mouseDragged(...)	For internal use only.
mouseEntered(...)	For internal use only
mouseExited(...)	For internal use only
mouseMoved(...)	For internal use only
mousePressed(...)	For internal use only
mouseReleased(...)	For internal use only
out(...)	For internal use only
paint(...)	For internal use only
propertyChange(...)	For internal use only
readExternal(...)	For internal use only
writeExternal(...)	For internal use only

Methods that Affect the Hierarchy

expandNode(...)	Expands the child nodes that depend on the parent node in the hierarchy.
getLayouterHelper()	Retrieves the JGLayouterHelper object allocated to this instance of VARCHART JGantt.
getParentNode(...)	Retrieves the parent node

Methods to Handle Date Lines

addGanttDateLine(...)	Adds a date line to the Gantt graph
removeGanttDateLine(...)	Removes a date line from the Gantt graph

Methods to Handle Groups of Activities

getLayouterGroupForNode(...)	Retrieves the group to which an activity belongs.
identifyLayouterGroup(...)	Retrieves the group object at the coordinates given

Methods to Handle Nodes and Links

addLinkChangeListener(...)	Adds a listener for link change events
addNodeChangeListener(...)	Adds a listener for node change events
collapseNode(...)	Collapses a node in the hierarchy
deleteSelectedNodes()	All nodes selected are removed
editSelectedEntities()	Invokes an editor dialog on the selected entities
getRowIndexForNode(...)	Retrieves the row index of the entity passed
identifyEntities(...)	Retrieves the entities of all objects located at the coordinates given
identifyEntityAttribute(...)	Retrieves the attribute name of the table column located at given coordinates
identifyLabels(...)	Identifies all NePicture objects at a defined position of the Gantt Graph.
identifyLayers(...)	Identifies all NeLayer objects at a defined position of the Gantt Graph.
identifyRow(...)	Retrieves the row of the Gantt diagram in which the mouse is located.
identifyTime(...)	Retrieves the time of the time scale at the coordinates given
indentNode(...)	Indents a node in the node hierarchy by one level towards the right
isIndentNodeAllowed(...)	Defines whether or not within the node hierarchy indenting of a node is allowed
isOutdentNodeAllowed(...)	Defines whether or not within the node hierarchy outdenting of a node is allowed
outdentNode(...)	Outdents a node in the node hierarchy by one level towards the left
removeLinkChangeListener(...)	Removes an existing listener for link change events
removeNodeChangeListener(...)	Removes an existing listener for node change events
selectedEntitiesIterator()	Retrieves the entities of all objects selected

Methods to Handle Page Layout and Printing

openPageLayoutDialog(...)	Opens the page layout dialog
openPrintDialog()	Opens the print dialog
openPrintPreview()	Opens the print preview

Methods to Handle Properties

addPropertyChangeListener(...)	Adds a listener for change events in the JGantt object
getPropertyEditor()	Retrieves the property editor belonging to this JGantt instance.

Methods to Handle the Export

export(...)	Exports a bitmap by specifying the target and the export format.
export(...)	Exports a bitmap by specifying the target, the export format, the width, the height and the degree of compression.
export(...)	Exports a bitmap by specifying the target, the export format, the resolution and the degree of compression.

Methods to Handle the Gantt Diagram

getLeafNodeFilter()	Retrieves a filter for leaf nodes.
getUserAction(...)	Generates an instance of a user action.
getVertLineGridsEx()	Returns the vertical line grids of the first timescale section.
iterateLayouterGroups()	Returns an iterator object on the node entites that exist in the Gantt graph.
iterateNodeEntitiesInRow (...)	Returns an iterator object on the node entites that exist in a row.
markAll(...)	Handles the marking of all nodes or links in a Gantt diagram
markEntity(...)	Handles the marking of a node or a link

Methods to Handle the JGantt Object

copyPropertiesFrom(...)	Copies all properties from a template JGantt object.
removePropertyChangeListener(...)	Removes an existing event listener for events on changes of the JGantt object.

setDiagramTitlePicture(...)	Picture object to the title line of the diagram
setDocumentBase(...)	Sets the path of the current directory.
synchronizeDiagramTableWidthPercent(...)	Synchronizes several instances of VARCHART JGantt

Methods to Handle the Table

calcOptimizedTableColumnWidth(...)	Calculates the optimum width of a table column

Methods to Handle the Time Scale

addTimeScaleSection(...)	Adds a new time scale section.
getTimeScaleDateFormats(...)	Retrieves user-defined date formats.
getTimeScaleTypes()	Retrieves the list of available time scale types
nextCoarserTimeScaleSectionType(...)	Sets the next coarser time scale section type.
nextCoarserTimeScaleType()	Sets the next coarser time scale type.
nextFinerTimeScaleSectionType(...)	Sets the next finer time scale section type.
nextFinerTimeScaleType()	Sets the next finer time scale type.
removeTimeScaleSection(...)	Removes a time scale section.
setTimeScaleDateFormats(...)	Sets user-defined date formats for timescale labels.

Constructors of the Class

These constructors allow to generate a JGantt object with or without a license string. After you have purchased a license string at NETRONIC Software GmbH, you can validate the VARCHART JGantt application permanently by setting the licence string. Otherwise the software will expire after a limited time.

JGantt

Constructor of JGantt

For internal use only!

Declaration
JGantt (java.lang.Object object, java.lang.String string1, java.lang.String string2)

Parameter	Data Type	Description
object	java.lang.Object	For internal use only!
string1	java.lang.String	For internal use only!
string2	java.lang.String	For internal use only!

JGantt

Constructor of JGantt

This constructor serves to generate a JGantt object not using a license code. The VARCHART JGantt application therefore will expire after three months at maximum.

Declaration
JGantt ()

JGantt

Constructor of JGantt

In case you purchased a license code, this constructor serves to generate a JGantt object that includes the license code, in order to validate your VARCHART JGantt application for an unlimited time. Otherwise the time limit implemented for protection will expire after 3 months at maximum.

Declaration
JGantt (java.lang.String theLicenceString)

Parameter	Data Type	Description
theLicenceString	java.lang.String	License code to be passed

Properties of the Class

ActiveNodeFilter

Property of JGantt

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

This filter lets you select nodes to be visible in the Gantt chart. The nodes may have been created by a NodeEntitySet or by a GroupNodeEntitySet.

Accessing Methods
void setActiveNodeFilter (de.netronic.common.intface.NeIFilter newValue)
de.netronic.common.intface.NeIFilter getActiveNodeFilter ()

AppData

Property of JGantt

Typ	**de.netronic.common.intface.NeIAppData**
Bound	**no**
Vetoable	**no**
Exposure Level	**hidden**

This property handles the application data object of the JGantt. Each JGantt bean receives its own application data object on creation, where the data of the nodes and links are stored.

To enable two different JGantt beans to use the same appData object, this property can replace the appData object in a JGantt bean after it has been created.

Accessing Methods
void setAppData (de.netronic.common.intface.NeIAppData newValue)
de.netronic.common.intface.NeIAppData getAppData ()

Example Code

```
private void fillAppdata ()
   {
   // try to load data as ressource first
   URL testDataURL = JGanttBeispiel.class.getResource(dataFileName);
   NeIAppData theAppData = jGantt1.getAppData();
   if (testDataURL != null)
     {
     theAppData.loadXML(testDataURL.toString());
     }
   else
     {
     // load as file if not successful
     try
       {
       java.io.FileInputStream file = new java.io.FileInputStream (dataFileName);
       file.close();
       theAppData.loadXML(dataFileName);
       }
     catch (Exception ex)
       {
              if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
        {
```

```
            theAppData.loadXML(fc.getSelectedFile().getPath());
                    }
        }
    }
}
```

BuildNumber

Read Only Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the build number of the JGantt object

Accessing Methods
java.lang.String getBuildNumber()

Calendar

Property of JGantt

Typ	**de.netronic.common.intface.NeICalendar**
Bound	**no**
Vetoable	**no**
Exposure Level	**hidden**

This property handles the calendar of the JGantt object. Each JGantt bean receives its own calendar object on creation. To enable two different JGantt objects to use the same calendar, this property can replace the calendar of a JGantt after its creation.

Accessing Methods
void setCalendar (de.netronic.common.intface.NeICalendar newValue)
de.netronic.common.intface.NeICalendar getCalendar ()

CanonicalHierarchyCode

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**intern erzeugt**

When dealing with incomplete hierarchies using **KeepIncompleteHierarchy**, VARCHART JGantt generates in the NodeEntitySet an additional attribute to which the internal canonical hierarchy code is stored. To be able to react to object changes more appropriately, you can specify an existing attribute.

Accessing Methods
void setCanonicalHierarchyCode (java.lang.String newValue)
java.lang.String getCanonicalHierarchyCode ()

Also see KeepIncompleteHierarchy
NodeHierarchyBy

Example Code

```
jGantt1.setNodeHierarchyBy("Input_HCCode");
jGantt1.setCanonicalHierarchyCode("Canonical_HCCode");
jGantt1.setKeepIncompleteHierarchy(true);
```

CollapseFilter

Read Only Property of JGantt

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Filter for collapsed nodes of the node hierarchy or of collapsed nodes. Nodes of the node hierarchy are identified by their hierarchy node, whereas groups are formed by values of their grouping attribute.

The filter will return **true** if the entity belongs to a collapsed node. If the node is expanded, it will retrun **false**.

Accessing Methods
de.netronic.common.intface.NeIFilter getCollapseFilter()

DataDefinitionName

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the name of an XML file that is used by the customizer dialog to store the definitions of the entity sets.

Accessing Methods
void setDataDefinitionName (java.lang.String newValue)
java.lang.String getDataDefinitionName ()

DiagramAnnotationsOnScreen

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not the diagram title is displayed on the screen. If it does not, it will appear on printouts only.

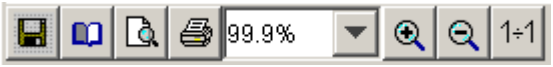
Accessing Methods
void setDiagramAnnotationsOnScreen (boolean newValue)
boolean hasDiagramAnnotationsOnScreen ()

DiagramControlBar

Property of JGantt

Typ	**javax.swing.JComponent**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the control panel for the user interactions. It offers a number of buttons for user interactions concerning file management, printing and the diagram. You can add or delete buttons in the task bar.



Control panel holding basic functions

Accessing Methods
void setDiagramControlBar (javax.swing.JComponent newValue)
javax.swing.JComponent getDiagramControlBar ()

DiagramControlBarVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines, whether the control panel for the ScreenManager functions is visible.

Accessing Methods
void setDiagramControlBarVisible (boolean newValue)
boolean isDiagramControlBarVisible ()

DiagramEmptyRow

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not an empty row is available at the bottom of the diagram. An empty row may be needed to add nodes at the bottom of the diagram.

Accessing Methods
void setDiagramEmptyRow (boolean newValue)
boolean hasDiagramEmptyRow ()

DiagramHistogramHeightPercent

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**25%**

This property defines the portion of the total diagram height that is occupied by the histogram.

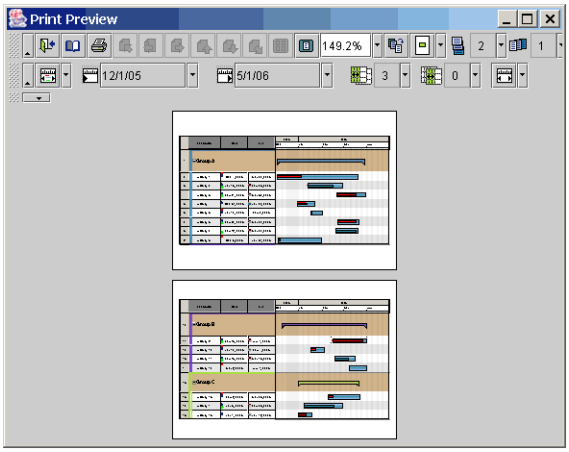
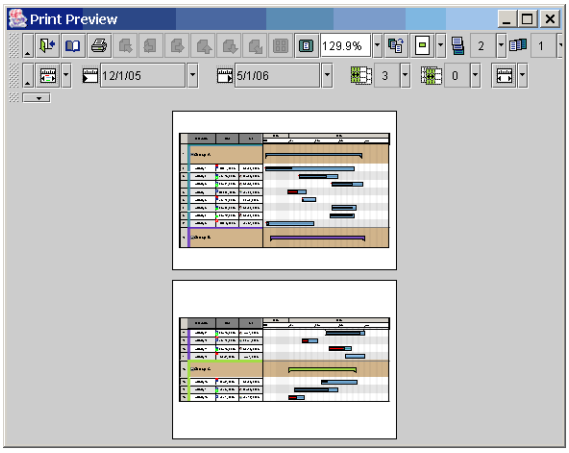
Accessing Methods
void setDiagramHistogramHeightPercent (int newValue)
int getDiagramHistogramHeightPercent ()

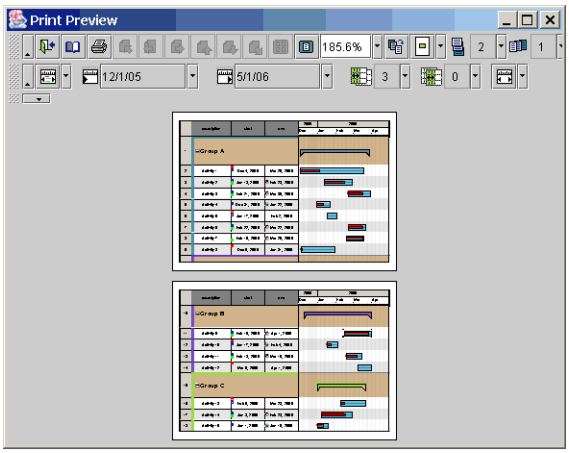
DiagramPageBreakMode

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**PAGE_BREAK_KEEP_ROWS**

This property defines the mode of page breaking for printing, if charts vertically extend over more than a single page. If for the group level specified a page does not offer sufficient space, page breaks are tried with a higher level.

Possible Values	Description
PAGE_BREAK_KEEP_FIRST_LEVEL_GROUPS	The page breaks on group limits of first level. 
PAGE_BREAK_KEEP_GROUPS	Deprecated, see PAGE_BREAK_KEEP_FIRST_LEVEL_GROUPS!
PAGE_BREAK_KEEP_ROWS	The page breaks on row limits. Lines cannot be devided. 
PAGE_BREAK_KEEP_SECOND_LEVEL_GROUPS	The page breaks on group limits of second level.
PAGE_BREAK_KEEP_SUBGROUPS	Deprecated, see PAGE_BREAK_KEEP_-SECOND_LEVEL_GROUPS!
PAGE_BREAK_KEEP_THIRD_LEVEL_GROUPS	The page breaks on group limits of third level.
PAGE_BREAK_NO_RESTRICTION	A page breaks when it is full. This may entail lines to be divided.



Accessing Methods
void setDiagramPageBreakMode (int newValue)
int getDiagramPageBreakMode ()

DiagramRowMargin

Property of JGantt

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**2.0**

This property defines the minimum offset between the row border and a node. The unit is millimeters.

Accessing Methods
void setDiagramRowMargin (double newValue)
double getDiagramRowMargin ()

Also see DiagramRowMinimumHeight

DiagramRowMinimumHeight

Property of JGantt

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**6.0**

This property defines the minimum height of the rows in the Gantt diagram. The unit is millimeters.

Accessing Methods
void setDiagramRowMinimumHeight (double newValue)
double getDiagramRowMinimumHeight ()

Also see DiagramRowMargin

DiagramTableWidthPercent

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the width of the table as percentage of the component width.

Accessing Methods
void setDiagramTableWidthPercent (int newValue)
int getDiagramTableWidthPercent ()

DiagramTitleColorScheme

Property of JGantt

Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the color scheme for the title of the Gantt diagram. The title itself will show the textColor, whereas its background will adopt the mainColor.

Accessing Methods
void setDiagramTitleColorScheme (JGColorScheme newValue)
JGColorScheme getDiagramTitleColorScheme ()

Also see DiagramTitleFont
DiagramTitleOnScreen
DiagramTitleText

DiagramTitleFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the font of the diagram title.

Accessing Methods
void setDiagramTitleFont (java.awt.Font newValue)
java.awt.Font getDiagramTitleFont ()

Also see DiagramTitleColorScheme
DiagramTitleOnScreen
DiagramTitleText

DiagramTitleOnScreen

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not the diagram title is displayed on the screen. If it does not, it will appear on printouts only.

Accessing Methods
void setDiagramTitleOnScreen (boolean newValue)
boolean hasDiagramTitleOnScreen ()

Also see DiagramTitleColorScheme
DiagramTitleFont
DiagramTitleText

DiagramTitleText

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the title of the Gantt diagram.

Accessing Methods
void setDiagramTitleText (java.lang.String newValue)
java.lang.String getDiagramTitleText ()

Also see DiagramTitleColorScheme
DiagramTitleFont
DiagramTitleOnScreen

DiagramZoomFactor

Property of JGantt

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the zoom factor of the Gantt diagram.

Accessing Methods
void setDiagramZoomFactor (double newValue)
double getDiagramZoomFactor ()

DocumentBase

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Path of the current directory.

Accessing Methods
void setDocumentBase (java.lang.String newValue)
java.lang.String getDocumentBase ()

EntityEditorDialog

Read Only Property of JGantt

Typ	**de.netronic.bean.entityeditor.NeEntityEditorDialog**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Dialog to edit entities.

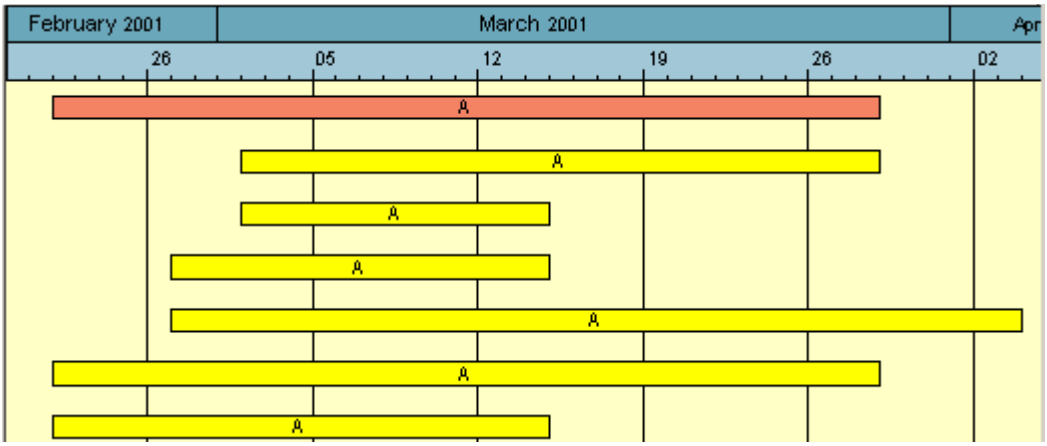
Accessing Methods
de.netronic.bean.entityeditor.NeEntityEditorDialog getEntityEditorDialog()

GanttCalendarGrid

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines, whether or not a calendar grid is displayed in the Gantt graph. The colors of the calendar grid are controlled by the ganttColorScheme assigned. Time spans where the spanID equals 0 are displayed in shadedAlternateColor of the colorScheme, all other time spans are displayed in alternateColor. More variety in colors can be achived by using NeMappedColors.



Calendar grid showing weekly periods

Accessing Methods
void setGanttCalendarGrid (boolean newValue)
boolean hasGanttCalendarGrid ()

GanttColorScheme

Property of JGantt

Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the color scheme of the Gantt graph. The shadedColor is used for the background of the Gantt graph. The main color dyes the date line while the alternateColor and the shadedAlternateColor are used for the calendar grid (ganttCalendarGrid). If you use a dynamic color (NeIDynamicColor) as shadedColor, you can set individual colors to the rows. The reference object for dynamic colors is the entity in the line; if there are several entities, the reference object is the group entity of the group (also see NeIDynamicColor and JGDynamicRowColor). The lineColor applies to the lines of the line grid (ganttGrid).

Accessing Methods
void setGanttColorScheme (JGColorScheme newValue)
JGColorScheme getGanttColorScheme ()

GanttDateLine

Property of JGantt

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property draws or retrieves a date line in the Gantt graph at the date passed. You can disable the line by setting the date to 0. The date line is expressed as milliseconds from 1.1.1970 onward.

Accessing Methods
void setGanttDateLine (long newValue)
long getGanttDateLine ()

GanttDisplayProfile

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property handles the name of a profile that is used to display the calendar grid in the Gantt graph. The calendar grid is displayed if the property GanttCalendarGrid was set to **true**.
If you do not set a profile name via the **set** method, the current node profile will be used as a default to display the calendar grid. If no Gantt display profile has been set when retrieving the name, an empty string will be returned.
The colors of the calendar grid are controlled by the GanttColorScheme property. TimeSpans of the spanID 0 will adopt the ShadedAlternateColor, all other timeSpans will be displayed in the AlternateColor. Varying colors you can set by using NeMappedColor.

Accessing Methods
void setGanttDisplayProfile (java.lang.String newValue)
java.lang.String getGanttDisplayProfile ()

Also see GanttGlassProfile

GanttFixedTopRow

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Fixes the first row at the top of the gantt chart. This row keeps its top position, even when vertically scrolling the diagram.

Accessing Methods
void setGanttFixedTopRow (boolean newValue)
boolean getGanttFixedTopRow ()

GanttGlassProfile

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property handles the name of a profilethat is used to additionally mark certain time spans in the Gantt graph.

The colors of the calendar grid are controlled by the GanttColorScheme property. TimeSpans of the spanID 0 will adopt the AlternateTextColor, all other timeSpans will be displayed in the ShadedAlternateTextColor. Varying colors you can set by using NeMappedColor.
Normally you will use colors with alpha value, thereby you can still see the calendar grid and the nodes and links.

Accessing Methods
void setGanttGlassProfile (java.lang.String newValue)
java.lang.String getGanttGlassProfile ()

Also see GanttDisplayProfile

GanttGraph

Read Only Property of JGantt

Typ	**JGIGanttGraph**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves an instance of the Gantt graph, that is automatically created when a JGantt object is generated. The Gantt graph is the non-table part of a Gantt chart situated below the time scale.

Accessing Methods
JGIGanttGraph getGanttGraph()

GanttGrid

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**VERT_GRID_NONE**

This property handles the line grid of the Gantt graph. If there are several time scale sections, then this property only sets the grid of the first section.

Possible Values	Description
VERT_GRID_AUTO	The distance of the grid lines depends on the type of time scale.
VERT_GRID_BY_MINUTES	The grid lines are displayed at periods of a minute.
VERT_GRID_BY_SECONDS	The grid lines are displayed at periods of a second.
VERT_GRID_DAILY	The grid lines are displayed at periods of a day.
VERT_GRID_HOURLY	The grid lines are displayed at periods of an hour.
VERT_GRID_MONTHLY	The grid lines are displayed at periods of a month.
VERT_GRID_NONE	No grid is displayed.
VERT_GRID_QUARTERLY	The grid lines are displayed at periods of a quarter.
VERT_GRID_WEEKLY	The grid lines are displayed at periods of a week.
VERT_GRID_YEARLY	The grid lines are displayed at periods of a year.

Accessing Methods
void setGanttGrid (int newValue)
int getGanttGrid ()

Also see GanttSectionGrids

Example Code

```
// Line grid, periods weekly
jGantt1.setGanttGrid(de.netronic.jgantt.JGantt.GANTT_GRID_WEEKLY);
```

GanttGridLineStyle

Property of JGantt

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color and pattern of the lines of the line grid in the gantt chart. To define the properties of the line style object please see class NeLineStyle.

Accessing Methods
void setGanttGridLineStyle (java.awt.Color newValue)
java.awt.Color getGanttGridLineStyle ()

Also see GanttSectionGridLineStyles

GanttSectionGridLineStyles

Property of JGantt

Typ	**java.awt.Color[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the Appearance of the lines of the gantt chart.line grids of all time scale sections in the Gantt graph.

Accessing Methods
void setGanttSectionGridLineStyles (integer index, java.awt.Color newValues)
void setGanttSectionGridLineStyles (java.awt.Color[] newValue)
java.awt.Color getGanttSectionGridLineStyles (integer index)
java.awt.Color[] getGanttSectionGridLineStyles ()

Also see GanttGridLineStyle

GanttSectionGrids

Property of JGantt

Typ	**int[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Values	**VERT_GRID_NONE**

This property handles the line grids of all time scale sections in the Gantt graph.

Possible Values	Description
VERT_GRID_AUTO	The distance of the grid lines depends on the type of time scale.
VERT_GRID_BY_MINUTES	The grid lines are displayed at periods of a minute.
VERT_GRID_BY_SECONDS	The grid lines are displayed at periods of a second.
VERT_GRID_DAILY	The grid lines are displayed at periods of a day.
VERT_GRID_HOURLY	The grid lines are displayed at periods of an hour.
VERT_GRID_MONTHLY	The grid lines are displayed at periods of a month.
VERT_GRID_NONE	No grid is displayed.
VERT_GRID_QUARTERLY	The grid lines are displayed at periods of a quarter.
VERT_GRID_WEEKLY	The grid lines are displayed at periods of a week.
VERT_GRID_YEARLY	The grid lines are displayed at periods of a year.

Accessing Methods
void setGanttSectionGrids (integer index, int newValues)
void setGanttSectionGrids (int[] newValue)
int getGanttSectionGrids (integer index)
int[] getGanttSectionGrids ()

Also see GanttGrid

Example Code

```
// Line grids, periods weekly at section 0
jGantt1.setGanttSectionGrids(0,
```

```
de.netronic.jgantt.JGantt.GANTT_GRID_WEEKLY);
```

GroupBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the attribute that the nodes are grouped by.

Accessing Methods
void setGroupBy (java.lang.String newValue)
java.lang.String getGroupBy ()

GroupHierarchyBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the name of the node attribute that holds the hierarchy code of that entity.

Accessing Methods
void setGroupHierarchyBy (java.lang.String newValue)
java.lang.String getGroupHierarchyBy ()

GroupLayout

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**GROUP_ROWLAYOUT_SINGLE_OPTIMIZED**

This property defines the way in which groups are layouted. The setting made by this attribute will affect all groups of the chart. Settings of single groups can be made by the GanttGraph property "RowLayout".

Possible Values	Description
GROUP_ROWLAYOUT_MULTIPLE	Each activity of a group is placed in a row of its own.
GROUP_ROWLAYOUT_SINGLE	All activities are placed in the same row, allowing them to overlap.
GROUP_ROWLAYOUT_SINGLE_OPTIMIZED	All activities are placed in the same row. If nodes overlap, they are shifted vertically within the same row, making the row wider.

Accessing Methods
void setGroupLayout (int newValue)
int getGroupLayout ()

Also see GroupsInitiallyCollapsed

GroupMode

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**GROUP_BY_HIERARCHY**

This property defines the grouping mode.

Possible Values	Description
GROUP_BY_HIERARCHY	Grouping will be performed via the hierarchy and allows for more than two grouping levels.
GROUP_BY_VALUE	Grouping is performed via the value of an attribute.

Accessing Methods
void setGroupMode (int newValue)
int getGroupMode ()

GroupNodeAnnotations

Property of JGantt

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property via attribute names defines an array of attributes of a group node entity that are designed to hold annotations, or retrieve the names previously set. 9 possible annotation attributes exist, so the index and the size of the array are limited to this number. The indexes occupy certain postions on a node, that depend on the node design assigned.

Accessing Methods
void setGroupNodeAnnotations (integer index, java.lang.String newValues)
void setGroupNodeAnnotations (java.lang.String[] newValue)
java.lang.String getGroupNodeAnnotations (integer index)
java.lang.String[] getGroupNodeAnnotations ()

Also see NodeAnnotations

GroupNodeColorScheme

Property of JGantt

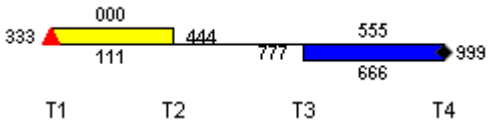
Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the color scheme of the group nodes. The way the colors of the scheme are applied depends on the node design used:

2Rects:

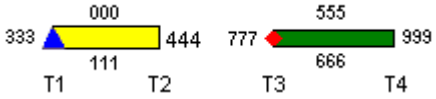
The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor
The bar between T3 and T4 receives shadedAlternateColor

2RectsConnected:



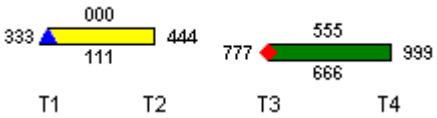
The bar between T3 and T4 receives mainColor.
The bar between T1 and T2 receives shadedColor
The line between T2 and T3 receives lineColor

2 RectsInterlocked:



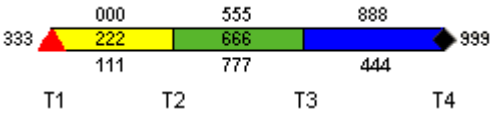
The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor
The bar between T3 and T4 receives shadedAlternateColor

2RectsShifted:



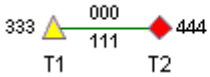
The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor
The bar between T3 and T4 receives shadedAlternateColor

3Rects:



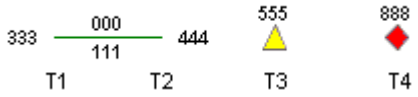
The bar between T3 and T4 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T1 receives alternateColor
The bar between T2 and T3 receives shadedAlternateColor
The symbol on T4 receives lineColor

Line:



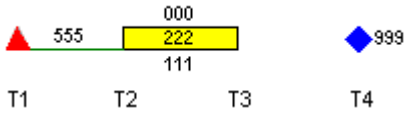
The symbol on T1 receives shadedColor
The symbol on T2 receives alternateColor
The line between T1 and T2 receives shadedAlternateColor

Line2Symbols:



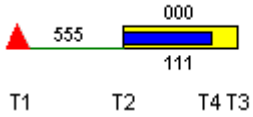
The line between T1 and T2 receives shadedAlternateColor
The symbol on T3 receives shadedColor
The symbol on T4 receives alternateColor

LineBottomRectSymbol:



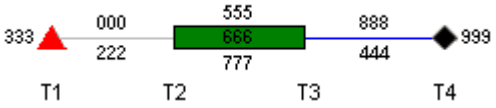
The symbol on T4 receives mainColor
The bar between T2 and T3 receives shadedColor
The Symbol on T1 receives alternateColor
The line between T1 and T2 receives shadedAlternateColor

LineCompletionRect:



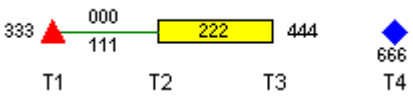
The narrow bar between T2 and T4 receives mainColor
The wide bar between T2 and T3 receives shadedColor
The symbol on T1 receives alternateColor
The line between T1 and T2 receives shadedAlternateColor

LineRectLine:



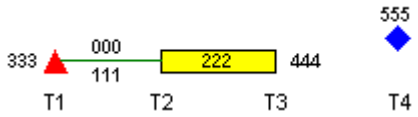
The line between T3 and T4 receives mainColor
The line between T1 and T2 receives shadedColor
The symbol on T1 receives alternateColor
The bar between T2 and T3 receives shadedAlternateColor
The symbol on T4 receives lineColor

LineRectSymbol:



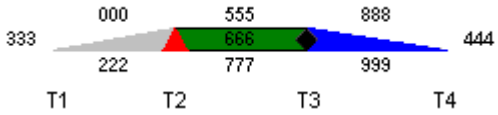
The symbol on T4 receives mainColor
The bar between T2 and T3 receives shadedColor
The symbol on T1 receives alternateColor
The line between T1 and T2 receives shadedAlternateColor

LineRectSymbolAbove:



As Line RectSymbol

Ramps:



The descending ramp between T3and T4 receives mainColor
The ascending ramp between T1 and T2 receives shadedColor
The symbol on T2 receives alternateColor
The bar between T2 and T3 receives shadedAlternateColor
The symbol on T3 receives lineColor

Rect:

The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T2 receives alternateColor

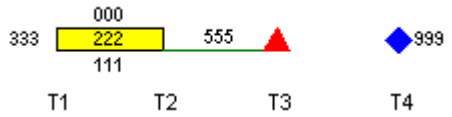
Rect2Symbols:

The bar between T1 and T2 receives shadedColor
The symbol on T2 receives alternateColor
The symbol on T3 receives shadedAlternateColor

Rect2Symbols above:

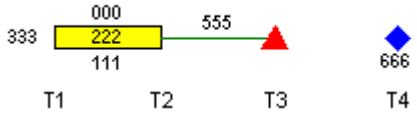
As Rect2Symbols

RectLineBottomSymbol:



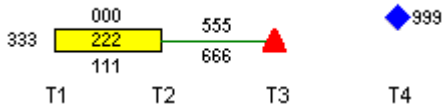
The symbol on T4 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor
The line between T2 and T3 receives shadedAlternateColor

RectLineSymbol:



As RectLineBottomSymbol

RectLineSymbolAbove:



As RectLineBottomSymbol

Summary:

The bar between T1 and T2 receives mainColor

Symbol

The symbol on T1 receives shadedColor

Accessing Methods
void setGroupNodeColorScheme (JGColorScheme newValue)
JGColorScheme getGroupNodeColorScheme ()

Also see NodeColorScheme
TableColorScheme
GanttColorScheme
TimeScaleColorScheme

GroupNodeDateFormat

Property of JGantt

Typ	**java.text.DateFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the format of the dates in annotations of group nodes. If no format is set here, the dates will be displayed in a format correspondig to **DateFormat.MEDIUM**.

Accessing Methods
void setGroupNodeDateFormat (java.text.DateFormat newValue)
java.text.DateFormat getGroupNodeDateFormat ()

Also see NodeDateFormat
TableDateFormat

GroupNodeDates

Property of JGantt

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines an array of attribute names of a group node entity that are designed to hold the dates of the group node. 4 possible date attributes exist, so the index and the size of the array are limited by this number.

Accessing Methods
void setGroupNodeDates (integer index, java.lang.String newValues)
void setGroupNodeDates (java.lang.String[] newValue)
java.lang.String getGroupNodeDates (integer index)
java.lang.String[] getGroupNodeDates ()

Also see NodeDates

GroupNodeDesign

Property of JGantt

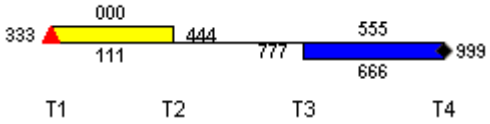
Typ	**JGNodeDesign**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines a node design object for the group nodes. Design objects are reponsible for what nodes look like. The possible values to be set you can retrieve by the method JGNodeDesign.getNames():

2Rects:

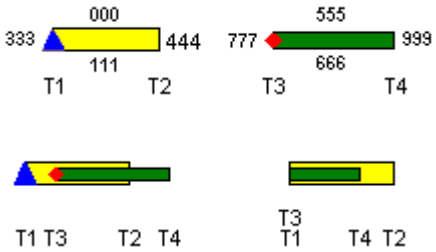
The space between the dates T1-T2 is filled by a broad bar of the shadedColor. Four different annotations can be assigned by the indexes 0, 1, 2 and 3. The space between the dates T3-T4 is filled by a broad bar of the shadedAlternateColor. Four different annotations can be assigned by the indexes 5, 6, 7 and 9. An existing symbol is displayed in mainColor on T1, another existing symbol is displayed on T3 in alternateColor.

2RectsConnected:



The space between the dates T1-T2 is filled by a narrow bar of the shadedColor and can be annotated by the indexes 0 and 1. The space between the dates T2-T3 is filled by a line of the lineColor and can be annotated by the indexes 4 and 7. The space between the dates T3-T4 is filled by a bar of the mainColor and can be annotated by the indexes 5 and 6.On T1 an existing symbol is displayed in alternateColor. It can be annotated on the left via index 3. On T4 an existing symbol is displayed in lineColor. It can be annotated below via index 9.

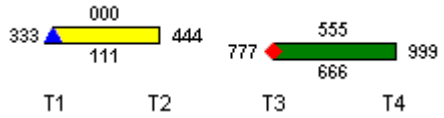
2 RectsInterlocked:



As node design "2Rects", except that the bars appear interlocked as soon as the dates allow for it. The narrow bar can act as an "progress" bar, representing a

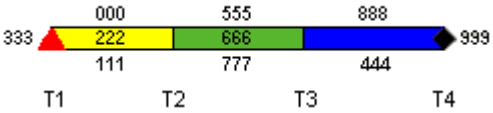
defined percentage of length of the broad bar, thus indicating the degree of completion of an activity.

2RectsShifted:



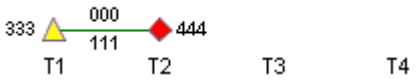
The space between the dates T1-T2 is filled by a narrow bar of the shadedColor. Four different annotations can be assigned by the indexes 0, 1, 3 and 4. The space between the dates T3-T4 is filled by a narrow bar of the shadedAlternateColor. Its position is lower than the one of the first bar. Four different annotations can be assigned by the indexes 5, 6, 7 and 9. An existing symbol is displayed in mainColor on T1, another existing symbol is displayed on T3 in alternateColor.

3Rects:



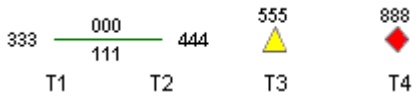
The space between the dates T1-T2 is filled by a bar of the shadedColor and can be annotated by the indexes 0 through 2. The space between the dates T2-T3 is filled by a bar of the shadedAlter-nateColor and can be annotated by the indexes 5 through 7. The space between the dates T3-T4 is filled by a bar of the mainColor and can be annotated by the indexes 8 and 4. On T1 an existing symbol is displayed in alternateColor. It can be annotated on the left via index 3. On T4 an existing symbol is displayed in lineColor. It can be annotated below via index 9.

Line:



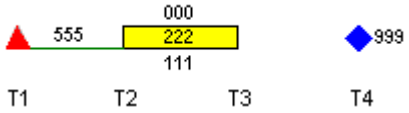
The space between the dates T1-T2 is filled by a line of the shadedAlternateColor. Two annotations can be assigned by the indexes 0 and 1. On T1 and T2 existing symbols are displayed in shadedColor and alternateColor, respectively. They can be assigned annotations via the indexes 3 and 4.

Line2Symbols:



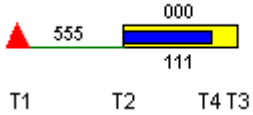
The space between the dates T1-T2 is filled by a line of the shadedAlternateColor. Four different annotations can be assigned by the indexes 0, 1, 3 and 4. On T3 and T4 existing symbols are displayed in shadedColor and alternateColor, respectively. They can be annotated via the indexes 5 and 8.

LineBottomRectSymbol:



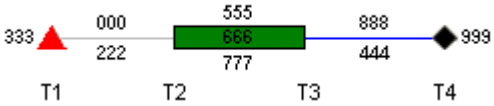
The space between the dates T1-T2 is filled by a subscript line of the shadedAlternateColor and can be annotated by the index 5. On T1 an existing symbol is displayed in alternateColor. The space between the dates T2-T3 is filled by a bar of the shadedColor and can be annotated by the indexes 0 through 2. On T4 an existing symbol is displayed in mainColor. It can be annotated on the right via index 9.

LineCompletionRect:



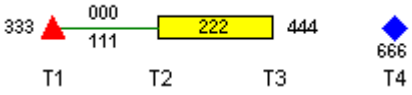
The space between the dates T1-T2 is filled by a subscript line of the shadedAlternateColor and can be annotated by the index 5. On T1 an existing symbol is displayed in alternateColor. The space between the dates T2-T3 is filled by a broad bar of the shadedColor and can be annotated by the indexes 0 through 2. The space between T2-T4 is filled with a narrow bar, which may act as a progress bar that indicates the degree of completion of an activity.

LineRectLine:



The space between the dates T1-T2 is filled by a line of the shadedColor and can be annotated by the indexes 0 and 2. The space between the dates T2-T3 is filled by a bar of the shadedAlter-nateColor and can be annotated by the indexes 5 through 7. The space between the dates T3-T4 is filled by a line of the mainColor and can be annotated by the indexes 8 and 4. On T1 an existing symbol is displayed in alternateColor. It can be annotated on the left via index 3. On T4 an existing symbol is displayed in lineColor. It can be annotated below via index 9.

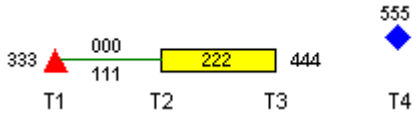
LineRectSymbol:



The space between the dates T1-T2 is filled by a line of the shadedAlternateColor and can be annotated by the indexes 0 and 1. On T1 an existing symbol is displayed in alternateColor. It can be annotated on the left via index 3. The space between the dates T2-T3 is filled by a bar of the shadedColor and can be

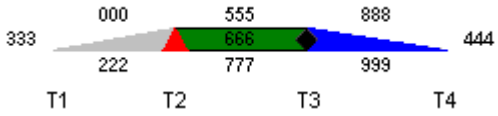
annotated by the indexes 2 and 4. On T4 an existing symbol is displayed in mainColor. It can be annotated below via index 6.

LineRectSymbolAbove:



As node design "LineRectSymbol", except that the symbol on T4 appears superscript and its annotation is placed above via index 5.

Ramps:



The space between the dates T1-T2 is filled by an ascending ramp of the shadedColor and can be annotated by the indexes 0, 2 and 3. The space between the dates T2-T3 is filled by a bar of the shadedAlternateColor and can be annotated by the indexes 5 through 7. The space between the dates T3-T4 is filled by a descending ramp of the mainColor and can be annotated by the indexes 8, 9 and 4. On T2 an existing symbol is displayed in alternateColor. On T3 an existing symbol is displayed in lineColor.

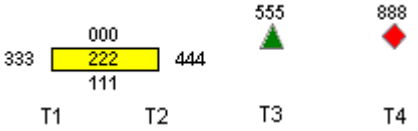
Rect:

The space between the dates T1-T2 is filled by a broad bar of the shadedColor. Five different annotations can be assigned by the indexes 0 through 4. Existing symbols will be displayed on T1 and T2 in the mainColor and in the alternateColor, respectively. The spaces between the dates T2-T3-T4 remain empty.

Rect2Symbols:

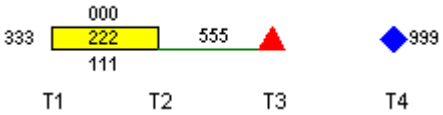
As node design "Rect", except that on T3 and T4 existing symbols are displayed in the ShadedAlternateColor and alternateColor, respectively. The symbols can be annotated via the indexes 5 and 8.

Rect2Symbols above:



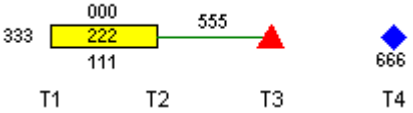
As node design "Recht2Symbols", except that the symbols and their annotations appear to be superscript.

RectLineBottomSymbol:



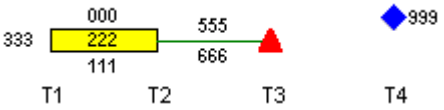
As design "RectLineSymbolAbove", except that the line is subscript and can only be annotated above via index 5.

RectLineSymbol:



The space between the dates T1-T2 is filled by a broad bar of the shadedColor and can be annotated by the indexes 0 through 3. The space between the dates T2-T3 is filled by a line of the shadedAlternateColor and can be annotated by the index 5. On T3 an existing symbol is displayed in alternateColor. On T4 an existing symbol is displayed in mainColor. It can be annotated below via index 6.

RectLineSymbolAbove:



As node design "RectLineSymbol", except that the line between T2-T3 can be annotated in addition by index 6 and that the symbol on T4 is superscript which can be annotated on its right by index 9.

Summary:

The space between the dates T1-T2 is filled by a summary bar of the mainColor and can be annotated by the indexes 0 and 2.

Symbol

An existing symbol of the shadedColor is displayed on T1. You can place an annotation right of the symbol by index 4.

Accessing Methods
void setGroupNodeDesign (JGNodeDesign newValue)
JGNodeDesign getGroupNodeDesign ()

Also see NodeDesign

GroupNodeFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the font of group node annotations.

Accessing Methods
void setGroupNodeFont (java.awt.Font newValue)
java.awt.Font getGroupNodeFont ()

Also see NodeFont
TableFont
TableRowTitleFont
TableGroupFont
TableColumnTitleFont
TimeScaleFont

GroupNodeProfile

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the default profile for group nodes. The default profile will be used in group nodes that were not assigned an individual profile.

Accessing Methods
void setGroupNodeProfile (java.lang.String newValue)
java.lang.String getGroupNodeProfile ()

Also see GroupNodeProfileBy

NodeProfile

GroupNodeProfileBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the group node attribute that holds the name of a calendar profile to be used individually for the groupNode passed. If the specified profile cannot be found, the group node default profile will be used.

Accessing Methods
void setGroupNodeProfileBy (java.lang.String newValue)
java.lang.String getGroupNodeProfileBy ()

Also see GroupNodeProfile
NodeProfileBy

GroupNodePropertiesEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not properties designed for group nodes are to apply. If this property is set to **false**, the group nodes will adopt the look of simple nodes.

Accessing Methods
void setGroupNodePropertiesEnabled (boolean newValue)
boolean hasGroupNodePropertiesEnabled ()

GroupNodeSetName

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the entity set that holds the data of the group nodes.

Accessing Methods
void setGroupNodeSetName (java.lang.String newValue)
java.lang.String getGroupNodeSetName ()

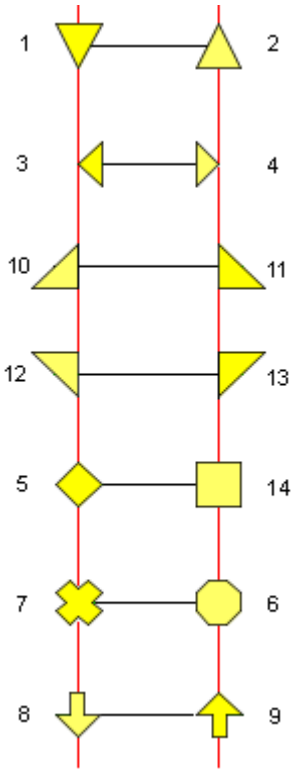
Also see NodeSetName
LinkSetName

GroupNodeSymbols

Property of JGantt

Typ	**de.netronic.common.intface.NelLabel[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines an array of symbols that, apart from the design, will be used to visualize the node. 2 possible symbol attributes exist, so the index, that is, the size of the array is limited to this number.
As values, the figures 1...14 are valid. They represent the below shapes:



Accessing Methods
void setGroupNodeSymbols (integer index, de.netronic.common.intface.NeILabel newValues)
void setGroupNodeSymbols (de.netronic.common.intface.NeILabel[] newValue)
de.netronic.common.intface.NeILabel getGroupNodeSymbols (integer index)
de.netronic.common.intface.NeILabel[] getGroupNodeSymbols ()

Also see NodeSymbols

GroupNodeZeroLengthSymbol

Property of JGantt

Typ	**de.netronic.common.intface.NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the symbol that is displayed in the place of group nodes that have no extent, i.e. where the start date equals the end date. If you do not set this property, the zero length node will be displayed as a short vertical line.

Accessing Methods
void setGroupNodeZeroLengthSymbol (de.netronic.common.intface.NeILabel newValue)
de.netronic.common.intface.NeILabel getGroupNodeZeroLengthSymbol ()

Also see NodeZeroLengthSymbol

GroupNodeZeroLengthVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not bars in group nodes, that have no length, i.e. where the start date equals the end date, will be displayed on the screen.

Accessing Methods
void setGroupNodeZeroLengthVisible (boolean newValue)
boolean hasGroupNodeZeroLengthVisible ()

Also see NodeZeroLengthVisible

GroupSetFilter

Read Only Property of JGantt

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves a filter for entities present in the group entity set. The filter will return **true** if the entity is contained in the group entity set, otherwise it will retrun **false**.

Accessing Methods
de.netronic.common.intface.NeIFilter getGroupSetFilter()

GroupsInitiallyCollapsed

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not group nodes will appear collapsed when the program is started or when the property GroupLayout is set. **True**: the group nodes appear collapsed, **false**: the group nodes appear expanded.

Accessing Methods
void setGroupsInitiallyCollapsed (boolean newValue)
boolean isGroupsInitiallyCollapsed ()

Also see GroupLayout

GroupsSortedBy

Property of JGantt

Typ	**java.util.Comparator**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

Groups at the top level and sub-groups on lower levels can be sorted. The sequence of the groups or sub-groups can be defined by a comparator object. At runtime, two objects of the type NeILayoutGroup need to be passed to the comparator for comparison.
If the sorting is intended to be dynamical, that is, if on the modification of data the goups should be sorted anew, the comparator passed in addition needs to implement the interface NeIGroupComparator. The class de.netronic.bean.layouter.NeGroupComparator provides a pre-defined default implementation of a comparator.
This property defines the sorting of all groups in the chart. It overwrites the values set by the JGantt property "GroupsSortedBy", which sets the sorting for individual groups of the Gantt chart. Setting the NeILayouterGroup property "GroupComparator" property again will overwrite the settings of the JGantt property "GroupsSortedBy".

Accessing Methods
void setGroupsSortedBy (java.util.Comparator newValue)
java.util.Comparator getGroupsSortedBy ()

Example Code

```
jGantt1.setGroupsSortedBy(new NeGroupComparator("PresentStart",false));
```

GroupValueUpdater

Read Only Property of JGantt

Typ	**NeIGroupValueUpdater**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the group value updater object. You can assigne it to a variable to create your own object of this kind.

Accessing Methods
NeIGroupValueUpdater getGroupValueUpdater()

Histogram

Read Only Property of JGantt

Typ	**JGIHistogram**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the histogram object, that is automatically created when a JGantt object is generated.

Accessing Methods
JGIHistogram getHistogram()

HistogramVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

The histogram can be visible or invisible in the diagram. If this property is set to **true**, the histogram will be visble, if set to **false**, it will be invisible.

Accessing Methods
void setHistogramVisible (boolean newValue)
boolean isHistogramVisible ()

Id

Read Only Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the identification that is automatically generated when a JGantt object comes into existence.

Accessing Methods
java.lang.String getId()

InteractionAutoScrollEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not the diagram automatically scrolls when the mouse reaches the boundary of the Gantt graph during an interaction.

Accessing Methods
void setInteractionAutoScrollEnabled (boolean newValue)
boolean isInteractionAutoScrollEnabled ()

InteractionDataEditingEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property allows to interactively edit data, if the data-editing license is available. If the license is available, this property will return **true**; if it is not, the property will return **false**.

Accessing Methods
void setInteractionDataEditingEnabled (boolean newValue)
boolean isInteractionDataEditingEnabled ()

Example Code

```
jGantt1.setInteractionDataEditingEnabled(true);
```

InteractionDragEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not the JGantt object acts as a drag source during a drag&drop action.

Accessing Methods
void setInteractionDragEnabled (boolean newValue)
boolean isInteractionDragEnabled ()

Also see InteractionDropEnabled
InteractionMoveMultipleNodesEnabled

InteractionDropEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not the JGantt object acts as a drop target during a drag&drop action.

Accessing Methods
void setInteractionDropEnabled (boolean newValue)
boolean isInteractionDropEnabled ()

Also see InteractionDragEnabled
InteractionMoveMultipleNodesEnabled

InteractionEditLinksEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not a dialog can be invoked by the pop up menu of the Gantt graph that lets the user modify link data.
This property can only be set if the property **InteractionDataEditingEnabled** was set to **true**.

Accessing Methods
void setInteractionEditLinksEnabled (boolean newValue)
boolean isInteractionEditLinksEnabled ()

Also see InteractionDataEditingEnabled
InteractionEditNewLinkEnabled

InteractionEditNewLinkEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines whether or not a dialog appears on the interactive generation of a link to modify link data.

Accessing Methods
void setInteractionEditNewLinkEnabled (boolean newValue)
boolean isInteractionEditNewLinkEnabled ()

Also see InteractionEditNewNodeEnabled

InteractionEditNewNodeEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines whether or not a dialog is to appear after a node was interactively created, to edit the data of the new node.

Accessing Methods
void setInteractionEditNewNodeEnabled (boolean newValue)
boolean isInteractionEditNewNodeEnabled ()

Also see InteractionEditNewLinkEnabled

InteractionEditNodesEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not a dialog can be invoked via the pop up menu of the Gantt graph that lets the user modify node data.
Diese Eigenschaft kann nur eingeschaltet werden, wenn die Eigenschaft InteractionDataEditingEnabled **true** ist.

Accessing Methods
void setInteractionEditNodesEnabled (boolean newValue)
boolean isInteractionEditNodesEnabled ()

Also see InteractionDataEditingEnabled
InteractionEditNewNodeEnabled

InteractionGroupNodeMoveMode

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_MOVE_MODE_NODES_BARS**

This property defines, whether or not group nodes or parts of group nodes can be moved.

Possible Values	Description
INTERACTION_MOVE_MODE_DISABLED	Nodes can be moved neither as a whole nor as parts.
INTERACTION_MOVE_MODE_NODES	Nodes can be moved as a whole only.
INTERACTION_MOVE_MODE_NODES_HORIZONTALLY	Only complete nodes can be moved, and they can be moved horizontally only.
INTERACTION_MOVE_MODE_NODES_VERTICALLY	Only complete nodes can be moved, and they can be moved vertically only.
INTERACTION_MOVE_MODE_NODES_BARS	Both complete nodes and parts of nodes can be moved.
INTERACTION_MOVE_MODE_NODES_BARS_-HORIZONTALLY	Both complete nodes and parts of nodes can be moved horizontally only.

Accessing Methods
void setInteractionGroupNodeMoveMode (int newValue)
int getInteractionGroupNodeMoveMode ()

Also see InteractionNodeMoveMode

InteractionGroupNodeResizeMode

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_RESIZE_MODE_LEFT_RIGHT**

This property defines, whether or not group nodes or parts of group nodes can be resized.

Possible Values	Description
INTERACTION_RESIZE_MODE_DISABLED	No resizing for parts of nodes and for complete nodes.
INTERACTION_RESIZE_MODE_LEFT	Resizing enabled on left end only.
INTERACTION_RESIZE_MODE_LEFT_RIGHT	Resizing enabled on both ends.
INTERACTION_RESIZE_MODE_RIGHT	Resizing enabled on right end only.

Accessing Methods
void setInteractionGroupNodeResizeMode (int newValue)
int getInteractionGroupNodeResizeMode ()

Also see InteractionNodeResizeMode

InteractionInfoWindowEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not an information window is displayed when an interaction is performed, for example when moving a node, indicating the current position.

Accessing Methods
void setInteractionInfoWindowEnabled (boolean newValue)
boolean isInteractionInfoWindowEnabled ()

InteractionLinksAtLeafNodesOnly

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not links can be drawn interactively between leaf nodes only. Leaf nodes exist in hierarchies only, they are postioned on the bottom level.

Accessing Methods
void setInteractionLinksAtLeafNodesOnly (boolean newValue)
boolean isInteractionLinksAtLeafNodesOnly ()

InteractionLinkSelectionRange

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**5**

Since links usually are objects that have the shape of a line, visual skills to some degree are required to select a link. To facilitate the selection of links, clicking in an area near the link also allows to mark it. This property lets you define the width of a selection area (one-sided) within which a mouse click should select a link. Unit: pixels.

Accessing Methods
void setInteractionLinkSelectionRange (int newValue)
int getInteractionLinkSelectionRange ()

InteractionMode

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_SELECT_NODES**

This property defines an interaction mode for the JGantt object.

Possible Values	Description
INTERACTION_CREATE_NODE	Nodes and links can be created.
INTERACTION_DRAGDROP_NODE	For internal use only!
INTERACTION_SELECT_NODES	Nodes and links can be selected.

Accessing Methods
void setInteractionMode (int newValue)
int getInteractionMode ()

InteractionMoveHorizontalWithChildren

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**INTERACTION_MOVE_WITH_CHILDREN_DISABLED**

When moving a non-leaf node in a hierarchy (GroupBy Hierarchy), this method lets you also move its child nodes. The child nodes may be moved
a) never
b) only when the node to be moved was collapsed
c) always.
The child nodes are moved only if the movement is a horizontal one.

Possible Values	Description
INTERACTION_MOVE_WITH_CHILDREN_ALWAYS	When moving a non-leave node horizontally its children will be moved also.
INTERACTION_MOVE_WITH_CHILDREN_DISABLED	When moving a non-leave node horizontally its children will not be moved.
INTERACTION_MOVE_WITH_CHILDREN_WHEN_COLLAPSED	When moving a non-leave node horizontally its children will only be moved if the non-leave node is collapsed.

Accessing Methods
void setInteractionMoveHorizontalWithChildren (int newValue)
int getInteractionMoveHorizontalWithChildren ()

Example Code

```
jGantt1.setInteractionMoveHorizontalWithChildren
  (JGantt.INTERACTION_MOVE_WITH_CHILDREN_ALWAYS);
```

InteractionMoveMultipleNodesEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property lets you move or copy several selected nodes in one go. If several nodes are moved/copied vertically in a grouped chart, all nodes will be moved/copied to the group where the mouse pointer is staying when the mouse button is released.

Accessing Methods
void setInteractionMoveMultipleNodesEnabled (boolean newValue)
boolean isInteractionMoveMultipleNodesEnabled ()

Also see InteractionDragEnabled
InteractionDropEnabled

Example Code

```
jGantt1.setInteractionMoveMultipleNodesEnabled(true);
```

InteractionNodeMoveMode

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**INTERACTION_MOVE_MODE_NODES_BARS**

This property defines, whether or not nodes or parts of nodes can be moved.

Possible Values	Description
INTERACTION_MOVE_MODE_DISABLED	Nodes can be moved neither as a whole nor as parts.
INTERACTION_MOVE_MODE_NODES	Nodes can be moved as a whole only.
INTERACTION_MOVE_MODE_NODES_HORIZONTALLY	Only complete nodes can be moved, and they can be moved horizontally only.
INTERACTION_MOVE_MODE_NODES_VERTICALLY	Only complete nodes can be moved, and they can be moved vertically only.
INTERACTION_MOVE_MODE_NODES_BARS	Both complete nodes and parts of nodes can be moved.
INTERACTION_MOVE_MODE_NODES_BARS_-HORIZONTALLY	Both complete nodes and parts of nodes can be moved horizontally only.

Accessing Methods
void setInteractionNodeMoveMode (int newValue)
int getInteractionNodeMoveMode ()

InteractionNodeResizeMode

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_RESIZE_MODE_LEFT_RIGHT**

This property defines, whether or not nodes or parts of group can be resized.

Possible Values	Description
INTERACTION_RESIZE_MODE_DISABLED	No resizing for parts of nodes and for complete nodes.
INTERACTION_RESIZE_MODE_LEFT	Resizing enabled on left end only.
INTERACTION_RESIZE_MODE_LEFT_RIGHT	Resizing enabled on both ends.
INTERACTION_RESIZE_MODE_RIGHT	Resizing enabled on right end only.

Accessing Methods
void setInteractionNodeResizeMode (int newValue)
int getInteractionNodeResizeMode ()

Also see InteractionGroupNodeResizeMode

InteractionPopupEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not the menu that appears on pressing the right mouse button is enabled to pop up. This property can be set only if the property **InteractionDataEditingEnabled** was set to **true**.

Accessing Methods
void setInteractionPopupEnabled (boolean newValue)
boolean isInteractionPopupEnabled ()

Also see InteractionDataEditingEnabled

InteractionResizeMultipleNodesEnabled

Property of JGantt

Typ	**boolean**
Bound	**yes**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property lets you re-size several selected nodes in one go.

Accessing Methods
void setInteractionResizeMultipleNodesEnabled (boolean newValue)
boolean isInteractionResizeMultipleNodesEnabled ()

Example Code

```
jGantt1.setInteractionResizeMultipleNodesEnabled(true);
```

InteractionSelectionSynchronized

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not the selection of nodes in the table and the Gantt graph will take place snychronously. **True**: the selection takes place synchronously, **false**: selected nodes are marked in only one of the two diagram sections. Default value: "true".

Accessing Methods
void setInteractionSelectionSynchronized (boolean newValue)
boolean isInteractionSelectionSynchronized ()

InteractionSelectTopNodeOrLinkOnly

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

After clicking a group of overlaying nodes only the node at the top will be selected.

Accessing Methods
void setInteractionSelectTopNodeOrLinkOnly (boolean newValue)
boolean isInteractionSelectTopNodeOrLinkOnly ()

Example Code

```
jGantt1.setInteractionSelectTopNodeOrLinkOnly(true)
```

InteractionTableMouseWheelScrolling

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_MOUSEWHEEL_SCROLLING_HORIZONTAL**

This property lets you set or retrieve in which way the table should react to mouse wheel events.

Possible Values	Description
INTERACTION_MOUSEWHEEL_SCROLLING_HORIZONTAL	After mouse wheel events the component will scroll horizontally.
INTERACTION_MOUSEWHEEL_SCROLLING_NO	The component will not scroll after mouse wheel events.
INTERACTION_MOUSEWHEEL_SCROLLING_VERTICAL	After mouse wheel events the component will scroll vertically.

Accessing Methods
void setInteractionTableMouseWheelScrolling (int newValue)
int getInteractionTableMouseWheelScrolling ()

Example Code

```
jGantt1.setInteractionTableMouseWheelScrolling(JGantt.INTERACTION_MOUSEWHEEL_SCROLLING_NO);
```

InteractionTimeQuantum

Property of JGantt

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_TIMEQUANTUM_SECOND**

This property defines the time quantum which represents the size of the intervals that a node leaps by when moved horizontally. The unit of the return value is milliseconds and may adopt any value, or one of the below constants:

Possible Values	Description
INTERACTION_TIMEQUANTUM_DAY	The number of milliseconds of a day.
INTERACTION_TIMEQUANTUM_HOUR	The number of milliseconds of an hour.
INTERACTION_TIMEQUANTUM_MINUTE	The number of milliseconds of a minute.
INTERACTION_TIMEQUANTUM_NONE	An activity will move smoothly without leaping when moved.
INTERACTION_TIMEQUANTUM_SECOND	The number of milliseconds of a second.
INTERACTION_TIMEQUANTUM_WEEK	The number of milliseconds of a week.

Accessing Methods
void setInteractionTimeQuantum (long newValue)
long getInteractionTimeQuantum ()

KeepIncompleteHierarchy

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

If you are dealing with incomplete hierarchies (for example, if from a very large database that has a hierarchy only a limited number of nodes was selected and if then root nodes are missing and gaps occur in the hierarchy code), you can still perform interactions complying with the hierarchy, if you set this property. VARCHART JGantt internally completes the code as far as possible and this way maintains the incomplete hierarchy.

Accessing Methods
void setKeepIncompleteHierarchy (boolean newValue)
boolean isKeepIncompleteHierarchy ()

Also see CanonicalHierarchyCode
NodeHierarchyBy

Example Code

```
jGantt1.setNodeHierarchyBy(HCODE_INPUT);
jGantt1.setKeepIncompleteHierarchy(true);
```

LinkNodeDateIndexes

Property of JGantt

Typ	**int[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines via names or indexes an array of node attributes that hold the dates for links to join the source and the target node. 2 possible date attributes exist, so the index and the size of the array are limited to this number. The default indexes are 0 and 1. The node dates can be retrieved via the call **getNodeDates**.

Accessing Methods
void setLinkNodeDateIndexes (integer index, int newValues)
void setLinkNodeDateIndexes (int[] newValue)
int getLinkNodeDateIndexes (integer index)
int[] getLinkNodeDateIndexes ()

LinkSetName

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the link entity set.

Accessing Methods
void setLinkSetName (java.lang.String newValue)
java.lang.String getLinkSetName ()

Also see GroupNodeSetName

NodeSetName

LinkSourceNodeBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the link attribute that holds the user-generated identification tag of the source node (the node preceding the link).

Accessing Methods
void setLinkSourceNodeBy (java.lang.String newValue)
java.lang.String getLinkSourceNodeBy ()

Also see LinkTargetNodeBy

LinkTargetNodeBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the link attribute that holds the user-generated identification tag of the target node (the node which succeeds the link).

Accessing Methods
void setLinkTargetNodeBy (java.lang.String newValue)
java.lang.String getLinkTargetNodeBy ()

Also see LinkSourceNodeBy

LinkTypeBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the link attribute that holds the link type. The link type can be set to the below values:

- "FS" = finish-start (default)
- "FF" = finish-finish
- "SS" = start-start
- "SF" = start-finish

These values specify the positions where the link joins the source node or the target node. Other values than the ones listed above are interpreted as "FS".

Accessing Methods
void setLinkTypeBy (java.lang.String newValue)
java.lang.String getLinkTypeBy ()

NodeAnnotations

Property of JGantt

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property via names defines an array of attributes of a node entity that are designed to hold annotations. 9 possible annotation attributes exist, so the index and the size of the array are limited to this number. The indexes occupy certain postions on a node, that depend on the node design assigned.

Accessing Methods
void setNodeAnnotations (integer index, java.lang.String newValues)
void setNodeAnnotations (java.lang.String[] newValue)
java.lang.String getNodeAnnotations (integer index)
java.lang.String[] getNodeAnnotations ()

Also see GroupNodeAnnotations

NodeColorScheme

Property of JGantt

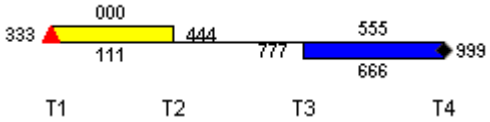
Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color scheme of the nodes in the Gantt graph. The way the colors of the scheme are applied depends on the node design used:

2Rects:

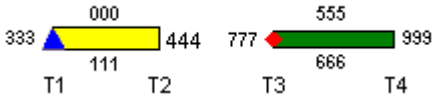
The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor
The bar between T3 and T4 receives shadedAlternateColor

2RectsConnected:



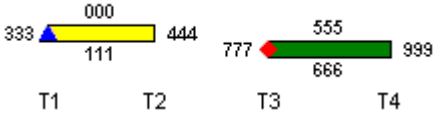
The bar between T3 and T4 receives mainColor.
The bar between T1 and T2 receives shadedColor
The line between T2 and T3 receives lineColor

2 RectsInterlocked:



The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor
The bar between T3 and T4 receives shadedAlternateColor

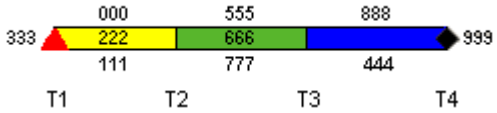
2RectsShifted:



The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor

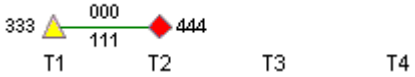
The bar between T3 and T4 receives shadedAlternateColor

3Rects:



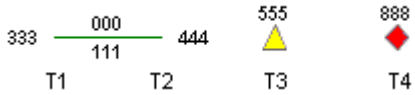
The bar between T3 and T4 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T1 receives alternateColor
The bar between T2 and T3 receives shadedAlternateColor
The symbol on T4 receives lineColor

Line:



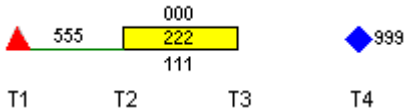
The symbol on T1 receives shadedColor
The symbol on T2 receives alternateColor
The line between T1 and T2 receives shadedAlternateColor

Line2Symbols:



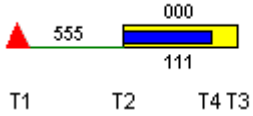
The line between T1 and T2 receives shadedAlternateColor
The symbol on T3 receives shadedColor
The symbol on T4 receives alternateColor

LineBottomRectSymbol:



The symbol on T4 receives mainColor
The bar between T2 and T3 receives shadedColor
The Symbol on T1 receives alternateColor
The line between T1 and T2 receives shadedAlternateColor

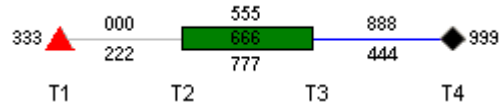
LineCompletionRect:



The narrow bar between T2 and T4 receives mainColor
The wide bar between T2 and T3 receives shadedColor
The symbol on T1 receives alternateColor

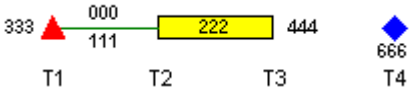
The line between T1 and T2 receives shadedAlternateColor

LineRectLine:



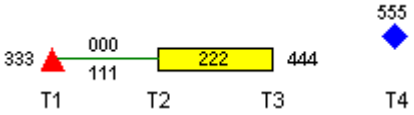
The line between T3 and T4 receives mainColor
The line between T1 and T2 receives shadedColor
The symbol on T1 receives alternateColor
The bar between T2 and T3 receives shadedAlternateColor
The symbol on T4 receives lineColor

LineRectSymbol:



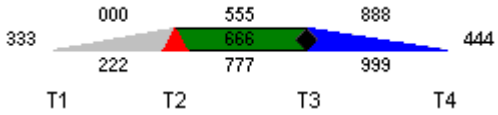
The symbol on T4 receives mainColor
The bar between T2 and T3 receives shadedColor
The symbol on T1 receives alternateColor
The line between T1 and T2 receives shadedAlternateColor

LineRectSymbolAbove:



As Line RectSymbol

Ramps:



The descending ramp between T3and T4 receives mainColor
The ascending ramp between T1 and T2 receives shadedColor
The symbol on T2 receives alternateColor
The bar between T2 and T3 receives shadedAlternateColor
The symbol on T3 receives lineColor

Rect:

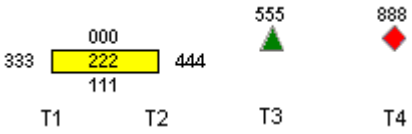
The symbol on T1 receives mainColor
The bar between T1 and T2 receives shadedColor

The symbol on T2 receives alternateColor

Rect2Symbols:

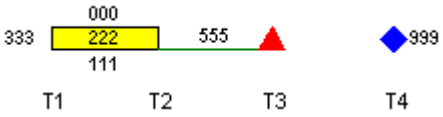
The bar between T1 and T2 receives shadedColor
The symbol on T2 receives alternateColor
The symbol on T3 receives shadedAlternateColor

Rect2Symbols above:



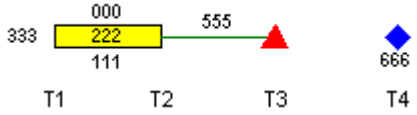
As Rect2Symbols

RectLineBottomSymbol:



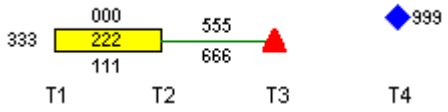
The symbol on T4 receives mainColor
The bar between T1 and T2 receives shadedColor
The symbol on T3 receives alternateColor
The line between T2 and T3 receives shadedAlternateColor

RectLineSymbol:



As RectLineBottomSymbol

RectLineSymbolAbove:



As RectLineBottomSymbol

Summary:

The bar between T1 and T2 receives mainColor

Symbol

The symbol on T1 receives shadedColor

Accessing Methods
void setNodeColorScheme (JGColorScheme newValue)
JGColorScheme getNodeColorScheme ()

Also see GroupNodeColorScheme
TableColorScheme
TimeScaleColorScheme
GanttColorScheme

NodeDateFormat

Property of JGantt

Typ	**java.text.DateFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the format of the dates in node annotations. If no format is set here, the dates will be displayed in a format correspondig to **DateFormat.MEDIUM**.

Accessing Methods
void setNodeDateFormat (java.text.DateFormat newValue)
java.text.DateFormat getNodeDateFormat ()

Also see GroupNodeDateFormat
TableDateFormat

NodeDates

Property of JGantt

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property via attribute names defines an array of attributes of a node entity that are designed to hold dates. 4 possible date attributes exist, so the index and

the size of the array are limited to this number. Two default names exist: "start" and "end".

Accessing Methods
void setNodeDates (integer index, java.lang.String newValues)
void setNodeDates (java.lang.String[] newValue)
java.lang.String getNodeDates (integer index)
java.lang.String[] getNodeDates ()

Also see GroupNodeDates

NodeDesign

Property of JGantt

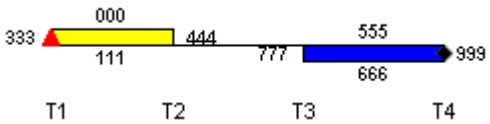
Typ	**JGNodeDesign**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines a design object for the nodes of the Gantt graph. The available values you can retrieve by the method JGNodeDesign.getNames():

2Rects:

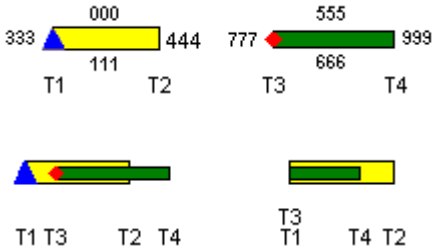
The space between the dates T1-T2 is filled by a broad bar of the shadedColor. Four different annotations can be assigned by the indexes 0, 1, 2 and 3. The space between the dates T3-T4 is filled by a broad bar of the shadedAlternateColor. Four different annotations can be assigned by the indexes 5, 6, 7 and 9. An existing symbol is displayed in mainColor on T1, another existing symbol is displayed on T3 in alternateColor.

2RectsConnected:



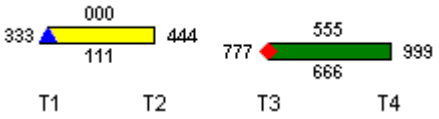
The space between the dates T1-T2 is filled by a narrow bar of the shadedColor and can be annotated by the indexes 0 and 1. The space between the dates T2-T3 is filled by a line of the lineColor and can be annotated by the indexes 4 and 7. The space between the dates T3-T4 is filled by a bar of the mainColor and can be annotated by the indexes 5 and 6.On T1 an existing symbol is displayed in alternateColor. It can be annotated on the left via index 3. On T4 an existing symbol is displayed in lineColor. It can be annotated below via index 9.

2 RectsInterlocked:



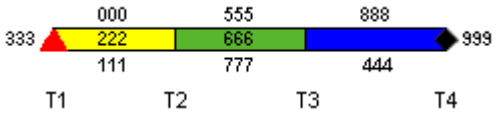
As node design "2Rects", except that the bars appear interlocked as soon as the dates allow for it. The narrow bar can act as an "progress" bar, representing a defined percentage of length of the broad bar, thus indicating the degree of completion of an activity.

2RectsShifted:



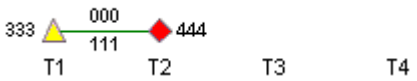
The space between the dates T1-T2 is filled by a narrow bar of the shadedColor. Four different annotations can be assigned by the indexes 0, 1, 3 and 4. The space between the dates T3-T4 is filled by a narrow bar of the shadedAlternateColor. Its position is lower than the one of the first bar. Four different annotations can be assigned by the indexes 5, 6, 7 and 9. An existing symbol is displayed in mainColor on T1, another existing symbol is displayed on T3 in alternateColor.

3Rects:



The space between the dates T1-T2 is filled by a bar of the shadedColor and can be annotated by the indexes 0 through 2. The space between the dates T2-T3 is filled by a bar of the shadedAlter-nateColor and can be annotated by the indexes 5 through 7. The space between the dates T3-T4 is filled by a bar of the mainColor and can be annotated by the indexes 8 and 4. On T1 an existing symbol is displayed in alternateColor. It can be annotated on the left via index 3. On T4 an existing symbol is displayed in lineColor. It can be annotated below via index 9.

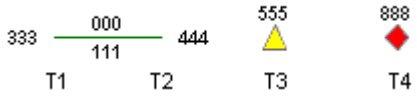
Line:



The space between the dates T1-T2 is filled by a line of the shadedAlternateColor. Two annotations can be assigned by the indexes 0 and 1.

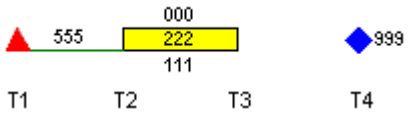
On T1 and T2 existing symbols are displayed in shadedColor and alternateColor, respectively. They can be assigned annotations via the indexes 3 and 4.

Line2Symbols:



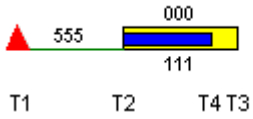
The space between the dates T1-T2 is filled by a line of the shadedAlternateColor. Four different annotations can be assigned by the indexes 0, 1, 3 and 4. On T3 and T4 existing symbols are displayed in shadedColor and alternateColor, respectively. They can be annotated via the indexes 5 and 8.

LineBottomRectSymbol:



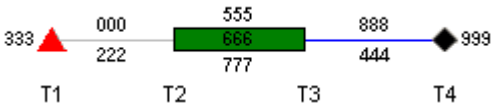
The space between the dates T1-T2 is filled by a subscript line of the shadedAlternateColor and can be annotated by the index 5. On T1 an existing symbol is displayed in alternateColor. The space between the dates T2-T3 is filled by a bar of the shadedColor and can be annotated by the indexes 0 through 2. On T4 an existing symbol is displayed in mainColor. It can be annotated on the right via index 9.

LineCompletionRect:



The space between the dates T1-T2 is filled by a subscript line of the shadedAlternateColor and can be annotated by the index 5. On T1 an existing symbol is displayed in alternateColor. The space between the dates T2-T3 is filled by a broad bar of the shadedColor and can be annotated by the indexes 0 through 2. The space between T2-T4 is filled with a narrow bar, which may act as a progress bar that indicates the degree of completion of an activity.

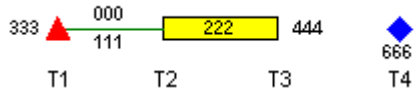
LineRectLine:



The space between the dates T1-T2 is filled by a line of the shadedColor and can be annotated by the indexes 0 and 2. The space between the dates T2-T3 is filled by a bar of the shadedAlter-nateColor and can be annotated by the indexes 5 through 7. The space between the dates T3-T4 is filled by a line of the mainColor and can be annotated by the indexes 8 and 4. On T1 an existing symbol is

displayed in alternateColor. It can be annotated on the left via index 3. On T4 an existing symbol is displayed in lineColor. It can be annotated below via index 9.

LineRectSymbol:

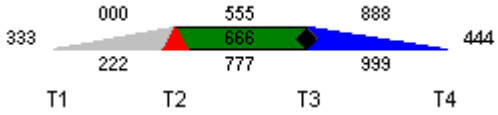


The space between the dates T1-T2 is filled by a line of the shadedAlternateColor and can be annotated by the indexes 0 and 1. On T1 an existing symbol is displayed in alternateColor. It can be annotated on the left via index 3. The space between the dates T2-T3 is filled by a bar of the shadedColor and can be annotated by the indexes 2 and 4. On T4 an existing symbol is displayed in mainColor. It can be annotated below via index 6.

LineRectSymbolAbove:

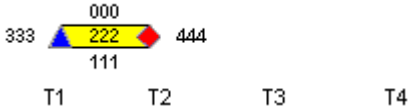
As node design "LineRectSymbol", except that the symbol on T4 appears superscript and its annotation is placed above via index 5.

Ramps:



The space between the dates T1-T2 is filled by an ascending ramp of the shadedColor and can be annotated by the indexes 0, 2 and 3. The space between the dates T2-T3 is filled by a bar of the shadedAlternateColor and can be annotated by the indexes 5 through 7. The space between the dates T3-T4 is filled by a descending ramp of the mainColor and can be annotated by the indexes 8, 9 and 4. On T2 an existing symbol is displayed in alternateColor. On T3 an existing symbol is displayed in lineColor.

Rect:



The space between the dates T1-T2 is filled by a broad bar of the shadedColor. Five different annotations can be assigned by the indexes 0 through 4. Existing symbols will be displayed on T1 and T2 in the mainColor and in the alternateColor, respectively. The spaces between the dates T2-T3-T4 remain empty.

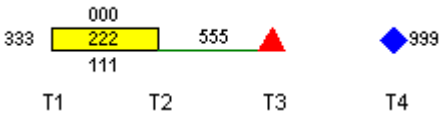
Rect2Symbols:

As node design "Rect", except that on T3 and T4 existing symbols are displayed in the ShadedAlternateColor and alternateColor, respectively. The symbols can be annotated via the indexes 5 and 8.

Rect2Symbols above:

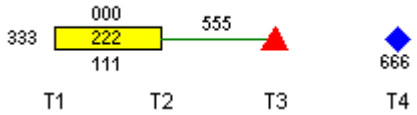
As node design "Recht2Symbols", except that the symbols and their annotations appear to be superscript.

RectLineBottomSymbol:



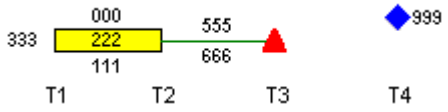
As design "RectLineSymbolAbove", except that the line is subscript and can only be annotated above via index 5.

RectLineSymbol:



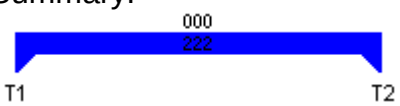
The space between the dates T1-T2 is filled by a broad bar of the shadedColor and can be annotated by the indexes 0 through 3. The space between the dates T2-T3 is filled by a line of the shadedAlternateColor and can be annotated by the index 5. On T3 an existing symbol is displayed in alternateColor. On T4 an existing symbol is displayed in mainColor. It can be annotated below via index 6.

RectLineSymbolAbove:



As node design "RectLineSymbol", except that the line between T2-T3 can be annotated in addition by index 6 and that the symbol on T4 is superscript which can be annotated on its right by index 9.

Summary:



The space between the dates T1-T2 is filled by a summary bar of the mainColor and can be annotated by the indexes 0 and 2.

Symbol

An existing symbol of the shadedColor is displayed on T1. You can place an annotation right of the symbol by index 4.

Accessing Methods
void setNodeDesign (JGNodeDesign newValue)
JGNodeDesign getNodeDesign ()

Also see GroupNodeDesign

NodeFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the font of node annotations.

Accessing Methods
void setNodeFont (java.awt.Font newValue)
java.awt.Font getNodeFont ()

Also see GroupNodeFont
TableFont
TableGroupFont
TableRowTitleFont
TableColumnTitleFont
TimeScaleFont

NodeHierarchyBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property via the name defines an attribute, that is designed to hold the hierarchy code of the node.

Accessing Methods
void setNodeHierarchyBy (java.lang.String newValue)
java.lang.String getNodeHierarchyBy ()

NodePhantomPositionLine

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_NODE_PHANTOM_NO_LINE**

This property defines the place of the positioning line displayed in node phantoms.

Possible Values	Description
INTERACTION_NODE_PHANTOM_BOTH_LINES	A vertical position line is shown at the beginning and at the end of the node phantom.
INTERACTION_NODE_PHANTOM_END_LINE	A vertical position line is shown at the end of the node phantom.
INTERACTION_NODE_PHANTOM_NO_LINE	No position line is shown.
INTERACTION_NODE_PHANTOM_START_LINE	A vertical position line is shown at the beginning of the node phantom.

Accessing Methods
void setNodePhantomPositionLine (int newValue)
int getNodePhantomPositionLine ()

Example Code

```
  public void setNodePhantomPositionLine(int newNodePhantomPositionLine)
    {
    int  oldNodePhantomPositionLine = nodePhantomPositionLine;
    nodePhantomPositionLine = newNodePhantomPositionLine;
    myGanttGraph.setPhantomStyle(newNodePhantomPositionLine);
    propertyChangeListeners.firePropertyChange("nodePhantomPositionLine", new
Integer(oldNodePhantomPositionLine), new Integer(newNodePhantomPositionLine));
   }
```

NodeProfile

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the default profile for the nodes of the Gantt graph. The default profile will apply if no individual profile (please see **nodeProfileBy**) has been assigned.

Accessing Methods
void setNodeProfile (java.lang.String newValue)
java.lang.String getNodeProfile ()

Also see GroupNodeProfile
NodeProfileBy

NodeProfileBy

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the node attribute that holds the name of a calendar profile to be used individually for the node. If the profile specified in the attribute cannot be found, the default node profile (please see **nodeProfile**) will be used.

Accessing Methods
void setNodeProfileBy (java.lang.String newValue)
java.lang.String getNodeProfileBy ()

Also see GroupNodeProfileBy
NodeProfile

NodeSetName

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the name of the entity set that holds the node entities. Default name: "nodes".

Accessing Methods
void setNodeSetName (java.lang.String newValue)
java.lang.String getNodeSetName ()

Also see GroupNodeSetName
LinkSetName

NodesSortedBy

Property of JGantt

Typ	**java.util.Comparator**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines a comparator object used to sort nodes inside the groups. The comparator object must be able to compare two NeIEntities. If no comparator is given, nodes will not be sorted. An appropriate implementation of Comparator is NeEntityComparator.

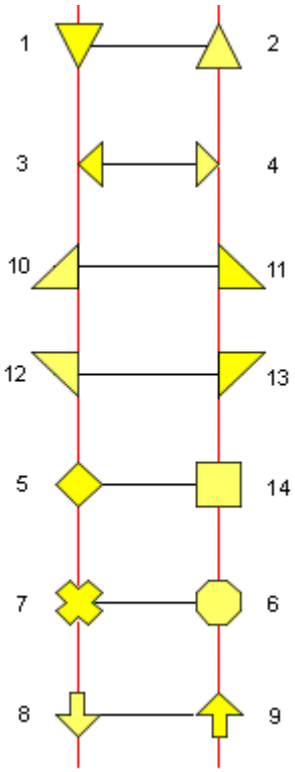
Accessing Methods
void setNodesSortedBy (java.util.Comparator newValue)
java.util.Comparator getNodesSortedBy ()

NodeSymbols

Property of JGantt

Typ	**de.netronic.common.intface.NeILabel[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines an array of node symbols which, as for example the design, are used for the graphical impression of the node. For a node, 2 possible symbol attributes exist, so the index and the size of the array are limited to this number. As values, the figures 1...14 are valid. They represent the below shapes:



Accessing Methods
void setNodeSymbols (integer index, de.netronic.common.intface.NeILabel newValues)
void setNodeSymbols (de.netronic.common.intface.NeILabel[] newValue)
de.netronic.common.intface.NeILabel getNodeSymbols (integer index)
de.netronic.common.intface.NeILabel[] getNodeSymbols ()

Also see GroupNodeSymbols

NodeZeroLengthSymbol

Property of JGantt

Typ	**de.netronic.common.intface.NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines a symbol, that is displayed in the place of a node that has no extent, i.e. where the start date equals the end date. If you do not set this property, the zero-length node will be displayed as a short vertical line.

Accessing Methods
void setNodeZeroLengthSymbol (de.netronic.common.intface.NeILabel newValue)
de.netronic.common.intface.NeILabel getNodeZeroLengthSymbol ()

Also see GroupNodeZeroLengthSymbol

NodeZeroLengthVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not bars in nodes, that have no length, i.e. where the start date equals the end date, will be displayed on the screen. On **true** they will be displayed as a vertical line.

Accessing Methods
void setNodeZeroLengthVisible (boolean newValue)
boolean hasNodeZeroLengthVisible ()

Also see GroupNodeZeroLengthVisible

NumberOfRows

Read Only Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**hidden**

Holds the number of rows currently existing in JGantt.

Accessing Methods
int getNumberOfRows()

PrintManager

Read Only Property of JGantt

Typ	**JGIPrintManager**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the print manager object.

Accessing Methods
JGIPrintManager getPrintManager()

RootGroup

Read Only Property of JGantt

Typ	**de.netronic.common.intface.NeILayouterGroup**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the root group (top group) in multiple grouping.

Accessing Methods
de.netronic.common.intface.NeILayouterGroup getRootGroup()

Scheduler

Read Only Property of JGantt

Typ	**de.netronic.common.intface.NeIScheduler**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the instance of the scheduling module (or scheduler), which automatically is generated with a JGantt object. To each one JGantt object, one scheduler object is allocated.

Accessing Methods
de.netronic.common.intface.NeIScheduler getScheduler()

SectionVertLineGridsEx

Read Only Property of JGantt

Typ	**de.netronic.jgantt.JGVertLineGrids[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the vertical line grids of a time scale section.

Accessing Methods
de.netronic.jgantt.JGVertLineGrids SectionVertLineGridsEx (integer index)
de.netronic.jgantt.JGVertLineGrids[] SectionVertLineGridsEx()

Also see VertLineGridsEx

Example Code

```
jGantt1.getSectionVertLineGridsEx(1);
```

Table

Read Only Property of JGantt

Typ	**de.netronic.common.intface.NeITable**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the table object, that is automatically created when a JGantt object is generated.

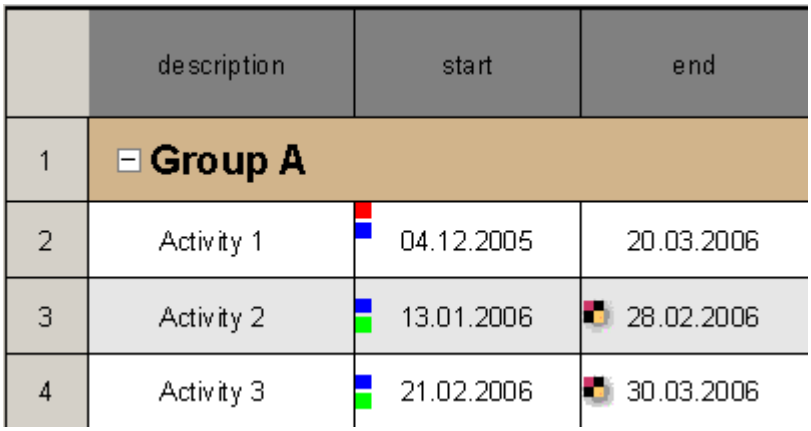
Accessing Methods
de.netronic.common.intface.NeITable getTable()

Table3D

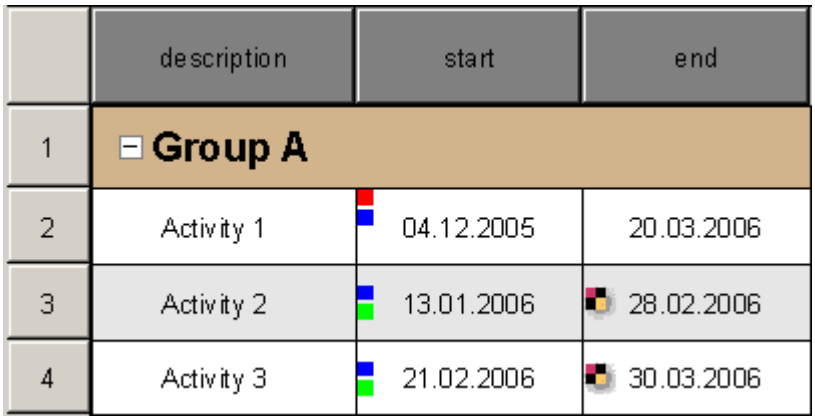
Property of JGantt

Typ	**boolean**
Bound	**yes**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not the header line and row are displayed three-dimensionally.



Above: 3D display switched off.



Above: 3D display switched on.

Accessing Methods
void setTable3D (boolean newValue)
boolean isTable3D ()

Also see TimeScale3D

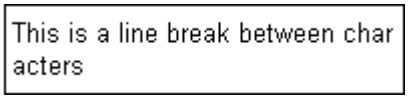
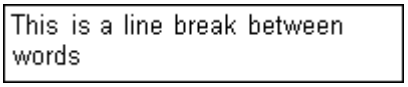
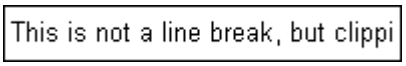
TableAutoWrap

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO_WRAP_OFF**

This property lets you set different types of automatic line breaks to head lines in the default table. For manually defined tables, of which the rows were defined by

NeRowDefinitions, please use the property **set/getAutoWrapMode** of the object **NeFieldStyle**.

Possible Values	Description
AUTO_WRAP_AT_CHARACTER	Automatic line breaks occur between characters: 
AUTO_WRAP_AT_WORD	Automatic line breaks occur between words: 
AUTO_WRAP_OFF	There are no automatic line breaks. To avoid the text to be clipped, line breaks can be set manually by **\n!**. 

Accessing Methods
void setTableAutoWrap (int newValue)
int getTableAutoWrap ()

Example Code

```
jGantt1.setTableAutoWrap( JGantt.AUTO_WRAP_AT_WORD );
```

TableColorScheme

Property of JGantt

Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color scheme for the table of the Gantt diagram. The mainColor is used for the leftmost column. The shadedColor applies to the trunk of the table, the alternateColor appears in the header row and the shadedAlternateColor dyes group rows. If you use a dynamic color (NeIDynamicColor) as shadedColor or as shadedAlternateColor, you can set

individual colors to the rows. The reference object for the dynmaic color is the entity in the row (also see NeIDynamicColor, JGDynamicRowColor).

Accessing Methods
void setTableColorScheme (JGColorScheme newValue)
JGColorScheme getTableColorScheme ()

Also see GanttColorScheme
GroupNodeColorScheme
NodeColorScheme
TimeScaleColorScheme

TableColumns

Property of JGantt

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines via attribute names an array of attributes of a node entity that are designed to hold the data of table columns. There is no limit to the size of the array, so the index can adopt any value.

Accessing Methods
void setTableColumns (integer index, java.lang.String newValues)
void setTableColumns (java.lang.String[] newValue)
java.lang.String getTableColumns (integer index)
java.lang.String[] getTableColumns ()

TableColumnTitleFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the font of the column titles (top row) of the table.

Accessing Methods
void setTableColumnTitleFont (java.awt.Font newValue)
java.awt.Font getTableColumnTitleFont ()

Also see NodeFont
GroupNodeFont

TableFont
TableGroupFont
TableRowTitleFont
TimeScaleFont

TableColumnTitlesSource

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**ENTITY_ATTRIBUTE_DISPLAY_NAME**

This property defines the source that the column titles (top row) of the table are taken from.

Possible Values	Description
ENTITY_ATTRIBUTE_DISPLAY_NAME	The display name of the entity attribute, which is assigned to the column, is used as the column title.
ENTITY_ATTRIBUTE_NAME	The name of the entity attribute, which is assigned to the column, is used as the column title.
GROUP_ENTITY_ATTRIBUTE_DISPLAY_NAME	The display name in the group node entity set of the entity attribute, which is assigned to the column, is used as the column title.
NODE_ENTITY_ATTRIBUTE_DISPLAY_NAME	The display name in the node entity set of the entity attribute, which is assigned to the column, is used as the column title.

Accessing Methods
void setTableColumnTitlesSource (int newValue)
int getTableColumnTitlesSource ()

Also see TableCornerText

Example Code

```
jGantt1.setTableColumnTitlesSource
(JGantt.ENTITY_ATTRIBUTE_DISPLAY_NAME);
```

TableColumnWidths

Property of JGantt

Typ	**double[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Values	**15.0**

This property defines values for the widths of the table columns. The unit is millimeters.

Accessing Methods
void setTableColumnWidths (integer index, double newValues)
void setTableColumnWidths (double[] newValue)
double getTableColumnWidths (integer index)
double[] getTableColumnWidths ()

TableCornerText

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property holds a text string for the cell in the top left corner of the table.

Accessing Methods
void setTableCornerText (java.lang.String newValue)
java.lang.String getTableCornerText ()

Also see TableColumnTitlesSource

TableDateFormat

Property of JGantt

Typ	**java.text.DateFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the format of the dates in the table. If no format is set here, the dates will be displayed in a format correspondig to **DateFormat.MEDIUM**.

Accessing Methods
void setTableDateFormat (java.text.DateFormat newValue)
java.text.DateFormat getTableDateFormat ()

Also see GroupNodeDateFormat
NodeDateFormat

TableEditable

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines, whether or not the table can be edited.

Accessing Methods
void setTableEditable (boolean newValue)
boolean isTableEditable ()

TableFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the font for the node rows in the table.

Accessing Methods
void setTableFont (java.awt.Font newValue)
java.awt.Font getTableFont ()

Also see GroupNodeFont
NodeFont
TableGroupFont
TableRowTitleFont
TableColumnTitleFont
TimeScaleFont

TableGroupFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines a font for group rows in the table.

Accessing Methods
void setTableGroupFont (java.awt.Font newValue)
java.awt.Font getTableGroupFont ()

Also see GroupNodeFont
NodeFont
TableFont
TableColumnTitleFont
TableRowTitleFont
TimeScaleFont

TableHierarchyColumn

Property of JGantt

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines a table column to display the node hierarchy.

Accessing Methods
void setTableHierarchyColumn (int newValue)
int getTableHierarchyColumn ()

Also see TableHierarchyIndentWidth

TableHierarchyIndentWidth

Property of JGantt

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**3.0**

This property defines the width of the indent of the string in the table column that has been assigned the hierarchical indent. The unit is millimeters.

Accessing Methods
void setTableHierarchyIndentWidth (double newValue)
double getTableHierarchyIndentWidth ()

Also see TableHierarchyColumn

TableRowTitleFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the font of the row titles (fields of the leftmost column) in the table. Also see **tableColumnTitleFont**.

Accessing Methods
void setTableRowTitleFont (java.awt.Font newValue)
java.awt.Font getTableRowTitleFont ()

Also see GroupNodeFont
NodeFont
TableFont
TableGroupFont
TableColumnTitleFont
TimeScaleFont

TableRowTitlesVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property defines, whether the column in the most left position (row titles) should be visible or hidden.

Accessing Methods
void setTableRowTitlesVisible (boolean newValue)
boolean isTableRowTitlesVisible ()

TableSashOneTouchExpandable

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

If you set this property to false, the arrows at the table sash (the separator between table and gantt graph), which enable collapsing or expanding of the table, will disappear.

Accessing Methods
void setTableSashOneTouchExpandable (boolean newValue)
boolean hasTableSashOneTouchExpandable ()

Example Code

```
jGantt1.setTableSashOneTouchExpandable(false);
```

TableVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property defines, whether the table is visible or hidden.

Accessing Methods
void setTableVisible (boolean newValue)
boolean isTableVisible ()

TimeScale

Read Only Property of JGantt

Typ	**JGTimeScale**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves an instance of the time scale, that is automatically created when a JGantt object is generated. The time scale may appear above or beneath the Gantt graph.

Accessing Methods
JGTimeScale getTimeScale()

TimeScale3D

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines, whether or not the time scale is displayed three-dimensionally.

Accessing Methods
void setTimeScale3D (boolean newValue)
boolean isTimeScale3D ()

Also see Table3D

TimeScaleAbsoluteResolution

Property of JGantt

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the absolute resolution of the time scale. It equals the width of 1mm per day.

Accessing Methods
void setTimeScaleAbsoluteResolution (double newValue)
double getTimeScaleAbsoluteResolution ()

TimeScaleAdvancedMouseInteractions

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Hereby you can specify an alternative reaction to mouse interactions in the timecale:

With a mouse drag you change the visible part of the timescale.
With the mousewheel you specify the resolution of the timescale.

Accessing Methods
void setTimeScaleAdvancedMouseInteractions (boolean newValue)
boolean hasTimeScaleAdvancedMouseInteractions ()

Example Code

```
jGantt1.setTimeScaleAdvancedMouseInteractions(true);
```

TimeScaleAntialiasText

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Antialiasing on text strings can be switched on or off.

Accessing Methods
void setTimeScaleAntialiasText (boolean newValue)
boolean isTimeScaleAntialiasText ()

TimeScaleAtBottomVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not the bottom time scale should be visible.

Accessing Methods
void setTimeScaleAtBottomVisible (boolean newValue)
boolean isTimeScaleAtBottomVisible ()

Also see TimeScaleAtTopVisible

TimeScaleAtTopVisible

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property defines whether or not the top time scale should be visible.

Accessing Methods
void setTimeScaleAtTopVisible (boolean newValue)
boolean isTimeScaleAtTopVisible ()

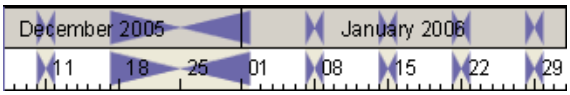
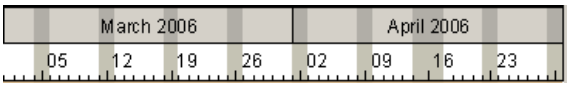
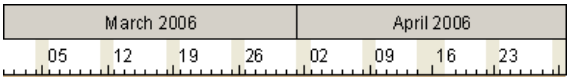
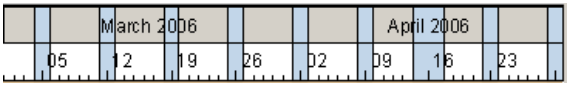
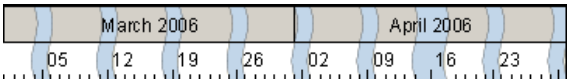
Also see TimeScaleAtBottomVisible

TimeScaleCollapseDisplayMode

Property of JGantt

Typ	**int**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**COLLAPSE_DISPLAY_WAVE**

This property defines, in which way collapsed non-working times are visualized.

Possible Values	Description
COLLAPSE_DISPLAY_BOWTIE	Collapsed non-working times are displayed as colored bowties. The color is set by the **alternateColor** of the **timeScaleColorScheme**. 
COLLAPSE_DISPLAY_DARKEN	Collapsed non-working times are displayed as dimmed rectangles. The degree of dimming is proportional to the size of the factor set by the property **TimeScaleCollapseFactor**. 
COLLAPSE_DISPLAY_NONE	Collapsed non-working times are not displayed. The picture below shows merely the calendar, which was graphically visualized. 
COLLAPSE_DISPLAY_RECT	Collapsed non-working times are displayed as colored rectangles. The color is set by the **alternateColor** of the **timeScaleColorScheme**. 
COLLAPSE_DISPLAY_WAVE	Collapsed non-working times are displayed as a vertical, colored wave. The color is set by the **alternateColor** of the **timeScaleColorScheme**. 

Accessing Methods
void setTimeScaleCollapseDisplayMode (int newValue)
int getTimeScaleCollapseDisplayMode ()

Also see TimeScaleCollapseFactor
TimeScaleCollapseProfile

TimeScaleCollapseFactor

Property of JGantt

Typ	**double**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0**

This property sets the factor, by which the time scale is downsized by collapsing non-working times. Permitted values: {0.0...1.0}

Accessing Methods
void setTimeScaleCollapseFactor (double newValue)
double getTimeScaleCollapseFactor ()

Also see TimeScaleCollapseDisplayMode
TimeScaleCollapseProfile

TimeScaleCollapseProfile

Property of JGantt

Typ	**java.lang.String**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**

By this property you can set a collapse profile to the time scale, by which non-working times can be collapsed. The profile is assigned by its name.

Accessing Methods
void setTimeScaleCollapseProfile (java.lang.String newValue)
java.lang.String getTimeScaleCollapseProfile ()

Also see TimeScaleCollapseDisplayMode
TimeScaleCollapseFactor

TimeScaleColorScheme

Property of JGantt

Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color scheme for the time scale. The mainColor and the shadedColor apply to the time ribbons in the scale. The alternateColor and the shadedAlternateColor are used in an additional calendar ribbon. The lineColor is used for ticks and framing lines of the time scale.

Accessing Methods
void setTimeScaleColorScheme (JGColorScheme newValue)
JGColorScheme getTimeScaleColorScheme ()

Also see GanttColorScheme
GroupNodeColorScheme
NodeColorScheme
TableColorScheme
TimeScaleSectionColorSchemes

TimeScaleDisplayProfile

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the name of the calendar profile for the time scale. This profile is displayed in the time scale and visualizes a pattern of working periods and non-working periods in the time scale. The way it does this depends on the type of time scale used. It may show different colors in week-ends or display a separate ribbon that shows a color pattern of working periods and nonworking periods, etc.

Setting this property to a value != **null** will switch the visualization of working patterns on.
The colors used for the visualization of the calendar are controlled by the timescaleColorScheme property. TimeSpans of the spanID 0 will adopt the shadedAlternateColor, all other timeSpans will be displayed in the alternateColor. Different colors you can set by using NeMappedColors.

Accessing Methods
void setTimeScaleDisplayProfile (java.lang.String newValue)
java.lang.String getTimeScaleDisplayProfile ()

TimeScaleDynamic

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not the time scale adapts dynamically to the change of size when zooming, that is, whether it changes its units, for example from a daily scale to a weekly or monthly scale. **True**: the time scale adapts dynamically, **false**: the time scale does not adapt dynamically.

Accessing Methods
void setTimeScaleDynamic (boolean newValue)
boolean isTimeScaleDynamic ()

TimeScaleEnd

Property of JGantt

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the final date of the time scale as the number of milliseconds since January, 1st 1970.

Accessing Methods
void setTimeScaleEnd (long newValue)
long getTimeScaleEnd ()

Also see TimeScaleStart
TimeScaleViewEnd

TimeScaleFont

Property of JGantt

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines a font for the time scale.

Accessing Methods
void setTimeScaleFont (java.awt.Font newValue)
java.awt.Font getTimeScaleFont ()

Also see GroupNodeFont
NodeFont
TableColumnTitleFont
TableFont
TableGroupFont
TableRowTitleFont
TimeScaleSectionFonts

TimeScalePopupEnabled

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not the menu that appears on pressing the right mouse button is enabled to pop up on the time scale.

Accessing Methods
void setTimeScalePopupEnabled (boolean newValue)
boolean isTimeScalePopupEnabled ()

TimeScaleResolution

Property of JGantt

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the resolution of the time scale. The resolution is the width (in mm) of a single base unit of the time scale.

Accessing Methods
void setTimeScaleResolution (double newValue)
double getTimeScaleResolution ()

Also see TimeScaleResolutionUserModifiable

TimeScaleResolutionUserModifiable

Property of JGantt

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property enables the user to adjust the time scale resolution at runtime by setting the **resolution modify interaction** to **true** or **false**.

Accessing Methods
void setTimeScaleResolutionUserModifiable (boolean newValue)
boolean isTimeScaleResolutionUserModifiable ()

Also see TimeScaleResolution

TimeScaleSectionAbsoluteResolutions

Property of JGantt

Typ	**double[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the absolute resolutions of the time scale sections. It equals the width of 1mm per day.

Accessing Methods
void setTimeScaleSectionAbsoluteResolutions (integer index, double newValues)
void setTimeScaleSectionAbsoluteResolutions (double[] newValue)
double getTimeScaleSectionAbsoluteResolutions (integer index)
double[] getTimeScaleSectionAbsoluteResolutions ()

Also see TimeScaleAbsoluteResolution
TimeScaleResolution
TimeScaleSectionResolutions

TimeScaleSectionColorSchemes

Property of JGantt

Typ	**de.netronic.jgantt.JGColorScheme[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the JGColorSchemes of the time scale sections.

Accessing Methods
void setTimeScaleSectionColorSchemes (integer index, de.netronic.jgantt.JGColorScheme newValues)
void setTimeScaleSectionColorSchemes (de.netronic.jgantt.JGColorScheme[] newValue)
de.netronic.jgantt.JGColorScheme getTimeScaleSectionColorSchemes (integer index)
de.netronic.jgantt.JGColorScheme[] getTimeScaleSectionColorSchemes ()

Also see TimeScaleColorScheme

TimeScaleSectionFonts

Property of JGantt

Typ	**java.awt.Font[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the fonts of the time scale sections.

Accessing Methods
void setTimeScaleSectionFonts (integer index, java.awt.Font newValues)
void setTimeScaleSectionFonts (java.awt.Font[] newValue)
java.awt.Font getTimeScaleSectionFonts (integer index)
java.awt.Font[] getTimeScaleSectionFonts ()

Also see TimeScaleFont

TimeScaleSectionResolutions

Property of JGantt

Typ	**double[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the resolutions of the time scale sections. The resolution are the widths (in mm) of the single base units of the corresponding time scale sections.

Accessing Methods
void setTimeScaleSectionResolutions (integer index, double newValues)
void setTimeScaleSectionResolutions (double[] newValue)
double getTimeScaleSectionResolutions (integer index)
double[] getTimeScaleSectionResolutions ()

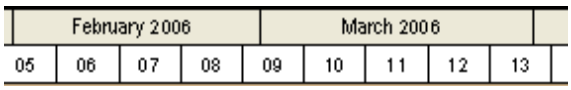
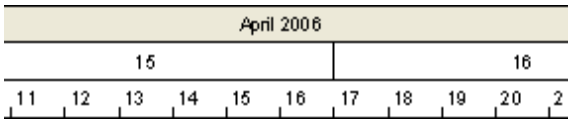
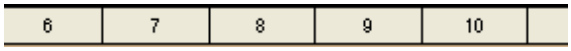
Also see TimeScaleAbsoluteResolution
TimeScaleResolution
TimeScaleSectionAbsoluteResolutions

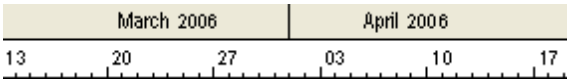
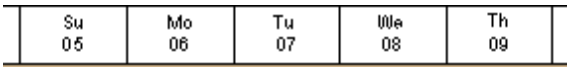
TimeScaleSectionTypes

Property of JGantt

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Values	**TIMESCALE_TYPE_WEEKDAY_CALENDARWEEK**

This property defines the types of the time scale sections.

Possible Values	Description
TIMESCALE_TYPE_CALENDARWEEK	Time scale of calendar week graduation 
TIMESCALE_TYPE_DAY	Time scale of day graduation
TIMESCALE_TYPE_DAY_CALENDARWEEK	Time scale of a days' and a calendar week graduation 
TIMESCALE_TYPE_HOUR	Time scale of hour graduation 
TIMESCALE_TYPE_MINUTE	Time scale of minute graduation
TIMESCALE_TYPE_MONTH	Time scale of month graduation 
TIMESCALE_TYPE_MONTH_CENTERED_MEDIUM	Time scale of month graduation. The designation of a month is abbreviated by three characters and its alignment is centered. 
TIMESCALE_TYPE_MONTH_CENTERED_SHORT	Time scale of month graduation. The designation of a month is abbreviated by a single character and its alignment is centered. 
TIMESCALE_TYPE_QUARTER	Time scale of quarter graduation 
TIMESCALE_TYPE_RELATIVE_DAYS	Time scale of relative day graduation 

TIMESCALE_TYPE_RELATIVE_HOURS	Time scale of relative hour graduation 
TIMESCALE_TYPE_SECOND	Time scale of second graduation
TIMESCALE_TYPE_SHIFT	Time scale of shift graduation
TIMESCALE_TYPE_WEEK	Time scale of week graduation 
TIMESCALE_TYPE_WEEKDAY	Time scale of week day graduation 
TIMESCALE_TYPE_WEEKDAY_CALENDARWEEK	Time scale of a weekdays' and a calendar week graduation

Accessing Methods
void setTimeScaleSectionTypes (integer index, java.lang.String newValues)
void setTimeScaleSectionTypes (java.lang.String[] newValue)
java.lang.String getTimeScaleSectionTypes (integer index)
java.lang.String[] getTimeScaleSectionTypes ()

Also see TimeScaleType

Example Code

```
jGantt.setTimeScaleSectionTypes(0, JGantt.TIMESCALE_TYPE_CALENDARWEEK);
```

TimeScaleStart

Property of JGantt

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the start date of the time scale. The date is specified in milliseconds since January 1st, 1970.

Accessing Methods
void setTimeScaleStart (long newValue)
long getTimeScaleStart ()

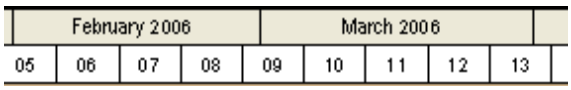
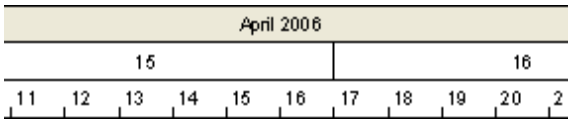
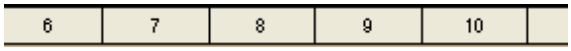
Also see TimeScaleEnd
TimeScaleViewStart

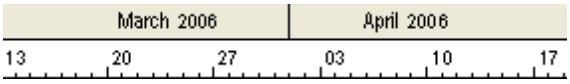
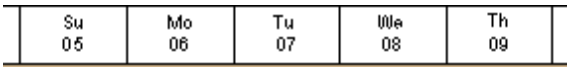
TimeScaleType

Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**TIMESCALE_TYPE_WEEK**

This property defines the type of the time scale.

Possible Values	Description
TIMESCALE_TYPE_CALENDARWEEK	Time scale of calendar week graduation 
TIMESCALE_TYPE_DAY	Time scale of day graduation
TIMESCALE_TYPE_DAY_CALENDARWEEK	Time scale of a days' and a calendar week graduation 
TIMESCALE_TYPE_HOUR	Time scale of hour graduation 
TIMESCALE_TYPE_MINUTE	Time scale of minute graduation
TIMESCALE_TYPE_MONTH	Time scale of month graduation 
TIMESCALE_TYPE_MONTH_CENTERED_MEDIUM	Time scale of month graduation. The designation of a month is abbreviated by three characters and its alignment is centered. 
TIMESCALE_TYPE_MONTH_CENTERED_SHORT	Time scale of month graduation. The designation of a month is abbreviated by a single character and its alignment is centered. 
TIMESCALE_TYPE_QUARTER	Time scale of quarter graduation 
TIMESCALE_TYPE_RELATIVE_DAYS	Time scale of relative day graduation 

TIMESCALE_TYPE_RELATIVE_HOURS	Time scale of relative hour graduation 
TIMESCALE_TYPE_SECOND	Time scale of second graduation
TIMESCALE_TYPE_SHIFT	Time scale of shift graduation
TIMESCALE_TYPE_WEEK	Time scale of week graduation 
TIMESCALE_TYPE_WEEKDAY	Time scale of week day graduation 
TIMESCALE_TYPE_WEEKDAY_CALENDARWEEK	Time scale of a weekdays' and a calendar week graduation

Accessing Methods
void setTimeScaleType (java.lang.String newValue)
java.lang.String getTimeScaleType ()

Also see TimeScaleSectionTypes

Example Code

```
jGantt.setTimeScaleType(JGantt.TIMESCALE_TYPE_CALENDARWEEK);
```

TimeScaleViewEnd

Read Only Property of JGantt

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property handles the start date of the window that the time scale and the Gantt graph are displayed by when the window is taken to the screen. This property cannot be set, because the end is calculated automatically from the size of the view and the resolution of the time scale.

Accessing Methods
long getTimeScaleViewEnd()

Also see TimeScaleEnd
TimeScaleViewStart

TimeScaleViewStart

Property of JGantt

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the start date of the window that the time scale and the Gantt graph are displayed by when the window is taken to the screen. The start date set here will be ignored in case it was set too close to the end of the view. Then the start date will be replaced by an automatically calculated one that considers the size of the view and the resolution of the time scale.

Accessing Methods
void setTimeScaleViewStart (long newValue)
long getTimeScaleViewStart ()

Also see TimeScaleViewEnd

Version

Read Only Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the version number of the JGantt object

Accessing Methods
java.lang.String getVersion()

VersionBuildNumber

Read Only Property of JGantt

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the version and build number of the JGantt object

Accessing Methods
java.lang.String getVersionBuildNumber()

VertLineGridsEx

Read Only Property of JGantt

Typ	**de.netronic.jgantt.JGVertLineGrids**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property etrieves the vertical line grids of the first time scale section.

Accessing Methods
de.netronic.jgantt.JGVertLineGrids getVertLineGridsEx()

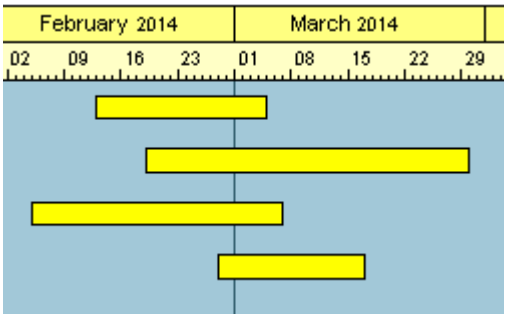
Also see SectionVertLineGridsEx

Methods of the Class

addGanttDateLine

Method of JGantt

This method adds a date line to the Gantt graph.



Declaration
void addGanttDateLine (de.netronic.common.beanbase.NeDateLine dateLine)

	Data Type	Description
Parameter		
dateLine	de.netronic.common.beanbase.-NeDateLine	Date line to be added
Return Value	void	

Also see removeGanttDateLine

addLinkChangeListener

Method of JGantt

Adds a listener for link change events. The listener will be informed if a link is about to be or has been changed, created or deleted.

Declaration
void addLinkChangeListener (de.netronic.common.event.NeObjectChangeListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeObjectChangeListener	Listener to be added.
Return Value	void	

Also see removeLinkChangeListener

addNodeChangeListener

Method of JGantt

This method adds a listener for node change events. The listener will be informed if a node is about to be or has been changed, created or deleted.

Declaration
void addNodeChangeListener (de.netronic.common.event.NeObjectChangeListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeObjectChangeListener	Listener to be added.
Return Value	void	

Also see removeNodeChangeListener

addPropertyChangeListener

Method of JGantt

This method adds a listener for change events in the JGantt object. The listener will be informed each time a property of the JGantt has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

addTimeScaleSection

Method of JGantt

Using this method the time scale is subdivided at a given date. In doing so an existing section or the entire time scale is divided in two sections. The index of the new generated section is returned.

Declaration
int addTimeScaleSection (long separationDate)

	Data Type	Description
Parameter		
separationDate	long	
Return Value	int	Index of the new time sacle section

Also see removeTimeScaleSection

Example Code

```
int sectionIndex;
// The following call adds a new section which starts
// on the 15th of May, 2008
sectionIndex = jGantt.addTimeScaleSection(new GregorianCalendar(2008, 4,
10).getTime().getTime());
```

calcOptimizedTableColumnWidth

Method of JGantt

By this method you can calculate and set the optimum width of a table column. The strings in the cells of the column will be completely visible while the column will be as narrow as possible. The index of the column affected is to be passed via the parameter. The method will return the optimum width calculated.

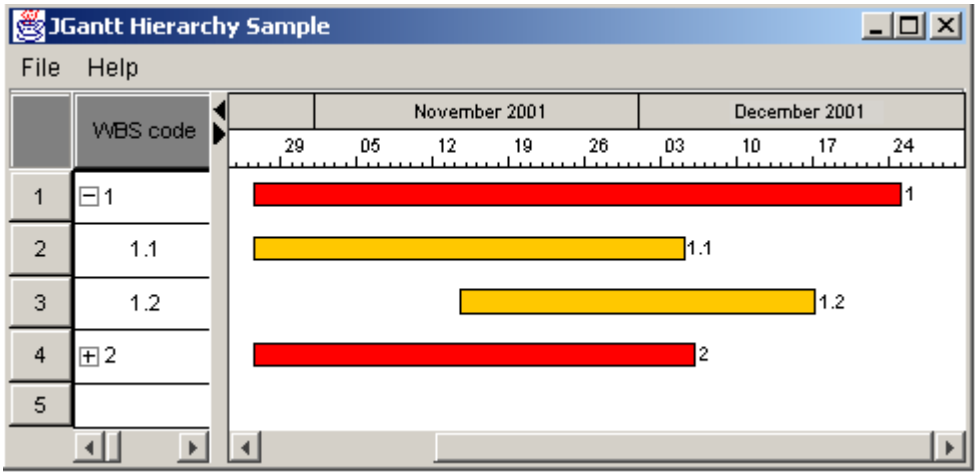
Declaration
double calcOptimizedTableColumnWidth (int columnIndex)

	Data Type	Description
Parameter		
columnIndex	int	Index of the column, the optimum width of which is to be calculated.
Return Value	double	Table width calculated (in 1/100 millimeters)

collapseNode

Method of JGantt

This method lets you collapse, i.e. hide, the nodes that are subordnated to a node in the hierarchy. Nodes in the node hierarchy are identified by their hierarchy code. If the subordinated nodes are visible, the top node is called "expanded", otherwise it is a "collapsed" node.



Collapsed node: subordinated nodes are invisible.

Declaration
void collapseNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of the node the child nodes of which are to be collapsed.
Return Value	void	

Also see expandNode

copyPropertiesFrom

Method of JGantt

This method lets you copy all properties from a template JGantt object.

Declaration
void copyPropertiesFrom (JGantt template)

	Data Type	Description
Parameter		
template	JGantt	Template that the properties are copied from.
Return Value	void	

deleteSelectedNodes

Method of JGantt

All nodes selected are removed from the Gantt diagram and are deleted from the AppData object associated with the JGantt class.
In the AppData object the data objects (entities) are stored that describe the activities.

Declaration
void deleteSelectedNodes ()

	Data Type	Description
Return Value	void	

doLayout

Method of JGantt

For internal use only

Declaration
void doLayout ()

	Data Type	Description
Return Value	void	

editSelectedEntities

Method of JGantt

This method lets you invoke an editor dialog for the user to edit the selected entities.

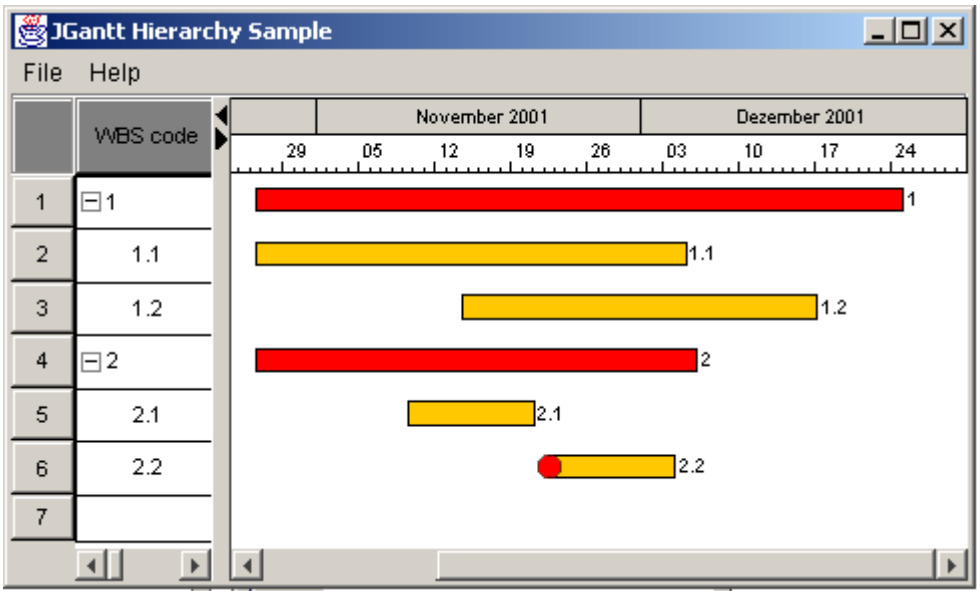
Declaration
void editSelectedEntities ()

	Data Type	Description
Return Value	void	

expandNode

Method of JGantt

This method lets you expand, i.e. display, the nodes that are subordnated to a node in the hierarchy. Nodes in the node hierarchy are identified by their hierarchy code. If the subordinated nodes are visible, the top node is called "expanded", otherwise it is a "collapsed" node.



Expanded node: subordinated nodes are visible.

Declaration
void expandNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of the node the child nodes of which are to be expanded.
Return Value	void	

Also see collapseNode

export

Method of JGantt

This method lets you export a bitmap by specifying the target, the export format, the width and the height and the degree of compression.

Declaration
void export (java.io.OutputStream outputStream, int exportFormat, int width, int height, int compressionLevel)

	Data Type	Description
Parameter		
outputStream	java.io.OutputStream	Target of the export.
exportFormat	int	Format that the Gantt chart is to be exported to.
	Possible Values:	
	EXPORT_PNG_FORMAT	PNG export format
	EXPORT_SVG_FORMAT	SVG export format
	EXPORT_VMF_FORMAT	VMF export format
width	int	Width of the target file in pixels.
height	int	Height of the target file in pixels.
compressionLevel	int	Degree of compression. Possible values: -1Default value, depends of the export format. 0 no compression 1-9 ascending degree of compression. The high values may impair the quality.
Return Value	void	

Also see export
export

export

Method of JGantt

This method lets you export a bitmap by specifying the target, the export format, the resolution and the degree of compression.

Declaration
void export (java.io.OutputStream outputStream, int exportFormat, int resolution, int compressionLevel)

	Data Type	Description
Parameter		
outputStream	java.io.OutputStream	Target of the export.
exportFormat	int	Format that the Gantt chart is to be exported to.
	Possible Values:	
	EXPORT_PNG_FORMAT	PNG export format
	EXPORT_SVG_FORMAT	SVG export format
	EXPORT_VMF_FORMAT	VMF export format
resolution	int	Resolution in dots per inch.
compressionLevel	int	Degree of compression. Possible values: -1Default value, depends of the export format. 0 no compression 1-9 ascending degree of compression. The high values may impair the quality.
Return Value	void	

Also see export
export

export

Method of JGantt

This method lets you export a bitmap by specifying the target and the export format.

Declaration
void export (java.io.OutputStream outputStream, int exportFormat)

	Data Type	Description
Parameter		
outputStream	java.io.OutputStream	Target of the export.
exportFormat	int	Format that the Gantt chart is to be exported to.
	Possible Values:	
	EXPORT_PNG_FORMAT	PNG export format
	EXPORT_SVG_FORMAT	SVG export format
	EXPORT_VMF_FORMAT	VMF export format
Return Value	void	

Also see export
export

finalActionsOfPropertySetting

Method of JGantt

For internal use only.

Declaration
void finalActionsOfPropertySetting ()

	Data Type	Description
Return Value	void	

generateMouseClicked

Method of JGantt

For internal use only

Declaration
void generateMouseClicked (int x, int y)

	Data Type	Description
Parameter		
x	int	
y	int	
Return Value	void	

getDiagramAnnotation

Method of JGantt

This method lets you retrieve the diagram annotation in the header or footer.

Declaration
de.netronic.jgantt.JGDiagramAnnotation getDiagramAnnotation (int position)

	Data Type	Description
Parameter		
position	int	Position of the diagram annotation to be retrieved
	Possible Values:	
	DIAGRAM_FOOTER	Footer of the diagram
	DIAGRAM_HEADER	Header of the diagram
Return Value	de.netronic.jgantt.JGDiagramAnnotation	Diagram annotation

Also see setDiagramAnnotation

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
```

getLayouterGroupForNode

Method of JGantt

Retrieves the group to which an activity belongs. The activity is to be passed by the parameter.

Declaration
de.netronic.common.intface.NeILayouterGroup getLayouterGroupForNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity, of which the group that it belongs to is to be retrieved.
Return Value	de.netronic.common.intface.-NeILayouterGroup	Group, to which the activity belongs.

getLayouterHelper

Method of JGantt

This method lets you retrieve the JGLayouterHelper object allocated to this instance of VARCHART JGantt.

Declaration
JGLayouterHelper getLayouterHelper ()

	Data Type	Description
Return Value	JGLayouterHelper	Retrieves the JGLayouterHelper object allocated to this instance of VARCHART JGantt.

Example Code

```
JGLayouterHelper helper = jGantt1.getLayouterHelper();
```

getLeafNodeFilter

Method of JGantt

This method lets you retrieve a filter for leaf nodes. A leaf node filter is a filter which returns **false** if the entity has child nodes in a hierarchical structure. Leaf nodes are the base nodes in a hierarchy.

Declaration
de.netronic.common.intface.NeIFilter getLeafNodeFilter ()

	Data Type	Description
Return Value	de.netronic.common.intface.NeIFilter	Filter object returned

getParentNode

Method of JGantt

If the property **GroupMode** was set to **GROUP_BY_HIERARCHY**, this method will return the parent node of the node passed, otherwise it will return null.

Deprecated, see JGLayouterHelper.getParentNode !

Declaration
de.netronic.common.intface.NeIEntity getParentNode (de.netronic.common.intface.NeIEntity node)

	Data Type	Description
Parameter		
node	de.netronic.common.intface.NeIEntity	Get the parent for this entity
Return Value	de.netronic.common.intface.NeIEntity	Entity of the parent node

getPersistenceManager

Method of JGantt

Returns the persistence manager of this jgantt instance.

Declaration
de.netronic.jgantt.JGIPersistenceManager getPersistenceManager ()

	Data Type	Description
Return Value	de.netronic.jgantt.JGIPersistenceManager	The persistence manager of this JGantt instance.

getPropertyEditor

Method of JGantt

This method retrieves the property editor belonging to this JGantt instance.

Declaration
JPEIJGanttPropertyEditor getPropertyEditor ()

	Data Type	Description
Return Value	JPEIJGanttPropertyEditor	The property editor belonging to this JGantt instance

Example Code

```
jGantt1.getPropertyEditor().show();
```

getRowIndexForNode

Method of JGantt

This method lets you retrieve the row index of the entity passed.

Deprecated, see JGLayouterHelper.getRowIndexForNode!

Declaration
int getRowIndexForNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity the row index of which is to be retrieved.
Return Value	int	Row index returned

Also see iterateNodeEntitiesInRow

getTimeScaleDateFormats

Method of JGantt

Retrieves for a given timescale type and a ribbon number all user-defined date formats.

Declaration
java.text.DateFormat[] getTimeScaleDateFormats (java.lang.String timeScaleType, int ribbonIndex)

	Data Type	Description
Parameter		
timeScaleType	java.lang.String	Type of timescale
ribbonIndex	int	Ribbon index, starts with 0 beginning at topmost ribbon.
Return Value	java.text.DateFormat[]	Date formats

Also see setTimeScaleDateFormats

getTimeScaleTypes

Method of JGantt

By this method you can retrieve the list of available time scale types. Possible values returned:

hour
shift
day
weekday
week
calendarweek
month
month centered short
month centered mediumquarter
quarter
relative hours
relative days
day+calendarweek

weekday+calendarweek

Declaration
java.lang.String[] getTimeScaleTypes ()

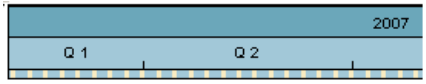
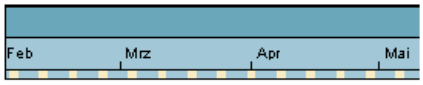
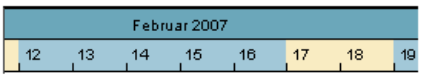
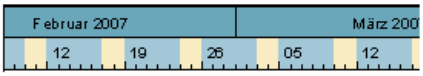
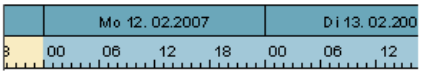
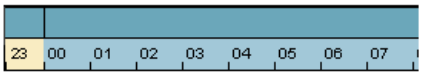
	Data Type	Description
Return Value	java.lang.String[]	Array of strings holding the list of available types.

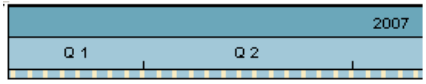
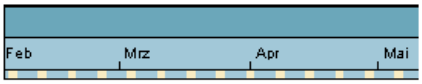
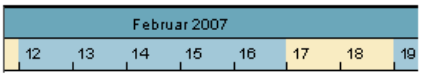
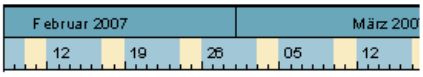
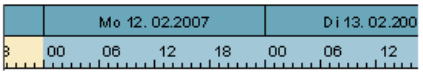
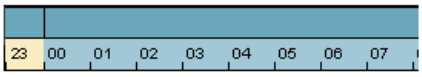
getUserAction

Method of JGantt

This method lets you generate the instance of a user action.

Declaration
de.netronic.common.beanbase.NeUserAction getUserAction (java.lang.Object name)

	Data Type	Description
Parameter		
name	java.lang.Object	Type of user action to be generated. the below user actions are available:
	Possible Values:	
	ACTION_COARSER_TIME_SCALE	This action can be used to reduce the resolution of the time scale. You can set the next coarser, that is, the next smaller (less mm per time unit) resolution. Starting from the type of an hours' scale, five increasingly coarser types are available:  Time scale displaying quarter units  Time scale displaying month units  Time scale displaying week units  Time scale displaying day units  Time scale displaying shifts  Time scale displaying hours
	ACTION_COLLAPSE_GROUP	This action collapses a group.
	ACTION_DELETE_NODES_-AND_LINKS	This action deletes all marked nodes and links.
	ACTION_EDIT_ENTITIES	This action starts the editor for the selected entities.
	ACTION_EXPAND_GROUP	This action expands a group.

	ACTION_FINER_TIME_SCALE	This action can be used to increase the resolution of the time scale. You can set the next finer, that is, the next larger (more millimeters per time unit) resolution. Starting from the type of a quarters' scale, five increasingly finer types are available:  Time scale displaying quarter units  Time scale displaying month units  Time scale displaying week units  Time scale displaying day units  Time scale displaying shifts  Time scale displaying hours
	ACTION_INDENT_NODE	This action indents a node of the hierarchy.
	ACTION_OUTDENT_NODE	This action outdents a node of the hierarchy.
	ACTION_SET_CREATE_MODE	This action sets the **create** mode
	ACTION_SET_SELECT_MODE	This action sets the **select** mode
Return Value	de.netronic.common.beanbase.NeUserAction	Generated instance of the user action.

getVertLineGridsEx

Method of JGantt

Returns the vertical line grids of the first timescale section.

Declaration
de.netronic.jgantt.JGVertLineGrids getVertLineGridsEx ()

	Data Type	Description
Return Value	de.netronic.jgantt.JGVertLineGrids	The vertical line grids of the first timescale section.

Example Code

```
jGantt1.getVertLineGridsEx(0).addGrid(gridWeeklyBeforeToday);
```

identifyEntities

Method of JGantt

This method lets you retrieve the entities of all objects located at the device coordinates xDV and yDV. This includes entities of for example nodes in the Gantt graph as well as of rows in the table.

Declaration
java.util.Iterator identifyEntities (int xDV, int yDV)

	Data Type	Description
Parameter		
xDV	int	X value of the coordinates to be passed.
yDV	int	Y value of the coordinates to be passed.
Return Value	java.util.Iterator	Iterator object returned that iterates over the entities found.

Also see identifyEntityAttribute
identifyLayouterGroup

identifyEntityAttribute

Method of JGantt

This method lets you identify the name of the attribute, that holds the data of the column located at the device coordinates xDV and yDV.

Declaration
java.lang.String identifyEntityAttribute (int xDV, int yDV)

	Data Type	Description
Parameter		
xDV	int	X value of the coordinates to be passed.
yDV	int	Y value of the coordinates to be passed.
Return Value	java.lang.String	Attribute name returned or an empty string, in case there is no column at the coordinates.

Also see identifyEntities
identifyLayouterGroup

identifyLabels

Method of JGantt

This method lets you retrieve all NePicture objects at the device coordinates XDV and yDV in the Gantt graph. This for example allows you to react to a mouse click on a symbol.

Declaration
java.util.Iterator identifyLabels (int xDV, int yDV)

	Data Type	Description
Parameter		
xDV	int	X value of the device coordinates to be passed.
yDV	int	Y value of the device coordinates to be passed.
Return Value	java.util.Iterator	Iterator object returned that iterates over the NePicture objects found.

Also see identifyEntities
identifyEntityAttribute
identifyLayers
identifyLayouterGroup

Example Code

```
Iterator iter = jGantt1.identifyLabels(e.getX(), e.getY());
```

identifyLayers

Method of JGantt

This method lets you retrieve all NeLayer objects at the device coordinates XDV and yDV in the Gantt graph. This for example allows you to react to a mouse click on a layer definition.

Declaration
java.util.Iterator identifyLayers (int xDV, int yDV)

	Data Type	Description
Parameter		
xDV	int	X value of the device coordinates to be passed.
yDV	int	Y value of the device coordinates to be passed.
Return Value	java.util.Iterator	Iterator object returned that iterates over the NeLayer objects found.

Also see identifyEntities
identifyEntityAttribute
identifyLabels
identifyLayouterGroup

Example Code

```
Iterator iter = jGantt1.identifyLayers(e.getX(), e.getY());
```

identifyLayouterGroup

Method of JGantt

This method lets you retrieve the group object at the device coordinates xDV and yDV.

Declaration
de.netronic.common.intface.NeILayouterGroup identifyLayouterGroup (int xDV, int yDV)

	Data Type	Description
Parameter		
xDV	int	X value of the coordinates to be passed.
yDV	int	Y value of the coordinates to be passed.
Return Value	de.netronic.common.intface.NeILayouterGroup	Group object returned

Also see identifyEntities
identifyEntityAttribute

identifyRow

Method of JGantt

Returns the index of the row in the Gantt diagram, in which the y coordinate of the mouse position is located. Please remember that the top row of the diagram is No. 0. If the mouse is located above or below the Gantt graph, -1 will be returned.

Declaration
long identifyRow (int xDV, int yDV)

	Data Type	Description
Parameter		
xDV	int	X value of the mouse coordinates to be passed.
yDV	int	Y value of the mouse coordinates to be passed.
Return Value	long	Index of the row

identifyTime

Method of JGantt

This method lets you retrieve the time of the time scale located at the device coordinates xDV and yDV.

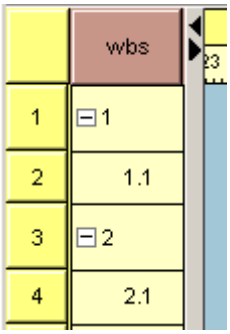
Declaration
long identifyTime (int xDV, int yDV)

	Data Type	Description
Parameter		
xDV	int	X value of the coordinates to be passed.
yDV	int	Y value of the coordinates to be passed.
Return Value	long	The numbers of milliseconds since 1.1.1970 are returned.

indentNode

Method of JGantt

This method lets you indent a node of the node hierarchy by one level towards the right.



Nodes on two different levels in the column wbs.

Declaration
void indentNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of the node to be indented.
Return Value	void	

Also see isIndentNodeAllowed
isOutdentNodeAllowed
outdentNode

isIndentNodeAllowed

Method of JGantt

This method retrieves, whether or not within the table hierarchy indenting the specified node is allowed.

Declaration
boolean isIndentNodeAllowed (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of the node which is to be checked.
Return Value	boolean	Returns **true**, if it is allowed to indent the node passed, otherwise **false** will be returned.

Also see indentNode
isOutdentNodeAllowed
outdentNode

isMarked

Method of JGantt

This method retrieves whether or not a node is marked.

Declaration
boolean isMarked (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity that is checked for marking.
Return Value	boolean	Will eturn **true** if the node is marked, otherwise **false** will returned.

isOutdentNodeAllowed

Method of JGantt

This method retrieves, whether or not within the table hierarchy outdenting the specified node is allowed.

Declaration
boolean isOutdentNodeAllowed (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of the node which is to be checked.
Return Value	boolean	Returns **true**, if it is allowed to outdent the node passed, otherwise **false** will returned.

Also see indentNode
isIndentNodeAllowed
outdentNode

iterateLayouterGroups

Method of JGantt

This method returns an iterator object on the node entities that exist in the Gantt graph.

Declaration
java.util.Iterator iterateLayouterGroups ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned.

iterateNodeEntitiesInRow

Method of JGantt

This method returns an iterator object on the node entities that exist in a row.

Declaration
java.util.Iterator iterateNodeEntitiesInRow (int rowIndex)

	Data Type	Description
Parameter		
rowIndex	int	Index of the row the node enitities of which are to be iterated. First row has index 0.
Return Value	java.util.Iterator	Iterator object returned.

Also see getRowIndexForNode

makeImage

Method of JGantt

For internal use only

Declaration
boolean makeImage (java.lang.String fileName, java.lang.String format)

	Data Type	Description
Parameter		
fileName	java.lang.String	
format	java.lang.String	
Return Value	boolean	

Also see makeImage

makeImage

Method of JGantt

For internal use only

Declaration
boolean makeImage (java.io.OutputStream outputStream, java.lang.String format)

	Data Type	Description
Parameter		
outputStream	java.io.OutputStream	
format	java.lang.String	
Return Value	boolean	

Also see makeImage

markAll

Method of JGantt

This method lets you mark or unmark all nodes or links present in the Gantt diagram. The marking mode and the type of objects (nodes or links) need to be specified via the parameters.

Declaration
void markAll (int kindOfEntity, int theMode)

	Data Type	Description
Parameter		
kindOfEntity	int	Type of entity
	Possible Values:	
	LINK	Entity of the type "link"
	NODE	Entity of the type "node"
theMode	int	Marking mode
	Possible Values:	
	MARK	An unmarked object will be marked
	MARK_AND_SCROLL_-ENTITY_TO_CENTER	An unmarked node will be marked. In addition, the chart will scroll to make the node become visible in the center of the screen.
	MARK_AND_SCROLL_TO_ENTITY	An unmarked node will be marked. If it is located in a chart section not visible on the screen, the chart will scroll to make the marked node become visible.
	TOGGLE_MARK	If the object was unmarked, it will be marked. If it was marked, it will be unmarked.
	TOGGLE_MARK_AND_-SCROLL_ENTITY_TO_CENTER	If the node was unmarked, it will be marked. If it was marked, it will be unmarked. In addition, the chart will scroll to make the node become visible in the center of the screen.
	TOGGLE_MARK_AND_-SCROLL_TO_ENTITY	If the node was unmarked, it will be marked. If it was marked, it will be unmarked. In addition, if it is located in a chart section not visible on the screen, the chart will scroll to make the node become visible.
	UNMARK	A marked object will be unmarked
Return Value	void	

markEntity

Method of JGantt

This method lets you mark or unmark a node or a link. The entity of the node or link and the marking mode need to be specified via the parameters.

Declaration
void markEntity (de.netronic.common.intface.NeIEntity theEntity, int theMode)

	Data Type	Description
Parameter		
theEntity	de.netronic.common.intface.NeIEntity	Entity of the node or link, to which the marking treatment applies
theMode	int	Marking mode
	Possible Values:	
	MARK	An unmarked object will be marked
	MARK_AND_SCROLL_-ENTITY_TO_CENTER	An unmarked node will be marked. In addition, the chart will scroll to make the node become visible in the center of the screen.
	MARK_AND_SCROLL_TO_ENTITY	An unmarked node will be marked. If it is located in a chart section not visible on the screen, the chart will scroll to make the marked node become visible.
	TOGGLE_MARK	If the object was unmarked, it will be marked. If it was marked, it will be unmarked.
	TOGGLE_MARK_AND_-SCROLL_ENTITY_TO_CENTER	If the node was unmarked, it will be marked. If it was marked, it will be unmarked. In addition, the chart will scroll to make the node become visible in the center of the screen.
	TOGGLE_MARK_AND_-SCROLL_TO_ENTITY	If the node was unmarked, it will be marked. If it was marked, it will be unmarked. In addition, if it is located in a chart section not visible on the screen, the chart will scroll to make the node become visible.
	UNMARK	A marked object will be unmarked
Return Value	void	

mouseClicked

Method of JGantt

For internal use only.

Declaration
void mouseClicked (java.awt.event.MouseEvent event)

	Data Type	Description
Parameter		
event	java.awt.event.MouseEvent	For internal use only.
Return Value	void	

mouseDragged

Method of JGantt

For internal use only.

Declaration
void mouseDragged (java.awt.event.MouseEvent event)

	Data Type	Description
Parameter		
event	java.awt.event.MouseEvent	For internal use only.
Return Value	void	

mouseEntered

Method of JGantt

For internal use only

Declaration
void mouseEntered (java.awt.event.MouseEvent event)

	Data Type	Description
Parameter		
event	java.awt.event.MouseEvent	For internal use only.
Return Value	void	

mouseExited

Method of JGantt

For internal use only

Declaration
void mouseExited (java.awt.event.MouseEvent event)

	Data Type	Description
Parameter		
event	java.awt.event.MouseEvent	For internal use only.
Return Value	void	

mouseMoved

Method of JGantt

For internal use only

Declaration
void mouseMoved (java.awt.event.MouseEvent event)

	Data Type	Description
Parameter		
event	java.awt.event.MouseEvent	For internal use only.
Return Value	void	

mousePressed

Method of JGantt

For internal use only

Declaration
void mousePressed (java.awt.event.MouseEvent event)

	Data Type	Description
Parameter		
event	java.awt.event.MouseEvent	For internal use only.
Return Value	void	

mouseReleased

Method of JGantt

For internal use only

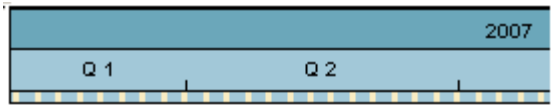
Declaration
void mouseReleased (java.awt.event.MouseEvent event)

	Data Type	Description
Parameter		
event	java.awt.event.MouseEvent	For internal use only.
Return Value	void	

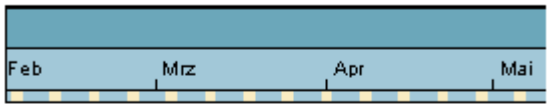
nextCoarserTimeScaleSectionType

Method of JGantt

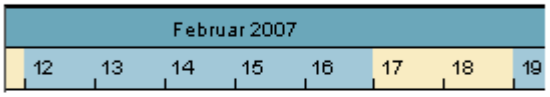
This method can be used to reduce the resolution of a time scale section. You can set the next coarser, that is, the next smaller (less mm per time unit) resolution. Starting from the type of an hours' scale, five increasingly coarser types are available:



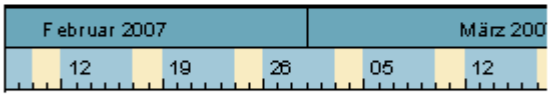
Time scale displaying quarter units



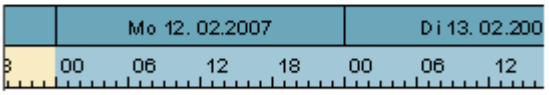
Time scale displaying month units



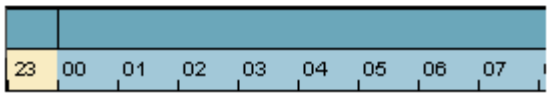
Time scale displaying week units



Time scale displaying day units



Time scale displaying shifts



Time scale displaying hours

Declaration
void nextCoarserTimeScaleSectionType (int sectionIndex)

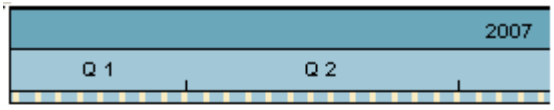
	Data Type	Description
Parameter		
sectionIndex	int	Zero-based index of the time scale section
Return Value	void	

Also see nextCoarserTimeScaleType
nextFinerTimeScaleSectionType
nextFinerTimeScaleType

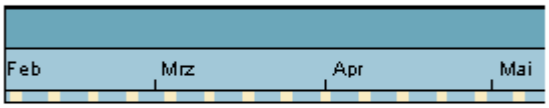
nextCoarserTimeScaleType

Method of JGantt

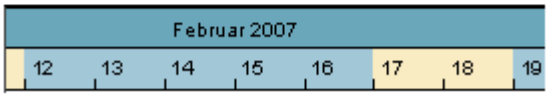
This method can be used to reduce the resolution of the time scale. You can set the next coarser, that is, the next smaller (less mm per time unit) resolution. Starting from the type of an hours' scale, five increasingly coarser types are available:



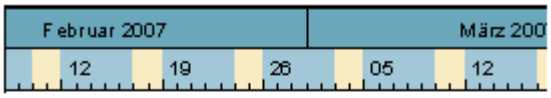
Time scale displaying quarter units



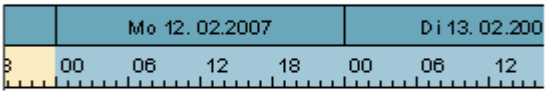
Time scale displaying month units



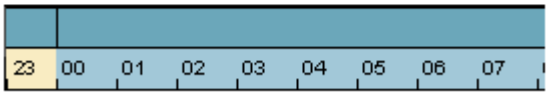
Time scale displaying week units



Time scale displaying day units



Time scale displaying shifts



Time scale displaying hours

If the time scale consists of several sections, then a call of this method affects only the first section.

Declaration
void nextCoarserTimeScaleType ()

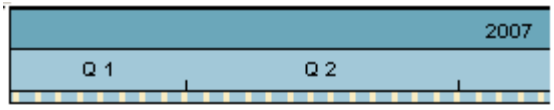
	Data Type	Description
Return Value	void	

Also see nextCoarserTimeScaleSectionType
nextFinerTimeScaleSectionType
nextFinerTimeScaleType

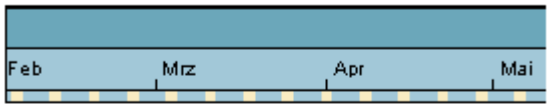
nextFinerTimeScaleSectionType

Method of JGantt

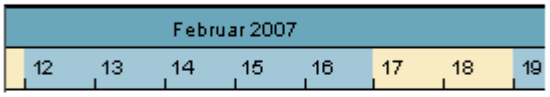
This method can be used to increase the resolution of a time scale section. You can set the next finer, that is, the next larger (more millimeters per time unit) resolution. Starting from the type of a quarters' scale, five increasingly finer types are available:



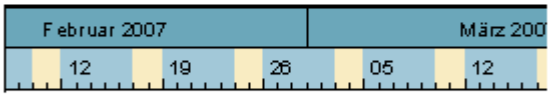
Time scale displaying quarter units



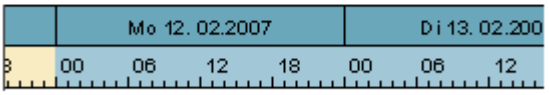
Time scale displaying month units



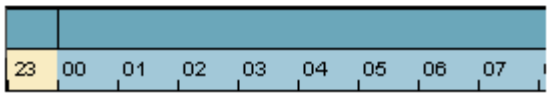
Time scale displaying week units



Time scale displaying day units



Time scale displaying shifts



Time scale displaying hours

Declaration
void nextFinerTimeScaleSectionType (int sectionIndex)

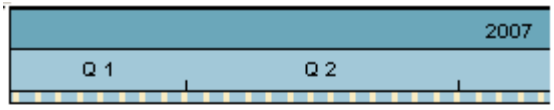
	Data Type	Description
Parameter		
sectionIndex	int	Zero-based index of the time scale section
Return Value	void	

Also see nextCoarserTimeScaleSectionType
nextCoarserTimeScaleType
nextFinerTimeScaleType

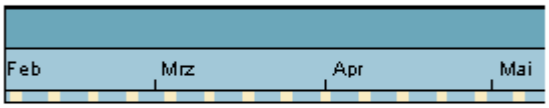
nextFinerTimeScaleType

Method of JGantt

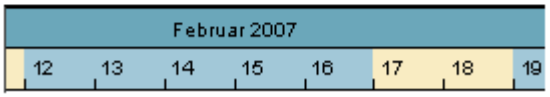
This method can be used to increase the resolution of the time scale. You can set the next finer, that is, the next larger (more millimeters per time unit) resolution. Starting from the type of a quarters' scale, five increasingly finer types are available:



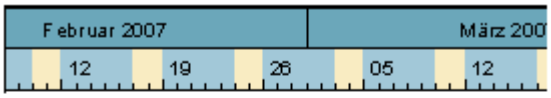
Time scale displaying quarter units



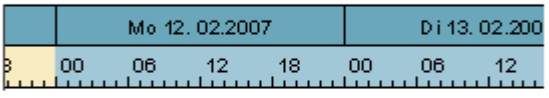
Time scale displaying month units



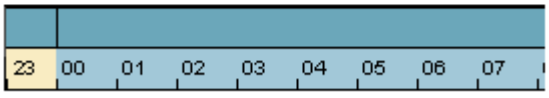
Time scale displaying week units



Time scale displaying day units



Time scale displaying shifts



Time scale displaying hours

If the time scale consists of several sections, then a call of this method affects only the first section.

Declaration
void nextFinerTimeScaleType ()

	Data Type	Description
Return Value	void	

Also see nextCoarserTimeScaleSectionType
nextCoarserTimeScaleType
nextFinerTimeScaleSectionType

openPageLayoutDialog

Method of JGantt

This method lets you open the page layout dialog.

Declaration
void openPageLayoutDialog (java.awt.Frame frame)

	Data Type	Description
Parameter		
frame	java.awt.Frame	Window to display the dialog.
Return Value	void	

openPrintDialog

Method of JGantt

This method lets you open the print dialog.

Declaration
void openPrintDialog ()

	Data Type	Description
Return Value	void	

openPrintPreview

Method of JGantt

This method lets you open the print preview.

Declaration
void openPrintPreview ()

	Data Type	Description
Return Value	void	

out

Method of JGantt

For internal use only

Declaration
java.io.ObjectOutput out (java.io.ObjectInput in)

	Data Type	Description
Parameter		
in	java.io.ObjectInput	For internal use only
Return Value	java.io.ObjectOutput	

outdentNode

Method of JGantt

This property lets you outdent a node of the node hierarchy by one level towards the left.

Declaration
void outdentNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of the node to be outdented.
Return Value	void	

Also see indentNode

paint

Method of JGantt

For internal use only

Declaration
void paint (java.awt.Graphics g)

	Data Type	Description
Parameter		
g	java.awt.Graphics	For internal use only
Return Value	void	

propertyChange

Method of JGantt

For internal use only

Declaration
void propertyChange (java.beans.PropertyChangeEvent evt)

	Data Type	Description
Parameter		
evt	java.beans.PropertyChangeEvent	For internal use only
Return Value	void	

readExternal

Method of JGantt

For internal use only

Declaration
void readExternal (java.io.ObjectInput in)

	Data Type	Description
Parameter		
in	java.io.ObjectInput	For internal use only
Return Value	void	

removeGanttDateLine

Method of JGantt

This method removes a date line from the Gantt graph.

Declaration
void removeGanttDateLine (de.netronic.common.beanbase.NeDateLine dateLine)

	Data Type	Description
Parameter		
dateLine	de.netronic.common.beanbase.-NeDateLine	Date line to be removed
Return Value	void	

Also see addGanttDateLine

removeLinkChangeListener

Method of JGantt

Removes an existing listener for link change events in the JGantt object.

Declaration
void removeLinkChangeListener (de.netronic.common.event.NeObjectChangeListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeObjectChangeListener	Listener to be removed.
Return Value	void	

Also see addLinkChangeListener

removeNodeChangeListener

Method of JGantt

Removes an existing listener for node change events in the JGantt object.

Declaration
void removeNodeChangeListener (de.netronic.common.event.NeObjectChangeListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeObjectChangeListener	Listener to be removed.
Return Value	void	

Also see addNodeChangeListener

removePropertyChangeListener

Method of JGantt

Removes an existing listener for events on changes of the JGantt object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

removeTimeScaleSection

Method of JGantt

This method lets you remove a time scale section given by an index.

Declaration
void removeTimeScaleSection (int sectionIndex, int replacementMode)

	Data Type	Description
Parameter		
sectionIndex	int	Zero-based index of the time scale section which should be removed
replacementMode	int	This parameter describes how the time scale should behave after a section has been removed from the scale. The settings - e.g. the resolution or the section type - for the area of the old section are inherited from one of the adjacent sections.
	Possible Values:	
	TIMESCALE_SECTION_-REPLACE_BY_NEXT	When removing the time scale section it will be replaced by the next section.
	TIMESCALE_SECTION_-REPLACE_BY_PREVIOUS	When removing the time scale section it will be replaced by the previous section.
Return Value	void	

Also see addTimeScaleSection

Example Code

```
jGantt.removeTimeScaleSection(1, jGantt.TIMESCALE_SECTION_REPLACE_BY_NEXT);
```

selectedEntitiesIterator

Method of JGantt

This method lets you retrieve the entities of all objects selected in the diagram.

Declaration
java.util.Iterator selectedEntitiesIterator ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned, that iterates over the entities found.

setDiagramAnnotation

Method of JGantt

This method lets you display or hide a diagram annotation in the header or footer, respectively.

Declaration
void setDiagramAnnotation (int position, de.netronic.jgantt.JGDiagramAnnotation diagramAnnotation)

	Data Type	Description
Parameter		
position	int	Position of the diagram annotation to be set
	Possible Values:	
	DIAGRAM_FOOTER	Footer of the diagram
	DIAGRAM_HEADER	Header of the diagram
diagramAnnotation	de.netronic.jgantt.JGDiagramAnnotation	The diagram annotation to be set
Return Value	void	Diagram annotation (header or footer) to be displayed or hidden.

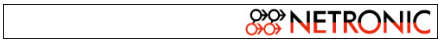
Also see getDiagramAnnotation

setDiagramTitlePicture

Method of JGantt

Sets a picture object to be displayed in the title line of the diagram. The picture may be a logo or anything similar.

Declaration
void setDiagramTitlePicture (de.netronic.common.beanbase.NePicture picture, int position)

	Data Type	Description
Parameter		
picture	de.netronic.common.beanbase.-NePicture	Picture object to be displayed by the diagram title.
position	int	Horizotal alignment of the picture object
	Possible Values:	
	NE_PICTURE_CENTER	The picture object is placed in the center of the title line. 
	NE_PICTURE_LEFT	The picture object is placed left-aligned. 
	NE_PICTURE_RIGHT	The picture object is placed right-aligned. 
Return Value	void	

setDocumentBase

Method of JGantt

Sets the path of the current directory.

Declaration
void setDocumentBase (java.lang.String stringValue)

	Data Type	Description
Parameter		
stringValue	java.lang.String	Character string that holds the path
Return Value	void	

setTimeScaleDateFormats

Method of JGantt

Sets for a given timescale type and a ribbon number user-defined date formats.

Declaration
void setTimeScaleDateFormats (java.lang.String timeScaleType, int ribbonIndex, java.text.DateFormat[] dateFormats)

	Data Type	Description
Parameter		
timeScaleType	java.lang.String	Type of timescale
ribbonIndex	int	Ribbon index, starts with 0 beginning at the topmost ribbon.
dateFormats	java.text.DateFormat[]	Array of date formats. The formats should be ordered in the array by decreasing length. JGantt tests in sequence the formats and selects the first one which produces labels of matching length.
Return Value	void	

Also see getTimeScaleDateFormats

Example Code

```
jGantt1.setTimeScaleDateFormats(JGantt.TIMESCALE_TYPE_MONTH,
                        1,
                        new DateFormat[]{new NeSimpleDateFormat("MMMM yyyy"),
                                new NeSimpleDateFormat("MMMM"),
                                new NeSimpleDateFormat("MMM"),
                                new NeSimpleDateFormat("MM")});
```

synchronizeDiagramTableWidthPercent

Method of JGantt

Synchronizes several JGantt instances in a way to flush-fit the dividers between all tables and all Gantt diagrams. Moving one divider interactivly will cause the other dividers to move synchronously.

The widest table will define the maximum visible area for all tables. The width of the last column in a table may be compressed or stretched to some degree if necessary.

Even if column widths are modified interactivly (or by API calls), the synchronization will remain. Column widths that were generated this way cannot fall below their pre-defined value which avoids the last column of a table to decrease to disappearance when column widths are modified in other tables.

Synchronization also works with zooming, provided that
1. the diagrams have the same zooming factors
2. the timescales have identical resolutions.

Declaration
void synchronizeDiagramTableWidthPercent (de.netronic.JGantt.JGantt[] jGantts)

	Data Type	Description
Parameter		
jGantts	de.netronic.JGantt.JGantt[]	Array containing the JGantt instances to be synchronized with the current JGantt object.
Return Value	void	

Example Code

```
jGantt1.synchronizeDiagramTableWidthPercent(new JGantt[] {jGantt2, jGantt3});
```

writeExternal

Method of JGantt

For internal use only

Declaration
void writeExternal (java.io.ObjectOutput out)

	Data Type	Description
Parameter		
out	java.io.ObjectOutput	For internal use only
Return Value	void	

6.2 JGanttSynchronizerPanel

Belongs to JGantt

Package name **de.netronic.jgantt**
Extends **javax.swing.JPanel**

The JGanttSynchronizerPanel class places two instances of VARCHART JGantt on top of each other and synchronizes their time-related sections and their tables. The top instance is the dependent or slave instance. It has a time scale, while the bottom instance is the dominating one which doesn't have a time scale. Both of the JGantt instances access the same AppData. The nodes available are distributed to the instances by a separation filter in a disjunctive way. If the separation filter was not set, both instances will have access to all entities of the AppData. Synchronization is started by the method **synchronize**, with the property settings of the master instance initially copied to the slave instance. You can set a maximum height to the slave instance.

Properties to Manage the Synchronizer Panel

SeparationFilter	When complying with the filter conditions, the group or the node will be assigned to the slave instance, otherwise to the master instance.
SeparationLineColor	Color of the separation line between a master and a slave instance.
SeparationLineWidth	Width of the separation line between a master and a slave instance (pixels).
SlaveMaximumHeight	Defines the maximum height of the slave instance in pixels.

Constructors of the Class

JGanttSynchronizerPanel

Constructor of JGanttSynchronizerPanel

This constructor allows to create a JGanttSynchronizerPanel with a license string and a separation filter.

Declaration
JGanttSynchronizerPanel (java.lang.String licenceString, de.netronic.common.intface.NeIFilter filter)

Parameter	Data Type	Description
licenceString	java.lang.String	The license string to be used for the master and slave instances
filter	de.netronic.common.intface.NeIFilter	The filter specified here will be the separation filter of the JGanttSynchronizerPanel.

JGanttSynchronizerPanel

Constructor of JGanttSynchronizerPanel

This constructor allows to create a JGanttSynchronizerPanel with a separation filter.

Declaration
JGanttSynchronizerPanel (de.netronic.common.intface.NeIFilter filter)

Parameter	Data Type	Description
filter	de.netronic.common.intface.NeIFilter	The filter specified here will be the separation filter of the JGanttSynchronizerPanel.

JGanttSynchronizerPanel

Constructor of JGanttSynchronizerPanel

This constructor allows to create a JGanttSynchronizerPanel with a JGantt master instance and a separation filter.

Declaration
JGanttSynchronizerPanel (de.netronic.jgantt.JGantt master, de.netronic.common.intface.NeIFilter filter)

Parameter	Data Type	Description
master	de.netronic.jgantt.JGantt	The JGantt instance specified here is going to be the master instance in the JGanttSynchronizerPanel.
filter	de.netronic.common.intface.NeIFilter	The filter specified here will be the separation filter of the JGanttSynchronizerPanel.

JGanttSynchronizerPanel

Constructor of JGanttSynchronizerPanel

For internal use only.

Declaration
JGanttSynchronizerPanel ()

Properties of the Class

PrintMode

Property of JGanttSynchronizerPanel

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**JG_MASTER_ONLY**

With this property you can specify whether only the Master, only the slave or both master and slave will be printed.

Possible Values	Description
JG_MASTER_AND_SLAVE	Master and Slave are printed.
JG_MASTER_ONLY	Only the master is printed.
JG_SLAVE_ONLY	Only the slave is printed.

Accessing Methods
void setPrintMode (int newValue)
int getPrintMode ()

SeparationFilter

Property of JGanttSynchronizerPanel

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

When complying with the filter conditions, the group or the node will be assigned to the slave instance (upper instance), otherwise to the master instance (lower instance). If the filter equals null, all groups and nodes will be displayed in master and slave instance.

Accessing Methods
void setSeparationFilter (de.netronic.common.intface.NeIFilter newValue)
de.netronic.common.intface.NeIFilter getSeparationFilter ()

SeparationLineColor

Property of JGanttSynchronizerPanel

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.gray**

This property lets you set or retrieve the color of the separation line between a master and a slave instance.

Accessing Methods
void setSeparationLineColor (java.awt.Color newValue)
java.awt.Color getSeparationLineColor ()

Also see SeparationLineWidth

Example Code

```
jGSynchro.setSeparationLineColor(Color.blue);
```

SeparationLineWidth

Property of JGanttSynchronizerPanel

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**3**

This property lets you set or retrieve the width of the separation line between a master and a slave insatnce. Unit: pixels

Accessing Methods
void setSeparationLineWidth (int newValue)
int getSeparationLineWidth ()

Also see SeparationLineColor

Example Code

```
jGSynchro.setSeparationLineWidth(5);
```

SlaveMaximumHeight

Property of JGanttSynchronizerPanel

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**-1**

This property allows to define the maximum height of the slave instance in pixels, including the time scale and, if present, also the diagramControlBar. If you set the value to **-1**, the slave instance will be just as high as required.

Accessing Methods
void setSlaveMaximumHeight (int newValue)
int getSlaveMaximumHeight ()

Example Code

```
jGSynchro.setSlaveMaximumHeight(200);
```

Methods of the Class

synchronize

Method of JGanttSynchronizerPanel

By this method the property values of the master instance will be to the slave instance. After this you may want to add some individual settings to the upper or lower instance.

Declaration
void synchronize ()

	Data Type	Description
Return Value	void	

Example Code

```
((JGanttSynchronizerPanel)jGantt1).synchronize();
```

6.3 JGColorScheme

Belongs to JGantt

Package name	**de.netronic.jgantt**
Extends	**java.lang.Object**
Implements	**java.io.Serializable**

This class lets you handle a color scheme.

Properties to Handle Color Schemes

Name	Name of the color scheme

Properties to Handle Colors

AlternateColor	The alternate color is one out of four background colors.

AlternateTextColor	The alternate text color is one out of five foreground colors.
LineColor	The line color is one out of five foreground colors.
MainColor	The main color is one out of four background colors.
MainTextColor	The main text color is one out of five foreground colors.
ShadedAlternateColor	The shaded alternate color is one out of four background colors.
ShadedAlternateTextColor	The shaded alternate text color is one out of five foreground colors.
ShadedColor	The shaded color is one out of four background colors.
ShadedTextColor	The shaded text color is one out of five foreground colors.
TextColor	Deprecated. Not recommended to use.

Methods to Handle Color Schemes

addPropertyChangeListener(...)	Adds a listener for change events in the color scheme object.
getInstance(...)	Generates a color scheme and names it by the string passed.
getNames()	Retrieves the names of all color schemes.
removePropertyChangeListener(...)	Removes a listener for change events in the color scheme object.
toString()	Returns a string representation of the object.

Methods to Handle Colors

getColor(...)	Returns one out of the nine colors of the color scheme
setColor(...)	Sets one out of the nine colors of the color scheme.

Constructors of the Class

By the constructors you can generate a color scheme and define none, some or all of its colors.

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme from a template.

Declaration
JGColorScheme (JGColorScheme sourceScheme)

Parameter	Data Type	Description
sourceScheme	JGColorScheme	Color scheme that serves as the template

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has a name and some colors. The colors not set here can be assigned later on via properties.

Declaration
JGColorScheme (java.lang.String name, java.awt.Color main, java.awt.Color alternate, java.awt.Color shaded, java.awt.Color shadedAlternate, java.awt.Color line, java.awt.Color text)

Parameter	Data Type	Description
name	java.lang.String	Character string holding the name of the color scheme.
main	java.awt.Color	Color to be allocated to the scheme color "mainColor".
alternate	java.awt.Color	Color to be allocated to the scheme color "alternateColor".
shaded	java.awt.Color	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateColor".
line	java.awt.Color	Color to be allocated to the scheme color "lineColor".
text	java.awt.Color	Color to be allocated to the scheme color "textColor".

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has a name and some colors. The colors not set here can be assigned later on via properties.

Declaration

JGColorScheme (java.lang.String name, java.awt.Color main, java.awt.Color alternate, java.awt.Color shaded, java.awt.Color shadedAlternate)

Parameter	Data Type	Description
name	java.lang.String	Character string holding the name of the color scheme.
main	java.awt.Color	Color to be allocated to the scheme color "mainColor".
alternate	java.awt.Color	Color to be allocated to the scheme color "alternateColor".
shaded	java.awt.Color	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateColor".

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has some colors but no name. The name and the colors not set here can be assigned later on via properties.

Declaration
JGColorScheme (java.awt.Color main, java.awt.Color alternate, java.awt.Color shaded, java.awt.Color shadedAlternate)

Parameter	Data Type	Description
main	java.awt.Color	Color to be allocated to the scheme color "mainColor".
alternate	java.awt.Color	Color to be allocated to the scheme color "alternateColor".
shaded	java.awt.Color	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateColor".

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has some colors but no name. The name and the colors not set here can be assigned later on via properties.

Declaration
JGColorScheme (java.awt.Color main, java.awt.Color alternate, java.awt.Color shaded, java.awt.Color shadedAlternate, java.awt.Color line, java.awt.Color text)

Parameter	Data Type	Description
main	java.awt.Color	Color to be allocated to the scheme color "mainColor".
alternate	java.awt.Color	Color to be allocated to the scheme color "alternateColor".
shaded	java.awt.Color	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateColor".
line	java.awt.Color	Color to be allocated to the scheme color "lineColor".
text	java.awt.Color	Color to be allocated to the scheme color "textColor".

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has all colors (as integer values) but no name. The term by which the integer value is composed is the one used by Java for combined RGB components. The name can be assigned later on via the corresponding property.

Declaration
JGColorScheme (int main, int alternate, int shaded, int shadedAlternate, int line, int text)

Parameter	Data Type	Description
main	int	Color to be allocated to the scheme color "mainColor".
alternate	int	Color to be allocated to the scheme color "alternateColor".
shaded	int	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	int	Color to be allocated to the scheme color "shadedAlternateColor".
line	int	Color to be allocated to the scheme color "lineColor".
text	int	Color to be allocated to the scheme color "textColor".

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has all colors (as integer values) but no name. The term by which the integer value is composed is the one used by Java for combined RGB components. The name can be assigned later on via the corresponding property.

Declaration
JGColorScheme (int main, int alternate, int shaded, int shadedAlternate, int line, int text, int alternateText, int shadedText, int shadedAlternateText)

Parameter	Data Type	Description
main	int	Color to be allocated to the scheme color "mainColor".
alternate	int	Color to be allocated to the scheme color "alternateColor".
shaded	int	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	int	Color to be allocated to the scheme color "shadedAlternateColor".
line	int	Color to be allocated to the scheme color "lineColor".
text	int	Color to be allocated to the scheme color "textColor".
alternateText	int	Color to be allocated to the scheme color "alternateTextColor".
shadedText	int	Color to be allocated to the scheme color "shadedTextColor".
shadedAlternateText	int	Color to be allocated to the scheme color "shadedAlternateTextColor".

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has all colors but no name. The name can be assigned later on via the corresponding property.

Declaration
JGColorScheme (java.awt.Color main, java.awt.Color alternate, java.awt.Color shaded, java.awt.Color shadedAlternate, java.awt.Color line, java.awt.Color text, java.awt.Color alternateText, java.awt.Color shadedText, java.awt.Color shadedAlternateText)

Parameter	Data Type	Description
main	java.awt.Color	Color to be allocated to the scheme color "mainColor".
alternate	java.awt.Color	Color to be allocated to the scheme color "alternateColor".
shaded	java.awt.Color	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateColor".
line	java.awt.Color	Color to be allocated to the scheme color "lineColor".
text	java.awt.Color	Color to be allocated to the scheme color "textColor".
alternateText	java.awt.Color	Color to be allocated to the scheme color "alternateTextColor".
shadedText	java.awt.Color	Color to be allocated to the scheme color "shadedTextColor".
shadedAlternateText	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateTextColor".

JGColorScheme

Constructor of JGColorScheme

Generates a color scheme that has a name and all colors.

Declaration
JGColorScheme (java.lang.String name, java.awt.Color main, java.awt.Color alternate, java.awt.Color shaded, java.awt.Color shadedAlternate, java.awt.Color line, java.awt.Color text, java.awt.Color alternateText, java.awt.Color shadedText, java.awt.Color shadedAlternateText)

Parameter	Data Type	Description
name	java.lang.String	Character string holding the name of the color scheme.
main	java.awt.Color	Color to be allocated to the scheme color "mainColor".
alternate	java.awt.Color	Color to be allocated to the scheme color "alternateColor".
shaded	java.awt.Color	Color to be allocated to the scheme color "shadedColor".
shadedAlternate	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateColor".
line	java.awt.Color	Color to be allocated to the scheme color "lineColor".
text	java.awt.Color	Color to be allocated to the scheme color "textColor".
alternateText	java.awt.Color	Color to be allocated to the scheme color "alternateTextColor".
shadedText	java.awt.Color	Color to be allocated to the scheme color "shadedTextColor".
shadedAlternateText	java.awt.Color	Color to be allocated to the scheme color "shadedAlternateTextColor".

JGColorScheme

Constructor of JGColorScheme

Generates an empty color scheme without name and colors. The name and the colors you can set later on via properties.

Declaration
JGColorScheme ()

Properties of the Class

AlternateColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the alternate color, one out of four background colors of the color scheme. It dyes the background of certain elements in an object that the color scheme was assigned to. The background areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setAlternateColor (java.awt.Color newValue)
java.awt.Color getAlternateColor ()

Also see AlternateTextColor

AlternateTextColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the alternate text color, one out of five foreground colors of the color scheme. It dyes the font of text strings in areas that display the alternate color. The areas affected are listed for each object with the following properties of the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setAlternateTextColor (java.awt.Color newValue)
java.awt.Color getAlternateTextColor ()

Also see AlternateColor

LineColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the line color, one out of five foreground colors. It dyes the lines of the object that it is assigned to.

Accessing Methods
void setLineColor (java.awt.Color newValue)
java.awt.Color getLineColor ()

Also see AlternateColor
MainColor
ShadedAlternateColor
ShadedColor
TextColor

MainColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the main color, one out of four background colors of the color scheme. It dyes the background of certain elements in an object, that the color scheme was assigned to. The background areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setMainColor (java.awt.Color newValue)
java.awt.Color getMainColor ()

Also see MainTextColor

MainTextColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the main text color, one out of five foreground colors of the color scheme. It dyes the font of text strings in areas that display the main color. The areas affected are listed for each object with the following properties of the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setMainTextColor (java.awt.Color newValue)
java.awt.Color getMainTextColor ()

Also see MainColor

Name

Property of JGColorScheme

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the name of the color scheme.

Accessing Methods
void setName (java.lang.String newValue)
java.lang.String getName ()

ShadedAlternateColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the shaded alternate color, one out of four background colors of the color scheme. It dyes the background of certain elements in an object, that the color scheme was assigned to. The background areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setShadedAlternateColor (java.awt.Color newValue)
java.awt.Color getShadedAlternateColor ()

Also see ShadedAlternateTextColor

ShadedAlternateTextColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the shaded alternate text color, one out of five foreground colors of the color scheme. It dyes the font of text strings in areas that display the shaded alternate color. The areas affected are listed for each object with the following properties of the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setShadedAlternateTextColor (java.awt.Color newValue)
java.awt.Color getShadedAlternateTextColor ()

Also see ShadedAlternateColor

ShadedColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the shaded color, one out of four background colors of the color scheme. It dyes the background of certain elements in an object, that the color scheme was assigned to. The background areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setShadedColor (java.awt.Color newValue)
java.awt.Color getShadedColor ()

Also see ShadedTextColor

ShadedTextColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the shaded text color, one out of five foreground colors of the color scheme. It dyes the font of text strings in areas that display the shaded color. The areas affected are listed for each object with the following properties of the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.

Accessing Methods
void setShadedTextColor (java.awt.Color newValue)
java.awt.Color getShadedTextColor ()

Also see ShadedColor

TextColor

Property of JGColorScheme

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Deprecated. Not recommended to use.

Accessing Methods
void setTextColor (java.awt.Color newValue)
java.awt.Color getTextColor ()

Methods of the Class

addPropertyChangeListener

Method of JGColorScheme

This method adds a listener for change events in the color scheme object. The listener will be informed each time a property of the color scheme has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

getColor

Method of JGColorScheme

By this method you can retrieve one out of the nine colors of the color scheme.

Declaration
java.awt.Color getColor (int colorKey)

	Data Type	Description
Parameter		
colorKey	int	Color key from the color scheme, the color of which is to be retrieved. The color key can be chosen from the pre-defined constants:
	Possible Values:	
	ALTERNATE_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	ALTERNATE_TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the alternate color as their background color.
	LINE_COLOR	Defines the second out of two available foreground colors of a color scheme. It determines the line color in the object that the color scheme was assigned to.
	MAIN_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	MAIN_TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the main color as their background color.

	SHADED_ALTERNATE_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	SHADED_ALTERNATE_-TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the shaded alternate color as their background color.
	SHADED_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	SHADED_TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the shaded color as their background color.
Return Value	java.awt.Color	

getInstance

Method of JGColorScheme

This method generates an instance of an existing color scheme named by the string passed.

Declaration
static JGColorScheme getInstance (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	String passing the name of the instance. The names listed below represent the color schemes available. The pre-defined RGB values of the scheme colors mainColor, shadedColor, alternateColor and shadedAlternateColor are listed for each scheme. The pre-definition of the remaining scheme colors mainTextColor, shadedTextColor, alternateTextColor, shadedAlternateTextColor and lineColor is black for nearly all schemes, therefore they are not listed except for the exceptions.
	Possible Values:	
	BABY	Color scheme using the RGB values: mainColor(204, 153, 255) shadedColor(204, 204, 255) alternateColor(255, 204, 255) shadedAlternateColor(255, 255, 255) all other colors: (0, 0, 0)
	BANANA	Color scheme using the RGB values: mainColor(255, 255, 128) shadedColor(200, 150, 136) alternateColor(255, 255, 198) shadedAlternateColor(242, 209, 152) all other colors: (0, 0, 0)
	BEACH	Color scheme using the RGB values: mainColor(96, 83, 224) shadedColor(255, 206, 90) alternateColor(163, 159, 211) shadedAlternateColor(255, 222, 140) all other colors: (0, 0, 0)
	BLOOD	Color scheme using the RGB values: mainColor(179, 41, 0) shadedColor(0, 37, 132) alternateColorColor(116, 30, 30) shadedAlternate(44, 71, 113) mainTextColor(255, 255, 255) shadedTextColor(255, 255, 255) all other colors: (0, 0, 0)
	BLUES	Color scheme using the RGB values: mainColor(107, 167, 186) shadedColor(139, 170, 255) alternateColor(162, 200, 216) shadedAlternateColor(183, 220, 255) all other colors: (0, 0, 0)

BOUNTY	Color scheme using the RGB values: mainColor(128, 255, 128) shadedColor(155, 230, 251) alternateColor(224, 226, 228) shadedAlternateColor(225, 255, 225) all other colors: (0, 0, 0)
CONTRAST	Color scheme using the RGB values: mainColor(255, 0, 0) shadedColor(0, 0, 255) alternateColor(255, 255, 0) shadedAlternateColor(0, 255, 0) all other colors: (0, 0, 0)
DEVIL	Color scheme using the RGB values: mainColor(248, 26, 6) shadedColor(30, 6, 248) alternateColor(228, 171, 170) shadedAlternateColor(196, 188, 219) all other colors: (0, 0, 0)
GRAPE	Color scheme using the RGB values: mainColor(167, 230, 87) shadedColor(216, 173, 69) alternateColor(172, 168, 225) shadedAlternateColor(202, 184, 143) all other colors: (0, 0, 0)
GREENAPPLE	Color scheme using the RGB values: mainColor(111, 184, 1) shadedColor(244, 1, 1) alternateColor(162, 212, 89) shadedAlternateColor(244, 131, 99) all other colors: (0, 0, 0)
JUNGLE	Color scheme using the RGB values: mainColor(234, 23, 0) shadedColor(7, 132,17) alternateColor(119, 204, 40) shadedAlternateColor(229, 217, 84) all other colors: (0, 0, 0)
MOUNTAIN	Color scheme using the RGB values: mainColor(119, 122, 98) shadedColor(69, 124, 59) alternateColor(169, 179, 161) shadedAlternateColor(140, 187, 133) all other colors: (0, 0, 0)
MOUNTAIN2	Color scheme using the RGB values: mainColor(119, 122, 98) shadedColor(69, 124, 59) alternateColor(169, 179, 161) shadedAlternateColor(140, 187, 133) mainTextColorColor(255, 255, 255) shadedTextColor(255, 255, 255) all other colors: (0, 0, 0)

	MUD	Color scheme using the RGB values: mainColor(0, 0, 0) shadedColor(82, 80, 72) alternateColor(100, 8, 33) shadedAlternateColor(133, 99, 22) mainTextColor(255, 255, 255) shadedTextColor(255, 255, 255) all other colors: (0, 0, 0)
	NIGHT	Color scheme using the RGB values: mainColor(0, 0, 0) shadedColor(19,41 ,93) alternateColor(83, 89, 77) shadedAlternateColor(63, 72, 104) mainTextColor(255, 255, 255) shadedTextColor(255, 255, 255) all other colors: (0, 0, 0)
	REDAPPLE	Color scheme using the RGB values: mainColor(244, 1, 1) shadedColor(111, 184, 1) alternateColor(244, 131, 99) shadedAlternateColor(162, 212, 89) all other colors: (0, 0, 0)
	SAHARA	Color scheme using the RGB values: mainColor(255, 231, 150) shadedColor(255, 146, 84) alternateColor(255, 193, 158) shadedAlternateColor(255, 255, 225) all other colors: (0, 0, 0)
	SYSTEM	Color scheme using the RGB values: mainColor=SystemColor.control shadedColor=SystemColor.controlShadow alternateColor=SystemColor.window shadedAlternateColor=SystemColor.info all other colors: (0, 0, 0)
	SYSTEM2	Color scheme using the RGB values: mainColor=SystemColor.inactiveCaption shadedColor=SystemColor.activeCaption alternateColor=SystemColor.control shadedAlternateColor=SystemColor.info all other colors: (0, 0, 0)
	WINE	Color scheme using the RGB values: mainColor(198, 170, 185) shadedColor(170, 197, 165) alternateColor(160, 211, 222) shadedAlternateColor(225, 255, 225) all other colors: (0, 0, 0)
	WOODS	Color scheme using the RGB values: mainColor(167, 230, 87) shadedColor(84, 143, 104) alternateColor(85, 237, 192) shadedAlternateColor(225, 255, 225) all other colors: (0, 0, 0)
Return Value	JGColorScheme	Instance of the color scheme created.

getNames

Method of JGColorScheme

This property lets you retrieve the names of all existing color schemes.

Declaration
static java.lang.String[] getNames ()

	Data Type	Description
Return Value	java.lang.String[]	Array of strings returning the names of the color schemes.

removePropertyChangeListener

Method of JGColorScheme

This method lets you remove a listener for change events in the JGColorScheme object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

setColor

Method of JGColorScheme

By this method you can assign one out of the nine colors of the color scheme.

Declaration
void setColor (int colorKey, java.awt.Color color)

	Data Type	Description
Parameter		
colorKey	int	Color key from the color scheme, the color of which is to be set. The color key can be chosen from the pre-defined constants:
	Possible Values:	
	ALTERNATE_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	ALTERNATE_TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the alternate color as their background color.
	LINE_COLOR	Defines the second out of two available foreground colors of a color scheme. It determines the line color in the object that the color scheme was assigned to.
	MAIN_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	MAIN_TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the main color as their background color.

	SHADED_ALTERNATE_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	SHADED_ALTERNATE_-TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the shaded alternate color as their background color.
	SHADED_COLOR	Defines one out of four available background colors of a color scheme. It determines the background color of certain parts of the object that the color scheme was assigned to. The areas affected are listed for each object with the properties in the Gantt class: GanttColorScheme, GroupNodeColorScheme, NodeColorScheme, TableColorScheme and TimeScaleColorScheme.
	SHADED_TEXT_COLOR	Defines one out of five available foreground colors of a color scheme. It determines the font color of areas that display the shaded color as their background color.
color	java.awt.Color	Color to be assigned
Return Value	void	

toString

Method of JGColorScheme

This method returns a string representation of the object.

Declaration
java.lang.String toString ()

	Data Type	**Description**
Return Value	java.lang.String	String that the object was transformed to.

6.4 JGDiagramAnnotation

Belongs to JGantt

Package name **de.netronic.jgantt**
Extends **java.lang.Object**

This class contains properties and methods to design headers and footers of the diagram.

Constructors of the Class

This constructor allows to copy an existing header or footer of the diagram.

JGDiagramAnnotation

Constructor of JGDiagramAnnotation

This constructor lets you create a copy of the diagram annotation.

Declaration
JGDiagramAnnotation (de.netronic.jgantt.JGDiagramAnnotation SourceAnnotation)

Parameter	Data Type	Description
SourceAnnotation	de.netronic.jgantt.JGDiagramAnnotation	Diagram annotation to be copied

Properties of the Class

AntiAliasText

Property of JGDiagramAnnotation

Typ **boolean**
Bound **no**
Vetoable **no**
Exposure Level **regular**
Default Value **true**

This property lets you set or retrieve whether anti-aliasing of a diagram annotation (footer or header) is switched on.

Accessing Methods
void setAntiAliasText (boolean newValue)
boolean hasAntiAliasText ()

Also see Text

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
header.setAntiAliasText(true);
```

ColorScheme

Property of JGDiagramAnnotation

Typ	**de.netronic.jgantt.JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve a color scheme of the diagram annotation.

Accessing Methods
void setColorScheme (de.netronic.jgantt.JGColorScheme newValue)
de.netronic.jgantt.JGColorScheme getColorScheme ()

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
header.setColorScheme(hugo);
```

Font

Property of JGDiagramAnnotation

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the font of a diagram annotation, i.e. of a header or a footer.

Accessing Methods
void setFont (java.awt.Font newValue)
java.awt.Font getFont ()

Also see Text

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
 header.setFont(new Font ("Cambria", Font.ITALIC, 20));
```

Text

Property of JGDiagramAnnotation

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the text of a diagram annotation, i.e. of a header or a footer.

Accessing Methods
void setText (java.lang.String newValue)
java.lang.String getText ()

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
header.setText("New Algorithms\nAuthor: Vanessa Isolde von
Habsbergen.\nDatum:27.Februar 2014");
```

TextAlignment

Property of JGDiagramAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**ALIGNMENT_CENTER**

This property lets you set or retrieve the text alignment of a diagram annotation, i.e. of a header or a footer.

Possible Values	Description
ALIGNMENT_CENTER	The annotation is aligned in the center.
ALIGNMENT_LEFT	The annotation is aligned to the left hand side.
ALIGNMENT_RIGHT	The annotation is aligned to the right hand side.

Accessing Methods
void setTextAlignment (int newValue)
int getTextAlignment ()

Also see Text

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
header.setTextAlignment(JGDiagramAnnotaion.ALIGNMENT_LEFT);
```

Methods of the Class

getPicture

Method of JGDiagramAnnotation

This method lets you retrieve the picture object at the specified position.

Declaration
de.netronic.common.beanbase.NePicture getPicture (int picturePosition)

	Data Type	Description
Parameter		
picturePosition	int	Returns the position of the picture object
	Possible Values:	
	POSITION_CENTER	The annotation is positioned on the center.
	POSITION_LEFT	The annotation is positioned on the left hand side.
	POSITION_RIGHT	The annotation is positioned on the right hand side.
Return Value	de.netronic.common.beanbase.-NePicture	The picture object at the specified position

Also see setPicture

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
NePicture pic = header.getPicture(JGDiagramAnnotation.POSITION_LEFT);
```

setPicture

Method of JGDiagramAnnotation

This method lets you assign a picture object to a diagram annotation at a defined position.

Declaration
void setPicture (int picturePosition, de.netronic.common.beanbase.NePicture picture)

	Data Type	Description
Parameter		
picturePosition	int	Position of the picture object
	Possible Values:	
	POSITION_CENTER	The annotation is positioned on the center.
	POSITION_LEFT	The annotation is positioned on the left hand side.
	POSITION_RIGHT	The annotation is positioned on the right hand side.
picture	de.netronic.common.beanbase.-NePicture	Picture object to be set
Return Value	void	

Also see getPicture

Example Code

```
JGDiagramAnnotation header = jGantt1.getDiagramAnnotation(JGantt.DIAGRAM_HEADER);
header.setPicture(ourCompanyLogo, JGDiagramAnnotation.POSITION_LEFT);
```

6.5 JGEntitySetFilter

Belongs to JGantt

Package name **de.netronic.jgantt**
Implements **de.netronic.common.intface.NeIFilter**

This class lets you filter entities which are contained in the given entity set.

Constructors of the Class

These constructors allow to generate entity set filter objects.

JGEntitySetFilter

Constructor of JGEntitySetFilter

Creates an entity set filter object. With this constructor the name of the entity set is specified by parameter.

Declaration
JGEntitySetFilter (java.lang.String entitySetName)

Parameter	Data Type	Description
entitySetName	java.lang.String	Name of the entity set

JGEntitySetFilter

Constructor of JGEntitySetFilter

Creates an entity set filter object. The name of the entity set remains to be specified by setting the property **EntitySetName**.

Declaration
JGEntitySetFilter ()

Properties of the Class

EntitySetName

Property of JGEntitySetFilter

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the entity set which should be used as filter condition.

Accessing Methods
void setEntitySetName (java.lang.String newValue)
java.lang.String getEntitySetName ()

Methods of the Class

apply

Method of JGEntitySetFilter

When JGantt invokes this method, the filter object verifies whether the entity passed is member of the entity set.

Declaration
boolean apply (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity that the filter is to apply to.
Return Value	boolean	**true**: the entity is member of the entity set, **false**: the entity is not member of the entity set.

6.6 JGIPersistenceManager

Belongs to JGantt

Package name **de.netronic.jgantt**

This interface is the access to the persistency concept of VARCHART JGantt. You can store all or single objects of VARCHART JGantt and re-load them.

Properties to Handle the Persistence Manager

Methods to Handle the Persistence Manager

readPropertiesFromFile(...)	Loads the properties from a file into the current JGantt instance.
writePropertiesToFile(...)	This method lets you persist a VARCHART JGantt component.

Methods of the Interface

readPropertiesFromFile

Method of JGIPersistenceManager

This method lets you load your own persistency file, for example into an "empty" instance of VARCHART JGantt, which then will supply settings to all properties.

Declaration
void readPropertiesFromFile (java.lang.String fileName)

	Data Type	Description
Parameter		
fileName	java.lang.String	File from which the property settings are to be loaded into the VARCHART JGantt instance.
Return Value	void	

Example Code

```
jGantt1.getPersistenceManager().readPropertiesFromFile(filePath);
```

writePropertiesToFile

Method of JGIPersistenceManager

This method lets you save the current state of all or a single of the VARCHART JGantt objects to an XML file, such as the AppData, Calendar, GanttGraph or Histogram component.

Declaration
void writePropertiesToFile (java.lang.String fileName, java.lang.Object objectToBeStored)

	Data Type	Description
Parameter		
fileName	java.lang.String	Name of the file to which the object passed shall be stored.
objectToBeStored	java.lang.Object	Object, the properties of which are to stored for reasons of persistency..
Return Value	void	

Also see readPropertiesFromFile

Example Code

```
jGantt1.getPersistenceManager().writeToFile(filePath, jGantt1);
```

6.7 JGIPrintManager

Belongs to JGantt

Package name **de.netronic.jgantt**

This interface contains properties to handle the printing.

Properties to Handle Printing

Alignment	Alignment of the diagram on the page
AspectRatioAdaptedToPages	Adaptation of the aspect ratio to the output medium
FooterAlignment	Alignment of the footer
FooterText	Text of the footer
GanttOptionsToolBarVisible	Is the Gantt options tool bar visible?
HeaderAlignment	Alignment of the header
HeaderFooterFont	Font type of the header and footer.
HeaderFooterToolBarVisible	Is the tool bar for header and footer settings visible?
HeaderText	Text of the header
MainTableColumns	Number of table columns on the main page(s)

Margin	Width of the sheet margin (in mm)
Orientation	Orientation of the sheet
PageNumbersPosition	Position of the page numbers
PagesHeight	Number of pages high occupied by the chart
PagesWidth	Number of pages wide occupied by the chart
Paper	Physical characteristics of a sheet of paper
PrintDatePosition	Position of the print date
PrintEndTime	End date of the Gantt graph section to be printed (in milliseconds)
PrintSectionNodesOnly	Section of the Gantt graph to be printed
PrintStartTime	Start date of the Gantt graph section to be printed (in milliseconds)
RepeatedTableColumns	Number of table columns on repeated pages
ScaleFactor	Scaling factor to print the chart (in %).

Properties of the Interface

Alignment

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**ALIGNMENT_CENTER_CENTER**

This property sets the alignment of the diagram on the page.

Possible Values	Description
ALIGNMENT_BOTTOM_CENTER	The chart is placed at the bottom in the center.
ALIGNMENT_BOTTOM_LEFT	The chart is placed at the bottom on the left hand side.
ALIGNMENT_BOTTOM_RIGHT	The chart is placed at the bottom on the right hand side.
ALIGNMENT_CENTER_CENTER	The chart is placed in the vertical and horizontal center.
ALIGNMENT_CENTER_LEFT	The chart is placed in the vertical center on the left hand side.
ALIGNMENT_CENTER_RIGHT	The chart is placed in the vertical center on the right hand side.
ALIGNMENT_TOP_CENTER	The chart is placed at the top in the center.
ALIGNMENT_TOP_LEFT	The chart is placed at the top on the left hand side.
ALIGNMENT_TOP_RIGHT	The chart is placed at the top on the right hand side.

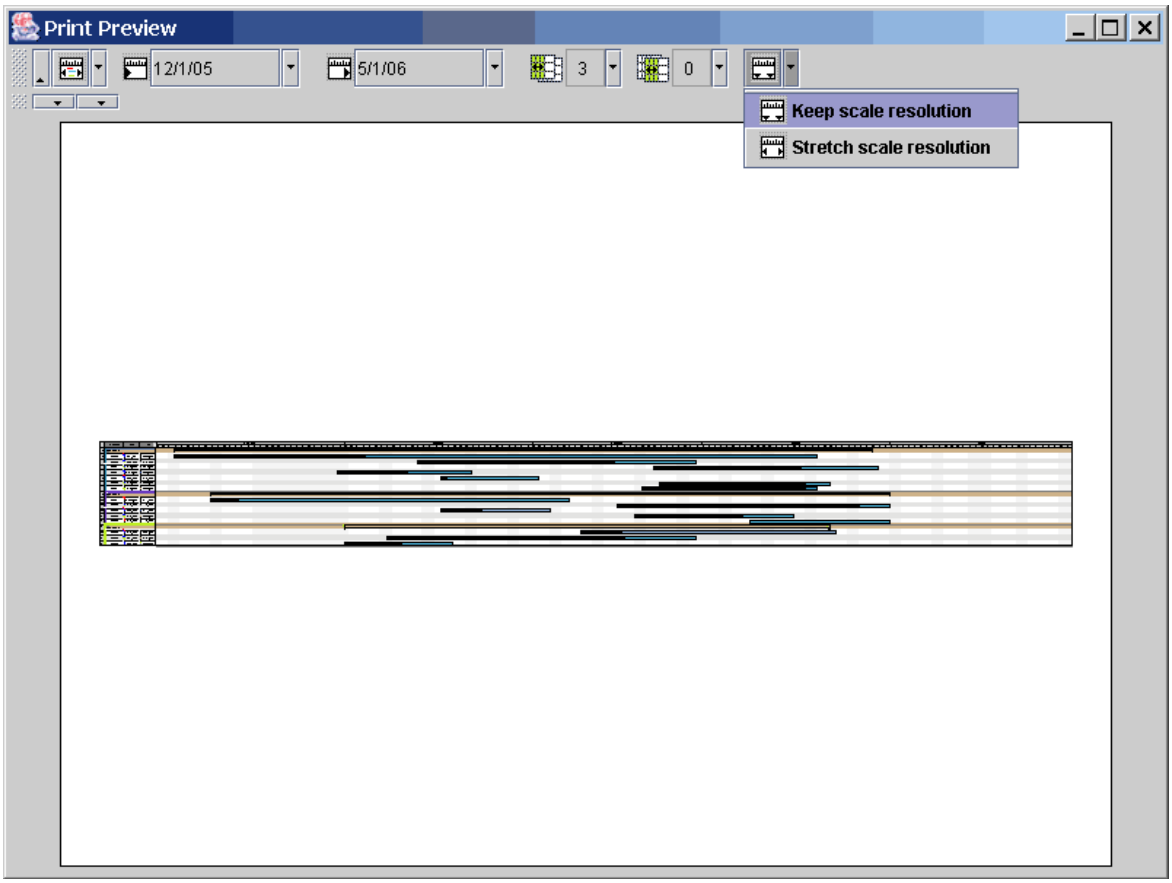
Accessing Methods
void setAlignment (int newValue)
int getAlignment ()

AspectRatioAdaptedToPages

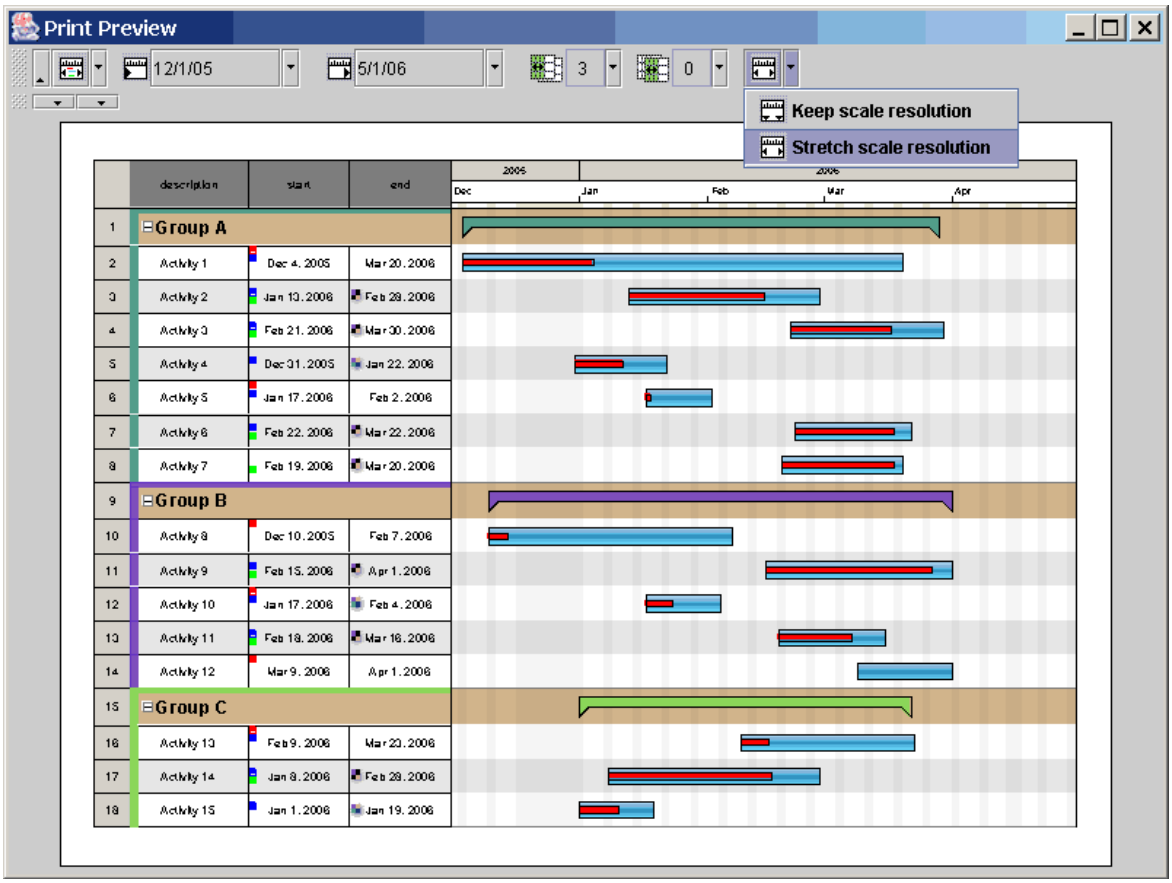
Property of JGIPrintManager

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you adapt the aspect ration of the chart to the aspect ration of the output medium. This is useful, for example if you intend to print a chart that has a very long time scale:



Above: No adaptation: the vertical dimension is hardly used; details cannot be recognized.



Above: the aspect ratio is adapted, the vertical dimension being fully used. Details can well be recognized.

This property is available for printing to a single page as well as to several pages. If you print to several pages, you can set the number of pages in horizontal and vertical direction independently of each other.

Accessing Methods
void setAspectRatioAdaptedToPages (boolean newValue)
boolean isAspectRatioAdaptedToPages ()

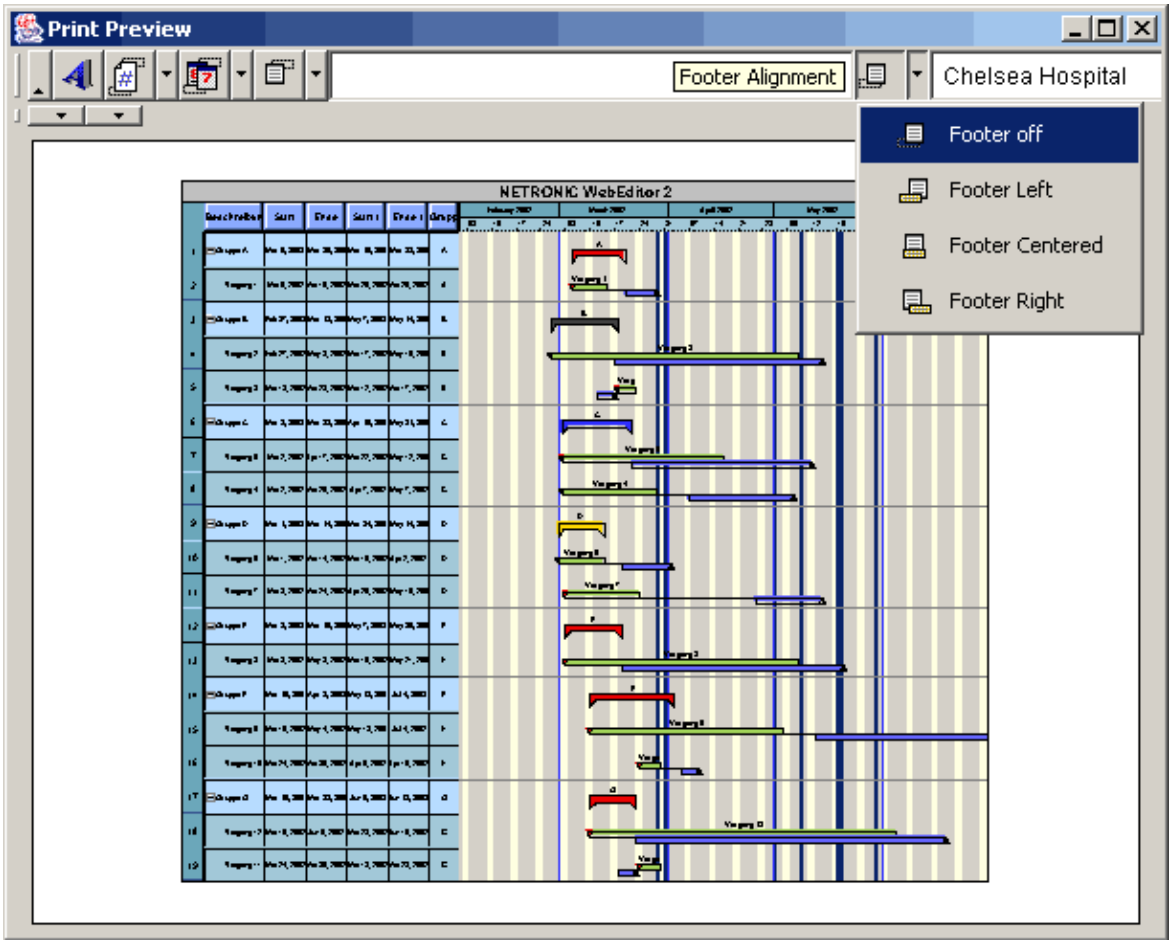
Also see PagesHeight
PagesWidth
ScaleFactor

FooterAlignment

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**FOOTER_OFF**

This property sets the alignment of the footer or hides it.



Possible Values	Description
FOOTER_CENTER	The footer is positioned in the center.
FOOTER_LEFT	The footer is aligned with the top left corner.
FOOTER_OFF	No footer is displayed.
FOOTER_RIGHT	The footer is aligned with the top right corner.

Accessing Methods
void setFooterAlignment (int newValue)
int getFooterAlignment ()

FooterText

Property of JGIPrintManager

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the text of the footer.

Accessing Methods
void setFooterText (java.lang.String newValue)
java.lang.String getFooterText ()

GanttOptionsToolBarVisible

Property of JGIPrintManager

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines, whether or not the Gantt options tool bar will be visible.
True: the tool bar will be visible; **false**: the tool bar will be invisible.

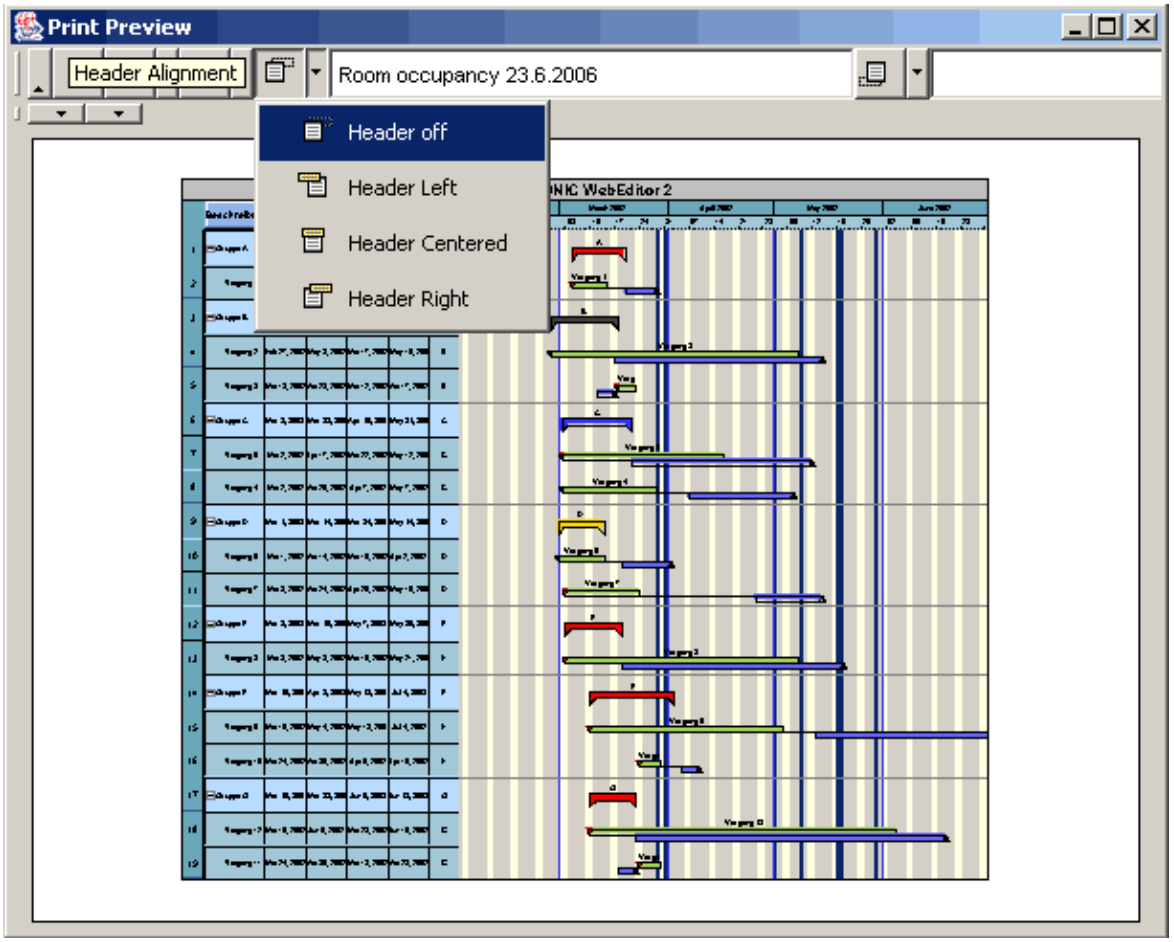
Accessing Methods
void setGanttOptionsToolBarVisible (boolean newValue)
boolean isGanttOptionsToolBarVisible ()

HeaderAlignment

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**HEADER_OFF**

This property sets the alignment of the header or hides it.



Possible Values	Description
HEADER_CENTER	The header is positioned in the center.
HEADER_LEFT	The header is aligned with the top left corner.
HEADER_OFF	No header is displayed.
HEADER_RIGHT	The header is aligned with the top right corner.

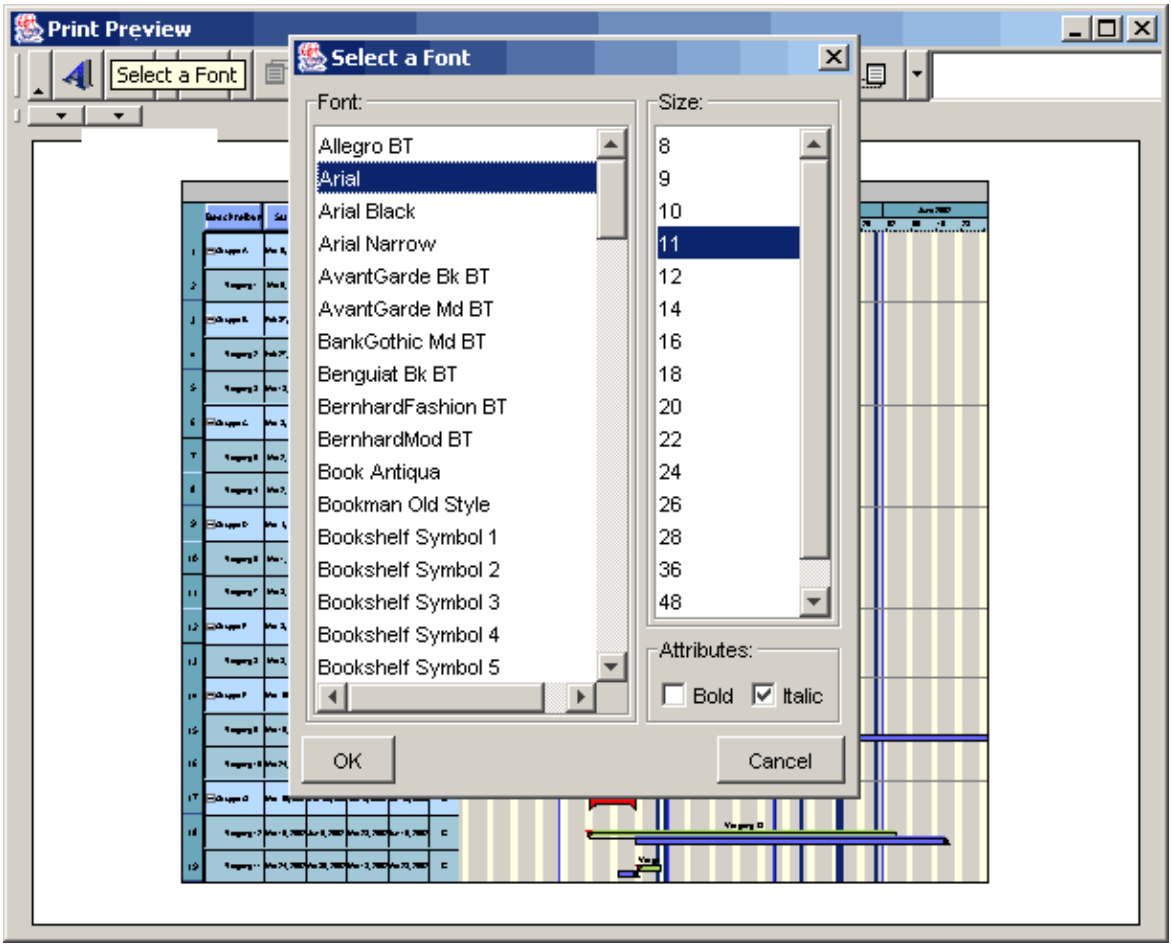
Accessing Methods
void setHeaderAlignment (int newValue)
int getHeaderAlignment ()

HeaderFooterFont

Property of JGIPrintManager

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Arial 11**

This property sets the font type for the header and footer.



Accessing Methods
void setHeaderFooterFont (java.awt.Font newValue)
java.awt.Font getHeaderFooterFont ()

HeaderFooterToolBarVisible

Property of JGIPrintManager

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines, whether or not the header and footer tool bar will be visible. **True**: the tool bar will be visible; **false**: the tool bar will be invisible.

Accessing Methods
void setHeaderFooterToolBarVisible (boolean newValue)
boolean isHeaderFooterToolBarVisible ()

HeaderText

Property of JGIPrintManager

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the text of the header.

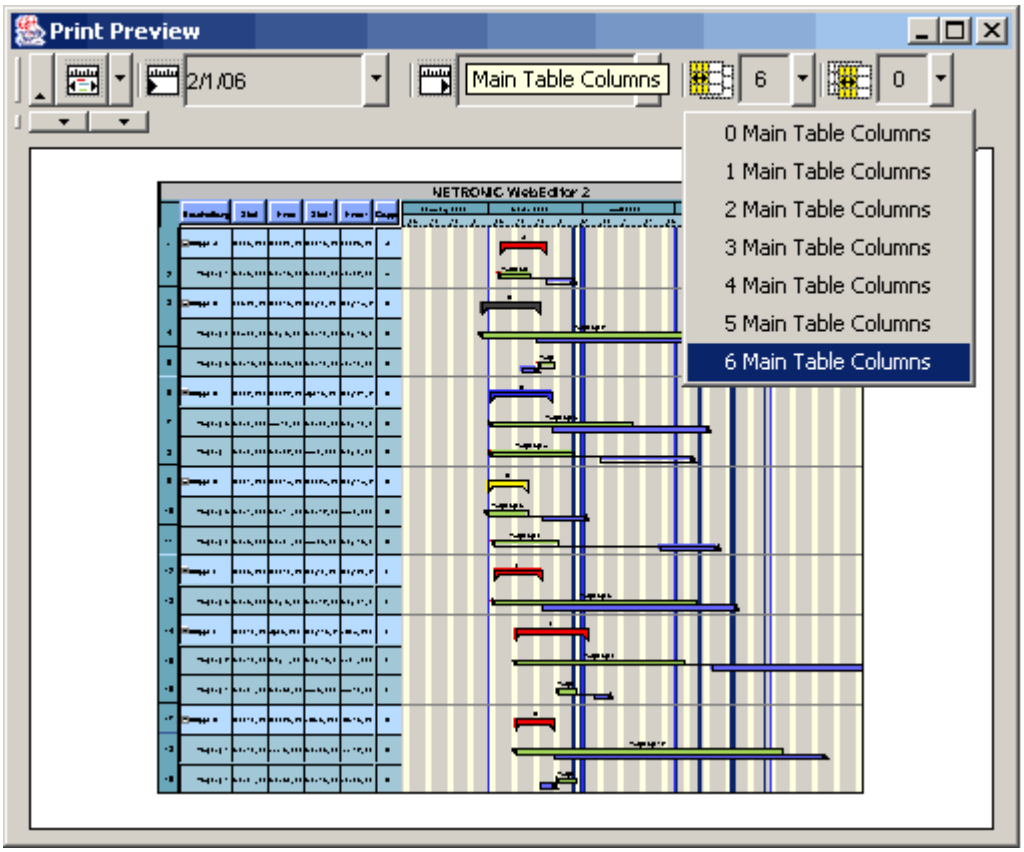
Accessing Methods
void setHeaderText (java.lang.String newValue)
java.lang.String getHeaderText ()

MainTableColumns

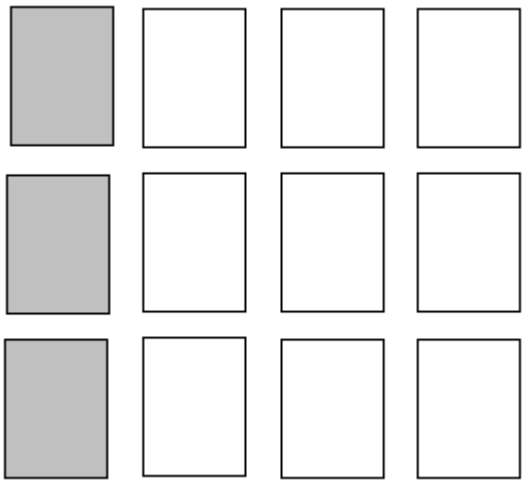
Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets the number of table columns on the main page(s), the columns being counted from left to right. On zero, no column will be repeated.



The main page is the first (far left) page of a row of pages. Several main pages will be generated if the diagram extends on several rows of sheets. The main pages then form the first "column" of pages on the left hand side of the matrix.



The size of the table, i.e. the number of columns in the table on the main pages, may differ from the size of the table on the repeated pages.

Accessing Methods
void setMainTableColumns (int newValue)
int getMainTableColumns ()

Also see RepeatedTableColumns

Margin

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**10**

This property sets the width of the sheet margin. Unit: mm.

Accessing Methods
void setMargin (int newValue)
int getMargin ()

Also see Paper

Orientation

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**ORIENTATION_LANDSCAPE**

This property sets the orientation of the sheet.

Possible Values	Description
ORIENTATION_LANDSCAPE	Landscape sheet orientation
ORIENTATION_PORTRAIT	Portrait sheet orientation

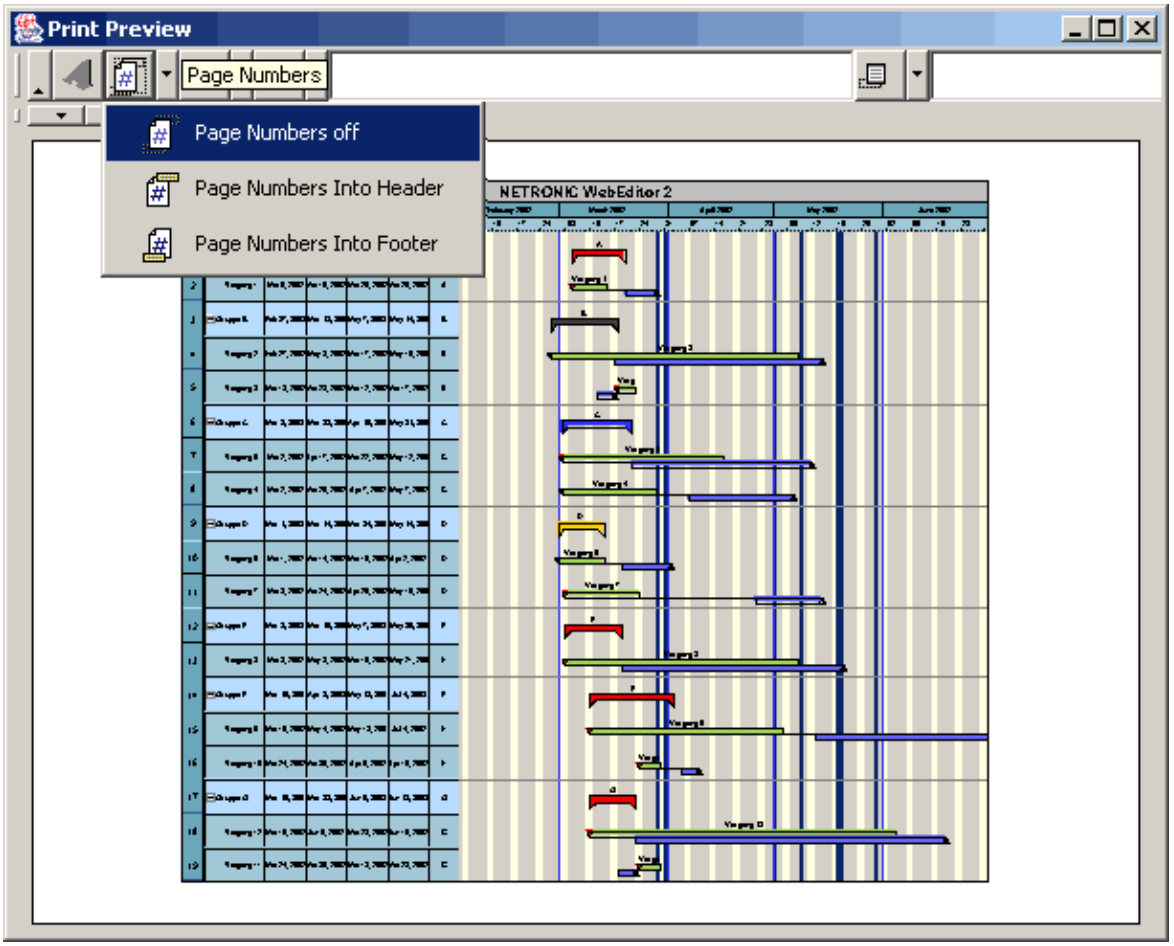
Accessing Methods
void setOrientation (int newValue)
int getOrientation ()

PageNumbersPosition

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**PAGE_NUMBERS_OFF**

This property sets the position of the page numbers or hides them.



Possible Values	Description
PAGE_NUMBERS_INTO_FOOTER	Page numbers are positioned in the footer
PAGE_NUMBERS_INTO_HEADER	Page numbers are positioned in the header
PAGE_NUMBERS_OFF	No page numbers are displayed

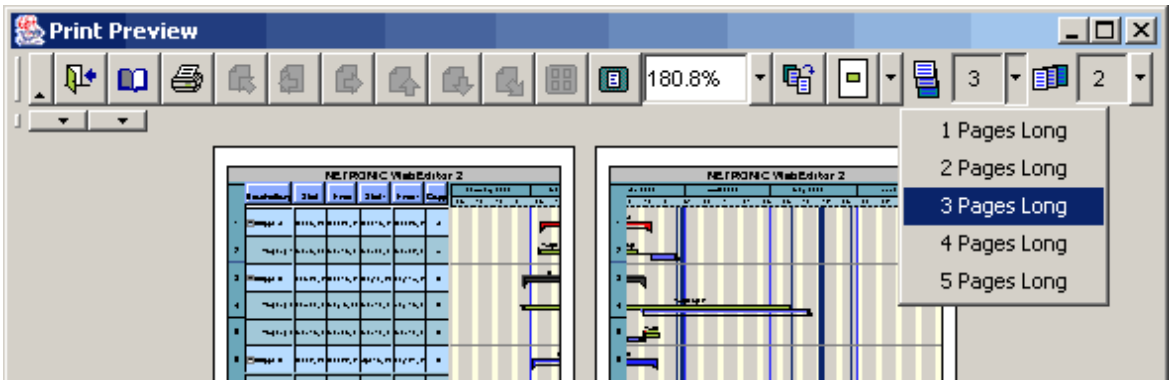
Accessing Methods
void setPageNumbersPosition (int newValue)
int getPageNumbersPosition ()

PagesHeight

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1**

This property sets the number of pages occupied by the chart in the dimension of height.



Accessing Methods
void setPagesHeight (int newValue)
int getPagesHeight ()

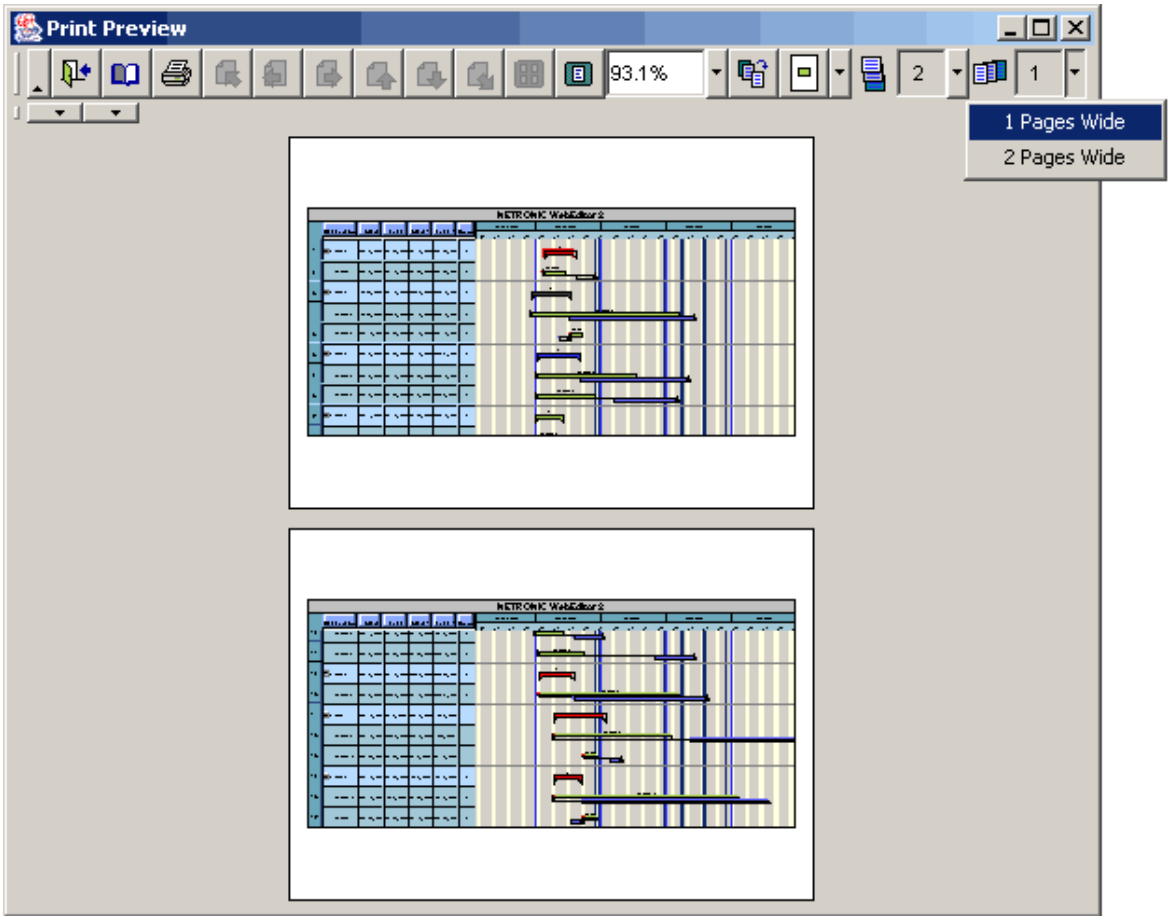
Also see AspectRatioAdaptedToPages
PagesWidth
ScaleFactor

PagesWidth

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1**

This property sets the number of pages occupied by the chart in the dimension of width.



Accessing Methods
void setPagesWidth (int newValue)
int getPagesWidth ()

Also see AspectRatioAdaptedToPages
PagesHeight
ScaleFactor

Paper

Property of JGIPrintManager

Typ	**java.awt.print.Paper**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property describes to the physical characteristics of a sheet of paper (e.g. the height and the width).

Accessing Methods
void setPaper (java.awt.print.Paper newValue)
java.awt.print.Paper getPaper ()

Also see Margin

Example Code

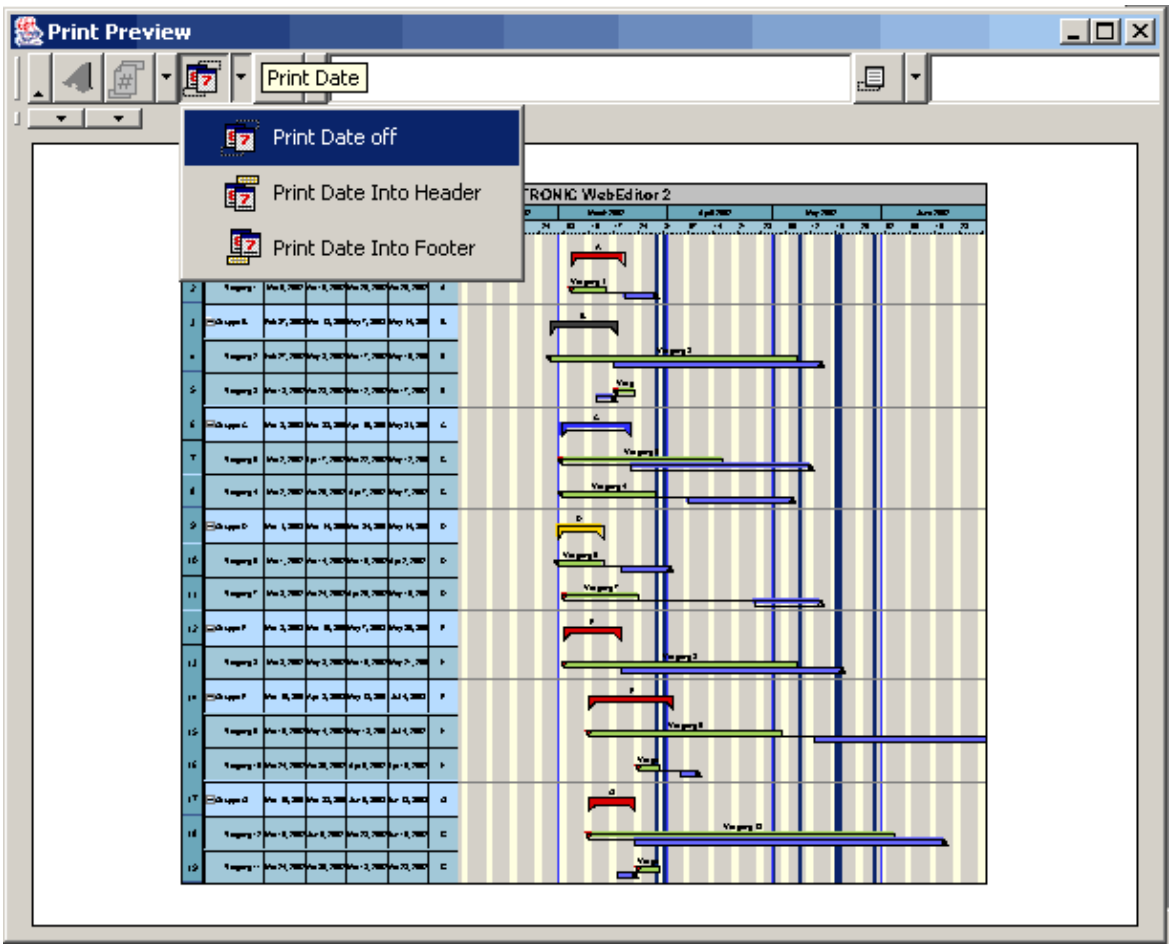
```
Paper paper = jGantt1.getPrintManager().getPaper();
paper.setSize(500.0, 500.0);
paper.setImageableArea(50.0, 50.0, 400.0, 400.0);
jGantt1.getPrintManager().setPaper(paper);
```

PrintDatePosition

Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**PRINT_DATE_OFF**

This property sets the position of the print date or hides it.



Possible Values	Description
PRINT_DATE_INTO_FOOTER	The print date is displayed in the footer.
PRINT_DATE_INTO_HEADER	The print date is displayed in the header.
PRINT_DATE_OFF	No print date is displayed.

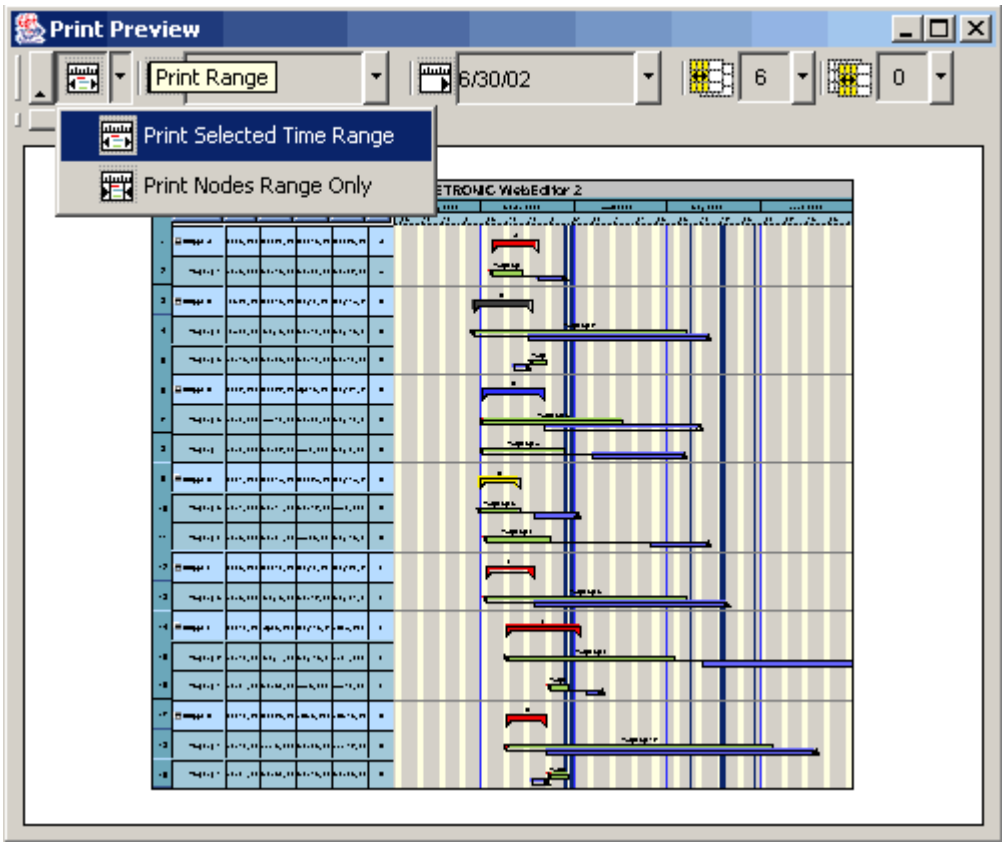
Accessing Methods
void setPrintDatePosition (int newValue)
int getPrintDatePosition ()

PrintEndTime

Property of JGIPrintManager

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property is needed to manually define a section of the Gantt graph to be printed. It sets the end date of the section. Unit: Number of milliseconds since 1.1.1970. The start you can set by set/getPrintStartTime.
Alternatively, you can restrict printing to the section of the Gantt graph that displays nodes (please see set/getPrintSectionNodesOnly).



Accessing Methods
void setPrintEndTime (long newValue)
long getPrintEndTime ()

Also see PrintSectionNodesOnly
PrintStartTime

PrintSectionNodesOnly

Property of JGIPrintManager

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property describes, whether the section of the Gantt graph to be printed is reduced to the area where nodes are present (**true**) or whether it comprises the chart completely (**false**). The limits (dates) of the section are set automatically.
If you wish to define a section by yourself, please see **set/getPrintStartTime** and **set/getPrintEndTime**.

Accessing Methods
void setPrintSectionNodesOnly (boolean newValue)
boolean isPrintSectionNodesOnly ()

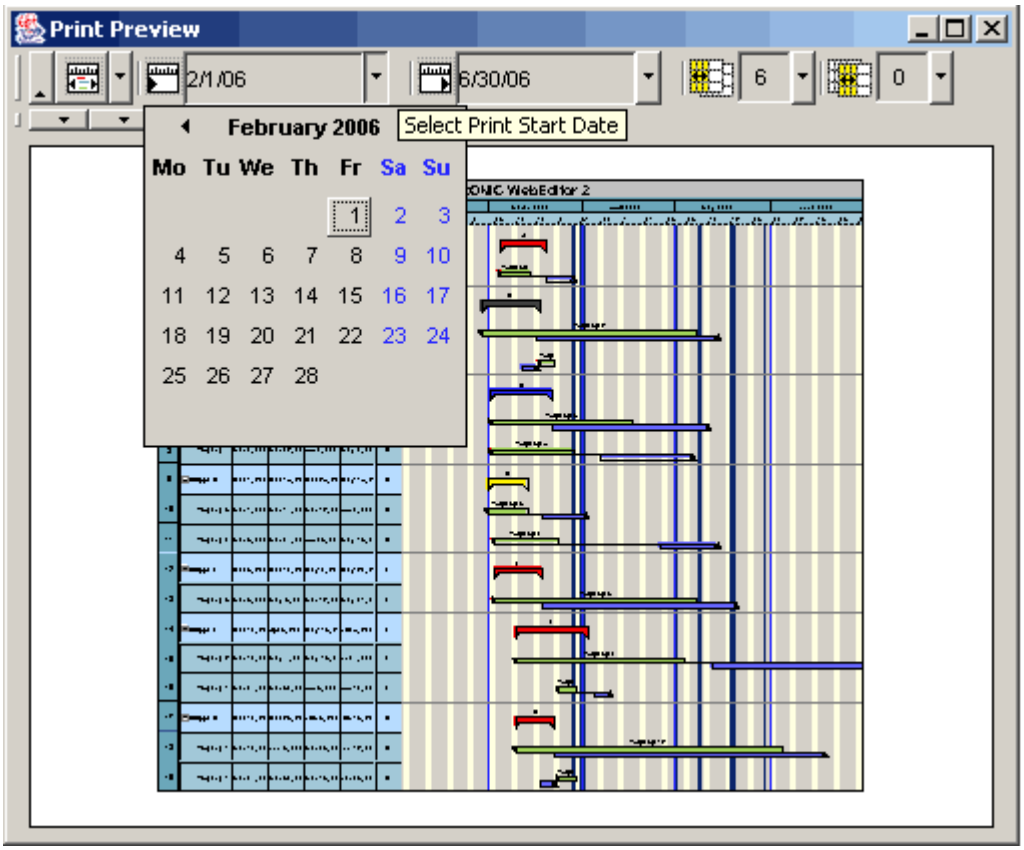
Also see PrintEndTime
PrintStartTime

PrintStartTime

Property of JGIPrintManager

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property is needed to manually define a section of the Gantt graph to be printed. It sets the start date of the section. Unit: Number of milliseconds since 1.1.1970. The end you can set by set/getPrintEndTime.
Alternatively, you can restrict printing to the section that displays nodes (please see set/getPrintSectionNodesOnly).



Accessing Methods
void setPrintStartTime (long newValue)
long getPrintStartTime ()

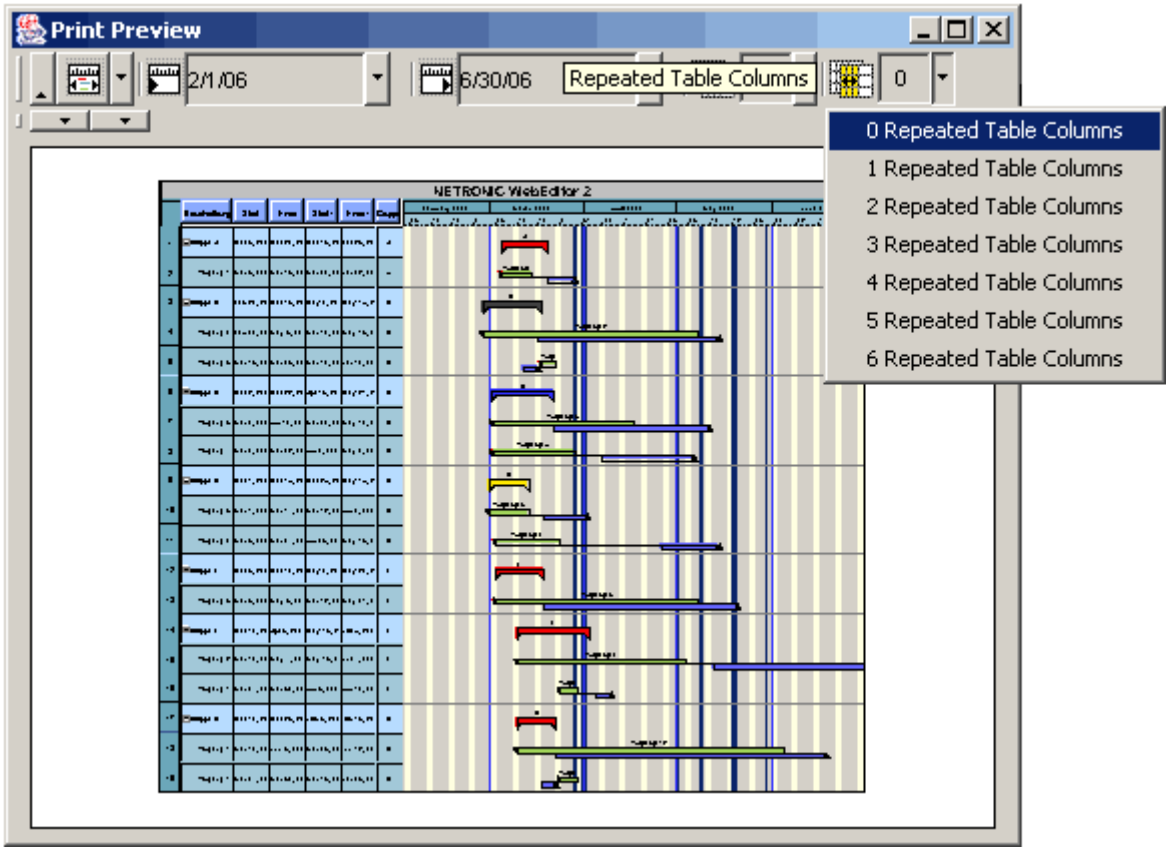
Also see PrintEndTime
PrintSectionNodesOnly

RepeatedTableColumns

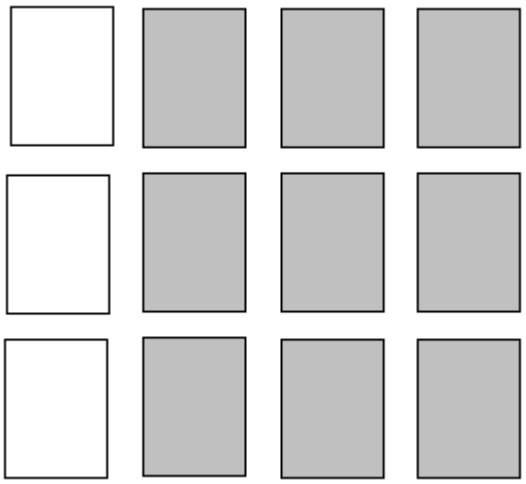
Property of JGIPrintManager

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0**

This property sets the number of table columns on the repeated page(s), the columns being counted from left to right. On zero, no column will be repeated.



Repeated pages are the ones that follow the first (far left) page in a row. A matrix of repeated pages will be generated if the diagram is printed on several rows of sheets.



The size of the table, i.e. the number of columns in the table on the main pages, may differ from the size of the table on the repeated pages.

Accessing Methods
void setRepeatedTableColumns (int newValue)
int getRepeatedTableColumns ()

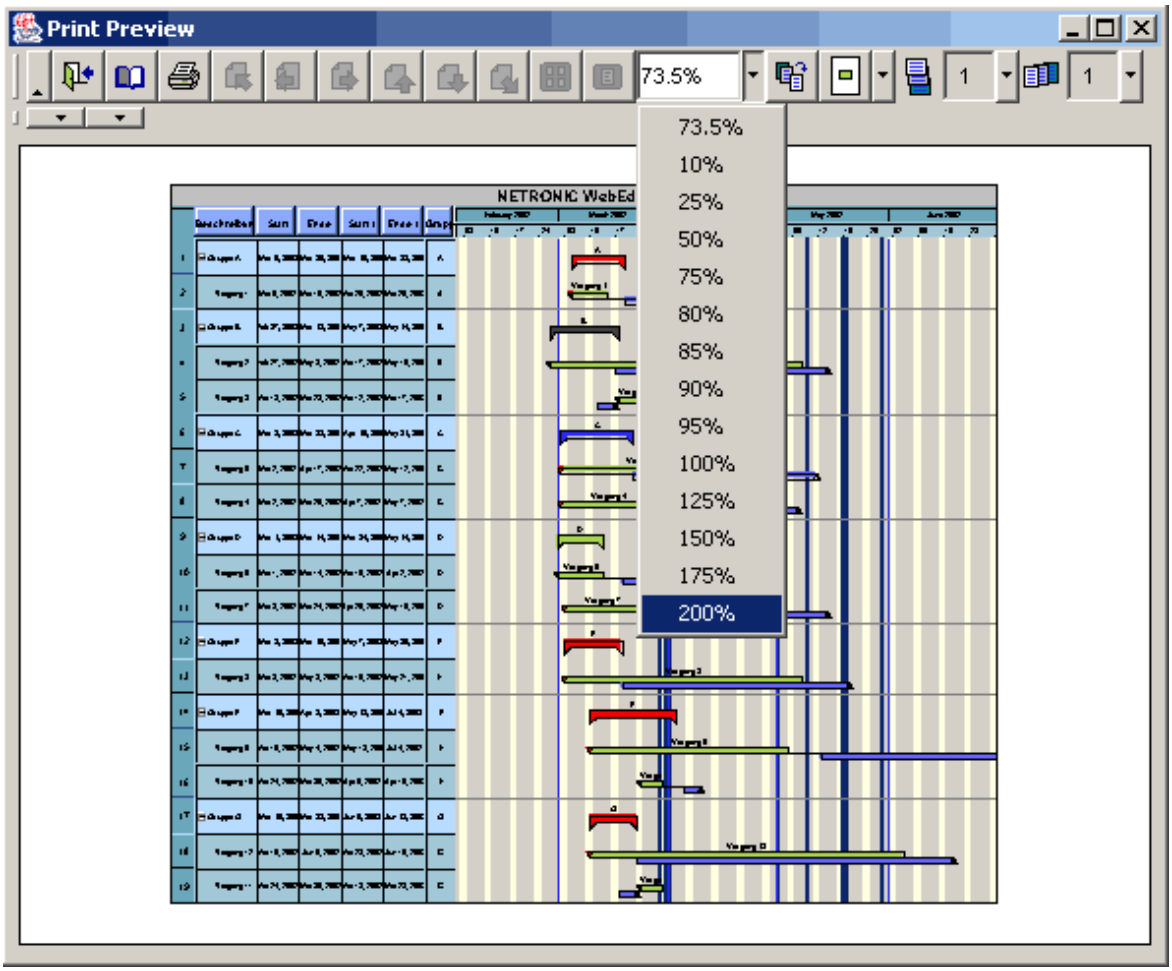
Also see MainTableColumns

ScaleFactor

Property of JGIPrintManager

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**100**

This property sets the scaling factor by which the chart is printed. Unit: percent. When opening the PrintPreview, the scaling factor is automatically set to a value that allows to display a page completely, overwriting the existing value if previously set. If you wish the user to open the Print Preview at a defined scaling factor, you need to set it after opening, as shown in the code sample.



Accessing Methods
void setScaleFactor (double newValue)
double getScaleFactor ()

Example Code

```
//Example of a scaling factor of 100 % on opening the PrintPreview:
jGantt1.openPrintPreview();
jGantt1.getPrintManager().setScaleFactor(100.0);
```

6.8 JGIsHierarchyFilter

Belongs to JGantt

Package name **de.netronic.jgantt**

Implements **de.netronic.common.intface.NeIFilter**

This class allows to check whether or not **grouping by hierarchy** was set to VARCHART JGantt. This filter may be required in cases where a NeCombinedFilter is used that retrieves the grouping type.

Constructors of the Class

This constructor allows to generate a filter which indicates whether or not **grouping by hierarchy** was set to VARCHART JGantt.

JGIsHierarchyFilter

Constructor of JGIsHierarchyFilter

A JGIsHierarchyFilter is created that verifies whether or not the VARCHART JGantt instance was set to the grouping type **hierarchy**. The filter needs to be activated by the method **apply**.

Declaration
JGIsHierarchyFilter (JGantt jGantt1)

Parameter	Data Type	Description
jGantt1	JGantt	The VARCHART JGantt instance that is to use the filter.

Methods of the Class

apply

Method of JGIsHierarchyFilter

This method verifies whether or not the VARCHART JGantt instance was set to the grouping type **hierarchy**.

Declaration
boolean apply (NeIEntity entity)

	Data Type	Description
Parameter		
entity	NeIEntity	Entity that the filter is to apply to.
Return Value	boolean	**true**: This VARCHART JGantt instance was set to the grouping type **hierarchy**. **false**: This VARCHART JGantt instance was not set to the grouping type **hierarchy**.

Example Code

```
filter = new NeCombinedFilter(new JGLevelFilter(jGantt1, 1), "&&", new
JGIsHierarchyFilter(jGantt1));
```

6.9 JGLayouterHelper

Belongs to JGantt

Package name **de.netronic.jgantt**

This class holds the methods of the layout of grouping and hierarchy structures in a Gantt chart. Each JGantt instance holds exactly one instance of this class. You can retrieve this instance by the JGantt method getLayouterHelper().

Common methods

getEntityAtRowIndex(...)	Retrieves the entity from the row specified.
getRowIndexOfNode(...)	Retrieves the row index of an entity.

Methodes related to the hierarchy structure

getChildNodes(...)	Retrieves the child nodes of an entity.
getEntityCountInHierarchy(...)	Retrieves the number of entities within a hierarchy
getEntityInHierarchyAtPreOrderIndex(...)	Retrieves the entity at the given index in a hierarchy tree structured according to the preorder.

getLevel(...)	Retrieves the grouping or hierarchy level of the entity.
getParentNode(...)	Retrieves the parent node of an entity if a hierarchy is present.
incrementHierarchyCodeAtIndex(...)	Increments the hierarchy code at a defined digit.
isCollapsed(...)	Verifies, whether or not the group belonging to the entity passed is collapsed.
isLeaf(...)	Retrieves, whether a node ist a leaf node/not a group node.

Methods of the Class

getChildNodes

Method of JGLayouterHelper

This method lets you retrieve the child nodes of an entity. If no hierarchy exists, null will be returned.

Declaration
java.util.Iterator getChildNodes (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of which the child nodes are to be retrieved.
Return Value	java.util.Iterator	Iterator object returned which iterates over the entities of the child nodes.

Also see getParentNode

Example Code

```
jGantt1.getLayouterHelper.getChildNodes (entity);
```

getEntityAtRowIndex

Method of JGLayouterHelper

This method lets you retrieve the entity of a row that is to be specified by the parameter.

Declaration
de.netronic.common.intface.NeIEntity getEntityAtRowIndex (int rowIndex)

	Data Type	Description
Parameter		
rowIndex	int	Index of the row the entity of which is to be retrieved.
Return Value	de.netronic.common.intface.NeIEntity	The entity in the row specified

Also see getRowIndexOfNode

Example Code

```
jGantt1.getLayouterHelper.getEntityAtRowIndex (rowIndex);
```

getEntityCountInHierarchy

Method of JGLayouterHelper

This method lets you retrieve the number of entities within a hierarchy if the grouping mode was set to **GROUP_BY_HIERARCHY** before; if it wasn't, **0** will be returned

Declaration
int getEntityCountInHierarchy (boolean ignoreCollapsedStates)

	Data Type	Description
Parameter		
ignoreCollapsedStates	boolean	This parameter indicates, whether (**true**) or not (**false**) entities in collapsed branches should be skipped.
Return Value	int	Number of entities within a hierarchy

getEntityInHierarchyAtPreOrderIndex

Method of JGLayouterHelper

By this method you can retrieve an entity at the index specified in a hierarchy tree structured according to the preorder. To obtain an entity, the gouping mode must have been set to **GROUP_BY_HIERARCHY**, otherwise **null** will be returned.

Declaration
de.netronic.common.intface.NeIEntity getEntityInHierarchyAtPreOrderIndex (int preOrderIndex, boolean ignoreCollapsedStates)

	Data Type	Description
Parameter		
preOrderIndex	int	Index according to the perorder, at which the entity should be retrieved.
ignoreCollapsedStates	boolean	This parameter indicates whether (**true**) or not (**false**) nodes in collapsed branches should be skipped.
Return Value	de.netronic.common.intface.NeIEntity	Entity within the hierarchy at the index specified

getLevel

Method of JGLayouterHelper

This method lets you retrieve the grouping or hierarchy level of the entity.

Declaration
int getLevel (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity the level of wich is to be retrieved.
Return Value	int	Hierarchy level of the entity specified

Example Code

```
jGantt1.getLayouterHelper.getLevel (entity);
```

getParentNode

Method of JGLayouterHelper

If the property **GroupMode** was set to **GROUP_BY_HIERARCHY**, this method will return the parent node of the entity passed, otherwise it will return null.

Declaration
de.netronic.common.intface.NeIEntity getParentNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of which the parent node is to be retrieved.
Return Value	de.netronic.common.intface.NeIEntity	Parent node of the entity specified

Also see getChildNodes

Example Code

```
jGantt1.getLayouterHelper.getParentNode (entity);
```

getRowIndexOfNode

Method of JGLayouterHelper

This method lets you retrieve the index of the row in which the entity passed is located.

Declaration
int getRowIndexOfNode (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	The entity of which the row index is to be retrieved
Return Value	int	Index of the row where the entity passed is located.

Also see getEntityAtRowIndex

Example Code

```
jGantt1.getLayouterHelper.getRowIndexForNode (entity);
```

incrementHierarchyCodeAtIndex

Method of JGLayouterHelper

This method lets you increment the hierarchy code at a defined digit. For this, The first parameter passes the digit while the second one passes the hierarchy code to be incremented.

Declaration
static int incrementHierarchyCodeAtIndex (int index, java.lang.String hierarchyCode)

	Data Type	Description
Parameter		
index	int	Index that describes the digit of the hierarchy code that is to be increased by 1. The index 0 identifies the first digit. If you wish to increment the last digit, you can retrieve it by index == JGLayouterHelper.LAST_INDEX and pass it by this parameter.
	Possible Values:	
	LAST_INDEX	Index that addresses the digit(s) on the ultimate right hand position of a hierarchy code; so in an example 1.2.3.14 LAST_INDEX will return the value 14.
hierarchyCode	java.lang.String	Hierarchy code to be incremented
Return Value	int	Hierarchy code that was incremented at the digit passed by the index.
	Possible Values:	
	LAST_INDEX	Index that addresses the digit(s) on the ultimate right hand position of a hierarchy code; so in an example 1.2.3.14 LAST_INDEX will return the value 14.

Example Code

```
//incrementing from 1.1.1 to 1.2.1
String hc = preEnt.getValueAsString(aktHCode);
hc = JGLayouterHelper.incrementHierarchyCodeAtIndex (1, "1.1.1")
```

isCollapsed

Method of JGLayouterHelper

This method lets you verify whether or not the group belonging to this entity is collapsed.

Declaration
boolean isCollapsed (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity of which is to be retrieved whether or not the associated group is collapsed.
Return Value	boolean	Returns **true**, if the group belonging to the entity is collapsed, otherwise **false** will be returned.

Example Code

```
jGantt1.getLayouterHelper().isCollapsed(entity);
```

isLeaf

Method of JGLayouterHelper

This method lets you retrieve whether a node is a leaf node/not a group node. o
In case of a hierarchy, i.e. the gruping mode was set to GROUP_BY_HIERARCHY, the method verifies if the entity represents a leaf node.
In case of grouping, i.e. the grouping mode was set to GROUP_BY_VALUE, the method verifies if the entity is not a group node.

Declaration
boolean isLeaf (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter entity	 de.netronic.common.intface.NeIEntity	 Entity of which is to be retrieved whether it represents a leaf node/not a group node.
Return Value	boolean	Returns **true**, if the entity passed is a leaf node or not a group node, otherwise **false** will be returned.

6.10 JGLegend

Belongs to JGantt

Package name **de.netronic.jgantt**
Extends **de.entronic.jgantt.JGDiagramAnnotation**

This class handles the legend for the Gantt diagram. The legend can contain arbitrary texts, rectangles or pictures, thus the legend can be customized to satisfy your needs.

By JGLegend legend = jGantt1.getLegend();
you'll get the JGLegend object of your JGantt object.

Properties of the Class

AreaStyle

Property of JGLegend

Typ **java.awt.Color**
Bound **no**
Vetoable **no**
Exposure Level **regular**

By this property you can specify or retrieve the background color of the legend.

Accessing Methods
void setAreaStyle (java.awt.Color newValue)
java.awt.Color getAreaStyle ()

Font

Property of JGLegend

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can specify the font of your legend

Accessing Methods
void setFont (java.awt.Font newValue)
java.awt.Font getFont ()

LayerRectangleWidthPercent

Property of JGLegend

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0**

By this property you can specify or retrieve the relative width of the layer rectangle in a legend cell in percent.

Accessing Methods
void setLayerRectangleWidthPercent (int newValue)
int getLayerRectangleWidthPercent ()

LineGrid

Property of JGLegend

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can specify or retrieve whether a line grid appears in te legend.

Accessing Methods
void setLineGrid (boolean newValue)
boolean getLineGrid ()

MinimumRowHeight

Property of JGLegend

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**800**

By this property you can specify or retrieve the minimum height of a legend row. Unit is [mm/100].

Accessing Methods
void setMinimumRowHeight (int newValue)
int getMinimumRowHeight ()

Mode

Property of JGLegend

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**LAYER_ONLY**

By this property you can specify the mode of your legend:

Possible Values	Description
LAYER_AND_LINKS	Layers and links are shown in the legend.
LAYER_ONLY	Only layers are shown in the legend.
LINKS_ONLY	Only links are shown in the legend.
USER_DEFINED	The legend is defined by the user.

Accessing Methods
void setMode (int newValue)
int getMode ()

NoOfColumns

Property of JGLegend

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can specify or retrieve the number of colums of the legend.

Accessing Methods
void setNoOfColumns (int newValue)
int getNoOfColumns ()

ShowZeroLengthLabels

Property of JGLegend

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can specify or retrieve whether the zero length label of a layerdefinition should appear in the legend.

Accessing Methods
void setShowZeroLengthLabels (boolean newValue)
boolean getShowZeroLengthLabels ()

TextColor

Property of JGLegend

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can specify or retrieve the text color of the legend.

Accessing Methods
void setTextColor (java.awt.Color newValue)
java.awt.Color getTextColor ()

Methods of the Class

addEmptyCell

Method of JGLegend

nly when legend is user defined: Introduction of an empty cell into legend.

Declaration
void addEmptyCell ()

	Data Type	Description
Return Value	void	

addLayerDefinition

Method of JGLegend

Only when legend is user defined: Introduction of a layer definition cell with special background color and special legend text into the legend.

Declaration
void addLayerDefinition (java.lang.String name, java.awt.Color color, java.lang.String legendtext)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the linkdefinition
color	java.awt.Color	Back ground color of the layer
legendtext	java.lang.String	Legend text of the linkdefinition
Return Value	void	

addLayerDefinition

Method of JGLegend

Only when legend is user defined: Introduction of a layer definition cell with special background color into the legend.

Declaration
void addLayerDefinition (java.lang.String name, java.awt.Color color)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the linkdefinition
color	java.awt.Color	Back ground color of the layer
Return Value	void	

addLayerDefinition

Method of JGLegend

Only when legend is user defined: Introduction of a layer definition cell with a special legend text into the legend.

Declaration
void addLayerDefinition (java.lang.String name, java.lang.String legendtext)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the linkdefinition
legendtext	java.lang.String	Legend text of the linkdefinition
Return Value	void	

addLayerDefinition

Method of JGLegend

Only when legend is user defined: Introduction of a layer definition cell into the legend.

Declaration
void addLayerDefinition (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the linkdefinition
Return Value	void	

addPicture

Method of JGLegend

Only when legend is user defined: Introduction of a graphics cell into legend.

Declaration
void addPicture (java.lang.String picturefile)

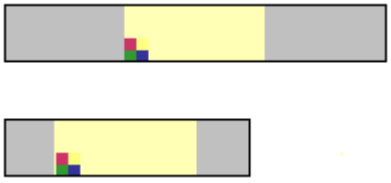
	Data Type	Description
Parameter		
picturefile	java.lang.String	Path to the graphics file.
Return Value	void	

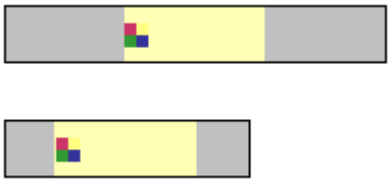
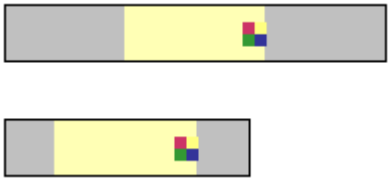
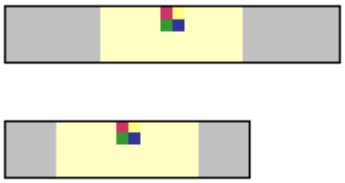
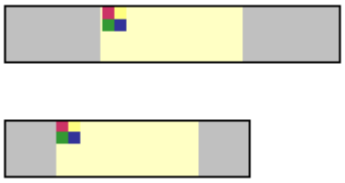
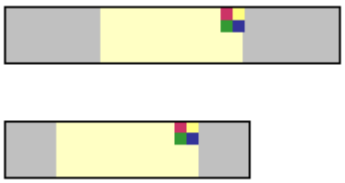
addPicture

Method of JGLegend

Only when legend is user defined: Introduction of a graphics cell with the specified alignment into the legend.

Declaration
void addPicture (java.lang.String picturefile, int alignment)

	Data Type	Description
Parameter		
picturefile	java.lang.String	Path to the graphics file.
alignment	int	Alignment of the graphics.
	Possible Values:	
	ALIGNMENT_BOTTOM_CENTER	The foreground theme is placed at the bottom in the center. 
	ALIGNMENT_BOTTOM_LEFT	The foreground theme is aligned at the bottom on the left hand side. 
	ALIGNMENT_BOTTOM_RIGHT	The foreground theme is aligned at the bottom on the right hand side. 
	ALIGNMENT_CENTER_CENTER	The foreground theme is placed in the vertical and horizontal center. 

	ALIGNMENT_CENTER_LEFT	The foreground theme is placed in the vertical center on the left hand side. 
	ALIGNMENT_CENTER_RIGHT	The foreground theme is placed in the vertical center on the right hand side. 
	ALIGNMENT_TOP_CENTER	The foreground theme is placed at the top in the center. 
	ALIGNMENT_TOP_LEFT	The foreground theme is placed at the top on the left hand side. 
	ALIGNMENT_TOP_RIGHT	The foreground theme is placed at the top on the right hand side. 
Return Value	void	

addRectangle

Method of JGLegend

Only when legend is user defined: Introduction of a legend cell with rectangle and legend text into the legend.
The width of the rectangle is specified by the property LayerRectangleWidthPercent.

Declaration
void addRectangle (java.awt.Color color, java.lang.String legendtext)

	Data Type	Description
Parameter		
color	java.awt.Color	Back ground color of the rectangle.
legendtext	java.lang.String	Legend text of the rectangle
Return Value	void	

addText

Method of JGLegend

Only when legend is user defined: Introduction of a text cell with special font into legend.

Declaration
void addText (java.lang.String text, java.awt.Font font)

	Data Type	Description
Parameter		
text	java.lang.String	Text of the legend cell.
font	java.awt.Font	Font of the legend text cell.
Return Value	void	

addText

Method of JGLegend

Only when legend is user defined: Introduction of a text cell into legend.

Declaration
void addText (java.lang.String text)

	Data Type	Description
Parameter		
text	java.lang.String	Text of the legend cell.
Return Value	void	

openNewRow

Method of JGLegend

Only when legend is user defined: Introduction of a new line into the legend.

Declaration
void openNewRow ()

	Data Type	Description
Return Value	void	

setVisible

Method of JGLegend

By this method you can make the legend visible.

Declaration
void setVisible (boolean visible)

	Data Type	Description
Parameter		
visible	boolean	
Return Value	void	

6.11 JGLevelFilter

Belongs to JGantt

Package name **de.netronic.jgantt**
Implements **de.netronic.common.intface.NelFilter**

This class lets you filter entities which have a specific hierarchy or grouping level.

Constructors of the Class

These constructors allow to generate a filter for a specific hierarchy or grouping level.

JGLevelFilter

Constructor of JGLevelFilter

This constructor allows to generate a filter for a specific hierarchy or grouping level.

Declaration
JGLevelFilter (de.netronic.jgantt.JGantt jGantt, int level)

Parameter	Data Type	Description
jGantt	de.netronic.jgantt.JGantt	The JGantt object, which shall use this filter.
level	int	The level, which is valid for this filter.

Methods of the Class

apply

Method of JGLevelFilter

When JGantt invokes this method, the filter object verifies whether the entity has the specified hierarchy or grouping level.

Declaration
boolean apply (NeIEntity entity)

	Data Type	Description
Parameter		
entity	NeIEntity	Entity that the filter is to apply to.
Return Value	boolean	**true**: the entity has the level level, **false**: the entity does not have the level level.

6.12 JGNodeDesign

Belongs to JGantt

Package name **de.netronic.jgantt**
Extends **java.lang.Object**
Implements **java.io.Externalizable**

This class lets you handle a node design. Together with the node symbols, a node design controls the appearance of nodes.

Properties to Handle the Node Design

Name The name of the node design

Methods to Handle the Node Design

equals(...)	Retrieves, whether or not the object passed equals the node design object.
getInstance(...)	Generates a node design and names it as passed by the string.
getNames()	Retrieves a list of names of all existing node designs.

Properties of the Class

Name

Read Only Property of JGNodeDesign

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the name of the node design.

Accessing Methods
java.lang.String getName()

Methods of the Class

equals

Method of JGNodeDesign

This method retrieves, whether or not the object passed equals the node design object.

Declaration
boolean equals (java.lang.Object parm1)

	Data Type	Description
Parameter		
parm1	java.lang.Object	Object that the node design object is compared to.
Return Value	boolean	Returns **true**, if the object passed equals the node design object or **false**, if it differs.

getInstance

Method of JGNodeDesign

This method generates an instance of the node design and names it as passed by the string.

Declaration
static JGNodeDesign getInstance (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	String passing the name of the instance.
Return Value	JGNodeDesign	Instance of the node design created.

getNames

Method of JGNodeDesign

By this method you can retrieve the names of all existing node designs.

Declaration
static java.lang.String [] getNames ()

	Data Type	Description
Return Value	java.lang.String []	List of names of all existing node designs

6.13 JGSymbol

Belongs to JGantt

Package name **de.netronic.jgantt**
Extends **java.lang.Object**

Implements	**de.netronic.common.intface.NelLabel**

This class lets you handle node symbols. Beside node designs, node symbols contribute essentially to the appearance of nodes.

Properties to Handle Symbols

Name	Name of the symbol

Methods to Handle Symbols

addPropertyChangeListener(...)	Adds a listener for change events of the JGSymbol object.
getInstance(...)	Generates a symbol and names it by the string passed.
getNames()	List of names of all existing symbols.
removePropertyChangeListener(...)	Removes a listener for change events of the JGSymbol object.

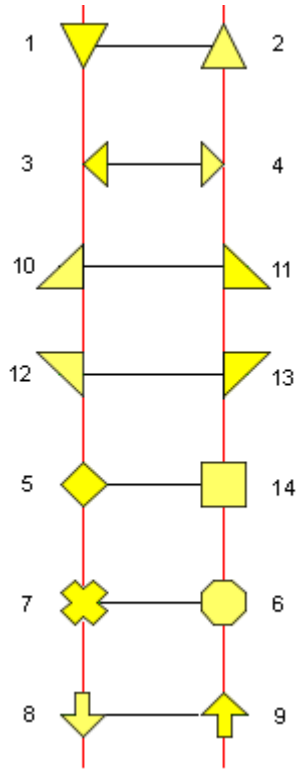
Properties of the Class

Name

Read Only Property of JGSymbol

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By the method of this property you can retrieve the name of the symbol. At present, 14 different names may be returned that are represented by figures:



Accessing Methods
java.lang.String getName()

Methods of the Class

addPropertyChangeListener

Method of JGSymbol

This method lets you add a listener for change events in the JGSymbol object. The listener will be informed each time a property of the JGSymbol object has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

getInstance

Method of JGSymbol

This method generates an instance of the symbol and names it by the string passed.

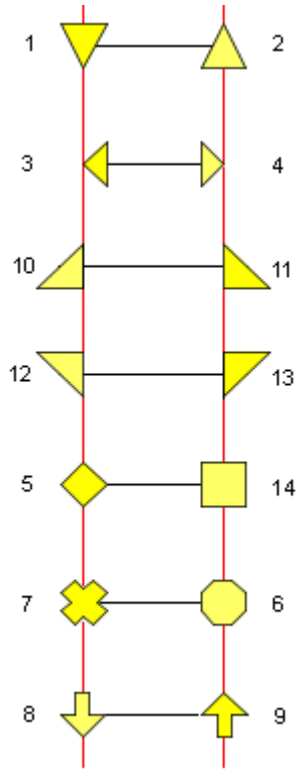
Declaration
static de.netronic.common.intface.NeILabel getInstance (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	String passing the name of the instance.
Return Value	de.netronic.common.intface.NeILabel	Instance of the symbol created.

getNames

Method of JGSymbol

By this method you can retrieve the names of all existing symbols. At present, 14 different names may be returned that are represented by figures:



Declaration
static java.lang.String [] getNames ()

	Data Type	Description
Return Value	java.lang.String []	Array of Strings holding the symbol names.

removePropertyChangeListener

Method of JGSymbol

This method lets you remove a listener for change events in the JGSymbol object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

6.14 JGVertLineGrid

Belongs to JGantt

Package name **de.netronic.jgantt**

This class describes a line grid in a section of the Gantt graph and the histogram. You can specify a line grid type (such as weeks or days etc.), a LineStyle and an imcrement (such as: after 3 weeks).

Constructors of the Class

JGVertLineGrid

Constructor of JGVertLineGrid

This constructor allows to create a vertical grid line of which the type, the line style and an increment need to be specified.

Declaration
JGVertLineGrid (int Type, int Increment, java.awt.Color LineStyle)

Parameter	Data Type	Description
Type	int	Type of the vertical line grid
	Possible Values:	
	VERT_GRID_AUTO	The distance of the grid lines depends on the type of time scale.
	VERT_GRID_BY_MINUTES	The grid lines are displayed at periods of a minute.
	VERT_GRID_BY_SECONDS	The grid lines are displayed at periods of a second.
	VERT_GRID_DAILY	The grid lines are displayed at periods of a day.
	VERT_GRID_HOURLY	The grid lines are displayed at periods of an hour.
	VERT_GRID_MONTHLY	The grid lines are displayed at periods of a month.
	VERT_GRID_NONE	No grid is displayed.
	VERT_GRID_QUARTERLY	The grid lines are displayed at periods of a quarter.
	VERT_GRID_WEEKLY	The grid lines are displayed at periods of a week.
	VERT_GRID_YEARLY	The grid lines are displayed at periods of a year.
Increment	int	Increment of the vertical line grid
LineStyle	java.awt.Color	LineStyle of the vertical line grid

JGVertLineGrid

Constructor of JGVertLineGrid

This constructor allows to create a vertical grid line of which the type and the line style need to be specified.

Declaration
JGVertLineGrid (int Type, java.awt.Color LineStyle)

Parameter	Data Type	Description
Type	int	Type of the vertical line grid
	Possible Values:	
	VERT_GRID_AUTO	The distance of the grid lines depends on the type of time scale.
	VERT_GRID_BY_MINUTES	The grid lines are displayed at periods of a minute.
	VERT_GRID_BY_SECONDS	The grid lines are displayed at periods of a second.
	VERT_GRID_DAILY	The grid lines are displayed at periods of a day.
	VERT_GRID_HOURLY	The grid lines are displayed at periods of an hour.
	VERT_GRID_MONTHLY	The grid lines are displayed at periods of a month.
	VERT_GRID_NONE	No grid is displayed.
	VERT_GRID_QUARTERLY	The grid lines are displayed at periods of a quarter.
	VERT_GRID_WEEKLY	The grid lines are displayed at periods of a week.
	VERT_GRID_YEARLY	The grid lines are displayed at periods of a year.
LineStyle	java.awt.Color	LineStyle of the vertical line grid

JGVertLineGrid

Constructor of JGVertLineGrid

This constructor allows to create a vertical grid line the type of which needs to be specified.

Declaration
JGVertLineGrid (int Type)

Parameter	Data Type	Description
Type	int	Type of the vertical line grid
	Possible Values:	
	VERT_GRID_AUTO	The distance of the grid lines depends on the type of time scale.
	VERT_GRID_BY_MINUTES	The grid lines are displayed at periods of a minute.
	VERT_GRID_BY_SECONDS	The grid lines are displayed at periods of a second.
	VERT_GRID_DAILY	The grid lines are displayed at periods of a day.
	VERT_GRID_HOURLY	The grid lines are displayed at periods of an hour.
	VERT_GRID_MONTHLY	The grid lines are displayed at periods of a month.
	VERT_GRID_NONE	No grid is displayed.
	VERT_GRID_QUARTERLY	The grid lines are displayed at periods of a quarter.
	VERT_GRID_WEEKLY	The grid lines are displayed at periods of a week.
	VERT_GRID_YEARLY	The grid lines are displayed at periods of a year.

Properties of the Class

Increment

Property of JGVertLineGrid

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you define the distance between the vertical grid lines. If you set this property for instance to 3, depending on the property Type the grid lines will appear any 3 minutes, 3 weeks etc.

Accessing Methods
void setIncrement (int newValue)
int getIncrement ()

Also see Type

LineStyle

Property of JGVertLineGrid

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you define the line style of the vertical line grid.

Accessing Methods
void setLineStyle (java.awt.Color newValue)
java.awt.Color getLineStyle ()

Type

Property of JGVertLineGrid

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**VERT_GRID_NONE**

This property lets you define the type of the vertical line grid.

Possible Values	Description
VERT_GRID_AUTO	The distance of the grid lines depends on the type of time scale.
VERT_GRID_BY_MINUTES	The grid lines are displayed at periods of a minute.
VERT_GRID_BY_SECONDS	The grid lines are displayed at periods of a second.
VERT_GRID_DAILY	The grid lines are displayed at periods of a day.
VERT_GRID_HOURLY	The grid lines are displayed at periods of an hour.
VERT_GRID_MONTHLY	The grid lines are displayed at periods of a month.
VERT_GRID_NONE	No grid is displayed.
VERT_GRID_QUARTERLY	The grid lines are displayed at periods of a quarter.
VERT_GRID_WEEKLY	The grid lines are displayed at periods of a week.
VERT_GRID_YEARLY	The grid lines are displayed at periods of a year.

Accessing Methods
void setType (int newValue)
int getType ()

Also see Increment

Visible

Property of JGVertLineGrid

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property defines, whether or not the this vertical line grid is to be displayed.

Accessing Methods
void setVisible (boolean newValue)
boolean isVisible ()

6.15 JGVertLineGrids

Belongs to JGantt

Package name **de.netronic.jgantt**

This class offers methods to handle a collection of the vertical line grids of a time scale section.

Methods of the Class

addGrid

Method of JGVertLineGrids

This method lets you add a vertical line grid to the collection of the time scale section.

Declaration
void addGrid (de.netronic.jgantt.JGVertLineGrid vertLineGrid)

	Data Type	Description
Parameter		
vertLineGrid	de.netronic.jgantt.JGVertLineGrid	Vertical line grid to be added to the time scale section.
Return Value	void	

clear

Method of JGVertLineGrids

This method lets you delete all the vertical line grids of a timescale section.

Declaration
void clear ()

	Data Type	Description
Return Value	void	

getGrid

Method of JGVertLineGrids

This method lets you retrieve a vertical line grid from the collection of the time scale section at the index to be specified. If there is no line grid at the index, null will be returned.

Declaration
de.netronic.jgantt.JGVertLineGrid getGrid (int index)

	Data Type	Description
Parameter		
index	int	Index of the line grid to be retrieved.
Return Value	de.netronic.jgantt.JGVertLineGrid	

getGridCount

Method of JGVertLineGrids

This method will return the number of the vertical line grids in the time scale section.

Declaration
int getGridCount ()

	Data Type	Description
Return Value	int	

removeGrid

Method of JGVertLineGrids

This method lets you delete a vertical line grid from the timescale section.

Declaration
void removeGrid (de.netronic.jgantt.JGVertLineGrid vertLineGrid)

	Data Type	Description
Parameter		
vertLineGrid	de.netronic.jgantt.JGVertLineGrid	Vertical line grid to be deleted from the time scale section.
Return Value	void	

6.16 JPEIJGanttPropertyEditor

Belongs to JGantt

Package name **de.netronic.jgantt.propertyeditor**

This interface contains methods and properties to handle the property editor

Properties of the Interface

DateFormatString

Property of JPEIJGanttPropertyEditor

Typ **java.lang.String**
Bound **no**
Vetoable **no**
Exposure Level **regular**
Default Value **dd.MM.yy**

DateFormat which is used in the property editor for showing date values.

Accessing Methods
void setDateFormatString (java.lang.String newValue)
java.lang.String getDateFormatString ()

Example Code

```
jGantt1.getPropertyEditor().setDateFormatString("dd.MM.yyyy");
```

DocPath

Property of JPEIJGanttPropertyEditor

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

You have to set the path of the JGantt online help by this method for having the online help available in the property editor.

Accessing Methods
void setDocPath (java.lang.String newValue)
java.lang.String getDocPath ()

Example Code

```
myJGantt.getPropertyEditor().setDocPath
("file:C:\\Programme\\Varchart\\JGantt\\doc_eng\\");
```

Enabled

Property of JPEIJGanttPropertyEditor

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines whether or not the property editor can be opened by the pop-up menu of the Gantt graph. It also defines whether a user can open the property editor by pressing <Ctrl><Shift><F10>.

Accessing Methods
void setEnabled (boolean newValue)
boolean isEnabled ()

Example Code

```
jGantt1.getPropertyEditor().setEnabled(true);
```

Methods of the Interface

addNeIDynamicColor

Method of JPEIJGanttPropertyEditor

If you use your own implementation of NeIDynamicColor, you can make known this implementation to the property editor by this method.

Declaration
void addNeIDynamicColor (NeIDynamicColor dynamicColor)

	Data Type	Description
Parameter		
dynamicColor	NeIDynamicColor	Instance of NeIDynamicColor to be made known to the property editor.
Return Value	void	

Example Code

```
jGantt1.getPropertyEditor().addNeIDynamicColor (myDynamicColor);
```

addNeIFilter

Method of JPEIJGanttPropertyEditor

If you use your own implementation of NeIFilter, you can make known this implementation to the property editor by this method.

Declaration
void addNeIFilter (NeIFilter filter)

	Data Type	Description
Parameter		
filter	NeIFilter	Instance of NeIFilter to be made known to the property editor.
Return Value	void	

Example Code

```
jGantt1.getPropertyEditor().addNeIFilter
        (jGantt1.getGanttGraph().getLayerDefinition("freeFloatLayer").getFilter());
```

show

Method of JPEIJGanttPropertyEditor

This method lets you invoke the property editor.

Declaration
void show ()

	Data Type	Description
Return Value	void	

Example Code

```
jGantt1.getPropertyEditor().show();
```

7 Scheduler

The scheduler component serves to calculate dates and floats of activities, mainly in the area of project management.

The Scheduler-Component consists of the below classes:

NeIScheduler	This interface represents the object for scheduling.
NeScheduleAdapter	This class is an adapter class for the interface NeScheduleListener.
NeScheduleEvent	This class represents an event for scheduling.
NeScheduleListener	This is the listener interface for receiving scheduler events.

7.1 NeIScheduler

Belongs to Scheduler

Package name **de.netronic.common.intface**

This interface represents the object for scheduling. It serves to calculate dates, such as the earliest possible date of the project finish, or other values such as the float of single activities etc. The properties allow to set different parameters to the scheduling.

Properties of Scheduling

ActualCompletionReference	Reference to an attribute that holds the degree of completion of an activity
ActualEndReference	Reference to an attribute that holds the actual final date of an activity
ActualRemainingDurationReference	Reference to an attribute that holds the pre-defined duration of an activity that is left until it finishes.
ActualStartReference	Reference to an attribute that holds the actual start date of an activity
AppData	Application data to be scheduled
AutoSchedule	Activating automatic scheduling
CalculatedProjectEnd	Calculated end of the project
CalculatedProjectStart	Calculated start of the project (ms)
Calendar	Calendar object

CompletionReference	Reference to an attribute that receives the calculated degree of completion.
DataDate	Default date for degree of completion
EarlyEndReference	Reference to an attribute that holds the calculated early end of the activity
EarlyStartReference	Reference to an attribute that holds the calculated early start of an activity
FreeFloatReference	Reference to an attribute that holds the calculated free float of the activity.
LateEndReference	Reference to an attribute that holds the calculated late end of the activity
LateStartReference	Reference to an attribute that holds the calculated late start of an activity
LinkLagReference	Reference to an attribute that holds a lag of a link.
LinkProfile	Name of the default profile for links
LinkProfileBy	Link attribute that holds the calendar profile for the lag of the link.
LinkSetName	Entity set that holds the links valid for the scheduling
LinkSourceNodeBy	Link attribute that holds the user-generated identification tag of the source node
LinkTargetNodeBy	Link attribute that holds the user-generated identification tag of the target node
LinkTypeBy	Link attribute that holds the link type.
MandatoryEndReference	Reference to an attribute that holds a mandatory date for the end of the activity.
MandatoryStartReference	Reference to an attribute that holds a mandatory date for the start of the activity.
NodeDurationReference	Reference to an attribute that holds the duration of an activity.
NodeProfile	Name of the default profile for nodes
NodeProfileBy	Node attribute that holds the calendar profile to be applied.
NodeSetName	Entity set that holds the nodes valid for the scheduling
ProjectEnd	End set to the project
ProjectStart	Start set to the project
RemainingDurationReference	Reference to an attribute that receives the calculated duration that remains for the acitvity to finish.

ScheduleNonLeafNodesInHierarchy	Calculation of non-leaf nodes in hierarchy
StartNotEarlierThanDataDate	Start of an activity not earlier than DataDate.
StartNotEarlierThanReference	Reference to an attribute that holds the date of the earliest possible start of the activity.
StartNotLaterThanReference	Reference to an attribute that holds the pre-defined latest possible start of the activity.
TotalFloatReference	Reference to an attribute that will receive the calculated total float.
ValidRangeExpansionStep	Step size by which the calendar is expanded if required

Methods of Scheduling

addScheduleListener(...)	Adds a NeScheduleListener
removeScheduleListener(...)	Removes an NeScheduleListener
schedule()	Performs the scheduling.

Properties of the Interface

ActualCompletionReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds the degree of completion set to an activity. From this figure (percentage), together with ActualEndReference and ActualStartReference the EarlyEnd and LateEnd are derived.

Accessing Methods
void setActualCompletionReference (NeIValueReference newValue)
NeIValueReference getActualCompletionReference ()

Also see ActualStartReference
ActualEndReference

ActualEndReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds the actual final date of an activity. Activities that show a value in this attribute are finished. EarlyEndReference and LateEndReference will be set to this value when scheduled.

Accessing Methods
void setActualEndReference (NeIValueReference newValue)
NeIValueReference getActualEndReference ()

Also see ActualStartReference

ActualRemainingDurationReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds the remaining duration. From the duration and from DataDate the dates of EarlyEnd and LateEnd are derived.

Accessing Methods
void setActualRemainingDurationReference (NeIValueReference newValue)
NeIValueReference getActualRemainingDurationReference ()

ActualStartReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds the actual start date of an activity that has already started. If this attribute contains a value, EarlyStartReference und LateStartReference will also be set to this value.

Accessing Methods
void setActualStartReference (NeIValueReference newValue)
NeIValueReference getActualStartReference ()

Also see ActualEndReference

AppData

Property of NeIScheduler

Typ	**NeIAppData**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Application data that the nodes and the links are taken from for the scheduling.

Accessing Methods
void setAppData (NeIAppData newValue)
NeIAppData getAppData ()

AutoSchedule

Property of NeIScheduler

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you switch automatic scheduling on or off.

Accessing Methods
void setAutoSchedule (boolean newValue)
boolean isAutoSchedule ()

CalculatedProjectEnd

Read Only Property of NeIScheduler

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the calculated end of the project. Unit: milliseconds since 1.1.1970.

Accessing Methods
long getCalculatedProjectEnd()

Also see CalculatedProjectStart

CalculatedProjectStart

Read Only Property of NeIScheduler

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the calculated start of the project. Unit: milliseconds since 1.1.1970.

Accessing Methods
long getCalculatedProjectStart()

Also see CalculatedProjectEnd

Calendar

Property of NeIScheduler

Typ	**NeICalendar**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Calendar object, the profiles of which with their working and non-working periods shall be considered during the scheduling procedure.

Accessing Methods
void setCalendar (NeICalendar newValue)
NeICalendar getCalendar ()

CompletionReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that receives the degree of completion calculated from actual Start, dataDate and earlyEnd. It is only calculated if actualStart was set, i.e., when the activity has started.

Accessing Methods
void setCompletionReference (NeIValueReference newValue)
NeIValueReference getCompletionReference ()

DataDate

Property of NeIScheduler

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**DATE_NOT_SET**

Default value for DataDate which is used for nodes that do not have a DataDate of their own. The DataDate is the date for which the degree of completion is to be calculated and that pre-defined values such as ActualCompletionReference and ActualRemainingDurationReference refer to.

Possible Values	Description
DATE_NOT_SET	Unsets a date.

Accessing Methods
void setDataDate (long newValue)
long getDataDate ()

EarlyEndReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute to which the scheduler will store the calculated early end of the activity.

Accessing Methods
void setEarlyEndReference (NeIValueReference newValue)
NeIValueReference getEarlyEndReference ()

EarlyStartReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute to which the scheduler will store the calculated early start of the activity.

Accessing Methods
void setEarlyStartReference (NeIValueReference newValue)
NeIValueReference getEarlyStartReference ()

FreeFloatReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute to which the scheduler writes the calculated free float of the activity. The free float of an activity is the time by which an activity can be moved without altering the position of other activities.

Accessing Methods
void setFreeFloatReference (NeIValueReference newValue)
NeIValueReference getFreeFloatReference ()

LateEndReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute to which the scheduler will store the calculated late end of the activity.

Accessing Methods
void setLateEndReference (NeIValueReference newValue)
NeIValueReference getLateEndReference ()

Also see LateStartReference

LateStartReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute to which the scheduler will store the calculated late start of the activity.

Accessing Methods
void setLateStartReference (NeIValueReference newValue)
NeIValueReference getLateStartReference ()

Also see LateEndReference

LinkLagReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds a lag (a duration) for a link.

Accessing Methods
void setLinkLagReference (NeIValueReference newValue)
NeIValueReference getLinkLagReference ()

Also see LinkProfileBy

LinkProfile

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the default profile, which is used for links that lack an own profile.

Accessing Methods
void setLinkProfile (java.lang.String newValue)
java.lang.String getLinkProfile ()

LinkProfileBy

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of an attribute, which contains the calendar profile with the lag of the link.

Accessing Methods
void setLinkProfileBy (java.lang.String newValue)
java.lang.String getLinkProfileBy ()

Also see LinkLagReference

LinkSetName

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the entity set that holds the links valid for the scheduling.

Accessing Methods
void setLinkSetName (java.lang.String newValue)
java.lang.String getLinkSetName ()

LinkSourceNodeBy

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the link attribute that holds the user-generated identification tag of the source node (the node preceding the link).

Accessing Methods
void setLinkSourceNodeBy (java.lang.String newValue)
java.lang.String getLinkSourceNodeBy ()

LinkTargetNodeBy

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the link attribute that holds the user-generated identification tag of the target node (the node succeeding the link).

Accessing Methods
void setLinkTargetNodeBy (java.lang.String newValue)
java.lang.String getLinkTargetNodeBy ()

LinkTypeBy

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the link attribute that holds the link type. The link type can be set to the below values:
FS = finish-start (default)
FF = finish-finish
SS = start-start
SF = start-finish
These values specify what dates of the target node and the source node depend on each other and how they do this. Other values than the ones listed above are interpreted as "FS".

Accessing Methods
void setLinkTypeBy (java.lang.String newValue)
java.lang.String getLinkTypeBy ()

MandatoryEndReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds a mandatory date for the end of the activity. It influences the scheduling process in a similar way as ActualEndReference. If the two dates differ, ActualEndReference will be given priority.

Accessing Methods
void setMandatoryEndReference (NeIValueReference newValue)
NeIValueReference getMandatoryEndReference ()

MandatoryStartReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds a mandatory date for the start of the activity. It influences the scheduling process in a similar way as ActualStartReference. If the two dates differ, ActualStartReference will be given priority.

Accessing Methods
void setMandatoryStartReference (NeIValueReference newValue)
NeIValueReference getMandatoryStartReference ()

NodeDurationReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds the duration of the node. The end of an activity is calculated from the start date plus the duration of the activity, considering the calendar profile present.

Accessing Methods
void setNodeDurationReference (NeIValueReference newValue)
NeIValueReference getNodeDurationReference ()

NodeProfile

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the default profile, which is used for nodes that lack an own profile.

Accessing Methods
void setNodeProfile (java.lang.String newValue)
java.lang.String getNodeProfile ()

NodeProfileBy

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the calendar profile the working and non-working periods of which are considered in the calculation.

Accessing Methods
void setNodeProfileBy (java.lang.String newValue)
java.lang.String getNodeProfileBy ()

NodeSetName

Property of NeIScheduler

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the entity set that holds the activities valid for the scheduling.

Accessing Methods
void setNodeSetName (java.lang.String newValue)
java.lang.String getNodeSetName ()

ProjectEnd

Property of NeIScheduler

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**DATE_NOT_SET**

End of the project. This date is to be set in order to calculate the project start. Unit: milliseconds since 1.1.1970.

You can set both, the project start and the project end. Redundant time will result in positive float values for the activities. If there is a lack of time, the float values will be negative.

Possible Values	Description
DATE_NOT_SET	Unsets a date.

Accessing Methods
void setProjectEnd (long newValue)
long getProjectEnd ()

Also see ProjectStart

ProjectStart

Property of NeIScheduler

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**DATE_NOT_SET**

Start of the project. This date is to be set in order to calculate the project finish. Unit: milliseconds since 1.1.1970.

You can set both, the project start and the project end. Redundant time will result in positive float values for the activities. If there is a lack of time, the float values will be negative.

Possible Values	Description
DATE_NOT_SET	Unsets a date.

Accessing Methods
void setProjectStart (long newValue)
long getProjectStart ()

Also see ProjectEnd

RemainingDurationReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that receives the calculated duration that remains for the acitvity to finish. If for example an activity starts on the 1st of April and finishes on the 30th, the calculation for a dataDate of 12th of April will deliver a result of 18, which represents the difference between earlyEnd and dataDate, if actualStart has been set.

Accessing Methods
void setRemainingDurationReference (NeIValueReference newValue)
NeIValueReference getRemainingDurationReference ()

ScheduleNonLeafNodesInHierarchy

Property of NeIScheduler

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property sets or retrieves whether in the case of a hierarchy the dates of nodes, which are not leaf nodes, will be calculated.

Accessing Methods
void setScheduleNonLeafNodesInHierarchy (boolean newValue)
boolean isScheduleNonLeafNodesInHierarchy ()

StartNotEarlierThanDataDate

Property of NeIScheduler

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property sets or retrieves whether an activity should not start earlier than DataDate. If you set this property to **true**, the calculated early start and late start dates will be set to DataDate, in case they are earlier than DataDate.

Accessing Methods
void setStartNotEarlierThanDataDate (boolean newValue)
boolean hasStartNotEarlierThanDataDate ()

StartNotEarlierThanReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that holds the date of the earliest possible start of the activity.

Accessing Methods
void setStartNotEarlierThanReference (NeIValueReference newValue)
NeIValueReference getStartNotEarlierThanReference ()

StartNotLaterThanReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute to which the date of the latest possible start of the activity was set.

Accessing Methods
void setStartNotLaterThanReference (NeIValueReference newValue)
NeIValueReference getStartNotLaterThanReference ()

TotalFloatReference

Property of NeIScheduler

Typ	**NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Reference to an attribute that will receive the calculated total float. The total float of an activity is the time by which an activity can be moved without altering the position of the project finish.

Accessing Methods
void setTotalFloatReference (NeIValueReference newValue)
NeIValueReference getTotalFloatReference ()

ValidRangeExpansionStep

Property of NeIScheduler

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**NO_VALID_RANGE_EXPANSION**

This property specifies the step size by which the scheduler expands the calendar if the valid range of the calendar is too small.
The unit of this property is milliseconds.

Possible Values	Description
NO_VALID_RANGE_EXPANSION	Calendar is not expanded automatically by the scheduler

Accessing Methods
void setValidRangeExpansionStep (long newValue)
long getValidRangeExpansionStep ()

Methods of the Interface

addScheduleListener

Method of NeIScheduler

This method lets you add a NeScheduleListener to the listener list.

Declaration
void addScheduleListener (de.netronic.common.event.NeScheduleListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.NeScheduleListener	Listener to be added
Return Value	void	

removeScheduleListener

Method of NeIScheduler

This method lets you remove a NeScheduleListener from the listener list.

Declaration
void removeScheduleListener (de.netronic.common.event.NeScheduleListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.NeScheduleListener	Listener to be removed
Return Value	void	

schedule

Method of NeIScheduler

This method lets you perform the scheduling. If the start of the project has been set, the finish will be calculated and vice versa: if the finish has been set, the start of the project will be calculated. For the activities, early and late dates as well as floats are calculated.

Declaration
void schedule ()

	Data Type	Description
Return Value	void	

7.2 NeScheduleAdapter

Belongs to Scheduler

Package name	**de.netronic.common.event**
Extends	**java.lang.Object**
Implements	**de.netronic.common.event.NeScheduleListener**

This class is an adapter class for the interface NeScheduleListener. Its methods are empty. This class exists as convenience for creating listener objects.

7.3 NeScheduleEvent

Belongs to Scheduler

Package name	**de.netronic.common.event**

This class represents an event for scheduling.

Properties that Define an Event

CalculatedEnd	End of the project calculated by the scheduler object.
CalculatedStart	Start of the project calculated by the scheduler object.
End	End of the project set.
Start	Start of the project set.
Status	State to indicate circular dependencies.

Constructors of the Class

NeScheduleEvent

Constructor of NeScheduleEvent

By this constructor you can create an event that has three out of six parameters. Those parameters have been omitted that only deliver values with the event NeScheduleListener.scheduleEnd.

Declaration
NeScheduleEvent (java.lang.Object source, long start, long end)

Parameter	Data Type	Description
source	java.lang.Object	The source of the event (which is the scheduler object).
start	long	Assigned start of the project.
end	long	Assigned end of the project.

NeScheduleEvent

Constructor of NeScheduleEvent

By this constructor you can create an event with all posible parameters set. The parameters calculatedStart, calculatedEnd and status only deliver values with the event NeScheduleListener.scheduleEnd.

Declaration
NeScheduleEvent (java.lang.Object source, long start, long end, long calculatedStart, long calculatedEnd, int status)

Parameter	Data Type	Description
source	java.lang.Object	The source of the event (which is the scheduler object).
start	long	Assigned start of the project.
end	long	Assigned end of the project.
calculatedStart	long	Start of the project calculated by the scheduler object. This parameter passes a value only on the event NeScheduleListener.scheduleEnd.
calculatedEnd	long	End of the project calculated by the scheduler object. This parameter passes a value only on the event NeScheduleListener.scheduleEnd.
status	int	This parameter delivers information whether or not in a project circular dependencies exist. It passes a value only on the event NeScheduleListener.scheduleEnd.
	Possible Values:	
	STATUS_OK	Circular dependencies do not exist in the project.
	STATUS_STRUCTURE_-HAS_CYCLES	Circular dependencies exist in the project.

NeScheduleEvent

Constructor of NeScheduleEvent

By this constructor you can create an event that delivers information on its source (that is, the scheduler object). Additional information you can set via the properties.

Declaration
NeScheduleEvent (java.lang.Object source)

Parameter	Data Type	Description
source	java.lang.Object	The source of the event (which is the scheduler object).

Properties of the Class

CalculatedEnd

Property of NeScheduleEvent

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This proeperty describes the end of the project calculated by the scheduler object. It passes a value only on the event NeScheduleListener.scheduleEnd.

Accessing Methods
void setCalculatedEnd (long newValue)
long getCalculatedEnd ()

Also see CalculatedStart

CalculatedStart

Property of NeScheduleEvent

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This proeperty describes the start of the project calculated by the scheduler object. It passes a value only on the event NeScheduleListener.scheduleEnd.

Accessing Methods
void setCalculatedStart (long newValue)
long getCalculatedStart ()

Also see CalculatedEnd

End

Property of NeScheduleEvent

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

End of the project set.

Accessing Methods
void setEnd (long newValue)
long getEnd ()

Also see Start

Start

Property of NeScheduleEvent

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Start of the project set.

Accessing Methods
void setStart (long newValue)
long getStart ()

Also see End

Status

Property of NeScheduleEvent

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property indicates whether or not circular dependencies are present in the project. It passes a value only on the event NeScheduleListener.scheduleEnd.

Possible Values	Description
STATUS_OK	Circular dependencies do not exist in the project.
STATUS_STRUCTURE_HAS_CYCLES	Circular dependencies exist in the project.

Accessing Methods
void setStatus (int newValue)
int getStatus ()

7.4 NeScheduleListener

Belongs to Scheduler

Package name **de.netronic.common.event**

This is the listener interface for receiving scheduler events. The class that is interested in processing a scheduler event either implements this interface, (including the methods it contains), or extends the SchedulerAdapter class (overriding only the methods of interest). The listener object is then registered with the scheduler using the scheduler's addListener method. A scheduler event is generated at the beginning and at the end of a scheduling process, independently of whether the scheduling is performed automatically or whether it is invoked by the method **schedule**.

Methods of the Schedule Listener

Methods of the Interface

scheduleEnd

Method of NeScheduleListener

This method is invoked after the scheduling has finished, independently of whether the scheduling is performed automatically or whether it is invoked by the method **schedule**.

Declaration
void scheduleEnd (NeScheduleEvent event)

	Data Type	Description
Parameter		
event	NeScheduleEvent	The event passes values of the project start and end that have been set, of the calculated values of the project start and end, and an information on whether the project does or does not contain circular dependencies.
Return Value	void	

scheduleStart

Method of NeScheduleListener

This method is invoked at the beginning of the scheduling, independently of whether the scheduling is performed automatically or whether it is invoked by the method **schedule**.

Declaration
void scheduleStart (NeScheduleEvent event)

	Data Type	Description
Parameter		
event	NeScheduleEvent	The event passes values of the project start and end that have been set.
Return Value	void	

8 Table

This component offers classes and interfaces to specify the table content. The table content comprises the structure as well as the appearance of the table.

A table is made of rows and columns. In a row, the values in the attributes of entities can be displayed. The content of a row can be described by a row definition. Several rows of the same sort form a row type and use the same row definition.

A row defintion has a priority and may hold a filter. Whether a row definition will apply to a row depends of the priority and of the result obtained by the filter conditions.

A row consist of fields, where a single field may stretch across several table columns. So a row definition is composed by several field definitions.

The Table-Component consists of the below classes:

NeContentsDefinition	This class serves to describe the contents of the table in a Gantt diagram.
NeContentsDefinitionEvent	This class offers properties and methods to handle events of calendars.
NeContentsDefinitionEventListener	This is the listener interface for receiving profile change events.
NeFieldDefinition	A field of a table basically represents the area of the intersection between a column and a row.
NeFieldStyle	The objects of this class serve to describe the appearance of fields.
NeIFieldDefinition	A field definition describes the content and the appearance of a field.
NeITable	This interface comprises methods and properties to handle the table.
NeRowDefinition	The contents of a row in a table can be described by a row definition.

8.1 NeContentsDefinition

Belongs to Table

Package name **de.netronic.common.contentsdefinition**

This class serves to describe the contents of the table in a Gantt diagram. Beside, it can be used to define the structure of the entity editor (see class NeEntityEditor).

A content definition comprises a set of row definitions. The set holds at least one row definition, that can be accessed by the property "DefaultRowDefinition". The latter in addition defines the number of table columns: the number of fields in the DefaultRowDefinition equals the number of columns in the table.

Common Properties of Contents Definitions

DatePickerEnabled	Toggles the date picker

Properties to Manage Row Definitions

DefaultRowDefinition	Retrieves the default row definition.
RowDefinitionCount	Number of row definition in a contents definition

Methods for Internal Use only

evt(...)	For internal use only
fieldDefinitionAdded(...)	For internal use only
fieldDefinitionChanged(...)	For internal use only
fieldDefinitionRemoved(...)	For internal use only
fieldDefinitionStripePropertyChanged(...)	For internal use only

Methods to Manage Row Definitions

addContentsDefinitionEventListener(...)	Adds a listener for change events in the content definition object.
addRowDefinition(...)	Adds a row definition to a contents definition
getRowDefinition(...)	Retrieves the row definition of an entity.
iterateRowDefinitions()	Retrieves an iterator object of all row definititions.
removeContentsDefinitionEventListener(...)	Removes an existing listener for events.
removeRowDefinition(...)	Removes a row definition from the contents definition.

Constructors of the Class

NeContentsDefinition

Constructor of NeContentsDefinition

This constructor lets you create a contents definition. It automatically generates a default row defintion of the priority **0**, which is added to the contents definition.

Declaration
NeContentsDefinition ()

Properties of the Class

DatePickerEnabled

Property of NeContentsDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you set or retrieve, whether on editing date fields the default text editor (**false**) or the integrated date picker is to be used (**true**).

Accessing Methods
void setDatePickerEnabled (boolean newValue)
boolean isDatePickerEnabled ()

DefaultRowDefinition

Read Only Property of NeContentsDefinition

Typ	**NeRowDefinition**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the default row definition. It applies if no other row definition can be found to apply. Its filter therefore set to **null** and cannto be modified. Its priority is also set to **0**, but can be raised.

The default row definition in addition defines the number of table columns: the number of fields in it equals the number of columns in the table.

Accessing Methods
NeRowDefinition getDefaultRowDefinition()

RowDefinitionCount

Read Only Property of NeContentsDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the number of row definitions in a contents definition. It includes the default row definition.

Accessing Methods
int getRowDefinitionCount()

Methods of the Class

addContentsDefinitionEventListener

Method of NeContentsDefinition

This method adds a listener for change events in the contents definition object. The listener will be informed each time the definition was changed.

Declaration
void addContentsDefinitionEventListener (de.netronic.common.event.-NeContentsDefinitionEventListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeContentsDefinitionEventListener	Listener to be added.
Return Value	void	

Also see removeContentsDefinitionEventListener

addRowDefinition

Method of NeContentsDefinition

This method adds a row definition to a contents definition

Declaration
boolean addRowDefinition (NeRowDefinition rowDefinition)

	Data Type	Description
Parameter		
rowDefinition	NeRowDefinition	Row definition to be added.
Return Value	boolean	**False** if the row definition was already present in the contents definition and therefore could not be added, otherwise **true**.

Also see removeRowDefinition

evt

Method of NeContentsDefinition

For internal use only

Declaration
void evt (de.netronic.common.event.NeFieldDefinitionStripeEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeFieldDefinitionStripeEvent	For internal use only
Return Value	void	

fieldDefinitionAdded

Method of NeContentsDefinition

For internal use only

Declaration
void fieldDefinitionAdded (de.netronic.common.event.NeFieldDefinitionStripeEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeFieldDefinitionStripeEvent	For internal use only
Return Value	void	

Also see fieldDefinitionChanged
fieldDefinitionRemoved

fieldDefinitionChanged

Method of NeContentsDefinition

For internal use only

Declaration
void fieldDefinitionChanged (de.netronic.common.event.NeFieldDefinitionStripeEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeFieldDefinitionStripeEvent	For internal use only
Return Value	void	

Also see fieldDefinitionAdded
fieldDefinitionRemoved

fieldDefinitionRemoved

Method of NeContentsDefinition

For internal use only

Declaration
void fieldDefinitionRemoved (de.netronic.common.event.NeFieldDefinitionStripeEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeFieldDefinitionStripeEvent	For internal use only
Return Value	void	

Also see fieldDefinitionAdded
fieldDefinitionChanged

fieldDefinitionStripePropertyChanged

Method of NeContentsDefinition

For internal use only

Declaration
void fieldDefinitionStripePropertyChanged (de.netronic.common.event.-NeFieldDefinitionStripeEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeFieldDefinitionStripeEvent	For internal use only
Return Value	void	

getRowDefinition

Method of NeContentsDefinition

This method retrieves the row definition of an entity that is used to display the entity in a table row. Priorities and the results of filters of all row definitions are considered. The row definition is selected according to the below criteria:

First, all filters are evaluated in the sequence of their descending priority, until one of them is found to return **true**. If no row definition returns **true**, then a row definition is searched for that has the highest priority but no filter. If no such row definition exists, the default row definition will be selected, ensuring that there is a row definition to apply.

Declaration
NeRowDefinition getRowDefinition (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity fo which a row definition is to be found.
Return Value	NeRowDefinition	

iterateRowDefinitions

Method of NeContentsDefinition

This method retrieves an iterator object of all row definititions of the content definition. The iterator also includes the default row definition.

Declaration
java.util.Iterator iterateRowDefinitions ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object of all row definitions, including the default row definition.

removeContentsDefinitionEventListener

Method of NeContentsDefinition

Removes an existing listener for events in the contents definition object.

Declaration
void removeContentsDefinitionEventListener (de.netronic.common.event.-NeContentsDefinitionEventListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeContentsDefinitionEventListener	Listener to be removed.
Return Value	void	

Also see addContentsDefinitionEventListener

removeRowDefinition

Method of NeContentsDefinition

This method removes a row definition from the contents definition. The default row definition cannot be removed.

Declaration
boolean removeRowDefinition (NeRowDefinition rowDefinition)

	Data Type	Description
Parameter		
rowDefinition	NeRowDefinition	Row definition to be removed. The default row definition must not be entered here.
Return Value	boolean	**False**, if the row definition could not be removed (for example if it was not contained in the contents definition or if the default row definition was addressed), otherwise **true**.

Also see addRowDefinition

8.2 NeContentsDefinitionEvent

Belongs to Table

Package name **de.netronic.common.event**
Extends **java.util.EventObject**

This class offers properties and methods to handle events of calendars.

Properties to Handle Events

FieldDefinition	Retrieves the field definition affected by the event.
NewValue	Retrieves the new value of the property changed.
OldValue	Retrieves the former value of the property changed.
PropertyName	Retrieves the property that has changed.
RowDefinition	Retrieves the row definition affected by the event.

Constructors of the Class

NeContentsDefinitionEvent

Constructor of NeContentsDefinitionEvent

This class lets you generate an event that holds information on the source of the event and the profile affected.

Declaration
NeContentsDefinitionEvent (java.lang.Object source, de.netronic.common.contentsdefinition.-NeRowDefinition rowDefinition, de.netronic.common.contentsdefinition.NeFieldDefinition fieldDefinition, java.lang.String propertyName, java.lang.Object oldValue, java.lang.Object newValue)

Parameter	Data Type	Description
source	java.lang.Object	Object that represents the source of the event.
rowDefinition	de.netronic.common.contentsdefinition.NeRowDefinition	Row definition affected
fieldDefinition	de.netronic.common.contentsdefinition.NeFieldDefinition	Field definition affected
propertyName	java.lang.String	Name of the property the value of which was changed
oldValue	java.lang.Object	Former value of the property
newValue	java.lang.Object	New value of the property

Properties of the Class

FieldDefinition

Read Only Property of NeContentsDefinitionEvent

Typ	**de.netronic.common.contentsdefinition.NeFieldDefinition**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the field defintion affected by the event.

Accessing Methods
de.netronic.common.contentsdefinition.NeFieldDefinition getFieldDefinition()

NewValue

Read Only Property of NeContentsDefinitionEvent

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the new value of the property changed.

Accessing Methods
java.lang.Object getNewValue()

OldValue

Read Only Property of NeContentsDefinitionEvent

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the former value of the property changed.

Accessing Methods
java.lang.Object getOldValue()

PropertyName

Read Only Property of NeContentsDefinitionEvent

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the property changed.

Accessing Methods
java.lang.String getPropertyName()

RowDefinition

Read Only Property of NeContentsDefinitionEvent

Typ	**de.netronic.common.contentsdefinition.NeRowDefinition**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the row defintion affected by the event.

Accessing Methods
de.netronic.common.contentsdefinition.NeRowDefinition getRowDefinition()

8.3 NeContentsDefinitionEventListener

Belongs to Table

Package name **de.netronic.common.event**
Extends **java.util.EventListener**

This is the listener interface for receiving profile change events. The class that is interested in processing a calendar change event will implement this interface, including the methods it contains.The listener object will then be registered with the object using the object's addListener method. A calendar change event is generated after creating, modifying or deleting an object.

Event Methods

fieldDefinitionAdded(...)	Event method on adding of a field definition
fieldDefinitionChanged(...)	Event method on change of a field definition
fieldDefinitionDeleted(...)	Event method on deleting a field definition
rowDefinitionAdded(...)	Event method on adding of a row definition

Methods of the Interface

fieldDefinitionAdded

Method of NeContentsDefinitionEventListener

This method is invoked on adding of a field definition

Declaration
void fieldDefinitionAdded (de.netronic.common.event.NeContentsDefinitionEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeContentsDefinitionEvent	Event triggered on adding a field definition
Return Value	void	

fieldDefinitionChanged

Method of NeContentsDefinitionEventListener

This method is invoked on change of a field definition

Declaration
void fieldDefinitionChanged (de.netronic.common.event.NeContentsDefinitionEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeContentsDefinitionEvent	Event triggered on the cahnge of a field definition
Return Value	void	

fieldDefinitionDeleted

Method of NeContentsDefinitionEventListener

This method is invoked on deleting a field definition

Declaration
void fieldDefinitionDeleted (de.netronic.common.event.NeContentsDefinitionEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeContentsDefinitionEvent	Event triggered on adding a field definition
Return Value	void	

rowDefinitionAdded

Method of NeContentsDefinitionEventListener

This method is invoked on adding of a row definition

Declaration
void rowDefinitionAdded (de.netronic.common.event.NeContentsDefinitionEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeContentsDefinitionEvent	Event triggered on adding a row definition
Return Value	void	

rowDefinitionChanged

Method of NeContentsDefinitionEventListener

This method is invoked on change of a row definition

Declaration
void rowDefinitionChanged (de.netronic.common.event.NeContentsDefinitionEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeContentsDefinitionEvent	Event triggered on changing a row definition
Return Value	void	

rowDefinitionDeleted

Method of NeContentsDefinitionEventListener

This method is invoked on deleting a row definition

Declaration
void rowDefinitionDeleted (de.netronic.common.event.NeContentsDefinitionEvent evt)

	Data Type	Description
Parameter		
evt	de.netronic.common.event.-NeContentsDefinitionEvent	Event triggered on deleting a row definition
Return Value	void	

8.4 NeFieldDefinition

Belongs to Table

Package name **de.netronic.common.contentsdefinition**

Implements **de.netronic.common.intface.NeIFieldDefinition**

A field of a table basically represents the area of the intersection between a column and a row. At the same time, a range can be allocated to a field that causes it to stretch across several columns.

A field definition describes the content and the appearance of a field. Objects of this class serve to define fields the contents of which depends of the attribute value of an entity. Therefore, the class contains the property "**AttributeName**".

In addition, to a field a field style and a text string can be assigned.

Common Properties of Field Definitions

AttributeName	Name of the attribute, of which the value is to be displayed.
Caption	Object to be used for text strings
Editable	Can fields be edited?
ExternalEditor	Self written TableCellEditor for this NeFieldDefinition
FieldStyle	Style to define the appearance of the field.
Picture	Object to be used as a picture in the field

Properties for Internal Use Only

Common Methods of Field Definitions

addPropertyChangeListener(...)	Adds a listener for change events in the field definition object.
getEditor(...)	For internal use only.
getRenderer(...)	For internal use only.
getValue(...)	For internal use only.
removePropertyChangeListener(...)	Removes a listener for change events.
setValue(...)	For internal use only.

Methods for Internal Use Only

propertyChange(...)	For internal use only

Constructors of the Class

NeFieldDefinition

Constructor of NeFieldDefinition

This constructor lets you create a field definition. It specifies the name of an entity attribute, of which the value will be displayed, and a field style.

Declaration
NeFieldDefinition (java.lang.String attributeName, NeFieldStyle fieldStyle)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute, of which the value is to be displayed.
fieldStyle	NeFieldStyle	Field style to define the appearance of the field.

NeFieldDefinition

Constructor of NeFieldDefinition

This constructor lets you create a field definition that contains the name of an entity attribute, of which the value is to be displayed.

Declaration
NeFieldDefinition (java.lang.String attributeName)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute, of which the value is to be displayed.

Properties of the Class

AttributeName

Property of NeFieldDefinition

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set the name of an entity attribute, of which the value is to be displayed.

Accessing Methods
void setAttributeName (java.lang.String newValue)
java.lang.String getAttributeName ()

Caption

Property of NeFieldDefinition

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

This property lets you set or retrieve an object, that provides text strings to be displayed in the field.

Accessing Methods
void setCaption (java.lang.Object newValue)
java.lang.Object getCaption ()

Editable

Property of NeFieldDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property lets you set or retrieve, whether or not the fields can be edited.

Accessing Methods
void setEditable (boolean newValue)
boolean isEditable ()

ExternalEditor

Property of NeFieldDefinition

Typ	**javax.swing.table.TableCellEditor**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

This property lets you specify an instance of a class which implements the interface **TableCellEditor** to be the editor of this NeFieldDefinition.
This way, you can set your own editor to the table and the NeEntityEditor.

Accessing Methods
void setExternalEditor (javax.swing.table.TableCellEditor newValue)
javax.swing.table.TableCellEditor getExternalEditor ()

Example Code

```
fieldDefinition.setExternalEditor(new LinkTypeEditor());
```

FieldStyle

Property of NeFieldDefinition

Typ	**NeFieldStyle**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

This property lets you set or retrieve the field style to define the appearance or the field. If a field style was not set in a field definition, the field will appear in the style set in the row definition.

Accessing Methods
void setFieldStyle (NeFieldStyle newValue)
NeFieldStyle getFieldStyle ()

Picture

Property of NeFieldDefinition

Typ	**de.netronic.common.intface.NeIPicture**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve a picture object that may serve as a "background" in table fields, similar to layers in the Gantt graph. As with the layer, "background" is not to be taken literally here as well. Since neither the picture nor the text have to fill the field completely, pictures can be placed beside texts. Also pictures produced by yourself can be used here.

Accessing Methods
void setPicture (de.netronic.common.intface.NeIPicture newValue)
de.netronic.common.intface.NeIPicture getPicture ()

Methods of the Class

addPropertyChangeListener

Method of NeFieldDefinition

This method adds a listener for change events in the field definition object. The listener will be informed each time a property was changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

getEditor

Method of NeFieldDefinition

For internal use only. Retrieves the field editor of an entity. The editor is returned anyway, even if the property "Editable" has been set to **false**.

Declaration
javax.swing.table.TableCellEditor getEditor (de.netronic.common.intface.NeIEntity entity, NeFieldStyle defaultFieldStyle)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity, of which the field editor is to be retrieved.
defaultFieldStyle	NeFieldStyle	Default field style
Return Value	javax.swing.table.TableCellEditor	

getRenderer

Method of NeFieldDefinition

For internal use only. Retrieves the field renderer.

Declaration
javax.swing.table.TableCellRenderer getRenderer (de.netronic.common.intface.NeIEntity entity, NeFieldStyle defaultFieldStyle)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity, of which the field renderer is to be retrieved.
defaultFieldStyle	NeFieldStyle	Default field style
Return Value	javax.swing.table.TableCellRenderer	

getValue

Method of NeFieldDefinition

For internal use only. Retrieves the value of an entity.

Declaration
java.lang.Object getValue (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity, of which the value is to be retrieved.
Return Value	java.lang.Object	

Also see setValue

propertyChange

Method of NeFieldDefinition

For internal use only

Declaration
void propertyChange (java.beans.PropertyChangeEvent propertyChangeEvent)

	Data Type	Description
Parameter		
propertyChangeEvent	java.beans.PropertyChangeEvent	For internal use only
Return Value	void	

removePropertyChangeListener

Method of NeFieldDefinition

Removes an existing listener for change events in the field definition object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

setValue

Method of NeFieldDefinition

For internal use only. Sets the value of an entity.

Declaration
void setValue (de.netronic.common.intface.NeIEntity entity, java.lang.Object value)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity, to which a value is to be set.
value	java.lang.Object	Value to be set to an entity.
Return Value	void	

Also see getValue

8.5 NeFieldStyle

Belongs to Table

Package name **de.netronic.common.contentsdefinition**
Extends **de.netronic.common.beanbase.NeAreaStyle**

The objects of this class serve to describe the appearance of fields. You can adjust the font, the alignment of the field contents, colors and patterns.

Properties to Define the Appearance of Fields

Alignment	Alignment of the field contents
AutoWrapMode	Different automatic break modes for text lines
Font	Font of the text strings in the field
FontWd	For internal use only.
MarginLeft	Left margin of the field
MarginRight	Right margin of the field
TextColor	Color of the texts

Methods to Handle the NeFieldStyle Object

addPropertyChangeListener(...)	Adds a listener for change events in the field style object.
removePropertyChangeListener(...)	Removes a listener for change events.

Constructors of the Class

NeFieldStyle

Constructor of NeFieldStyle

This constructor lets you create a field style by using a template of a different field style. The design of the backgound is set by the parameter **backgroundStyle**. It may be a simple color a well as an areaStyle. Except for the backgroundStyle, all other properties can be modified later on; the backgroundStyle parameter can only be set by a constructor.

Declaration
NeFieldStyle (NeFieldStyle template, java.awt.Color backgroundStyle)

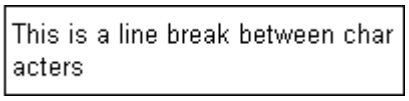
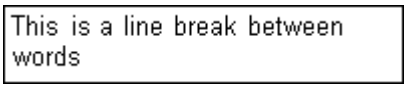
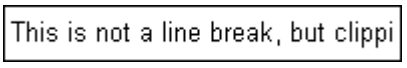
Parameter	Data Type	Description
template	NeFieldStyle	Constuctor template to be copied
backgroundStyle	java.awt.Color	Background color and pattern of the field

NeFieldStyle

Constructor of NeFieldStyle

This constructor lets you create a field style that has a font, an alignment, a wrap mode and colors.

Declaration
NeFieldStyle (int autoWrap, java.awt.Font font, int alignment, java.awt.Color textColor, java.awt.Color areaStyle)

Parameter	Data Type	Description
autoWrap	int	Types of line breaks
	Possible Values:	
	AUTO_WRAP_AT_CHARACTER	Automatic line breaks occur between characters: 
	AUTO_WRAP_AT_WORD	Automatic line breaks occur between words: 
	AUTO_WRAP_OFF	There are no automatic line breaks. To avoid the text to be clipped, line breaks can be set manually by **\n**. 
font	java.awt.Font	Font of the text strings in the field
alignment	int	Alignment of the field contents
	Possible Values:	
	ALIGNMENT_BOTTOM_CENTER	Vertical alignment at the bottom, horizontal alignment in the center.
	ALIGNMENT_BOTTOM_LEFT	Vertical alignment at the bottom, horizontal alignment on the left.
	ALIGNMENT_BOTTOM_RIGHT	Vertical alignment at the bottom, horizontal alignment on the right.
	ALIGNMENT_CENTER_CENTER	Vertical alignment and horizontal alignment in the center.
	ALIGNMENT_CENTER_LEFT	Vertical alignment in the center, horizontal alignment on the left.
	ALIGNMENT_CENTER_RIGHT	Vertical alignment in the center, horizontal alignment on the left
	ALIGNMENT_TOP_CENTER	Vertical alignment at the top, horizontal alignment in the center
	ALIGNMENT_TOP_LEFT	Vertical alignment at the top, horizontal alignment on the left

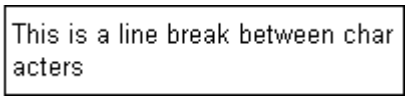
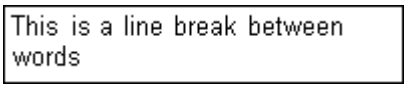
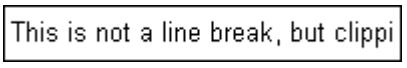
	ALIGNMENT_TOP_RIGHT	Vertical alignment at the top, horizontal alignment on the right
textColor	java.awt.Color	Color of the font in the field
areaStyle	java.awt.Color	Background color of the field

NeFieldStyle

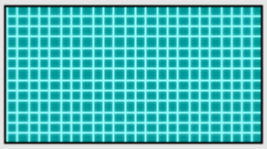
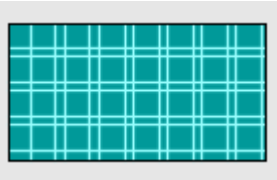
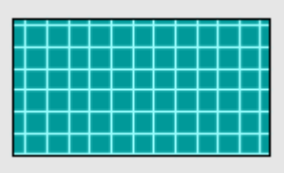
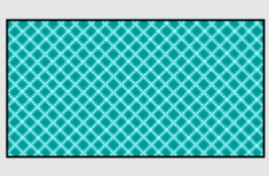
Constructor of NeFieldStyle

This constructor lets you create a field style that has a font, an alignment, a wrap mode, colors and a pattern.

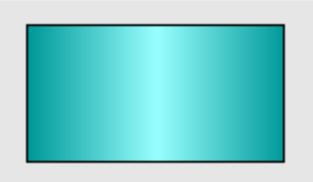
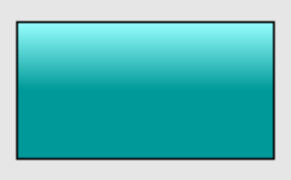
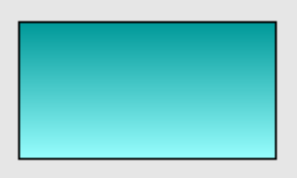
Declaration
NeFieldStyle (int autoWrap, java.awt.Font font, int alignment, java.awt.Color textColor, java.awt.Color backgroundColor, int pattern, java.awt.Color patternColor)

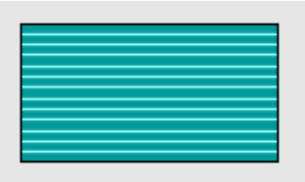
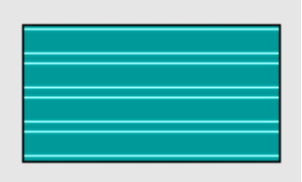
Parameter	Data Type	Description
autoWrap	int	Types of line breaks
	Possible Values:	
	AUTO_WRAP_AT_CHARACTER	Automatic line breaks occur between characters: 
	AUTO_WRAP_AT_WORD	Automatic line breaks occur between words: 
	AUTO_WRAP_OFF	There are no automatic line breaks. To avoid the text to be clipped, line breaks can be set manually by **\n**. 
font	java.awt.Font	Font of the text strings in the field
alignment	int	Alignment of the field contents
	Possible Values:	
	ALIGNMENT_BOTTOM_CENTER	Vertical alignment at the bottom, horizontal alignment in the center.
	ALIGNMENT_BOTTOM_LEFT	Vertical alignment at the bottom, horizontal alignment on the left.
	ALIGNMENT_BOTTOM_RIGHT	Vertical alignment at the bottom, horizontal alignment on the right.
	ALIGNMENT_CENTER_CENTER	Vertical alignment and horizontal alignment in the center.
	ALIGNMENT_CENTER_LEFT	Vertical alignment in the center, horizontal alignment on the left.
	ALIGNMENT_CENTER_RIGHT	Vertical alignment in the center, horizontal alignment on the left
	ALIGNMENT_TOP_CENTER	Vertical alignment at the top, horizontal alignment in the center
	ALIGNMENT_TOP_LEFT	Vertical alignment at the top, horizontal alignment on the left

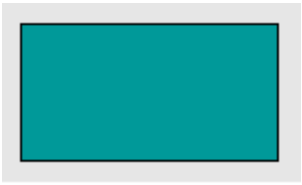
	ALIGNMENT_TOP_RIGHT	Vertical alignment at the top, horizontal alignment on the right
textColor	java.awt.Color	Color of the font in the field
backgroundColor	java.awt.Color	Background color of the field
pattern	int	Pattern to be displayed
	Possible Values:	
	FILL_PATTERN_BACKSLASH	Hatching pattern that shows backslash lines with spaces of intermediate width.
		
	FILL_PATTERN_-BACKSLASH_DOUBLE	Hatching pattern that shows double backslash lines with spaces of intermediate width.
		
	FILL_PATTERN_-BACKSLASH_WIDE	Hatching pattern that shows backslash lines with spaces of large width.
		

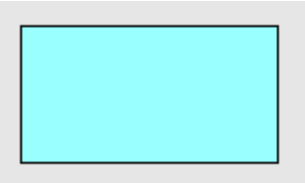
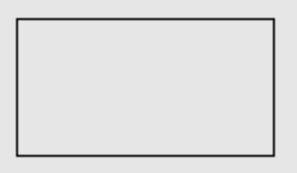
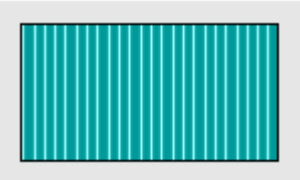
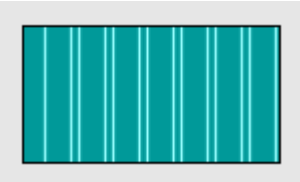
FILL_PATTERN_CROSS	Hatching pattern that shows horizontal and vertical cross lines with spaces of intermediate width. 
FILL_PATTERN_CROSS_DOUBLE	Hatching pattern that shows horizontal and vertical double cross lines with spaces of intermediate width. 
FILL_PATTERN_CROSS_WIDE	Hatching pattern that shows horizontal and vertical cross lines with spaces of large width. 
FILL_PATTERN_DIAGONAL	Hatching pattern that shows diagonal cross lines with spaces of intermediate width. 

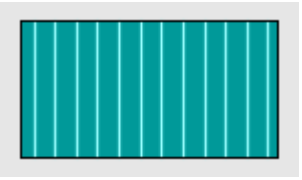
FILL_PATTERN_- DIAGONAL_DOUBLE	Hatching pattern that shows diagonal double cross lines with spaces of intermediate width. 
FILL_PATTERN_DIAGONAL_WIDE	Hatching pattern that shows diagonal cross lines with spaces of large width. 
FILL_PATTERN_- GRADIENT_DIAGONAL_DOWN	Vertical color gradient 
FILL_PATTERN_- GRADIENT_DIAGONAL_DOWN	Vertical color gradient 

FILL_PATTERN_- GRADIENT_HORIZONTAL	Horizontal color gradient 
FILL_PATTERN_- GRADIENT_HORIZONTAL_- CONVEX	Horizontally convex color gradient 
FILL_PATTERN_- GRADIENT_LEFT_LIGHTED	Color gradient lighted on the left 
FILL_PATTERN_- GRADIENT_TOP_LIGHTED	Color gradient lighted at the top 
FILL_PATTERN_- GRADIENT_VERTICAL	Vertical color gradient 

FILL_PATTERN_- GRADIENT_VERTICAL_CONVEX	Vertically convex color gradient 
FILL_PATTERN_HORIZONTAL	Hatching pattern that shows horizontal lines with spaces of intermediate width. 
FILL_PATTERN_- HORIZONTAL_DOUBLE	Hatching pattern that shows horizontal double lines with spaces of intermediate width. 
FILL_PATTERN_- HORIZONTAL_WIDE	Hatching pattern that shows horizontal lines with spaces of large width. 

FILL_PATTERN_NONE	Pattern that has no a hatching pattern, which therefore lacks the foreground color. The backgrund color will completely cover the area. 
FILL_PATTERN_SLASH	Hatching pattern that shows slash lines with spaces of intermediate width. 
FILL_PATTERN_SLASH_DOUBLE	Hatching pattern that shows double slash lines with spaces of intermediate width. 
FILL_PATTERN_SLASH_WIDE	Hatching pattern that shows slash lines with spaces of large width. 

FILL_PATTERN_SOLID	The hatching pattern is a solid area. Therefore the foregrund color will completely cover the area. 
FILL_PATTERN_TRANSPARENT	Empty pattern. It neither has a foreground color nor a backgrund color. 
FILL_PATTERN_VERTICAL	Hatching pattern that shows vertical lines with spaces of intermediate width. 
FILL_PATTERN_- VERTICAL_DOUBLE	Hatching pattern that shows vertical double lines with spaces of intermediate width. 

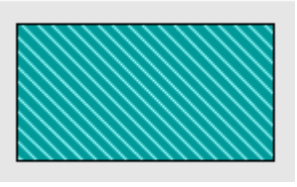
	FILL_PATTERN_VERTICAL_WIDE	Hatching pattern that shows vertical lines with spaces of large width. 
patternColor	java.awt.Color	Color of the pattern

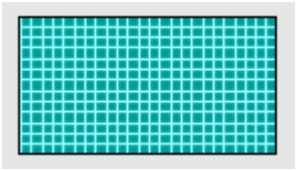
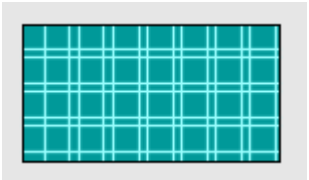
NeFieldStyle

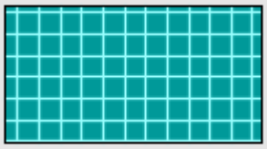
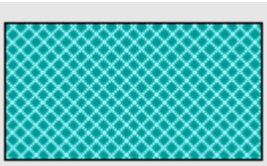
Constructor of NeFieldStyle

This constructor lets you create a field style that has a font, an alignment, colors and a pattern.

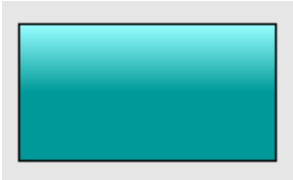
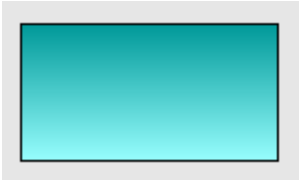
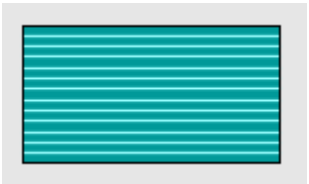
Declaration
NeFieldStyle (java.awt.Font font, int alignment, java.awt.Color textColor, java.awt.Color backgroundColor, int pattern, java.awt.Color patternColor)

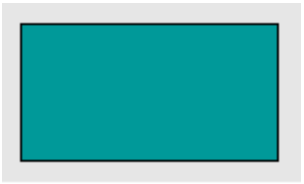
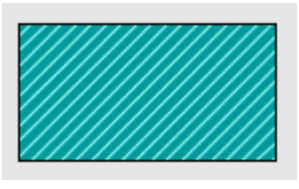
Parameter	Data Type	Description
font	java.awt.Font	Font of the text strings in the field
alignment	int	Alignment of the field contents
	Possible Values:	
	ALIGNMENT_BOTTOM_CENTER	Vertical alignment at the bottom, horizontal alignment in the center.
	ALIGNMENT_BOTTOM_LEFT	Vertical alignment at the bottom, horizontal alignment on the left.
	ALIGNMENT_BOTTOM_RIGHT	Vertical alignment at the bottom, horizontal alignment on the right.
	ALIGNMENT_CENTER_CENTER	Vertical alignment and horizontal alignment in the center.
	ALIGNMENT_CENTER_LEFT	Vertical alignment in the center, horizontal alignment on the left.
	ALIGNMENT_CENTER_RIGHT	Vertical alignment in the center, horizontal alignment on the left
	ALIGNMENT_TOP_CENTER	Vertical alignment at the top, horizontal alignment in the center
	ALIGNMENT_TOP_LEFT	Vertical alignment at the top, horizontal alignment on the left
	ALIGNMENT_TOP_RIGHT	Vertical alignment at the top, horizontal alignment on the right
textColor	java.awt.Color	Color of the font in the field
backgroundColor	java.awt.Color	Background color of the field
pattern	int	Pattern to be displayed
	Possible Values:	
	FILL_PATTERN_BACKSLASH	Hatching pattern that shows backslash lines with spaces of intermediate width.
		

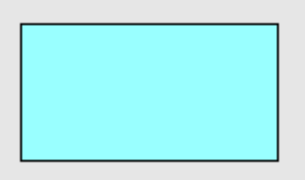
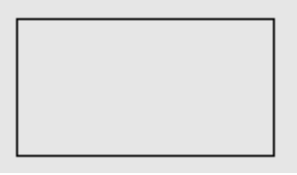
FILL_PATTERN_- BACKSLASH_DOUBLE	Hatching pattern that shows double backslash lines with spaces of intermediate width. 
FILL_PATTERN_- BACKSLASH_WIDE	Hatching pattern that shows backslash lines with spaces of large width. 
FILL_PATTERN_CROSS	Hatching pattern that shows horizontal and vertical cross lines with spaces of intermediate width. 
FILL_PATTERN_CROSS_DOUBLE	Hatching pattern that shows horizontal and vertical double cross lines with spaces of intermediate width. 

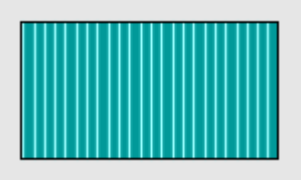
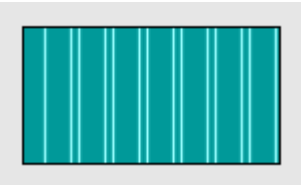
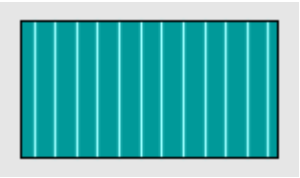
FILL_PATTERN_CROSS_WIDE	Hatching pattern that shows horizontal and vertical cross lines with spaces of large width. 
FILL_PATTERN_DIAGONAL	Hatching pattern that shows diagonal cross lines with spaces of intermediate width. 
FILL_PATTERN_- DIAGONAL_DOUBLE	Hatching pattern that shows diagonal double cross lines with spaces of intermediate width. 
FILL_PATTERN_DIAGONAL_WIDE	Hatching pattern that shows diagonal cross lines with spaces of large width. 

FILL_PATTERN_- GRADIENT_DIAGONAL_DOWN	Vertical color gradient 
FILL_PATTERN_- GRADIENT_DIAGONAL_DOWN	Vertical color gradient 
FILL_PATTERN_- GRADIENT_HORIZONTAL	Horizontal color gradient 
FILL_PATTERN_- GRADIENT_HORIZONTAL_- CONVEX	Horizontally convex color gradient 
FILL_PATTERN_- GRADIENT_LEFT_LIGHTED	Color gradient lighted on the left 

FILL_PATTERN_- GRADIENT_TOP_LIGHTED	Color gradient lighted at the top 
FILL_PATTERN_- GRADIENT_VERTICAL	Vertical color gradient 
FILL_PATTERN_- GRADIENT_VERTICAL_CONVEX	Vertically convex color gradient 
FILL_PATTERN_HORIZONTAL	Hatching pattern that shows horizontal lines with spaces of intermediate width. 

FILL_PATTERN_- HORIZONTAL_DOUBLE	Hatching pattern that shows horizontal double lines with spaces of intermediate width. 
FILL_PATTERN_- HORIZONTAL_WIDE	Hatching pattern that shows horizontal lines with spaces of large width. 
FILL_PATTERN_NONE	Pattern that has no a hatching pattern, which therefore lacks the foreground color. The backgrund color will completely cover the area. 
FILL_PATTERN_SLASH	Hatching pattern that shows slash lines with spaces of intermediate width. 

FILL_PATTERN_SLASH_DOUBLE	Hatching pattern that shows double slash lines with spaces of intermediate width. 
FILL_PATTERN_SLASH_WIDE	Hatching pattern that shows slash lines with spaces of large width. 
FILL_PATTERN_SOLID	The hatching pattern is a solid area. Therefore the foregrund color will completely cover the area. 
FILL_PATTERN_TRANSPARENT	Empty pattern. It neither has a foreground color nor a backgrund color. 

	FILL_PATTERN_VERTICAL	Hatching pattern that shows vertical lines with spaces of intermediate width. 
	FILL_PATTERN_- VERTICAL_DOUBLE	Hatching pattern that shows vertical double lines with spaces of intermediate width. 
	FILL_PATTERN_VERTICAL_WIDE	Hatching pattern that shows vertical lines with spaces of large width. 
patternColor	java.awt.Color	Color of the pattern

NeFieldStyle

Constructor of NeFieldStyle

This constructor lets you create a field style that has a font, an alignment and a background design. The design of the backgound is set by the parameter **backgroundStyle**. It may be a simple color or, alternatively, an areaStyle. Except for the backgroundStyle, all other properties can be modified later on; the backgroundStyle parameter can only be set by a constructor.

Declaration
NeFieldStyle (java.awt.Font font, int alignment, java.awt.Color textColor, java.awt.Color backgroundStyle)

Parameter	Data Type	Description
font	java.awt.Font	Font of the text strings in the field
alignment	int	Alignment of the field contents
	Possible Values:	
	ALIGNMENT_BOTTOM_CENTER	Vertical alignment at the bottom, horizontal alignment in the center.
	ALIGNMENT_BOTTOM_LEFT	Vertical alignment at the bottom, horizontal alignment on the left.
	ALIGNMENT_BOTTOM_RIGHT	Vertical alignment at the bottom, horizontal alignment on the right.
	ALIGNMENT_CENTER_CENTER	Vertical alignment and horizontal alignment in the center.
	ALIGNMENT_CENTER_LEFT	Vertical alignment in the center, horizontal alignment on the left.
	ALIGNMENT_CENTER_RIGHT	Vertical alignment in the center, horizontal alignment on the left
	ALIGNMENT_TOP_CENTER	Vertical alignment at the top, horizontal alignment in the center
	ALIGNMENT_TOP_LEFT	Vertical alignment at the top, horizontal alignment on the left
	ALIGNMENT_TOP_RIGHT	Vertical alignment at the top, horizontal alignment on the right
textColor	java.awt.Color	Color of the font in the field
backgroundStyle	java.awt.Color	Background color and pattern of the field

Properties of the Class

Alignment

Property of NeFieldStyle

Typ **int**
Bound **yes**
Vetoable **no**
Exposure Level **regular**
Default Value **ALIGNMENT_CENTER_CENTER**

This property sets or retrieves the horizontal and vertical alignment of the field contents.

Possible Values	Description
ALIGNMENT_BOTTOM_CENTER	Vertical alignment at the bottom, horizontal alignment in the center.
ALIGNMENT_BOTTOM_LEFT	Vertical alignment at the bottom, horizontal alignment on the left.
ALIGNMENT_BOTTOM_RIGHT	Vertical alignment at the bottom, horizontal alignment on the right.
ALIGNMENT_CENTER_CENTER	Vertical alignment and horizontal alignment in the center.
ALIGNMENT_CENTER_LEFT	Vertical alignment in the center, horizontal alignment on the left.
ALIGNMENT_CENTER_RIGHT	Vertical alignment in the center, horizontal alignment on the left
ALIGNMENT_TOP_CENTER	Vertical alignment at the top, horizontal alignment in the center
ALIGNMENT_TOP_LEFT	Vertical alignment at the top, horizontal alignment on the left
ALIGNMENT_TOP_RIGHT	Vertical alignment at the top, horizontal alignment on the right

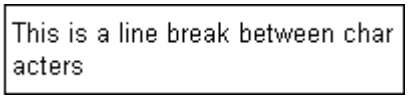
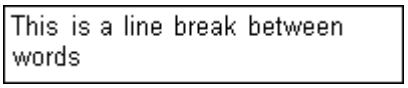
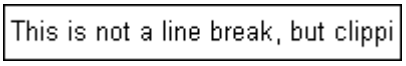
Accessing Methods
void setAlignment (int newValue)
int getAlignment ()

AutoWrapMode

Property of NeFieldStyle

Typ	**int**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO_WRAP_OFF**

This property lets you set different types of automatic line breaks to texts of manually defined tables. For the default table please use the property **set/getTableAutoWrapMode** of the object **JGantt**.

Possible Values	Description
AUTO_WRAP_AT_CHARACTER	Automatic line breaks occur between characters: 
AUTO_WRAP_AT_WORD	Automatic line breaks occur between words: 
AUTO_WRAP_OFF	There are no automatic line breaks. To avoid the text to be clipped, line breaks can be set manually by **\n**. 

Accessing Methods
void setAutoWrapMode (int newValue)
int getAutoWrapMode ()

Font

Property of NeFieldStyle

Typ	**java.awt.Font**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**

This property sets or retrieves the font of the text strings in the field.

Accessing Methods
void setFont (java.awt.Font newValue)
java.awt.Font getFont ()

Also see TextColor

FontWd

Read Only Property of NeFieldStyle

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

For internal use only. Retrieves the font (size in world coordinates) of the text strings in the field.

Accessing Methods
java.awt.Font getFontWd()

MarginLeft

Property of NeFieldStyle

Typ	**int**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**100**

This property sets or retrieves the left margin of a field. Unit: 1/100 mm.

Accessing Methods
void setMarginLeft (int newValue)
int getMarginLeft ()

Also see MarginRight

MarginRight

Property of NeFieldStyle

Typ	**int**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**100**

This property sets or retrieves the right margin of the field. Unit: 1/100 mm.

Accessing Methods
void setMarginRight (int newValue)
int getMarginRight ()

Also see MarginLeft

TextColor

Property of NeFieldStyle

Typ	**java.awt.Color**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**

This property sets or retrieves the color of the text.

Accessing Methods
void setTextColor (java.awt.Color newValue)
java.awt.Color getTextColor ()

Also see Font

Methods of the Class

addPropertyChangeListener

Method of NeFieldStyle

This method adds a listener for change events in the field style object. The listener will be informed each time a property was changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

removePropertyChangeListener

Method of NeFieldStyle

Removes an existing listener for change events in the field style object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

8.6 NeIFieldDefinition

Belongs to Table

Package name **de.netronic.common.intface**

A field definition describes the content and the appearance of a field. Objects of this class serve to define fields the contents of which depends of the attribute value of an entity. Therefore, the class contains the property "**AttributeName**".

In addition, to a field a field style and a text string can be assigned.

Common Properties of Field Definitions

Caption	Object to be used for text strings
Editable	Can fields be edited?
ExternalEditor	By this property you can specify a selfcoded TableCellEditor for a specific table field.
Picture	Object to be used as a picture in the field

Properties for Internal Use only

Common Methods of Field Definitions

getEditor(...)	For internal use only.
getRenderer(...)	For internal use only.

Properties of the Interface

Caption

Property of NeIFieldDefinition

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

This property lets you set or retrieve an object that provides text strings to be displayed in the field.

Accessing Methods
void setCaption (java.lang.Object newValue)
java.lang.Object getCaption ()

Editable

Property of NeIFieldDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property lets you set or retrieve, whether or not the fields can be edited.

Accessing Methods
void setEditable (boolean newValue)
boolean isEditable ()

ExternalEditor

Property of NeIFieldDefinition

Typ	**javax.swing.table.TableCellEditor**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

By this property you can specify a selfcoded TableCellEditor for a specific table field.

Accessing Methods
void setExternalEditor (javax.swing.table.TableCellEditor newValue)
javax.swing.table.TableCellEditor getExternalEditor ()

Example Code

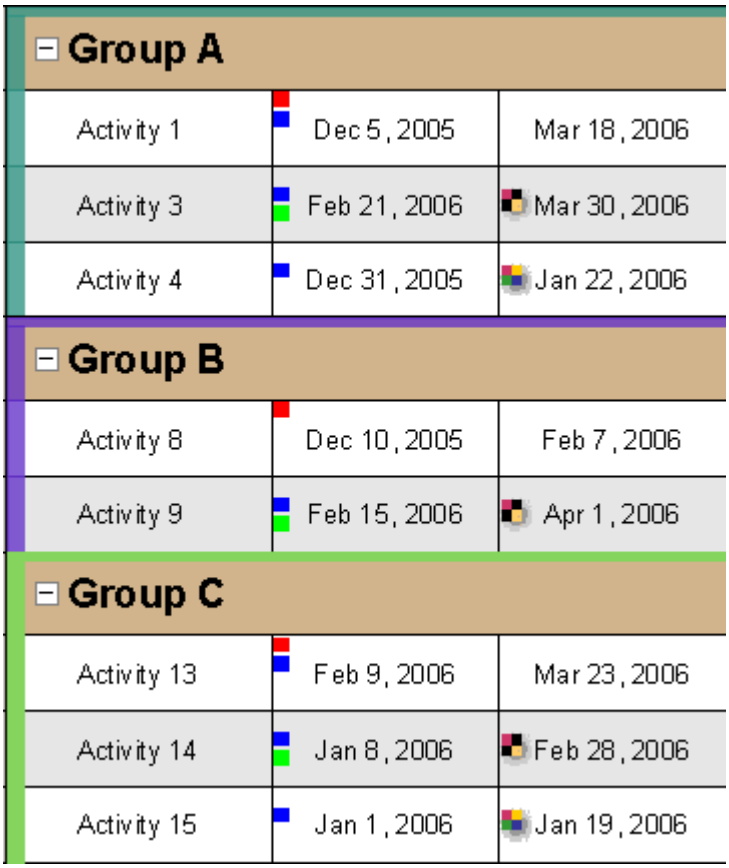
```
rowDef.getFieldDefinition(0).setExternalEditor(new PersonalAnimalCellEditor(this));
```

Picture

Property of NeIFieldDefinition

Typ	**de.netronic.common.intface.NeIPicture**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve pictures as a "background" in fields of tables, similar to layers in the Gantt graph. "Background" ist not to be taken literally. Since neither the picture object nor the text necessarily fill the field completely, pictures can be positioned near texts. Also, own pictures can be used.



Accessing Methods
void setPicture (de.netronic.common.intface.NeIPicture newValue)
de.netronic.common.intface.NeIPicture getPicture ()

Methods of the Interface

getEditor

Method of NeIFieldDefinition

For internal use only. Retrieves the field editor of an entity. The editor is returned anyway, even if the property "Editable" has been set to **false**.

Declaration
javax.swing.table.TableCellEditor getEditor (NeIEntity entity,
de.netronic.common.contentsdefinition.NeFieldStyle defaultFieldStyle)

	Data Type	Description
Parameter		
entity	NeIEntity	Entity, of which the field editor is to be retrieved.
defaultFieldStyle	de.netronic.common.contentsdefinition.-NeFieldStyle	Default field style
Return Value	javax.swing.table.TableCellEditor	

getRenderer

Method of NeIFieldDefinition

For internal use only. Retrieves the field renderer.

Declaration
javax.swing.table.TableCellRenderer getRenderer (NeIEntity entity,
de.netronic.common.contentsdefinition.NeFieldStyle defaultFieldStyle)

	Data Type	Description
Parameter		
entity	NeIEntity	Entity, of which the field renderer is to be retrieved.
defaultFieldStyle	de.netronic.common.contentsdefinition.-NeFieldStyle	Default field style
Return Value	javax.swing.table.TableCellRenderer	

8.7 NeITable

Belongs to Table

Package name **de.netronic.common.intface**

This interface comprises methods and properties to handle the table.

Common Properties of the Table

AntialiasText	Antialiasing on texts
ColumnReorderingAllowed	Re-ordering of table columns
ColumnResizingAllowed	Re-sizing the width of table columns
ContentsDefinition	Provides the contents definition of the table
InteractionRowMoveMode	Interactive moving of table rows.

Common Methods of the Table

addRowInteractionListener(...)	By this method you can add a NeRowInteractionListener to your table.
getColumnViewIndex(...)	Returns the position of a column index, where the column is located in the table.
setColumnViewIndex(...)	Sets the position of a column index, where the column is located in the table.

Properties of the Interface

AntialiasText

Property of NeITable

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Antialiasing on text strings can be switched on or off.

Accessing Methods
void setAntialiasText (boolean newValue)
boolean isAntialiasText ()

ColumnReorderingAllowed

Property of NeITable

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you allow or prohibit the user-interaction of re-ordering table columns.

Accessing Methods
void setColumnReorderingAllowed (boolean newValue)
boolean getColumnReorderingAllowed ()

Also see ColumnResizingAllowed

ColumnResizingAllowed

Property of NeITable

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you allow or prohibit the user interaction of re-sizing the width of table columns. Re-sizing table columns by the API remains unaffected by this property.

Accessing Methods
void setColumnResizingAllowed (boolean newValue)
boolean getColumnResizingAllowed ()

Also see ColumnReorderingAllowed

ContentsDefinition

Property of NeITable

Typ	**de.netronic.common.contentsdefinition.NeContentsDefinition**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property provides the contents definition of the table

Accessing Methods
void setContentsDefinition (de.netronic.common.contentsdefinition.NeContentsDefinition newValue)
de.netronic.common.contentsdefinition.NeContentsDefinition getContentsDefinition ()

InteractionRowMoveMode

Property of NeITable

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**INTERACTION_MOVE_MODE_DISABLED**

Interactive moving of table rows.

Possible Values	Description
INTERACTION_MOVE_MODE_DISABLED	It is not possible to move table rows interactively.
INTERACTION_MOVE_MODE_ROWS	Table rows can be moved interactively.

Accessing Methods
void setInteractionRowMoveMode (int newValue)
int getInteractionRowMoveMode ()

Example Code

```
jGantt1.getTable().setInteractionRowMoveMode
    (JGTable.INTERACTION_MOVE_MODE_ROWS);
```

Methods of the Interface

addRowInteractionListener

Method of NeITable

Here you can add a NeRowInteractionListener to your table and thus react to interactions in the table.

Declaration
void addRowInteractionListener (de.netronic.common.event.NeRowInteractionListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.NeRowInteractionListener	The row interaction listener you want to add.
Return Value	void	

Also see removeRowInteractionListener

Example Code

```
jGantt1.getTable().addRowInteractionListener(myRowInteractionListener );
```

getColumnViewIndex

Method of NeITable

Usually, table columns are displayed in the sequence that was defined by the contents definition (NeContentsDefinition) or set by the the property **TableColumns** of the **JGantt** class. The sequence originally defined can be modified by moving columns interactively or by the method **setColumnViewIndex**. This method retrieves the actual postion of a column in the table.

Declaration
int getColumnViewIndex (int columnIndex)

	Data Type	Description
Parameter		
columnIndex	int	Index of the column, of which the position is to be retrieved.
Return Value	int	

Also see setColumnViewIndex

removeRowInteractionListener

Method of NeITable

By this method you can remove a NeRowInteractionListener listening to your table interactions.

Declaration
void removeRowInteractionListener (de.netronic.common.event.NeRowInteractionListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.NeRowInteractionListener	The row interaction listener you want to remove.
Return Value	void	

Also see addRowInteractionListener

Example Code

```
jGantt1.getTable().removeRowInteractionListener(myRowInteractionListener );
```

setColumnViewIndex

Method of NeITable

Usually, table columns are displayed in the sequence that was defined by the contents definition (NeContentsDefinition) or set by the the property **TableColumns** of the **JGantt** class. The sequence originally defined can be modified by moving columns interactively or by this method.

Declaration
void setColumnViewIndex (int columnIndex, int viewIndex)

	Data Type	Description
Parameter		
columnIndex	int	Index of the column, of which the position is to be set.
viewIndex	int	Position, to which the column is to be set. The index starts by the value **0**.
Return Value	void	

Also see getColumnViewIndex

8.8 NeRowDefinition

Belongs to Table

Package name **de.netronic.common.contentsdefinition**

The contents of a row in a table can be described by a row definition. Uniform rows are of the same row type and use the same row definition.

A row definition represents a row of field definitions. It has a priority, a filter can be allocated.

When displaying a table row, from all row definitions the one will apply that has the highest priority plus of which the filter condition gives **true**.

If no filter is allocated to a row definition, it can apply to any row for which a row definition that has a filter was not found. In a set of several row definitions that lack a filter the one of highest priority will apply.

Beside, a row definition can have a field style. It will apply to field in case the field definition does not have a field style of its own.

Common Properties of Row Definitions

FieldStyle	Default field style
Filter	Filter to be retrieved
MarkType	Marking type of selected rows
Priority	Priority of the row definition

Properties to Manage Field Definitions

FieldDefinitionCount	Number of field definitions in a row definition
MultiColumnsFields	Retrieves, whether in this row fields exist that stretch across several columns.

Methods for Internal Use Only

addFieldDefinitionStripeEventListener(...)	For internal use only
propertyChange(...)	For internal use only
removeFieldDefinitionStripeEventListener(...)	For internal use only

Methods to Manage Field Definitions

addFieldDefinition(...)	Adds a field defintion to a row definition.
addFieldDefinition(...)	Adds a field definition and positions it in the row definition.
getFieldDefinitionByColumnIndex(...)	Retrieves the field definition at the column index
getFieldRange(...)	Retrieves the number of table columns that a field stretches across.
iterateFieldDefinitions()	Provides an iterator object of all field definitions.
removeFieldDefinition(...)	Removes a field defintion from the row definition.
setFieldRange(...)	Sets the number of table columns for a field to stretch across.

Constructors of the Class

NeRowDefinition

Constructor of NeRowDefinition

This constructor lets you create a row definition that has a filter and a priority.

Declaration
NeRowDefinition (de.netronic.common.intface.NeIFilter filter, int priority)

Parameter	Data Type	Description
filter	de.netronic.common.intface.NeIFilter	Filter to specify row types to which the row definition should apply.
priority	int	Priority of the row definition.

Properties of the Class

FieldDefinitionCount

Read Only Property of NeRowDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the number of field definitions in a row definition.

Accessing Methods
int getFieldDefinitionCount()

FieldStyle

Property of NeRowDefinition

Typ	**NeFieldStyle**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

This property lets you set or retrieve a field style to or from the row defintion. It is the default field style for all field definitions, that do not have a field style of their own.

Accessing Methods
void setFieldStyle (NeFieldStyle newValue)
NeFieldStyle getFieldStyle ()

Filter

Read Only Property of NeRowDefinition

Typ	**de.netronic.common.intface.NeIFilter**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the filter of a row definition. The filter can be assigned only once when a row definition is generated.

Accessing Methods
de.netronic.common.intface.NeIFilter getFilter()

MarkType

Property of NeRowDefinition

Typ	**int**
Bound	**yes**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**MARKTYPE_NONE**

This property lets you set or retrieve, in which way a selected table row is marked.

Possible Values	Description
MARKTYPE_DARK	The row colors are darkened.
MARKTYPE_NONE	The selected row is not marked

Accessing Methods
void setMarkType (int newValue)
int getMarkType ()

MultiColumnsFields

Read Only Property of NeRowDefinition

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This method retrieves, whether in this row fields exist that stretch across several columns.

Accessing Methods
boolean hasMultiColumnsFields()

Priority

Property of NeRowDefinition

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the priority of a row definition.

Accessing Methods
void setPriority (int newValue)
int getPriority ()

Methods of the Class

addFieldDefinition

Method of NeRowDefinition

This method adds a field definition to a row definition. The new field defintion is added to the end.

A field definition kann be included in several row definitions.

Declaration
boolean addFieldDefinition (de.netronic.common.intface.NeIFieldDefinition fieldDefinition)

	Data Type	Description
Parameter fieldDefinition	 de.netronic.common.intface.-NeIFieldDefinition	 Field definition to be added
Return Value	boolean	**False** if the field definition was already present in the row definition and therefore could not be added, otherwise **true**.

Also see removeFieldDefinition

addFieldDefinition

Method of NeRowDefinition

This method adds a field definition and positions it in the row definition, in relation to a reference field definition.

A field definition can be included in several row definitions.

Declaration
boolean addFieldDefinition (de.netronic.common.intface.NeIFieldDefinition fieldDefinition, de.netronic.common.intface.NeIFieldDefinition refFieldDefinition, int position)

	Data Type	Description
Parameter		
fieldDefinition	de.netronic.common.intface.-NeIFieldDefinition	Field definition to be added.
refFieldDefinition	de.netronic.common.intface.-NeIFieldDefinition	Reference field definition, before or after which the new field definition will be inserted.
position	int	Relative position, at which the new field definition is to be added
	Possible Values:	
	AFTER	Positioning after the reference object
	BEFORE	Positioning before a reference object
Return Value	boolean	**False** if the field definition was already present in the row definition and therefore could not be added, otherwise **true**.

Also see removeFieldDefinition

addFieldDefinitionStripeEventListener

Method of NeRowDefinition

For internal use only

Declaration
void addFieldDefinitionStripeEventListener (de.netronic.common.event.-NeFieldDefinitionStripeEventListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeFieldDefinitionStripeEventListener	For internal use only
Return Value	void	

getFieldDefinitionByColumnIndex

Method of NeRowDefinition

This method retrieves the field definition that applies to the field at the column index passed.

Declaration
void getFieldDefinitionByColumnIndex (int index)

	Data Type	Description
Parameter		
index	int	Column index of the field, of which the field definition is to be retrieved.
Return Value	void	

getFieldRange

Method of NeRowDefinition

Fields can stretch across several table columns. This method retrieves the number of table columns that the field (of a given field definition) stretches across. The range ist not a part of the field definition, because a field definition can belong to different row definitions and thus may show a different size in each one.

Declaration
int getFieldRange (de.netronic.common.intface.NeIFieldDefinition fieldDefinition)

	Data Type	Description
Parameter		
fieldDefinition	de.netronic.common.intface.-NeIFieldDefinition	Field definition, of which the range will be retrieved.
Return Value	int	Number of table columns

Also see setFieldRange

iterateFieldDefinitions

Method of NeRowDefinition

This method provides an iterator object of all field definitions.

Declaration
java.util.Iterator iterateFieldDefinitions ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object of all field definitions

propertyChange

Method of NeRowDefinition

For internal use only

Declaration
void propertyChange (java.beans.PropertyChangeEvent propertyChangeEvent)

	Data Type	Description
Parameter		
propertyChangeEvent	java.beans.PropertyChangeEvent	For internal use only
Return Value	void	

removeFieldDefinition

Method of NeRowDefinition

This method removes a field defintion from the row definition.

Declaration
boolean removeFieldDefinition (de.netronic.common.intface.NeIFieldDefinition fieldDefinition)

	Data Type	Description
Parameter		
fieldDefinition	de.netronic.common.intface.-NeIFieldDefinition	Field definition to be removed.
Return Value	boolean	**False**, if the field definition could not be removed (for example if it was not contained in the row definition), otherwise **true**.

Also see addFieldDefinition

removeFieldDefinitionStripeEventListener

Method of NeRowDefinition

For internal use only

Declaration
void removeFieldDefinitionStripeEventListener (de.netronic.common.event.-NeFieldDefinitionStripeEventListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeFieldDefinitionStripeEventListener	For internal use only
Return Value	void	

setFieldRange

Method of NeRowDefinition

Fields can stretch across several table columns. This method sets the number of table columns that the field (of a given field definition) stretches across. The range ist not a part of the field definition, because a field definition can belong to different row definitions and thus may show a different size in each one.

Declaration
boolean setFieldRange (de.netronic.common.intface.NeIFieldDefinition fieldDefinition, int range)

	Data Type	Description
Parameter		
fieldDefinition	de.netronic.common.intface.-NeIFieldDefinition	Field definition, of which the range will be set.
range	int	Number of table columns
Return Value	boolean	**True** if the range is larger than or equals 0 and if it differs from the range already set. Otherwise **false**.

Also see getFieldRange

9 TimeScale

This component offers classes and interfaces to handle the time scale.

A time scale (JGTimeScale) is composed of one or several sections (JGTimeScaleSection). In each section there is a visible scale (JGTimeScaleRibbonStripe), which is composed of several ribbons (JGTimeScaleRibbon). The appearance of a ribbon is determined by one or more tick rules.

Each of these time scale components can have a background color, a text color and a line color. If one of these colors is set to null, the correspondent color of the holder (e.g. JGTimeScaleRibbonStripe is the holder of a JGTimeScaleRibbon) is used.

The other properties of the time scale components are handled in the same manner.

The TimeScale-Component consists of the below classes:

JGTimeScale	This class administers the time scale.
JGTimeScaleElement	Base class for time scale elements.
JGTimeScaleRibbon	This class specifies a time scale ribbon.
JGTimeScaleRibbonStripe	This class specifies a time scale ribbon stripe, whch consists of several time scale ribbons.
JGTimeScaleSection	This class specifies a time scale section.
NeITimeScaleSegmentsGenerator	You only need this class, if you want to draw a time scale ribbon yourself.
NeTimeScaleDateFormats	For a time scale section this class specifies user defined date formats.
NeTimeScaleSegment	You need this class only, if you want to design a time scale ribbon using the interface NeITimeScaleSegmentsGenerator.

9.1 JGTimeScale

Belongs to TimeScale

Package name	**de.netronic.jgantt**
Extends	**de.netronic.jgantt.timescale.JGTimeScaleElement**

This class administers the time scale. Many of its properties are former JGantt properties. These JGantt properties are deprecated. Please use the JGTimeScale properties in future applications.

Properties of the Class

AdvancedMouseInteractions

Property of JGTimeScale

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Hereby you can specify an alternative reaction to mouse interactions in the timecale:

With a mouse drag you change the visible part of the timescale.
With the mousewheel you specify the resolution of the timescale.

Accessing Methods
void setAdvancedMouseInteractions (boolean newValue)
boolean hasAdvancedMouseInteractions ()

Example Code

AntialiasText

Property of JGTimeScale

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Antialiasing on text strings can be switched on or off.

Accessing Methods
void setAntialiasText (boolean newValue)
boolean isAntialiasText ()

CoarsestRibbonStripe

Property of JGTimeScale

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Coarsest step ribbon stripe allowed of the dynamic time scale.
This property only matters in a dynamic time scale.
The dynamic time scale contains a list of ribbon stripes, ordered by minimum absolute resolution. This property specifies the coarsest ribbon stripe which can be selected from this list. Thereby automatically a coarsest absolute resolution of the time scale section is set.

Possible Values	Description
RIBBONSTRIPE_10_MILLISECONDS	A time scale ribbon stripe with minor ticks every 10 milliseconds, major ticks every 50 milliseconds and texts every 100 milliseconds
RIBBONSTRIPE_10_YEARS	A time scale ribbon stripe with minor ticks every 10 years, major ticks every 50 years and texts every 100 years. This is the coarsest predefined ribbon stripe.
RIBBONSTRIPE_100_MILLISECONDS	A time scale ribbon stripe with minor ticks every 100 milliseconds, major ticks every 500 milliseconds and texts every second.
RIBBONSTRIPE_12_HOURS	A time scale ribbon stripe with minor ticks every 12 hours and texts every day.
RIBBONSTRIPE_15_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every 15 minutes, small texts every hour and big texts every three hours.
RIBBONSTRIPE_15_MINUTES_FINE	A time scale ribbon stripe with minor ticks every 15 minutes and texts every hour.
RIBBONSTRIPE_15_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 15 seconds, small texts every minute and big texts every 5 minutes.
RIBBONSTRIPE_15_SEONDS_FINE	A time scale ribbon stripe with minor ticks every 15 seconds and texts every minute.
RIBBONSTRIPE_3_HOURS	A time scale ribbon stripe with minor ticks every 3 hours and small texts every 6 hours.
RIBBONSTRIPE_3_MONTHS	A time scale ribbon stripe with minor ticks every 3 months, major ticks every 6 months and texts every year.
RIBBONSTRIPE_30_MINUTES	A time scale ribbon stripe with minor ticks every 30 minutes, major ticks every hour and texts every 3hours.
RIBBONSTRIPE_30_SECONDS	A time scale ribbon stripe with minor ticks every 30 seconds, major ticks every minute and texts every 5 minutes.
RIBBONSTRIPE_5_MILLISECONDS	A time scale ribbon stripe with minor ticks every 5 milliseconds, major ticks every 10 milliseconds and texts every 50 milliseconds.
RIBBONSTRIPE_5_MINUTES	A time scale ribbon stripe with minor ticks every 5 minutes, major ticks every 15 minutes and texts every 30 minutes.
RIBBONSTRIPE_5_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_SECONDS_FINE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.

RIBBONSTRIPE_5_YEARS	A time scale ribbon stripe with minor ticks every 5 years, major ticks every 10 years, small texts every 50 years and big tets every 100 years.
RIBBONSTRIPE_50_MILLISECONDS	A time scale ribbon stripe with minor ticks every 50 milliseconds, major ticks every 100 milliseconds and texts every 500 milliseconds.
RIBBONSTRIPE_500_MILLISECONDS	A time scale ribbon stripe with minor ticks every 500 milliseconds, major ticks every second and texts every 5 seconds.
RIBBONSTRIPE_6_HOURS	A time scale ribbon stripe with minor ticks every 6 hours and texts every day.
RIBBONSTRIPE_6_MONTHS	A time scale ribbon stripe with minor ticks every 6 months, major ticks every year, small texts every 5 years and big texts every 10 years.
RIBBONSTRIPE_DAY_COARSE	A time scale ribbon stripe with minor ticks every day and texts every week.
RIBBONSTRIPE_DAY_FINE	A time scale ribbon stripe with texts every day and thick full ticks every week.
RIBBONSTRIPE_HOUR	A time scale ribbon stripe with minor ticks every hour, major ticks every 3 hours and texts every 6 hours.
RIBBONSTRIPE_MILLISECOND	A time scale ribbon stripe with minor ticks every millisecond, major ticks every 5 milliseconds, small texts every 10 milliseconds and big texts every 50 milliseconds. This is the finest predefined ribbon stripe.
RIBBONSTRIPE_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every minute, major ticks every 5 minutes and texts every 15 minutes.
RIBBONSTRIPE_MINUTES_FINE	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MINUTES_MEDIUM	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MONTH_COARSE	A time scale ribbon stripe with minor ticks every month and texts every 3 months.
RIBBONSTRIPE_MONTH_FINE	A time scale ribbon stripe with month names three letters long.
RIBBONSTRIPE_MONTH_MEDIUM	A time scale ribbon stripe with month names one letter long.
RIBBONSTRIPE_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every second, major ticks every 5 seconds and texts every 15 seconds.
RIBBONSTRIPE_SECONDS_FINE	A time scale ribbon stripe with minor ticks every second, small texts every 5 seconds and large texts every 15 seconds.

RIBBONSTRIPE_WEEK_COARSE	A time scale ribbon stripe with minor ticks every week, major ticks every 5 calendar weeks and texts every 10 calendar weeks.
RIBBONSTRIPE_WEEK_FINE	A time scale ribbon stripe with days in month as text at beginning of every week.
RIBBONSTRIPE_WEEK_MEDIUM	A time scale ribbon stripe with minor ticks every week and texts every 5 calendar weeks.
RIBBONSTRIPE_YEAR	A time scale ribbon stripe with minor ticks every year, major ticks every 5 years, small texts every 10 years and big texts every 50 years:

Accessing Methods
void setCoarsestRibbonStripe (java.lang.String newValue)
java.lang.String getCoarsestRibbonStripe ()

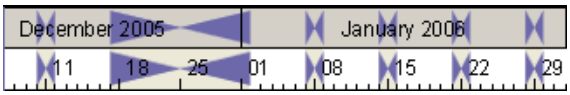
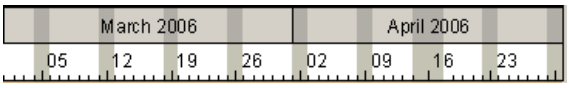
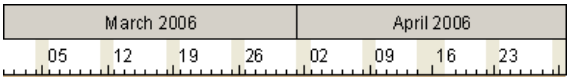
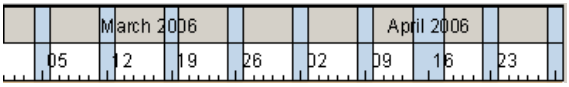
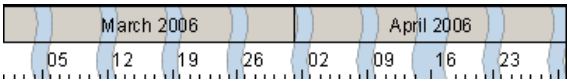
Also see FinestRibbonStripe

CollapseDisplayMode

Property of JGTimeScale

Typ	**int**
Bound	**yes**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**COLLAPSE_DISPLAY_WAVE**

This property defines, in which way collapsed non-working times are visualized.

Possible Values	Description
COLLAPSE_DISPLAY_BOWTIE	Collapsed non-working times are displayed as colored bowties. The color is set by the **alternateColor** of the **ColorScheme**. 
COLLAPSE_DISPLAY_DARKEN	Collapsed non-working times are displayed as dimmed rectangles. The degree of dimming is proportional to the size of the factor set by the property **CollapseFactor**. 
COLLAPSE_DISPLAY_NONE	Collapsed non-working times are not displayed. The picture below shows merely the calendar, which was graphically visualized. 
COLLAPSE_DISPLAY_RECT	Collapsed non-working times are displayed as colored rectangles. The color is set by the **alternateColor** of the **ColorScheme**. 
COLLAPSE_DISPLAY_WAVE	Collapsed non-working times are displayed as a vertical, colored wave. The color is set by the **alternateColor** of the **ColorScheme**. 

Accessing Methods
void setCollapseDisplayMode (int newValue)
int getCollapseDisplayMode ()

Also see CollapseFactor
CollapseProfile

CollapseFactor

Property of JGTimeScale

Typ	**double**
Bound	**yes**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**0**

This property sets the factor, by which the time scale is downsized by collapsing non-working times. Permitted values: {0.0...1.0}

Accessing Methods
void setCollapseFactor (double newValue)
double getCollapseFactor ()

Also see CollapseDisplayMode
CollapseProfile

CollapseProfile

Property of JGTimeScale

Typ	**java.lang.String**
Bound	**yes**
Vetoable	**no**
Exposure Level	**expert**

By this property you can set a collapse profile to the time scale, by which non-working times can be collapsed. The profile is assigned by its name.

Accessing Methods
void setCollapseProfile (java.lang.String newValue)
java.lang.String getCollapseProfile ()

Also see CollapseDisplayMode
CollapseFactor

ColorScheme

Property of JGTimeScale

Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color scheme for the time scale.
The color scheme determines the following properties of the time scale:

The mainColor sets the AreaStyle.
The shadedColor sets the AreaStyleForHigherRibbons,
The shadedAlternateColor sets the AreaStyleForTimeSpans.
The alternateColor sets the AreaStyleForHigherTimeSpans.
The lineColor sets the LineColor.
The mainTextColor sets the TextColor.

Accessing Methods
void setColorScheme (JGColorScheme newValue)
JGColorScheme getColorScheme ()

DateFormats

Property of JGTimeScale

Typ	**NeTimeScaleDateFormats**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

All user defined date formats for all ribbon stripe indexes and types or step units.

Accessing Methods
void setDateFormats (NeTimeScaleDateFormats newValue)
NeTimeScaleDateFormats getDateFormats ()

DisplayProfile

Property of JGTimeScale

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

This property defines the name of the calendar profile for the time scale. This profile is displayed in the time scale and visualizes a pattern of working periods and non-working periods in the time scale. The way it does this depends on the type of time scale used. It may show different colors in week-ends or display a separate ribbon that shows a color pattern of working periods and nonworking periods, etc.

Setting this property to a value != **null** will switch the visualization of working patterns on.
The colors used for the visualization of the calendar are controlled by the ColorScheme property. TimeSpans of the spanID 0 will adopt the shadedAlternateColor, all other timeSpans will be displayed in the alternateColor. Different colors you can set by using NeMappedColors.

Accessing Methods
void setDisplayProfile (java.lang.String newValue)
java.lang.String getDisplayProfile ()

Dynamic

Property of JGTimeScale

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property defines whether or not the time scale adapts dynamically to the change of size when zooming, that is, whether it changes its units, for example from a daily scale to a weekly or monthly scale. **True**: the time scale adapts dynamically, **false**: the time scale does not adapt dynamically.

Accessing Methods
void setDynamic (boolean newValue)
boolean isDynamic ()

FinestRibbonStripe

Property of JGTimeScale

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Finest step ribbon stripe allowed of the dynamic time scale.
This property only matters in a dynamic time scale.
The dynamic time scale contains a list of ribbon stripes, ordered by minimum absolute resolution. This property specifies the finest ribbon stripe which can be selected from this list. Thereby automatically a finest absolute resolution of the time scale section is set.

Possible Values	Description
RIBBONSTRIPE_10_MILLISECONDS	A time scale ribbon stripe with minor ticks every 10 milliseconds, major ticks every 50 milliseconds and texts every 100 milliseconds
RIBBONSTRIPE_10_YEARS	A time scale ribbon stripe with minor ticks every 10 years, major ticks every 50 years and texts every 100 years. This is the coarsest predefined ribbon stripe.
RIBBONSTRIPE_100_MILLISECONDS	A time scale ribbon stripe with minor ticks every 100 milliseconds, major ticks every 500 milliseconds and texts every second.
RIBBONSTRIPE_12_HOURS	A time scale ribbon stripe with minor ticks every 12 hours and texts every day.
RIBBONSTRIPE_15_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every 15 minutes, small texts every hour and big texts every three hours.
RIBBONSTRIPE_15_MINUTES_FINE	A time scale ribbon stripe with minor ticks every 15 minutes and texts every hour.
RIBBONSTRIPE_15_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 15 seconds, small texts every minute and big texts every 5 minutes.
RIBBONSTRIPE_15_SEONDS_FINE	A time scale ribbon stripe with minor ticks every 15 seconds and texts every minute.
RIBBONSTRIPE_3_HOURS	A time scale ribbon stripe with minor ticks every 3 hours and small texts every 6 hours.
RIBBONSTRIPE_3_MONTHS	A time scale ribbon stripe with minor ticks every 3 months, major ticks every 6 months and texts every year.
RIBBONSTRIPE_30_MINUTES	A time scale ribbon stripe with minor ticks every 30 minutes, major ticks every hour and texts every 3hours.
RIBBONSTRIPE_30_SECONDS	A time scale ribbon stripe with minor ticks every 30 seconds, major ticks every minute and texts every 5 minutes.
RIBBONSTRIPE_5_MILLISECONDS	A time scale ribbon stripe with minor ticks every 5 milliseconds, major ticks every 10 milliseconds and texts every 50 milliseconds.
RIBBONSTRIPE_5_MINUTES	A time scale ribbon stripe with minor ticks every 5 minutes, major ticks every 15 minutes and texts every 30 minutes.
RIBBONSTRIPE_5_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_SECONDS_FINE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.

RIBBONSTRIPE_5_YEARS	A time scale ribbon stripe with minor ticks every 5 years, major ticks every 10 years, small texts every 50 years and big tets every 100 years.
RIBBONSTRIPE_50_MILLISECONDS	A time scale ribbon stripe with minor ticks every 50 milliseconds, major ticks every 100 milliseconds and texts every 500 milliseconds.
RIBBONSTRIPE_500_MILLISECONDS	A time scale ribbon stripe with minor ticks every 500 milliseconds, major ticks every second and texts every 5 seconds.
RIBBONSTRIPE_6_HOURS	A time scale ribbon stripe with minor ticks every 6 hours and texts every day.
RIBBONSTRIPE_6_MONTHS	A time scale ribbon stripe with minor ticks every 6 months, major ticks every year, small texts every 5 years and big texts every 10 years.
RIBBONSTRIPE_DAY_COARSE	A time scale ribbon stripe with minor ticks every day and texts every week.
RIBBONSTRIPE_DAY_FINE	A time scale ribbon stripe with texts every day and thick full ticks every week.
RIBBONSTRIPE_HOUR	A time scale ribbon stripe with minor ticks every hour, major ticks every 3 hours and texts every 6 hours.
RIBBONSTRIPE_MILLISECOND	A time scale ribbon stripe with minor ticks every millisecond, major ticks every 5 milliseconds, small texts every 10 milliseconds and big texts every 50 milliseconds. This is the finest predefined ribbon stripe.
RIBBONSTRIPE_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every minute, major ticks every 5 minutes and texts every 15 minutes.
RIBBONSTRIPE_MINUTES_FINE	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MINUTES_MEDIUM	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MONTH_COARSE	A time scale ribbon stripe with minor ticks every month and texts every 3 months.
RIBBONSTRIPE_MONTH_FINE	A time scale ribbon stripe with month names three letters long.
RIBBONSTRIPE_MONTH_MEDIUM	A time scale ribbon stripe with month names one letter long.
RIBBONSTRIPE_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every second, major ticks every 5 seconds and texts every 15 seconds.
RIBBONSTRIPE_SECONDS_FINE	A time scale ribbon stripe with minor ticks every second, small texts every 5 seconds and large texts every 15 seconds.

RIBBONSTRIPE_WEEK_COARSE	A time scale ribbon stripe with minor ticks every week, major ticks every 5 calendar weeks and texts every 10 calendar weeks.
RIBBONSTRIPE_WEEK_FINE	A time scale ribbon stripe with days in month as text at beginning of every week.
RIBBONSTRIPE_WEEK_MEDIUM	A time scale ribbon stripe with minor ticks every week and texts every 5 calendar weeks.
RIBBONSTRIPE_YEAR	A time scale ribbon stripe with minor ticks every year, major ticks every 5 years, small texts every 10 years and big texts every 50 years:

Accessing Methods
void setFinestRibbonStripe (java.lang.String newValue)
java.lang.String getFinestRibbonStripe ()

Also see CoarsestRibbonStripe

NoOfSections

Read Only Property of JGTimeScale

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Retrieves the number of sections of this time scale.

Accessing Methods
int getNoOfSections()

PopupEnabled

Property of JGTimeScale

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines, whether or not the menu that appears on pressing the right mouse button is enabled to pop up on the time scale.

Accessing Methods
void setPopupEnabled (boolean newValue)
boolean isPopupEnabled ()

ResolutionUserModifiable

Property of JGTimeScale

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

This property enables the user to adjust the time scale resolution at runtime by setting the **resolution modify interaction** to **true** or **false**.

Accessing Methods
void setResolutionUserModifiable (boolean newValue)
boolean isResolutionUserModifiable ()

ViewEnd

Read Only Property of JGTimeScale

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the end date of the window that the time scale and the Gantt graph are displayed by when the window is taken to the screen. This property cannot be set, because the end is calculated automatically from the size of the view and the resolution of the time scale.

Accessing Methods
long getViewEnd()

Also see ViewStart

ViewStart

Property of JGTimeScale

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the start date of the window that the time scale and the Gantt graph are displayed by when the window is taken to the screen. The start date set here will be ignored in case it was set too close to the end of the view. Then the start date will be replaced by an automatically calculated one that considers the size of the view and the resolution of the time scale.

Accessing Methods
void setViewStart (long newValue)
long getViewStart ()

Also see ViewEnd

VisibleAtBottom

Property of JGTimeScale

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**false**

This property defines whether or not the bottom time scale should be visible.

Accessing Methods
void setVisibleAtBottom (boolean newValue)
boolean isVisibleAtBottom ()

VisibleAtTop

Property of JGTimeScale

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**true**

This property defines whether or not the top time scale should be visible.

Accessing Methods
void setVisibleAtTop (boolean newValue)
boolean isVisibleAtTop ()

VisibleRibbonStripe

Property of JGTimeScale

Typ	**JGTimeScaleRibbonStripe**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Visible ribbon stripe of this time scale section. In a dynamic time scale the visible ribbons stripe depends on the absolute resolution of time scale section.

Accessing Methods
void setVisibleRibbonStripe (JGTimeScaleRibbonStripe newValue)
JGTimeScaleRibbonStripe getVisibleRibbonStripe ()

Methods of the Class

getDateFormats

Method of JGTimeScale

This method lists for a time scale ribbon index and a type or step unit the specified date formats as patterns.

Declaration
java.lang.String[] getDateFormats (int ribbonIndex, java.lang.String typeOrStepUnit)

	Data Type	Description
Parameter		
ribbonIndex	int	Index of the time scale ribbon the date formats of which you want to change.
typeOrStepUnit	java.lang.String	You can specify here either a time scale type (as under JGantt.setTimeScaleType) or a step unit. If you have a JGTimeScaleTickRule with step rule "3H", which paints a text, you change the format of this text by specifing the StepUnit "H" in this method.
Return Value	java.lang.String[]	

setDateFormats

Method of JGTimeScale

Here you can change the date formats of a time scale ribbon. You must specify the ribbon index, the time scale type or step unit, and one or more date formats. If you specify more than one date formats the higher ones must be shorter and will be used when the text must be shortened due to lack of space.

Sample:
timeScale.setDateFormats(1, "H", "HH 'h'", "HH");

All TickRules with a hour step unit add a "h" to the hours text. If this text is too long, the second format without the "h" is used.

Declaration
void setDateFormats (int ribbonIndex, java.lang.String typeOrStepUnit, java.lang.String dateFormat)

	Data Type	Description
Parameter		
ribbonIndex	int	Index of the time scale ribbon the date formats of which you want to change.
typeOrStepUnit	java.lang.String	You can specify here either a time scale type (as under JGantt.setTimeScaleType) or a step unit. If you have a JGTimeScaleTickRule with step rule "3H", which paints a text, you change the format of this text by specifing the StepUnit "H" in this method.
dateFormat	java.lang.String	The pattern for a NeTimeScaleDateFormat, you can use all possible patterns for a SimpleDateFormat.
Return Value	void	

Example Code

```
timeScale.setDateFormats(0, jGantt1.getTimeScaleType(), "yyyy-MM", "yy-MM", "MM");
timeScale.setDateFormats(1, jGantt1.getTimeScaleType(), "dd", "");
```

setVisibleRibbonStripe

Method of JGTimeScale

This property specifies the visible ribbon stripe by the name of a predefined ribbon stripe.

Declaration
void setVisibleRibbonStripe (java.lang.String stripe)

	Data Type	Description
Parameter		
stripe	java.lang.String	Name of the time scale ribbon stripe.
	Possible Values:	
	RIBBONSTRIPE_10_MILLISECONDS	A time scale ribbon stripe with minor ticks every 10 milliseconds, major ticks every 50 milliseconds and texts every 100 milliseconds
	RIBBONSTRIPE_10_YEARS	A time scale ribbon stripe with minor ticks every 10 years, major ticks every 50 years and texts every 100 years. This is the coarsest predefined ribbon stripe.
	RIBBONSTRIPE_100_MILLISECONDS	A time scale ribbon stripe with minor ticks every 100 milliseconds, major ticks every 500 milliseconds and texts every second.
	RIBBONSTRIPE_12_HOURS	A time scale ribbon stripe with minor ticks every 12 hours and texts every day.
	RIBBONSTRIPE_15_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every 15 minutes, small texts every hour and big texts every three hours.
	RIBBONSTRIPE_15_MINUTES_FINE	A time scale ribbon stripe with minor ticks every 15 minutes and texts every hour.
	RIBBONSTRIPE_15_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 15 seconds, small texts every minute and big texts every 5 minutes.
	RIBBONSTRIPE_15_SEONDS_FINE	A time scale ribbon stripe with minor ticks every 15 seconds and texts every minute.
	RIBBONSTRIPE_3_HOURS	A time scale ribbon stripe with minor ticks every 3 hours and small texts every 6 hours.
	RIBBONSTRIPE_3_MONTHS	A time scale ribbon stripe with minor ticks every 3 months, major ticks every 6 months and texts every year.
	RIBBONSTRIPE_30_MINUTES	A time scale ribbon stripe with minor ticks every 30 minutes, major ticks every hour and texts every 3hours.
	RIBBONSTRIPE_30_SECONDS	A time scale ribbon stripe with minor ticks every 30 seconds, major ticks every minute and texts every 5 minutes.

RIBBONSTRIPE_5_MILLISECONDS	A time scale ribbon stripe with minor ticks every 5 milliseconds, major ticks every 10 milliseconds and texts every 50 milliseconds.
RIBBONSTRIPE_5_MINUTES	A time scale ribbon stripe with minor ticks every 5 minutes, major ticks every 15 minutes and texts every 30 minutes.
RIBBONSTRIPE_5_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_SECONDS_FINE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_YEARS	A time scale ribbon stripe with minor ticks every 5 years, major ticks every 10 years, small texts every 50 years and big tets every 100 years.
RIBBONSTRIPE_50_MILLISECONDS	A time scale ribbon stripe with minor ticks every 50 milliseconds, major ticks every 100 milliseconds and texts every 500 milliseconds.
RIBBONSTRIPE_500_MILLISECONDS	A time scale ribbon stripe with minor ticks every 500 milliseconds, major ticks every second and texts every 5 seconds.
RIBBONSTRIPE_6_HOURS	A time scale ribbon stripe with minor ticks every 6 hours and texts every day.
RIBBONSTRIPE_6_MONTHS	A time scale ribbon stripe with minor ticks every 6 months, major ticks every year, small texts every 5 years and big texts every 10 years.
RIBBONSTRIPE_CALENDARWEEK	A time scale ribbon stripe with calendar weeks.
RIBBONSTRIPE_DAY_CALENDARWEEK	A time scale ribbon stripe with days and calendar weeks.
RIBBONSTRIPE_DAY_COARSE	A time scale ribbon stripe with minor ticks every day and texts every week.
RIBBONSTRIPE_DAY_FINE	A time scale ribbon stripe with texts every day and thick full ticks every week.
RIBBONSTRIPE_HOUR	A time scale ribbon stripe with minor ticks every hour, major ticks every 3 hours and texts every 6 hours.
RIBBONSTRIPE_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every minute, major ticks every 5 minutes and texts every 15 minutes.
RIBBONSTRIPE_MINUTES_FINE	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.

	RIBBONSTRIPE_MINUTES_MEDIUM	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
	RIBBONSTRIPE_MONTH_COARSE	A time scale ribbon stripe with minor ticks every month and texts every 3 months.
	RIBBONSTRIPE_MONTH_FINE	A time scale ribbon stripe with month names three letters long.
	RIBBONSTRIPE_MONTH_MEDIUM	A time scale ribbon stripe with month names one letter long.
	RIBBONSTRIPE_QUARTER	A time scale ribbon stripe with texts for every quarter.
	RIBBONSTRIPE_RELATIVE_DAYS	A time scale ribbon stripe with relative days.
	RIBBONSTRIPE_RELATIVE_HOURS	A time scale ribbon stripe with relative hours.
	RIBBONSTRIPE_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every second, major ticks every 5 seconds and texts every 15 seconds.
	RIBBONSTRIPE_SECONDS_FINE	A time scale ribbon stripe with minor ticks every second, small texts every 5 seconds and large texts every 15 seconds.
	RIBBONSTRIPE_WEEK_COARSE	A time scale ribbon stripe with minor ticks every week, major ticks every 5 calendar weeks and texts every 10 calendar weeks.
	RIBBONSTRIPE_WEEK_FINE	A time scale ribbon stripe with days in month as text at beginning of every week.
	RIBBONSTRIPE_WEEK_MEDIUM	A time scale ribbon stripe with minor ticks every week and texts every 5 calendar weeks.
	RIBBONSTRIPE_WEEKDAY	A time scale ribbon stripe with days in week and days in month.
	RIBBONSTRIPE_WEEKDAY_CALENDARWEEK	A time scale ribbon stripe with days in week, days in month and calendar weeks.
	RIBBONSTRIPE_YEAR	A time scale ribbon stripe with minor ticks every year, major ticks every 5 years, small texts every 10 years and big texts every 50 years:
Return Value	void	

9.2 JGTimeScaleElement

Belongs to TimeScale

Package name **de.netronic.jgantt.timescale**

Base class for time scale elements. Time scale elements are e.g. time scale ribbons, time scale ribbon stripes, time scale sections and the whole time scale.

Properties of the Class

AreaStyle

Property of JGTimeScaleElement

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

Background color of the time scale element. If this property is not set, the active background color of the holder is used.

Accessing Methods
void setAreaStyle (java.awt.Color newValue)
java.awt.Color getAreaStyle ()

AreaStyleForHigherRibbons

Property of JGTimeScaleElement

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Background color of a time scale ribbon if this is the second or higher of a time scale ribbon stripe. If this property is not set, the ActiveAreaStyleForHigherRibbons of the holder is used.

This property matters only if the time scale element is a ribbon stripe, a section or the whole time scale.

Accessing Methods
void setAreaStyleForHigherRibbons (java.awt.Color newValue)
java.awt.Color getAreaStyleForHigherRibbons ()

AreaStyleForHigherTimeSpans

Property of JGTimeScaleElement

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Background color of the workfree times of the time scale element if the TimeSpanID>0.
If this property is not set, the ActiveAreaStyleForHigherTimeSpans of the holder is used.

This property only matters if the property DisplayProfile is set.

Accessing Methods
void setAreaStyleForHigherTimeSpans (java.awt.Color newValue)
java.awt.Color getAreaStyleForHigherTimeSpans ()

AreaStyleForTimeSpans

Property of JGTimeScaleElement

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Background color of the workfree times of the time scale element.
If this property is not set, the ActiveAreaStyleForTimeSpans of the holder is used.

This property only matters if the property DisplayProfile is set.

Accessing Methods
void setAreaStyleForTimeSpans (java.awt.Color newValue)
java.awt.Color getAreaStyleForTimeSpans ()

DisplayProfile

Property of JGTimeScaleElement

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

Calendar profile to be shown in this time scale element.
If this property is not set there will be no calendar information in the time scale element.

Accessing Methods
void setDisplayProfile (java.lang.String newValue)
java.lang.String getDisplayProfile ()

EndDate

Read Only Property of JGTimeScaleElement

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

End date of the time scale element. This property only matters for the classes JGTimeScale and JGTimeScaleSection

Accessing Methods
long getEndDate()

LineColor

Property of JGTimeScaleElement

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Line Color of the time scale element. If this property is not set, the active line color of the holder is used.

Accessing Methods
void setLineColor (java.awt.Color newValue)
java.awt.Color getLineColor ()

NumberSuffix

Property of JGTimeScaleElement

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Number suffix for relative time scale ribbons, e.g. "h" for relative time scale ribbons showing hours.
If this property is not set, it is determined by the ActiveNumberSuffix of the holder.

Accessing Methods
void setNumberSuffix (java.lang.String newValue)
java.lang.String getNumberSuffix ()

RibbonHeight

Property of JGTimeScaleElement

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Ribbon height of the time scale element.
If this property is set to -1 (which means not set), ActiveRibbonHeight is used.

Accessing Methods
void setRibbonHeight (int newValue)
int getRibbonHeight ()

StartDate

Read Only Property of JGTimeScaleElement

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Start date of the time scale element. This property only matters for the classes JGTimeScale and JGTimeScaleSection

Accessing Methods
long getStartDate()

TextColor

Property of JGTimeScaleElement

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Text color of the time scale element If this property is not set, the active text color of the holder is used.

Accessing Methods
void setTextColor (java.awt.Color newValue)
java.awt.Color getTextColor ()

TextColorWeak

Property of JGTimeScaleElement

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

The weak text color of the time scale element.
If this property is not set, the active weak text color of the holder is used.

This property is used to specify two kinds of texts (important and not important) in a time scale ribbon.

Accessing Methods
void setTextColorWeak (java.awt.Color newValue)
java.awt.Color getTextColorWeak ()

TextFont

Property of JGTimeScaleElement

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Font of the time scale element.
If this property is not set, the active text font of the holder is used.

Accessing Methods
void setTextFont (java.awt.Font newValue)
java.awt.Font getTextFont ()

Methods of the Class

getActiveAreaStyle

Method of JGTimeScaleElement

Retrieves the active area style of this time scale element.
If the area style of this time scale element is not set, the active area style will be determined by the active area style of the holder.

Declaration
java.awt.Color getActiveAreaStyle ()

	Data Type	Description
Return Value	java.awt.Color	

getActiveAreaStyleForHigherRibbons

Method of JGTimeScaleElement

Retrieves the active area style for higher ribbons of this time scale element.
If this property of the time scale element is not set, the active area style for higher ribbons is determined by the active correspondent property of the holder.

This method is only relevant for time scale ribbon stripes, time scale sections and the whole time scale.

Declaration
java.awt.Color getActiveAreaStyleForHigherRibbons ()

	Data Type	Description
Return Value	java.awt.Color	

getActiveAreaStyleForHigherTimeSpans

Method of JGTimeScaleElement

Retrieves the active area style for higher time spans of this time scale element.
If this property of the time scale element is not set, the active area style for higher timespans is determined by the active correspondent property of the holder.

This method is only relevant when a display profile is set.

Declaration
java.awt.Color getActiveAreaStyleForHigherTimeSpans ()

	Data Type	Description
Return Value	java.awt.Color	

getActiveAreaStyleForTimeSpans

Method of JGTimeScaleElement

Retrieves the active area style for time spans of this time scale element.
If this property of the time scale element is not set, the active area style for timespans is determined by the active correspondent property of the holder.

This method is only relevant when a display profile is set.

Declaration
java.awt.Color getActiveAreaStyleForTimeSpans ()

	Data Type	Description
Return Value	java.awt.Color	

getActiveDisplayProfile

Method of JGTimeScaleElement

Retrieves the active display profile of this time scale element.
If this property of the time scale element is not set, the active display profile is determined by the active display profile of the holder.

Declaration
java.lang.String getActiveDisplayProfile ()

	Data Type	Description
Return Value	java.lang.String	

getActiveLineColor

Method of JGTimeScaleElement

Retrieves the active line color of this time scale element.
If this property of the time scale element is not set, the active line color is determined by the active line color of the holder.

Declaration
java.awt.Color getActiveLineColor ()

	Data Type	Description
Return Value	java.awt.Color	

getActiveNumberSuffix

Method of JGTimeScaleElement

Retrieves the active number suffix for relative time scales of this time scale element.
If this property of the time scale element is not set, the active number suffix for relative time scales is determined by the active correspondent property of the holder

Declaration
java.lang.String getActiveNumberSuffix ()

	Data Type	Description
Return Value	java.lang.String	

getActiveRibbonHeight

Method of JGTimeScaleElement

Retrieves the active ribbon height of this time scale element.
If this property of the time scale element is not set, the active ribbon height is determined by the active ribbon height of the holder.

Declaration
int getActiveRibbonHeight ()

	Data Type	Description
Return Value	int	

getActiveTextColor

Method of JGTimeScaleElement

Retrieves the active text color of this time scale element.
If this property of the time scale element is not set, the active text color is determined by the active text color of the holder.

Declaration
java.awt.Color getActiveTextColor ()

	Data Type	Description
Return Value	java.awt.Color	

getActiveTextColorWeak

Method of JGTimeScaleElement

Retrieves the active weak text color of this time scale element.
If this property of the time scale element is not set, the active weak text color is determined by the weak active text color of the holder.

Declaration
java.awt.Font getActiveTextColorWeak ()

	Data Type	Description
Return Value	java.awt.Font	

getActiveTextFont

Method of JGTimeScaleElement

Retrieves the active text font of this time scale element.
If this property of the time scale element is not set, the active text font is determined by the active text font of the holder.

Declaration
java.awt.Font getActiveTextFont ()

	Data Type	Description
Return Value	java.awt.Font	

getHolder

Method of JGTimeScaleElement

Retrieves the holder of this time scale element.

The holder of a JGTimeScaleRibbon is a JGTimeScaleRibbonStripe, whose holder is a JGTimeScaleSection, whose holder is the JGTimeScale. Not specified properties are always determined via the holder, therefore the getActive methods are used.

Declaration
JGTimeScaleElement getHolder ()

	Data Type	Description
Return Value	JGTimeScaleElement	

9.3 JGTimeScaleRibbon

Belongs to TimeScale

Package name **de.netronic.jgantt.timescale**
Extends **JGTimeScaleElement**

This class specifies a time scale ribbon. A time scale ribbon is part of a time scale ribbon stripe.
You can specify the ticks in this time scale ribbon via the addTickRule methods. The adding of the tick rules must be ordered, the finer tick rule must be added before the coarser, e.g. hours before days, days before weeks.

Constructors of the Class

JGTimeScaleRibbon

Constructor of JGTimeScaleRibbon

Creates a time scale ribbon with a name.

Declaration
JGTimeScaleRibbon (java.lang.String name)

Parameter	Data Type	Description
name	java.lang.String	The name of this ribbon stripe.

Properties of the Class

ActiveRibbonHeight

Read Only Property of JGTimeScaleRibbon

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the actual height of the ribbon stripe.
This height may e.g. be determined by a very high other time scale section or by the font size of the ribbon stripe.

Accessing Methods
int getActiveRibbonHeight()

Name

Read Only Property of JGTimeScaleRibbon

Typ	**void**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

The name of the time scale ribbon. By this name you can specify a short description of the appearance or task of this time scale ribbon.

Accessing Methods
void getName()

RelativeTo

Property of JGTimeScaleRibbon

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**-1**

By this property you can specify whether this time scale ribbon shows time distances relative to a date. This date may lie outside the time scale span.

Accessing Methods
void setRelativeTo (long newValue)
long getRelativeTo ()

SegmentsGenerator

Property of JGTimeScaleRibbon

Typ	**NelTimeScaleSegmentsGenerator**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**

By this property you can specify a self coded generator for the time scale segments.
By this self coded generator you can design a time scale ribbon according exactly to your needs.

Accessing Methods
void setSegmentsGenerator (NelTimeScaleSegmentsGenerator newValue)
NelTimeScaleSegmentsGenerator getSegmentsGenerator ()

TickRules

Property of JGTimeScaleRibbon

Typ	**JGTimeScaleTickRule[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**hidden**

Only for internal use.

Accessing Methods
void setTickRules (integer index, JGTimeScaleTickRule newValues)
void setTickRules (JGTimeScaleTickRule[] newValue)
JGTimeScaleTickRule getTickRules (integer index)
JGTimeScaleTickRule[] getTickRules ()

Methods of the Class

addFullTickRule

Method of JGTimeScaleRibbon

By this method you can add a full tick rule. Full ticks use the full height of the time scale ribbon. In the code sample the ticks will appear each full hour, the text is formatted by the second parameter. The texts will appear centered between the ticks.

Declaration
JGTimeScaleRibbon addFullTickRule (java.lang.String stepRule, java.lang.String dateFormat)

	Data Type	Description
Parameter		
stepRule	java.lang.String	Step rule for this tick rule. A step rule consists of a factor and a unit, e.g. "6H" stands for a tick every 6 hours, "H" for a tick each hour. Possible units: S: milli seconds s: seconds m: minutes H: hours D: days W: weeks M: months Q: quarters y: years
dateFormat	java.lang.String	This is the pattern of the SimpleDateFormat that formats the texts of this tick rule. If this parameter equals null, no texts will appear.
Return Value	JGTimeScaleRibbon	The current time scale ribbon.

Also see
addFullTickRule
addMajorTickRule
addMajorTickRule
addMinorTickRule
addTickRule

Example Code

```
new JGTimeScaleRibbon("Day").addFullTickRule("H", "dd.MM.yyyy HH");
```

addFullTickRule

Method of JGTimeScaleRibbon

By this method you can add a full tick rule with centered texts, the text format is determined from the step rule. Full ticks use the full height of the time scale ribbon. In the code sample the ticks will appear each full day. The texts will appear centered between the ticks.

Declaration
JGTimeScaleRibbon addFullTickRule (java.lang.String stepRule)

	Data Type	Description
Parameter		
stepRule	java.lang.String	Step rule for this tick rule. A step rule consists of a factor and a unit, e.g. "6H" stands for a tick every 6 hours, "H" for a tick each hour. Possible units: S: milli seconds s: seconds m: minutes H: hours D: days W: weeks M: months Q: quarters y: years
Return Value	JGTimeScaleRibbon	The current time scale ribbon.

Also see addFullTickRule
addMajorTickRule
addMajorTickRule
addMinorTickRule
addTickRule

Example Code

```
new JGTimeScaleRibbon("Day").addFullTickRule("D");
```

addMajorTickRule

Method of JGTimeScaleRibbon

By this method you can add a major tick rule. Major ticks may have texts (centered above the ticks) and are drawn by long lines. In the code example the ticks will appear each five minutes, in the texts hours and minutes appear.

Declaration
JGTimeScaleRibbon addMajorTickRule (java.lang.String stepRule, java.lang.String dateFormat)

	Data Type	Description
Parameter		
stepRule	java.lang.String	Step rule for this tick rule. A step rule consists of a factor and a unit, e.g. "6H" stands for a tick every 6 hours, "H" for a tick each hour. Possible units: S: milli seconds s: seconds m: minutes H: hours D: days W: weeks M: months Q: quarters y: years
dateFormat	java.lang.String	This is the pattern of the SimpleDateFormat that formats the texts of this tick rule. If this parameter equals null, no texts will appear.
Return Value	JGTimeScaleRibbon	The current time scale ribbon.

Also see addFullTickRule
addFullTickRule
addMajorTickRule
addMinorTickRule
addTickRule

Example Code

```
new JGTimeScaleRibbon("Minute").addMajorTickRule("5m", "hh:mm");
```

addMajorTickRule

Method of JGTimeScaleRibbon

By this method you can add a major tick rule. Major ticks may have texts (centered above the ticks) and are drawn by long lines. In the code example the ticks will appear each five minutes, the text format is determined by the step rule, thus only minutes appear.

Declaration
JGTimeScaleRibbon addMajorTickRule (java.lang.String stepRule)

	Data Type	Description
Parameter		
stepRule	java.lang.String	Step rule for this tick rule. A step rule consists of a factor and a unit, e.g. "6H" stands for a tick every 6 hours, "H" for a tick each hour. Possible units: S: milli seconds s: seconds m: minutes H: hours D: days W: weeks M: months Q: quarters y: years
Return Value	JGTimeScaleRibbon	The current time scale ribbon.

Also see
addFullTickRule
addFullTickRule
addMajorTickRule
addMinorTickRule
addTickRule

Example Code

```
new JGTimeScaleRibbon("Minute").addMajorTickRule("5m");
```

addMinorTickRule

Method of JGTimeScaleRibbon

By this method you can add a minor tick rule. Minor ticks cannot have texts and are drawn by short lines. In the code example the ticks will appear each five minutes.

Declaration
JGTimeScaleRibbon addMinorTickRule (java.lang.String stepRule)

	Data Type	Description
Parameter		
stepRule	java.lang.String	Step rule for this tick rule. A step rule consists of a factor and a unit, e.g. "6H" stands for a tick every 6 hours, "H" for a tick each hour. Possible units: S: milli seconds s: seconds m: minutes H: hours D: days W: weeks M: months Q: quarters y: years
Return Value	JGTimeScaleRibbon	The current time scale ribbon.

Also see addFullTickRule
addFullTickRule
addMajorTickRule
addMajorTickRule
addTickRule

Example Code

```
new JGTimeScaleRibbon("Minute").addMinorTickRule("5m");
```

addTickRule

Method of JGTimeScaleRibbon

By this method you can add a arbitrary rule for ticks.
This is an "expert" method, if you have problems, please consult Netronic.

Declaration
JGTimeScaleRibbon addTickRule (java.lang.String stepRule, int tickHeight, java.awt.Color lineColor, NeTimeScaleDateFormat dateFormat, java.awt.Font font, java.awt.Color textColor, int textPosition, double relativeFontSize, int tickWidth)

	Data Type	Description
Parameter		
stepRule	java.lang.String	Step rule for this tick rule. A step rule consists of a factor and a unit, e.g. "6H" stands for a tick every 6 hours, "H" for a tick each hour. Possible units: S: milli seconds s: seconds m: minutes H: hours D: days W: weeks M: months Q: quarters y: years
tickHeight	int	Height of the tick. Possible values: NeITimeScale.TICKHEIGHT_MINOR NeITimeScale.TICKHEIGHT_MAJOR NeITimeScale.TICKHEIGHT_FULL An absolute tick height, e.g. 200 may also be specified, the unit is mm/100.

lineColor	java.awt.Color	Line color of this tick rule. If this parameter is null, the active line color of time scale ribbon will be used.
dateFormat	NeTimeScaleDateFormat	NeTimeScaleDateFormat of this tick rule.
font	java.awt.Font	Font for the texts of this tick rule. If this font is null, the active text font of the time scale ribbon is used.
textColor	java.awt.Color	Text color of this tick rule. If this is null, the active text color of the time scale ribbon is used.
textPosition	int	Text position Possible values: NeITimeScale.TICK_CENTERED NeITimeScale.AREA_CENTERED NeITimeScale.AREA_CENTERED_HIGH
	Possible Values:	
	AREA_CENTERED	The text is drawn centered within the segment.
	AREA_CENTERED_HIGH	The text is drawn centered within the segment. The text is slightly raised, thus minor ticks can be drawn below the text.
	TICK_CENTERED	The text is drawn centered above the ticks.
relativeFontSize	double	Relative font size of this tick rule.
tickWidth	int	Width of the tick, the unit is mm / 100.

Return Value	JGTimeScaleRibbon	The current time scale ribbon.

Also see addFullTickRule
addFullTickRule
addMajorTickRule
addMajorTickRule
addMinorTickRule

9.4 JGTimeScaleRibbonStripe

Belongs to TimeScale

Package name **de.netronic.jgantt.timescale**
Extends **JGTimeScaleElement**

This class specifies a time scale ribbon stripe, whch consists of several time scale ribbons.

Constructors of the Class

JGTimeScaleRibbonStripe

Constructor of JGTimeScaleRibbonStripe

Copy constructor.

Declaration
JGTimeScaleRibbonStripe (JGTimeScaleRibbonStripe source)

Parameter	Data Type	Description
source	JGTimeScaleRibbonStripe	The time scale ribbon stripe, from which the properties of the new time scale ribbon stripe will be copied.

JGTimeScaleRibbonStripe

Constructor of JGTimeScaleRibbonStripe

Declaration
JGTimeScaleRibbonStripe (java.lang.String name, double minimumAbsoluteResolution, JGTimeScaleRibbon[] ribbons)

Parameter	Data Type	Description
name	java.lang.String	The name of the time scale ribbon stripe
minimumAbsoluteResolution	double	Minimum absolute resolution of this time scale ribbon stripe
ribbons	JGTimeScaleRibbon[]	The time scale ribbons of this time scale ribbon stripe.

Properties of the Class

MinimumAbsoluteResolution

Property of JGTimeScaleRibbonStripe

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Minimum absolute resolution of this time scale ribbon stripe.

If the absolute resolution of the holding time scale section gets smaller than the minimum absolute resolution of this timescale ribbon, the ribbon cannot be sensibly drawn any more.
This property is used in the dynamic time scale to define resolutions where the visible ribbon stripe of a time scale section is changed.

Accessing Methods
void setMinimumAbsoluteResolution (double newValue)
double getMinimumAbsoluteResolution ()

Name

Property of JGTimeScaleRibbonStripe

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

The name of the time scale ribbon stripe

Accessing Methods
void setName (java.lang.String newValue)
java.lang.String getName ()

Ribbons

Property of JGTimeScaleRibbonStripe

Typ	**JGTimeScaleRibbon[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can administer the time scale ribbons of a time scale ribbon stripe.

In the upper time scale the ribbons will appear from the top down, in the lower time scale they will appear from the bottom up.

Accessing Methods
void setRibbons (integer index, JGTimeScaleRibbon newValues)
void setRibbons (JGTimeScaleRibbon[] newValue)
JGTimeScaleRibbon getRibbons (integer index)
JGTimeScaleRibbon[] getRibbons ()

9.5 JGTimeScaleSection

Belongs to TimeScale

Package name	**de.netronic.jgantt.timescale**
Extends	**JGTimeScaleElement**

This class specifies a time scale section. A time scale may consist of several sections.

Constructors of the Class

JGTimeScaleSection

Constructor of JGTimeScaleSection

Copy constructor.

Declaration
JGTimeScaleSection (JGTimeScaleSection Source)

Parameter	Data Type	Description
Source	JGTimeScaleSection	The time scale section which specifies the properties of the new section.

Properties of the Class

AbsoluteResolution

Property of JGTimeScaleSection

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the absolute resolution of the time scale section.
It equals the width of 1mm per day.
Normally the absolute resolutions of the timescale sections differ.
If the time scale is dynamic the absolute resolution of a time scale section determines the visible ribbon stripe.

Accessing Methods
void setAbsoluteResolution (double newValue)
double getAbsoluteResolution ()

CoarsestRibbonStripe

Property of JGTimeScaleSection

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Coarsest step ribbon stripe allowed of the dynamic time scale.
This property only matters in a dynamic time scale.
The dynamic time scale contains a list of ribbon stripes, ordered by minimum absolute resolution. This property specifies the coarsest ribbon stripe which can be selected from this list. Thereby automatically a coarsest absolute resolution of the time scale section is set.

Possible Values	Description
RIBBONSTRIPE_10_MILLISECONDS	A time scale ribbon stripe with minor ticks every 10 milliseconds, major ticks every 50 milliseconds and texts every 100 milliseconds
RIBBONSTRIPE_10_YEARS	A time scale ribbon stripe with minor ticks every 10 years, major ticks every 50 years and texts every 100 years. This is the coarsest predefined ribbon stripe.
RIBBONSTRIPE_100_MILLISECONDS	A time scale ribbon stripe with minor ticks every 100 milliseconds, major ticks every 500 milliseconds and texts every second.
RIBBONSTRIPE_12_HOURS	A time scale ribbon stripe with minor ticks every 12 hours and texts every day.
RIBBONSTRIPE_15_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every 15 minutes, small texts every hour and big texts every three hours.
RIBBONSTRIPE_15_MINUTES_FINE	A time scale ribbon stripe with minor ticks every 15 minutes and texts every hour.
RIBBONSTRIPE_15_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 15 seconds, small texts every minute and big texts every 5 minutes.
RIBBONSTRIPE_15_SEONDS_FINE	A time scale ribbon stripe with minor ticks every 15 seconds and texts every minute.
RIBBONSTRIPE_3_HOURS	A time scale ribbon stripe with minor ticks every 3 hours and small texts every 6 hours.
RIBBONSTRIPE_3_MONTHS	A time scale ribbon stripe with minor ticks every 3 months, major ticks every 6 months and texts every year.
RIBBONSTRIPE_30_MINUTES	A time scale ribbon stripe with minor ticks every 30 minutes, major ticks every hour and texts every 3hours.
RIBBONSTRIPE_30_SECONDS	A time scale ribbon stripe with minor ticks every 30 seconds, major ticks every minute and texts every 5 minutes.
RIBBONSTRIPE_5_MILLISECONDS	A time scale ribbon stripe with minor ticks every 5 milliseconds, major ticks every 10 milliseconds and texts every 50 milliseconds.
RIBBONSTRIPE_5_MINUTES	A time scale ribbon stripe with minor ticks every 5 minutes, major ticks every 15 minutes and texts every 30 minutes.
RIBBONSTRIPE_5_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_SECONDS_FINE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.

RIBBONSTRIPE_5_YEARS	A time scale ribbon stripe with minor ticks every 5 years, major ticks every 10 years, small texts every 50 years and big tets every 100 years.
RIBBONSTRIPE_50_MILLISECONDS	A time scale ribbon stripe with minor ticks every 50 milliseconds, major ticks every 100 milliseconds and texts every 500 milliseconds.
RIBBONSTRIPE_500_MILLISECONDS	A time scale ribbon stripe with minor ticks every 500 milliseconds, major ticks every second and texts every 5 seconds.
RIBBONSTRIPE_6_HOURS	A time scale ribbon stripe with minor ticks every 6 hours and texts every day.
RIBBONSTRIPE_6_MONTHS	A time scale ribbon stripe with minor ticks every 6 months, major ticks every year, small texts every 5 years and big texts every 10 years.
RIBBONSTRIPE_DAY_COARSE	A time scale ribbon stripe with minor ticks every day and texts every week.
RIBBONSTRIPE_DAY_FINE	A time scale ribbon stripe with texts every day and thick full ticks every week.
RIBBONSTRIPE_HOUR	A time scale ribbon stripe with minor ticks every hour, major ticks every 3 hours and texts every 6 hours.
RIBBONSTRIPE_MILLISECOND	A time scale ribbon stripe with minor ticks every millisecond, major ticks every 5 milliseconds, small texts every 10 milliseconds and big texts every 50 milliseconds. This is the finest predefined ribbon stripe.
RIBBONSTRIPE_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every minute, major ticks every 5 minutes and texts every 15 minutes.
RIBBONSTRIPE_MINUTES_FINE	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MINUTES_MEDIUM	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MONTH_COARSE	A time scale ribbon stripe with minor ticks every month and texts every 3 months.
RIBBONSTRIPE_MONTH_FINE	A time scale ribbon stripe with month names three letters long.
RIBBONSTRIPE_MONTH_MEDIUM	A time scale ribbon stripe with month names one letter long.
RIBBONSTRIPE_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every second, major ticks every 5 seconds and texts every 15 seconds.
RIBBONSTRIPE_SECONDS_FINE	A time scale ribbon stripe with minor ticks every second, small texts every 5 seconds and large texts every 15 seconds.

RIBBONSTRIPE_WEEK_COARSE	A time scale ribbon stripe with minor ticks every week, major ticks every 5 calendar weeks and texts every 10 calendar weeks.
RIBBONSTRIPE_WEEK_FINE	A time scale ribbon stripe with days in month as text at beginning of every week.
RIBBONSTRIPE_WEEK_MEDIUM	A time scale ribbon stripe with minor ticks every week and texts every 5 calendar weeks.
RIBBONSTRIPE_YEAR	A time scale ribbon stripe with minor ticks every year, major ticks every 5 years, small texts every 10 years and big texts every 50 years:

Accessing Methods
void setCoarsestRibbonStripe (java.lang.String newValue)
java.lang.String getCoarsestRibbonStripe ()

Also see FinestRibbonStripe

ColorScheme

Property of JGTimeScaleSection

Typ	**JGColorScheme**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the color scheme for the time scale section.
The color scheme determines the following properties of the time scale section:

The mainColor sets the AreaStyle.
The shadedColor sets the AreaStyleForHigherRibbons,
The shadedAlternateColor sets the AreaStyleForTimeSpans.
The alternateColor sets the AreaStyleForHigherTimeSpans.
The lineColor sets the LineColor.
The mainTextColor sets the TextColor.

Accessing Methods
void setColorScheme (JGColorScheme newValue)
JGColorScheme getColorScheme ()

DateFormats

Property of JGTimeScaleSection

Typ	**NeTimeScaleDateFormats**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

All user defined date formats for all ribbon stripe indexes and types or step untis.

Accessing Methods
void setDateFormats (NeTimeScaleDateFormats newValue)
NeTimeScaleDateFormats getDateFormats ()

FinestRibbonStripe

Property of JGTimeScaleSection

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**null**

Finest step ribbon stripe allowed of the dynamic time scale.
This property only matters in a dynamic time scale.
The dynamic time scale contains a list of ribbon stripes, ordered by minimum absolute resolution. This property specifies the finest ribbon stripe which can be selected from this list. Thereby automatically a finest absolute resolution of the time scale section is set.

Possible Values	Description
RIBBONSTRIPE_10_MILLISECONDS	A time scale ribbon stripe with minor ticks every 10 milliseconds, major ticks every 50 milliseconds and texts every 100 milliseconds
RIBBONSTRIPE_10_YEARS	A time scale ribbon stripe with minor ticks every 10 years, major ticks every 50 years and texts every 100 years. This is the coarsest predefined ribbon stripe.
RIBBONSTRIPE_100_MILLISECONDS	A time scale ribbon stripe with minor ticks every 100 milliseconds, major ticks every 500 milliseconds and texts every second.
RIBBONSTRIPE_12_HOURS	A time scale ribbon stripe with minor ticks every 12 hours and texts every day.
RIBBONSTRIPE_15_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every 15 minutes, small texts every hour and big texts every three hours.
RIBBONSTRIPE_15_MINUTES_FINE	A time scale ribbon stripe with minor ticks every 15 minutes and texts every hour.
RIBBONSTRIPE_15_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 15 seconds, small texts every minute and big texts every 5 minutes.
RIBBONSTRIPE_15_SEONDS_FINE	A time scale ribbon stripe with minor ticks every 15 seconds and texts every minute.
RIBBONSTRIPE_3_HOURS	A time scale ribbon stripe with minor ticks every 3 hours and small texts every 6 hours.
RIBBONSTRIPE_3_MONTHS	A time scale ribbon stripe with minor ticks every 3 months, major ticks every 6 months and texts every year.
RIBBONSTRIPE_30_MINUTES	A time scale ribbon stripe with minor ticks every 30 minutes, major ticks every hour and texts every 3hours.
RIBBONSTRIPE_30_SECONDS	A time scale ribbon stripe with minor ticks every 30 seconds, major ticks every minute and texts every 5 minutes.
RIBBONSTRIPE_5_MILLISECONDS	A time scale ribbon stripe with minor ticks every 5 milliseconds, major ticks every 10 milliseconds and texts every 50 milliseconds.
RIBBONSTRIPE_5_MINUTES	A time scale ribbon stripe with minor ticks every 5 minutes, major ticks every 15 minutes and texts every 30 minutes.
RIBBONSTRIPE_5_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_SECONDS_FINE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.

RIBBONSTRIPE_5_YEARS	A time scale ribbon stripe with minor ticks every 5 years, major ticks every 10 years, small texts every 50 years and big tets every 100 years.
RIBBONSTRIPE_50_MILLISECONDS	A time scale ribbon stripe with minor ticks every 50 milliseconds, major ticks every 100 milliseconds and texts every 500 milliseconds.
RIBBONSTRIPE_500_MILLISECONDS	A time scale ribbon stripe with minor ticks every 500 milliseconds, major ticks every second and texts every 5 seconds.
RIBBONSTRIPE_6_HOURS	A time scale ribbon stripe with minor ticks every 6 hours and texts every day.
RIBBONSTRIPE_6_MONTHS	A time scale ribbon stripe with minor ticks every 6 months, major ticks every year, small texts every 5 years and big texts every 10 years.
RIBBONSTRIPE_DAY_COARSE	A time scale ribbon stripe with minor ticks every day and texts every week.
RIBBONSTRIPE_DAY_FINE	A time scale ribbon stripe with texts every day and thick full ticks every week.
RIBBONSTRIPE_HOUR	A time scale ribbon stripe with minor ticks every hour, major ticks every 3 hours and texts every 6 hours.
RIBBONSTRIPE_MILLISECOND	A time scale ribbon stripe with minor ticks every millisecond, major ticks every 5 milliseconds, small texts every 10 milliseconds and big texts every 50 milliseconds. This is the finest predefined ribbon stripe.
RIBBONSTRIPE_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every minute, major ticks every 5 minutes and texts every 15 minutes.
RIBBONSTRIPE_MINUTES_FINE	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MINUTES_MEDIUM	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MONTH_COARSE	A time scale ribbon stripe with minor ticks every month and texts every 3 months.
RIBBONSTRIPE_MONTH_FINE	A time scale ribbon stripe with month names three letters long.
RIBBONSTRIPE_MONTH_MEDIUM	A time scale ribbon stripe with month names one letter long.
RIBBONSTRIPE_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every second, major ticks every 5 seconds and texts every 15 seconds.
RIBBONSTRIPE_SECONDS_FINE	A time scale ribbon stripe with minor ticks every second, small texts every 5 seconds and large texts every 15 seconds.

RIBBONSTRIPE_WEEK_COARSE	A time scale ribbon stripe with minor ticks every week, major ticks every 5 calendar weeks and texts every 10 calendar weeks.
RIBBONSTRIPE_WEEK_FINE	A time scale ribbon stripe with days in month as text at beginning of every week.
RIBBONSTRIPE_WEEK_MEDIUM	A time scale ribbon stripe with minor ticks every week and texts every 5 calendar weeks.
RIBBONSTRIPE_YEAR	A time scale ribbon stripe with minor ticks every year, major ticks every 5 years, small texts every 10 years and big texts every 50 years:

Accessing Methods
void setFinestRibbonStripe (java.lang.String newValue)
java.lang.String getFinestRibbonStripe ()

Also see CoarsestRibbonStripe

Index

Read Only Property of JGTimeScaleSection

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Index of this time scale section in the time scale.

Accessing Methods
int getIndex()

Resolution

Property of JGTimeScaleSection

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property defines the resolution of the time scale section. The resolution is the width (in mm) of a single base unit of the time scale.
Normally the resolutions of the timescale sections differ.

Accessing Methods
void setResolution (double newValue)
double getResolution ()

VisibleRibbonStripe

Property of JGTimeScaleSection

Typ	**JGTimeScaleRibbonStripe**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Visible ribbon stripe of this time scale section. In a dynamic time scale the visible ribbons stripe depends on the absolute resolution of time scale section.

Accessing Methods
void setVisibleRibbonStripe (JGTimeScaleRibbonStripe newValue)
JGTimeScaleRibbonStripe getVisibleRibbonStripe ()

Methods of the Class

getDateFormats

Method of JGTimeScaleSection

This method lists for a time scale ribbon index and a type or step unit the specified date formats as patterns.

Declaration
void getDateFormats (int ribbonIndex, java.lang.String typeOrStepUnit)

	Data Type	Description
Parameter		
ribbonIndex	int	Index of the time scale ribbon the date formats of which you want to change.
typeOrStepUnit	java.lang.String	You can specify here either a time scale type (as under JGantt.setTimeScaleType) or a step unit. If you have a JGTimeScaleTickRule with step rule "3H", which paints a text, you change the format of this text by specifing the StepUnit "H" in this method.
Return Value	void	

setDateFormats

Method of JGTimeScaleSection

Here you can change the date formats of a time scale ribbon. You must specify the ribbon index, the time scale type or step unit, and one or more date formats. If you specify more than one date formats the higher ones must be shorter and will be used when the text must be shortened due to lack of space.

Sample:
timeScale.setDateFormats(1, "H", "HH 'h'", "HH");

All TickRules with a hour step unit add a "h" to the hours text. If this text is too long, the second format without the "h" is used.

Declaration
void setDateFormats (int ribbonIndex, java.lang.String typeOrStepUnit, java.lang.String dateFormat)

	Data Type	Description
Parameter		
ribbonIndex	int	Index of the time scale ribbon the date formats of which you want to change.
typeOrStepUnit	java.lang.String	You can specify here either a time scale type (as under JGantt.setTimeScaleType) or a step unit. If you have a JGTimeScaleTickRule with step rule "3H", which paints a text, you change the format of this text by specifing the StepUnit "H" in this method.
dateFormat	java.lang.String	The pattern for a NeTimeScaleDateFormat, you can use all possible patterns for a SimpleDateFormat.
Return Value	void	

Example Code

```
timeScaleSection.setDateFormats(0, jGantt1.getTimeScaleType(), "yyyy-MM", "yy-MM", "MM");
timeScaleSection.setDateFormats(1, jGantt1.getTimeScaleType(), "dd", "");
```

setVisibleRibbonStripe

Method of JGTimeScaleSection

This property specifies the visible ribbon stripe by the name of a predefined ribbon stripe.

Declaration
void setVisibleRibbonStripe (java.lang.String stripe)

	Data Type	Description
Parameter		
stripe	java.lang.String	Name of the time scale ribbon stripe.
	Possible Values:	
	RIBBONSTRIPE_10_MILLISECONDS	A time scale ribbon stripe with minor ticks every 10 milliseconds, major ticks every 50 milliseconds and texts every 100 milliseconds
	RIBBONSTRIPE_10_YEARS	A time scale ribbon stripe with minor ticks every 10 years, major ticks every 50 years and texts every 100 years. This is the coarsest predefined ribbon stripe.
	RIBBONSTRIPE_100_MILLISECONDS	A time scale ribbon stripe with minor ticks every 100 milliseconds, major ticks every 500 milliseconds and texts every second.
	RIBBONSTRIPE_12_HOURS	A time scale ribbon stripe with minor ticks every 12 hours and texts every day.
	RIBBONSTRIPE_15_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every 15 minutes, small texts every hour and big texts every three hours.
	RIBBONSTRIPE_15_MINUTES_FINE	A time scale ribbon stripe with minor ticks every 15 minutes and texts every hour.
	RIBBONSTRIPE_15_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 15 seconds, small texts every minute and big texts every 5 minutes.
	RIBBONSTRIPE_15_SEONDS_FINE	A time scale ribbon stripe with minor ticks every 15 seconds and texts every minute.
	RIBBONSTRIPE_3_HOURS	A time scale ribbon stripe with minor ticks every 3 hours and small texts every 6 hours.
	RIBBONSTRIPE_3_MONTHS	A time scale ribbon stripe with minor ticks every 3 months, major ticks every 6 months and texts every year.
	RIBBONSTRIPE_30_MINUTES	A time scale ribbon stripe with minor ticks every 30 minutes, major ticks every hour and texts every 3hours.
	RIBBONSTRIPE_30_SECONDS	A time scale ribbon stripe with minor ticks every 30 seconds, major ticks every minute and texts every 5 minutes.

RIBBONSTRIPE_5_MILLISECONDS	A time scale ribbon stripe with minor ticks every 5 milliseconds, major ticks every 10 milliseconds and texts every 50 milliseconds.
RIBBONSTRIPE_5_MINUTES	A time scale ribbon stripe with minor ticks every 5 minutes, major ticks every 15 minutes and texts every 30 minutes.
RIBBONSTRIPE_5_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_SECONDS_FINE	A time scale ribbon stripe with minor ticks every 5 seconds, major ticks every 15 seconds and texts every 30 seconds.
RIBBONSTRIPE_5_YEARS	A time scale ribbon stripe with minor ticks every 5 years, major ticks every 10 years, small texts every 50 years and big tets every 100 years.
RIBBONSTRIPE_50_MILLISECONDS	A time scale ribbon stripe with minor ticks every 50 milliseconds, major ticks every 100 milliseconds and texts every 500 milliseconds.
RIBBONSTRIPE_500_MILLISECONDS	A time scale ribbon stripe with minor ticks every 500 milliseconds, major ticks every second and texts every 5 seconds.
RIBBONSTRIPE_6_HOURS	A time scale ribbon stripe with minor ticks every 6 hours and texts every day.
RIBBONSTRIPE_6_MONTHS	A time scale ribbon stripe with minor ticks every 6 months, major ticks every year, small texts every 5 years and big texts every 10 years.
RIBBONSTRIPE_CALENDARWEEK	A time scale ribbon stripe with calendar weeks.
RIBBONSTRIPE_DAY_CALENDARWEEK	A time scale ribbon stripe with days and calendar weeks.
RIBBONSTRIPE_DAY_COARSE	A time scale ribbon stripe with minor ticks every day and texts every week.
RIBBONSTRIPE_DAY_FINE	A time scale ribbon stripe with texts every day and thick full ticks every week.
RIBBONSTRIPE_HOUR	A time scale ribbon stripe with minor ticks every hour, major ticks every 3 hours and texts every 6 hours.
RIBBONSTRIPE_MILLISECOND	A time scale ribbon stripe with minor ticks every millisecond, major ticks every 5 milliseconds, small texts every 10 milliseconds and big texts every 50 milliseconds. This is the finest predefined ribbon stripe.

RIBBONSTRIPE_MINUTES_COARSE	A time scale ribbon stripe with minor ticks every minute, major ticks every 5 minutes and texts every 15 minutes.
RIBBONSTRIPE_MINUTES_FINE	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MINUTES_MEDIUM	A time scale ribbon stripe with minor ticks every minute, small texts every 5 minutes and big texts every 15 minutes.
RIBBONSTRIPE_MONTH_COARSE	A time scale ribbon stripe with minor ticks every month and texts every 3 months.
RIBBONSTRIPE_MONTH_FINE	A time scale ribbon stripe with month names three letters long.
RIBBONSTRIPE_MONTH_MEDIUM	A time scale ribbon stripe with month names one letter long.
RIBBONSTRIPE_QUARTER	A time scale ribbon stripe with texts for every quarter.
RIBBONSTRIPE_RELATIVE_DAYS	A time scale ribbon stripe with relative days.
RIBBONSTRIPE_RELATIVE_HOURS	A time scale ribbon stripe with relative hours.
RIBBONSTRIPE_SECONDS_COARSE	A time scale ribbon stripe with minor ticks every second, major ticks every 5 seconds and texts every 15 seconds.
RIBBONSTRIPE_SECONDS_FINE	A time scale ribbon stripe with minor ticks every second, small texts every 5 seconds and large texts every 15 seconds.
RIBBONSTRIPE_WEEK_COARSE	A time scale ribbon stripe with minor ticks every week, major ticks every 5 calendar weeks and texts every 10 calendar weeks.
RIBBONSTRIPE_WEEK_FINE	A time scale ribbon stripe with days in month as text at beginning of every week.
RIBBONSTRIPE_WEEK_MEDIUM	A time scale ribbon stripe with minor ticks every week and texts every 5 calendar weeks.
RIBBONSTRIPE_WEEKDAY	A time scale ribbon stripe with days in week and days in month.
RIBBONSTRIPE_WEEKDAY_CALENDARWEEK	A time scale ribbon stripe with days in week, days in month and calendar weeks.

	RIBBONSTRIPE_YEAR	A time scale ribbon stripe with minor ticks every year, major ticks every 5 years, small texts every 10 years and big texts every 50 years:
Return Value	void	

9.6 NeITimeScaleSegmentsGenerator

Belongs to TimeScale

Package name **de.netronic.bean.timescale**

You only need this class, if you want to draw a time scale ribbon yourself.

This interface together with the property SegmentsGenerator of the time scale ribbon enables you to draw ticks, texts and background colors of the time scale ribbon absolutely freely.

Methods of the Interface

createSegmentList

Method of NeITimeScaleSegmentsGenerator

This methods generates a list of NeTimeScaleSegments.

Declaration
List<NeTimeScaleSegment> createSegmentList (java.util.Date startDate, java.util.Date endDate)

	Data Type	Description
Parameter		
startDate	java.util.Date	Start date for the generation of time scale elements,
endDate	java.util.Date	End date for the generation of time scale elements,
Return Value	List<NeTimeScaleSegment>	

9.7 NeTimeScaleDateFormats

Belongs to TimeScale

Package name **de.netronic.bean.timescale**

For a time scale section this class specifies user defined date formats.

Constructors of the Class

NeTimeScaleDateFormats

Constructor of NeTimeScaleDateFormats

Creation of a new set of date formats.

Declaration
NeTimeScaleDateFormats ()

Methods of the Class

getDateFormatPatterns

Method of NeTimeScaleDateFormats

This method lists for a time scale ribbon index and a type or step unit the specified date formats as patterns.

Declaration
java.lang.String[] getDateFormatPatterns (int ribbonIndex, java.lang.String typeOrStepUnit)

	Data Type	Description
Parameter		
ribbonIndex	int	Index of the time scale ribbon.
typeOrStepUnit	java.lang.String	Time scale type (as under JGantt.setTimeScaleType) or a step unit.
Return Value	java.lang.String[]	Simple date format patterns for this ribbon index and this time scale type.

Example Code

```
dateFormats.getDateFormatPatterns(0, jGantt1.getTimeScaleType());
```

getSpecifiedTypesOrStepUnits

Method of NeTimeScaleDateFormats

Retrieves the specified types or step units for this ribbon index.

Declaration
java.lang.String[] getSpecifiedTypesOrStepUnits (int ribbonIndex)

	Data Type	Description
Parameter		
ribbonIndex	int	Index of the time scale ribbon the types or step units of which shall be retrieved.
Return Value	java.lang.String[]	

setDateFormatPatterns

Method of NeTimeScaleDateFormats

Here you can change the date formats of a time scale ribbon. You must specify the ribbon index, the time scale type or step unit, and one or more date formats. If you specify more than one date formats the higher ones must be shorter and will be used when the text must be shortened due to lack of space.

In the code sample all TickRules with a hour step unit add a "h" to the hours text. If this text is too long, the second format without the "h" is used.

Declaration
void setDateFormatPatterns (int ribbonIndex, java.lang.String typeOrStepUnit, java.lang.String dateFormat)

	Data Type	Description
Parameter		
ribbonIndex	int	Index of the time scale ribbon the date formats of which you want to change.
typeOrStepUnit	java.lang.String	Time scale type (as under JGantt.setTimeScaleType) or a step unit. If you have a JGTimeScaleTickRule with step rule "3H", which paints a text, you change the format of this text by specifing the StepUnit "H" in this method.
dateFormat	java.lang.String	The pattern for a NeTimeScaleDateFormat, all possible patterns for a SimpleDateFormat may be used.
Return Value	void	

Example Code

```
dateFormats.setDateFormatPatterns(1, "H", "HH 'h'", "HH");
```

9.8 NeTimeScaleSegment

Belongs to TimeScale

Package name **de.netronic.bean.timescale**

You need this class only, if you want to design a time scale ribbon using the interface NeITimeScaleSegmentsGenerator.

This class specifies all properties of a time scale segment, this is the area from one time scale tick to the next.

Constructors of the Class

NeTimeScaleSegment

Constructor of NeTimeScaleSegment

You only need this class, if you want to design a time scale ribbon using the interface NeITimeScaleSegmentsGenerator.

Declaration
NeTimeScaleSegment (java.util.Date start, java.util.Date end, java.util.Date endForAreaCentered, int tickHeight, int tickWidth, int textPosition, java.awt.Font textFont, NeTimeScaleDateFormat format, java.awt.Color textColor, java.lang.String text, java.lang.String text2, java.awt.Color lineColor, NeILabel label)

Parameter	Data Type	Description
start	java.util.Date	Start date of the segment, at this position the ticks are drawn.
end	java.util.Date	End date of the segment.
endForAreaCentered	java.util.Date	End date of the segment for area centered text. Necessary if the text reaches over several segments.
tickHeight	int	Height of the tick to be drawn. Unit is mm/100.
tickWidth	int	Width of the tick to be drawn. Unit is mm/100.
textPosition	int	Position of the text to be drawn.
	Possible Values:	
	AREA_CENTERED	The text is drawn centered within the segment.
	AREA_CENTERED_HIGH	The text is drawn centered within the segment. The text is slightly raised, thus minor ticks can be drawn below the text.
	TICK_CENTERED	The text is drawn centered above the ticks.
textFont	java.awt.Font	Font of the text to be drawn.
format	NeTimeScaleDateFormat	Format of the text to be drawn.
textColor	java.awt.Color	Color of the text to be drawn.
text	java.lang.String	Text to be drawn.
text2	java.lang.String	Text to be drawn in a second line.
lineColor	java.awt.Color	Color of the tick to be drawn.
label	NeILabel	NeILabel which has to be drawn in the time scale segment.

10 Various Classes

This chapter describes the classes that either are used by several components or that cannot be allocated to a component.

The Various Classes-Component consists of the below classes:

JGDynamicRowColor	This class serves to generate in a simple way alternating or rotating color backgounds in the lines of the Gantt graph or of the table.
NeAnnotation	This class offers methods to handle annotations in nodes.
NeAreaStyle	An NeAreaStyle is an extension of the class java.
NeColorMap	This class offers methods to map colors to key values for the handling of dynamic colors.
NeCombinedFilter	This class provides a constructor that serves to generate a filter which contains a condition composed of conditions of other filters.
NeDateLine	This class allows to add a data line to the Gantt graph via different constructors.
NeDynamicLabel	This abstract class defines a dynamic NeILabel object, which can be of the type **NeAnnotation** or **NeSymbol**.
NeDynamicPicture	This abstract class defines a dynamic NePicture object.
NeEntityAttributeColor	This class allows to use values of **color** type AppData attributes as colors in VARCHART JGantt.
NeEntityComparator	This class lets you to compare entities.
NeEntityEditorDialog	An NeEntityEditorDialog serves for simple editing or for displaying attribute values of one or several entities.
NeGroupComparator	The class NeGroupComparator compares groups after their group entity.
NeIDrawingConstants	This interface offers constants that define different ways of drawing elements.
NeIDrawingElement	This interface holds properties and methods to handle DrawingElement objects, mainly for internal administrative purposes.
NeIDynamicColor	This interface defines a color that may change at runtime, for example in dependence of the value of an attribute.

NeIDynamicLabel	This interface offers methods to handle dynamic labels.
NeIDynamicPicture	This interface defines a picture object, the actual contents of which is determined dynamically at run time.
NeIFilter	This interface defines a filter that applies to entites.
NeIGroupComparator	When for sorting groups (see JGantt.
NeIGroupValueUpdater	This interface offers methods properties to generate group nodes (summary bars) and to calculate dates of summary bars.
NeILabel	This interface represents a label object.
NeILabelAttachment	This interface represents the placement of a label at an object.
NeILineAttributes	This interface lets you define the properties of a grid line.
NeILinkLabelAttachment	This interface represents the placement of a label at a link.
NeIPainter	This interface offers properties to paint drawing elements.
NeIPicture	This interface represents a picture object.
NeITransaction	Transactions summarize methods concerning the applicationData into a comprehensive operation.
NeITransactionHandler	Transactions summarize methods concerning the applicationData into a comprehensive operation.
NeIUserActionSource	This interface offers methods to handle action listeners.
NeIValueReference	This interface offers properties and methods to handle the NeIValueReference object.
NeLabelMap	A labelMap object handles allocations of values (or range values) and label objects.
NeLineStyle	An NeLineStyle is an extension of the class java.
NeMappedColor	This class represents a color object, the actual color of which is retrieved from a color map.
NeMappedLabel	A mappedLabel object is an implementation of NeIDynamicLabel, where the NeILabel object displayed is selected from a label map (NeLabelMap).
NeMappedPicture	A mappedPicture object is an implementation of NeIDynamicPicture, where the NeIPicture object displayed is selected from a picture map (NePictureMap).

NeNotFilter	This class offers a constructor that serves to generate a filter which contains an inverted condition.
NeObjectChangeAdapter	This class is an adapter class for the interface NeObjectChangeListener.
NeObjectChangeEvent	This class offers events for changements in objects.
NeObjectChangeInfo	This class offers properties to handle information on changes in objects.
NeObjectChangeListener	This is the listener interface for receiving object change events.
NePicture	Objects of the class **NePicture** are decorative elements, the appearence of which is defined by a bit map.
NePictureMap	A picureMap object handles allocations of values (or range values) and picure objects.
NePictureStack	A picture stack (class NePictureStack) is an impementation of NeIPicture, by which you can pile picture objects in the third (spacial) dimension.
NePictureStripe	A picture stripe is an implementation of NeIPicture, by which you can combine several NeIPicture objects into a horizontal or vertical picture stripe.
NeRelativeValueReference	This class offers methods to handle a relative date attribute.
NeRowInteractionAdapter	This class is an adapter class for the interface NeRowInteractionListener.
NeRowInteractionEvent	Event which is created when moving table rows.
NeRowInteractionListener	This is the listener interface for receiving row interaction events.
NeSimpleDateFormat	Beyond the features of class java.
NeSumValueReference	This class offers methods to handle a composed date attribute of an entity.
NeSymbol	Objects of the NeSymbol class are vector based symbols that for example are used in table fields and in nodes.
NeTransaction	Abstract class containing methods that can be overwritten if they should do jobs different to their default jobs.
NeTransactionAdapter	This class is an adapter class for the interface NeTransactionListener.
NeTransactionEvent	This class contains methods to handle the events of transactions.

NeTransactionListener	This is the listener interface for receiving transaction events.
NeUserAction	Contains the implementations of the interface NeIUserActionSource.
NeUserActionAdapter	This class is an adapter class for the interface NeUserActionExtendedListener.
NeUserActionEvent	This class contains methods to handle the events of user actions.
NeUserActionExtendedListener	This listener interface extends the interface NeUserActionListener by the method onUserActionPerformed, which is called after a user action has been performed.
NeUserActionListener	This is the listener interface for receiving userAction events.
NeValueFilter	This filter compares the value of an attribute to a pre-defined value.
NeValueReference	ValueReference objects transmit dates between layers and attributes of entities.
NeVetoException	An NeVetoException can be thrown by an application during the handling of an interaction notification event (for example onObjectChanged ()) sent by a JComponent in order to cancel the interaction.

10.1 JGDynamicRowColor

Belongs to VariousClasses

Package name **de.netronic.jgantt**
Implements **de.netronic.common.intface.NeIDynamicColor**

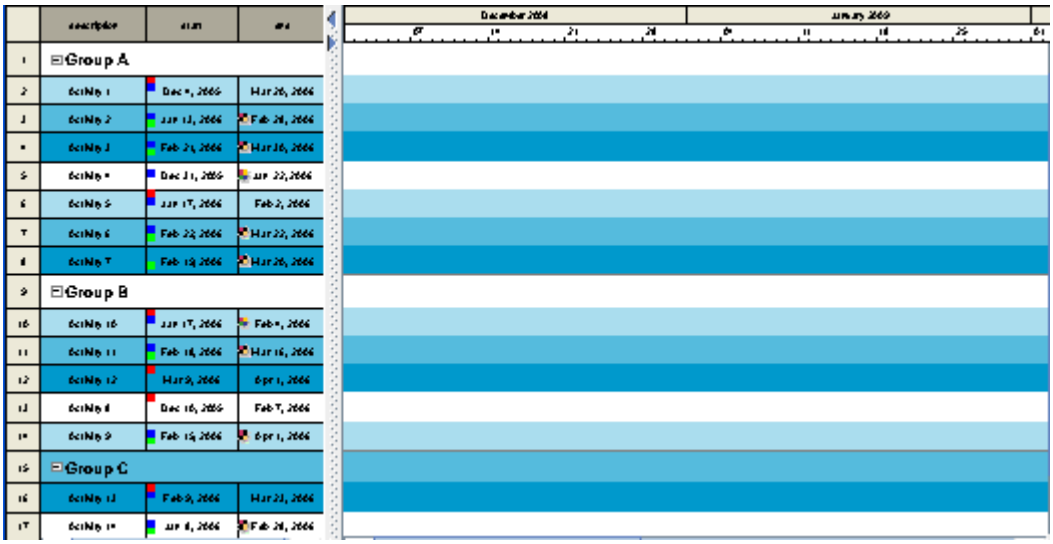
This class serves to generate in a simple way alternating or rotating color backgounds in the lines of the Gantt graph or of the table. By the simultaneous use as shadedColor in the ganttColorScheme and in the tableColorScheme the lines can be given the same color pattern in the Gantt graph and in the table.

Constructors of the Class

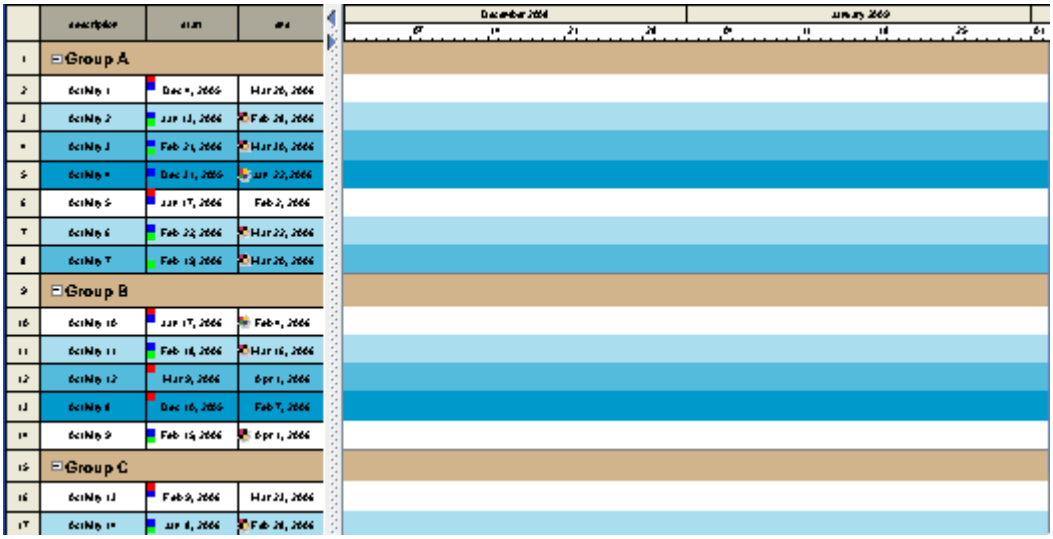
JGDynamicRowColor

Constructor of JGDynamicRowColor

This constructor allows to display a color gradient across severals rows of the Gantt graph. If you set a value to the parameter groupColor, the color will be used for group rows.



Color gradient, no color for group provided



Color gradient, color for group added

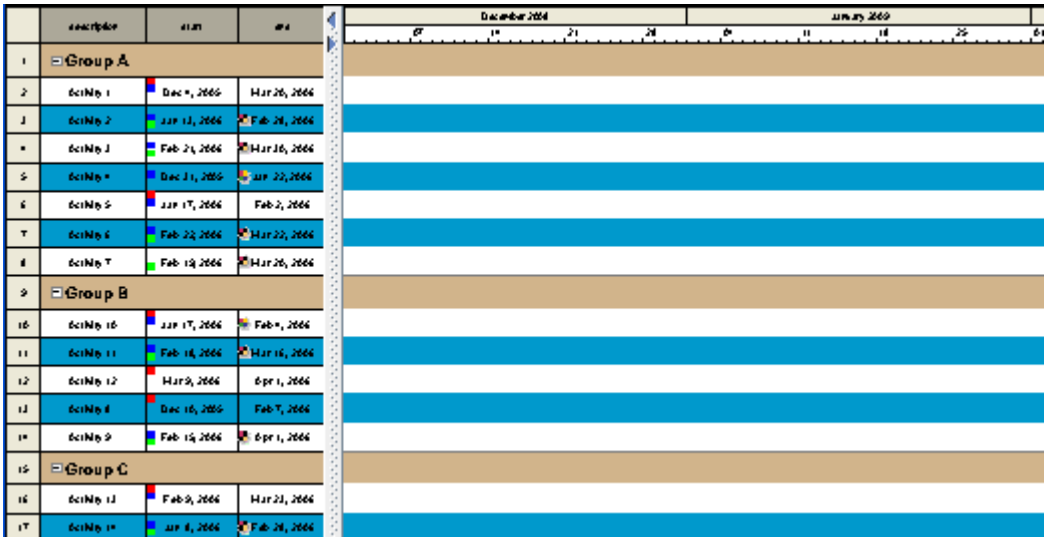
Declaration
JGDynamicRowColor (int noOfColors, JGantt jgantt, java.awt.Color oddColor, java.awt.Color evenColor, java.awt.Color groupColor)

Parameter	Data Type	Description
noOfColors	int	Number of color grades in the gradient.
jgantt	JGantt	Gantt object to which the colors apply.
oddColor	java.awt.Color	Starting color of the color gradient.
evenColor	java.awt.Color	Final color of the color gradient.
groupColor	java.awt.Color	Color for group lines

JGDynamicRowColor

Constructor of JGDynamicRowColor

This constructor allows to display the lines in a Gantt graph in two alternating colors, and to assign a separate color to groups.



Alternating colors plus additional group color (brown)

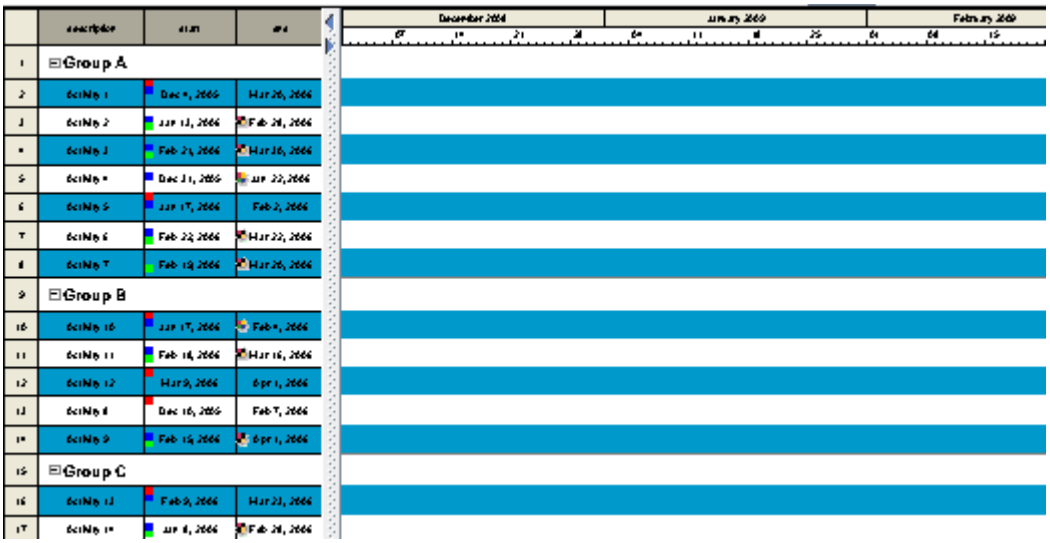
Declaration
JGDynamicRowColor (JGantt jgantt, java.awt.Color oddColor, java.awt.Color evenColor, java.awt.Color groupColor)

Parameter	Data Type	Description
jgantt	JGantt	Gantt object to which the colors apply.
oddColor	java.awt.Color	Color to be assgned to lines of odd numbers.
evenColor	java.awt.Color	Color to be assgned to lines of even numbers.
groupColor	java.awt.Color	Color for group lines

JGDynamicRowColor

Constructor of JGDynamicRowColor

This constructor allows to display the lines in a Gantt graph in two alternating colors.



Alternating colors

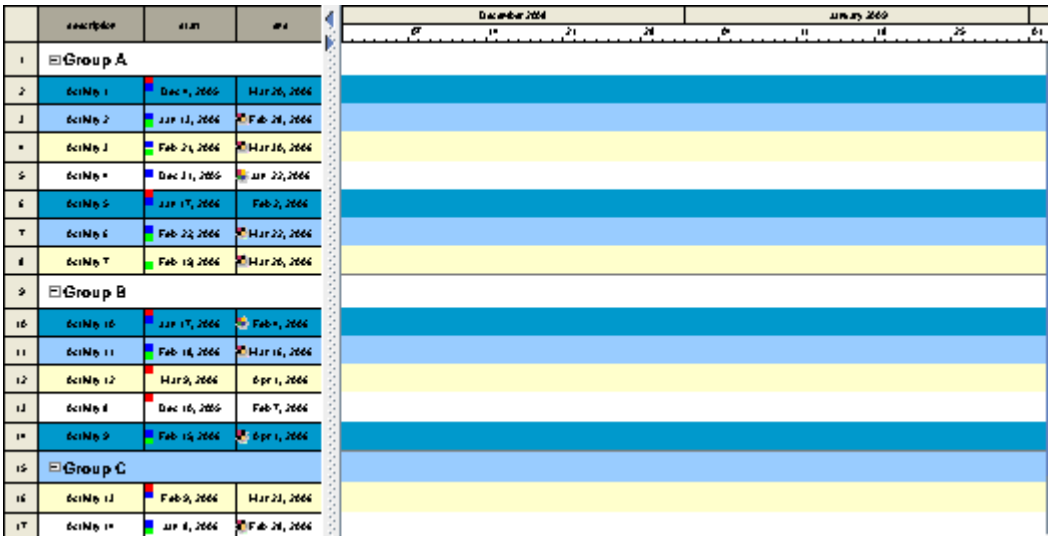
Declaration
JGDynamicRowColor (JGantt jgantt, java.awt.Color oddColor, java.awt.Color evenColor)

Parameter	Data Type	Description
jgantt	JGantt	Gantt object to which the colors apply.
oddColor	java.awt.Color	Color to be assigned to lines of odd numbers.
evenColor	java.awt.Color	Color to be assigned to lines of even numbers.

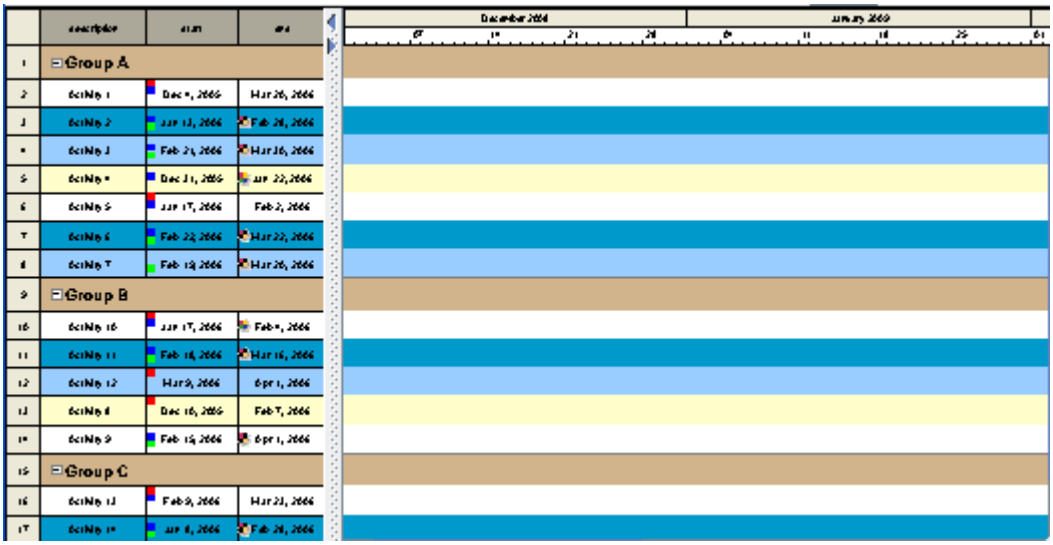
JGDynamicRowColor

Constructor of JGDynamicRowColor

This constructor allows to display different colors for several rows. The Gantt graph may show a color gradient this way, that can be repeated periodically, also for groups.



Recurring array of colors, no group color.



Recurring array of colors, group color provided.

Declaration
JGDynamicRowColor (JGantt jgantt, java.awt.Color colors [], boolean forGroup)

Parameter	Data Type	Description
jgantt	JGantt	Gantt object to which the colors apply.
colors []	java.awt.Color	Array of "rotating" colors. Of n colors, to the first line of the Gantt graph is assigned colors[0], to the second line is assigned colors[1] etc., up to [n-1]. Line number n receives colors[0] again.
forGroup	boolean	If this parameter is set to true, the array starts counting from the beginning by each group. Only n-1 colors at maximum are used for the subordinated lines within the group; color n is reserved for the line that holds the group node.

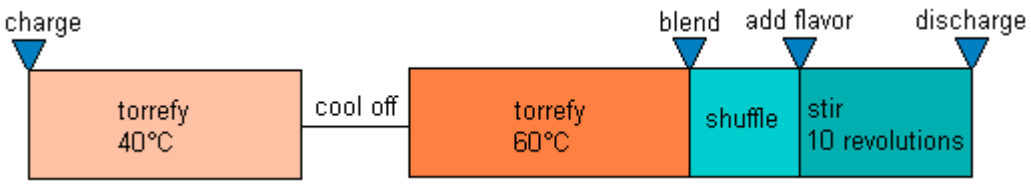
10.2 NeAnnotation

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIPicture**
I **de.netronic.common.intface.NeILabel**

This class offers methods to handle annotations in nodes.

Annotations for example can occur with nodes. Nodes consist of two types of graphical elements: "layers" and "decorations". The latter can be positioned in or on layers. Decorations are represented by picture objects (bitmaps), by symbols or by annotations. There are two different types of decorations, the NeIPicture type and the NeILabel type. So annotations can be used as both, as an NeIPicture type or an NeILabel type. Below, the node is composed of layers, symbols and annotations.



Please also see the classes NeSymbol and NePicture.

Properties to Compose the Appearance of an Annotation

3DMode	Annotation on a background of three-dimensional appearance
Alignment	Alignment of an annotation within the annotation rectangle
AutoWrapMode	Different automatic break modes for text lines
BackgroundStyle	Background style of the annotation
BorderStyle	Appearance of the surrounding border line of the annotation
ClipMode	Different clipping modes for the string if font size not adapted
DateFieldIDs	Attributes to hold annotations for the format string
DateFormat	Date format for the annotation
Extent	Extent of the annotation rectangle
Font	Font of the annotation
FormatString	Text modules to compose the annotation
Height	Heigth of the annotation
HorizontalPictureFillMode	Fill mode for the horizontal direction of the annotation
NumberFormat	Format for numbers
Offset	Offset between the reference points of the layer and the annotation
RefPointPosition	Reference point of the annotation
ScaleFontToFit	Fits the font to the size of the rectangle
TextColor	Font color
VerticalPictureFillMode	Fill mode for the vertical direction of the background rectangle

Constructors of the Class

NeAnnotation

Constructor of NeAnnotation

This constructor lets you generate an empty description.

Declaration

NeAnnotation ()

Properties of the Class

3DMode

Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**MODE_3D_NONE**

Background to the annotation, that can appear three-dimensionally raised or depressed.

Possible Values	Description
MODE_3D_LOWERED	Buttons and surfaces that are displayed in three-dimensional mode appear impressed.
MODE_3D_NONE	The three-dimensional display is switched off.
MODE_3D_RAISED	Buttons and surfaces that are displayed in three-dimensional mode appear raised.

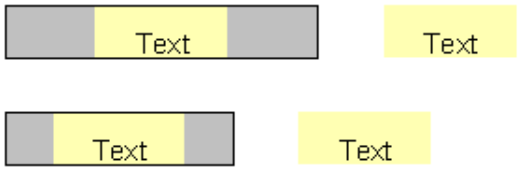
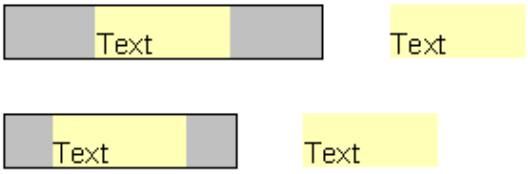
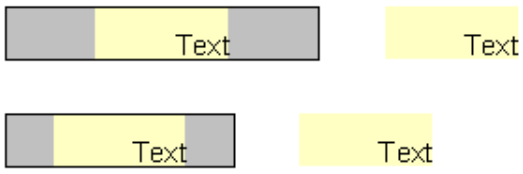
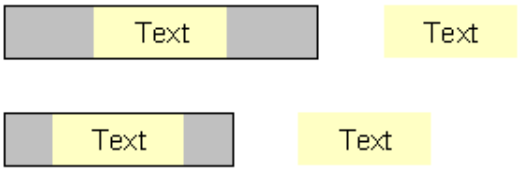
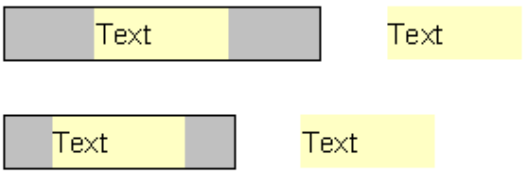
Accessing Methods
void set3DMode (int newValue)
int get3DMode ()

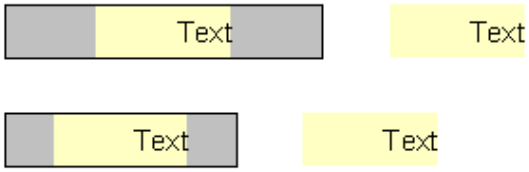
Alignment

Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**ALIGNMENT_CENTER_CENTER**

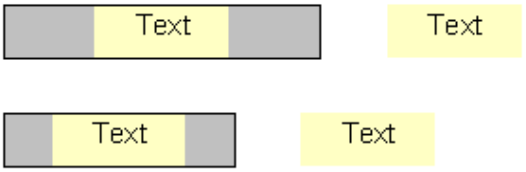
Alignment of an annotation within the annotation rectangle. If the annotation was assigned as an NeIPicture object, the size of its background rectangle is automatically set by the object that it was assigned to, for example by the layer. If it was assigned as an NeILabel, the size is set by the property **Extent** or derives from the size of the text, if the **Extent** property was not set.

Possible Values	Description
ALIGNMENT_BOTTOM_CENTER	The annotation is placed at the bottom in the center. 
ALIGNMENT_BOTTOM_LEFT	The annotation is aligned at the bottom on the left hand side. 
ALIGNMENT_BOTTOM_RIGHT	The annotation is aligned at the bottom on the right hand side. 
ALIGNMENT_CENTER_CENTER	The annotation is placed in the vertical and horizontal center. 
ALIGNMENT_CENTER_LEFT	The annotation is placed in the vertical center on the left hand side. 
ALIGNMENT_CENTER_RIGHT	The annotation is placed in the vertical center on the right hand side.



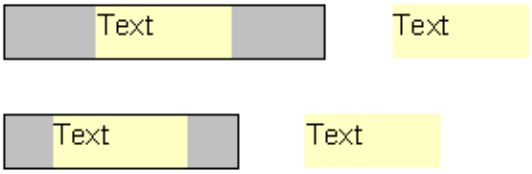
ALIGNMENT_TOP_CENTER

The annotation is placed at the top in the center.



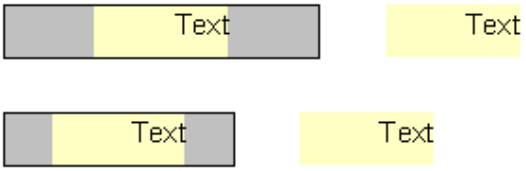
ALIGNMENT_TOP_LEFT

The annotation is placed at the top on the left hand side.



ALIGNMENT_TOP_RIGHT

The annotation is placed at the top on the right hand side.



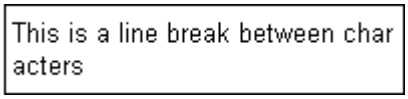
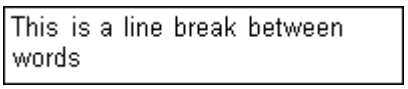
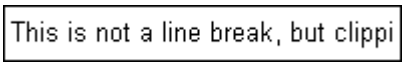
Accessing Methods
void setAlignment (int newValue)
int getAlignment ()

AutoWrapMode

Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO_WRAP_OFF**

This property lets you set different types of automatic line breaks to annotations.

Possible Values	Description
AUTO_WRAP_AT_CHARACTER	Automatic line breaks occur between characters: 
AUTO_WRAP_AT_WORD	Automatic line breaks occur between words: 
AUTO_WRAP_OFF	There are no automatic line breaks. To avoid the text to be clipped, line breaks can be set manually by <b\n!>: 

Accessing Methods
void setAutoWrapMode (int newValue)
int getAutoWrapMode ()

BackgroundStyle

Property of NeAnnotation

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Background color of the annotation. In the most simple case, this is the color of the background. Alternatively, you can use an object of the class NeAreaStyle

deriving from java.awt.Color, to generate patterns or color gradients for the background.

Accessing Methods
void setBackgroundStyle (java.awt.Color newValue)
java.awt.Color getBackgroundStyle ()

BorderStyle

Property of NeAnnotation

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Color object, that gives an appearance to the surrounding border line of the annotation. In the most simple case, the appearance consists of a color. Alternatively, you can set an object of the class NeLineStyle, which ist also is derived from java.awt.Color, in order to set a line style.

Accessing Methods
void setBorderStyle (java.awt.Color newValue)
java.awt.Color getBorderStyle ()

ClipMode

Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**CLIP_TO_RECTANGLE**

Different clipping modes for the annotation string, in case the font size is not adapted to the size of the positioning rectangle.

Possible Values	Description
CLIP_TO_RECTANGLE	Clips the string to the length of the rectangle.
COMPLETE_DISPLAY	The annotation string is not clipped and will be displayed only if the string fits completely in its rectangle.

Accessing Methods
void setClipMode (int newValue)
int getClipMode ()

Also see ScaleFontToFit

DateFieldIDs

Property of NeAnnotation

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Names of the attributes that hold annotations for the format strings. The content of the attributes will be used as variable text modules, probably combined with constant strings to compose the annotation.

Accessing Methods
void setDateFieldIDs (integer index, java.lang.String newValues)
void setDateFieldIDs (java.lang.String[] newValue)
java.lang.String getDateFieldIDs (integer index)
java.lang.String[] getDateFieldIDs ()

Also see FormatString

Example Code

```
//two attributes at index 0 and index 1
annotation.setAttributeNames ({"numOfActivity", "duration"});
//the two attributes are integrated via their indexes
annotation.setFormatString ("Activity no. {0}\n  will take {1} days");
```

DateFormat

Property of NeAnnotation

Typ	**java.text.DateFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Date format for the annotation, which will be used to format a date.

Accessing Methods
void setDateFormat (java.text.DateFormat newValue)
java.text.DateFormat getDateFormat ()

Extent

Property of NeAnnotation

Typ	**java.awt.Dimension**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

If you are using the annotation as a label object this property defines the extent of the rectangle, in which the annotation will appear.
If you are using the annotation as a picture object, this property will define the favorite size of the picture object.
Unit: 1/100 mm.

Accessing Methods
void setExtent (java.awt.Dimension newValue)
java.awt.Dimension getExtent ()

Font

Property of NeAnnotation

Typ	**java.awt.Font**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Font of the annotation

Accessing Methods
void setFont (java.awt.Font newValue)
java.awt.Font getFont ()

FormatString

Property of NeAnnotation

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Composes an annotation from variable and constant text modules. Before, texts in attributes have to be generated. The attributes need to be made known to the

annotation via the property setAttributeNames. After that, you can use their indexes in the format string. The contents of the attributes will appear in the places of the indexes.

Accessing Methods
void setFormatString (java.lang.String newValue)
java.lang.String getFormatString ()

Also see DateFieldIDs

Example Code

```
//two attributes at index 0 and index 1
annotation.setAttributeNames ({"numOfActivity", "duration"});
//the two attributes are integrated via their indexes
annotation.setFormatString ("Activity {0}\n  will take {1} days");
```

Height

Read Only Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the height of the annotation. Unit: 1/100 mm.

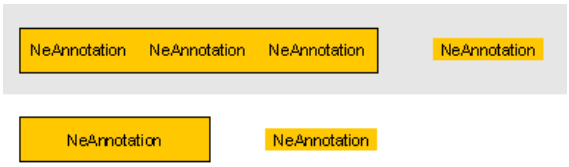
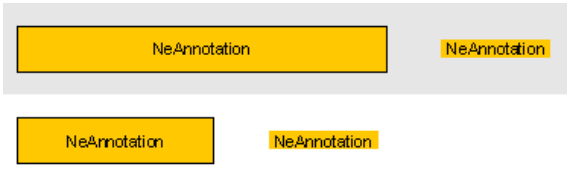
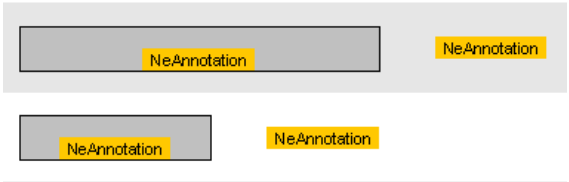
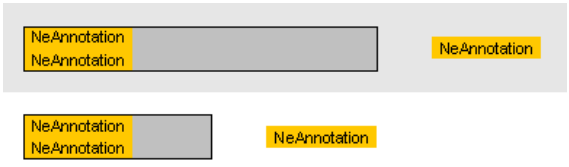
Accessing Methods
int getHeight()

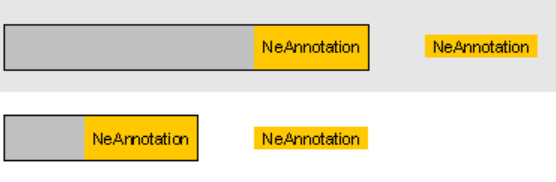
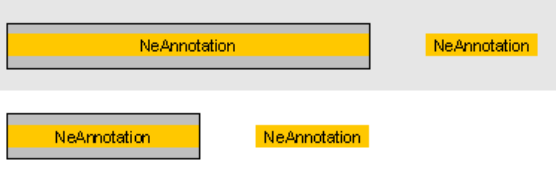
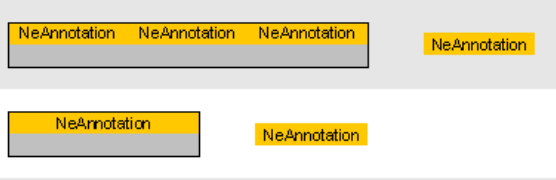
HorizontalPictureFillMode

Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the annotation fills its field in horizontal direction, if the annotation is used as an NeIPicture. Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the leftmost position and equals the constant LEFT; +100 describes the rightmost position and equals the constant RIGHT. If the annotation is used as an NeILabel, this property will not apply.

Possible Values	Description
AUTO	If by the property **Extent** a size of the width was specified, the annotation rectangle (background of the annotation) will be tiled in horizontal direction. If no extend was specified, the rectangle will be stretched in horizontal direction.  Extent specified: The rectangle ist tiled horizontally (while the vertical direction was set to stretching). Right of the layer there is the original size of the character string and its background.  With no extent set, the annotation rectangle is stretched to the size of the layer.
CENTER	The annotation is placed in the horizonatl center (value: 0) of the layer. At the same time, the vertical fill mode is set to "bottom": 
LEFT	The annotation is placed in the ultmate left position (value: -100) of the layer. In the picture below, at the same time the vertical fill mode was set to "tile": 
RIGHT	The annotation is placed in the rightmost position (value: +100) of the layer. In the picture below, at the same time the vertical fill mode was set to "STRETCH":

	
STRETCH	The layer is filled in horizontal direction by stretching the background of the annotation. In the picture below, at the same time the vertical fill mode was set to "center": 
TILE	The layer is filled in horizontal direction by tiling the annotation. In the picture below, at the same time the vertical fill mode was set to "top": 

Accessing Methods
void setHorizontalPictureFillMode (int newValue)
int getHorizontalPictureFillMode ()

Also see
- Extent
- Font
- ScaleFontToFit
- VerticalPictureFillMode

NumberFormat

Property of NeAnnotation

Typ	**java.text.NumberFormat**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Format that applies if the annotation consists of numbers.

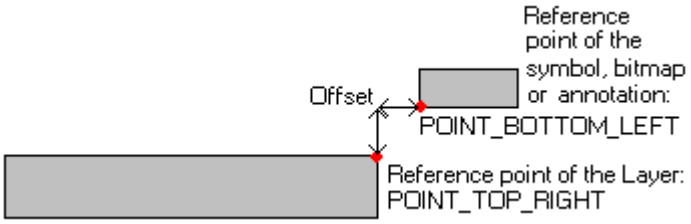
Accessing Methods
void setNumberFormat (java.text.NumberFormat newValue)
java.text.NumberFormat getNumberFormat ()

Offset

Property of NeAnnotation

Typ	**java.awt.Dimension**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

If an annotation, used as NeILabel, for example is positioned on the layer, the reference point of the layer and the reference point of the annotation are superimposed.
This property lets you add an offset by which their position will differ, or retrieve the offset added.



Accessing Methods
void setOffset (java.awt.Dimension newValue)
java.awt.Dimension getOffset ()

Also see RefPointPosition

RefPointPosition

Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**POSITION_CENTER_CENTER**

When annotation, used as NeILabel, is positioned at an object, the reference point of the object and the reference point of the annotation are superimposed.

This property lets you set the reference point to the anntation or retrieve the reference point set.

Possible Values	Description
POSITION_BOTTOM_CENTER	The reference point is situated in the center of the bottom border.
POSITION_BOTTOM_LEFT	The reference point is situated at the bottom left corner.
POSITION_BOTTOM_RIGHT	The reference point is situated at the bottom right corner.
POSITION_CENTER_CENTER	The reference point is situated in the vertical and horizontal center.
POSITION_CENTER_LEFT	The reference point is situated in the vertical center on the left hand side.
POSITION_CENTER_RIGHT	The refernce point is situated in the vertical center on the right hand side.
POSITION_TOP_CENTER	The reference point is situated in the center of the top border.
POSITION_TOP_LEFT	The refrence point is situated at the top left corner.
POSITION_TOP_RIGHT	The refernce point is situated at the top right corner.

Accessing Methods
void setRefPointPosition (int newValue)
int getRefPointPosition ()

Also see Offset

ScaleFontToFit

Property of NeAnnotation

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

Adapts the font to the size of the rectangle, that the annotation is placed in. **True**: the size of the font is adapted, **false**: the size is not adapted. If the string is not adapted, it will be clipped. Different clipping modes can be selected.

Accessing Methods
void setScaleFontToFit (boolean newValue)
boolean hasScaleFontToFit ()

Also see ClipMode

TextColor

Property of NeAnnotation

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Font color of the annotation.

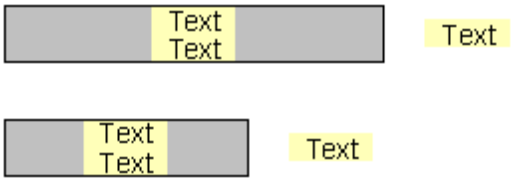
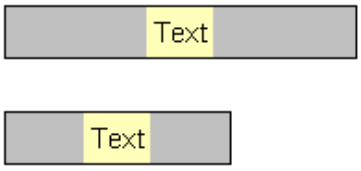
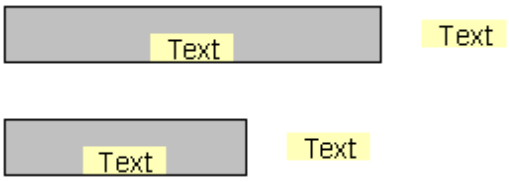
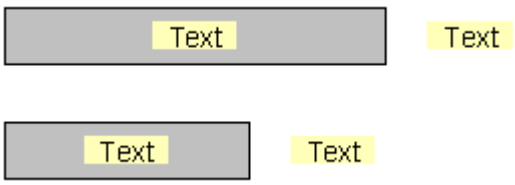
Accessing Methods
void setTextColor (java.awt.Color newValue)
java.awt.Color getTextColor ()

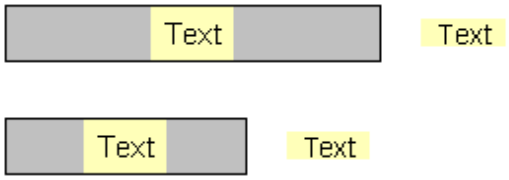
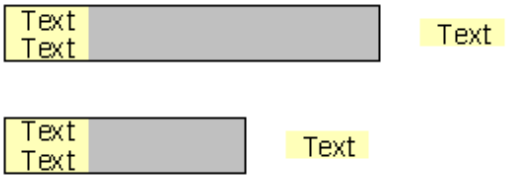
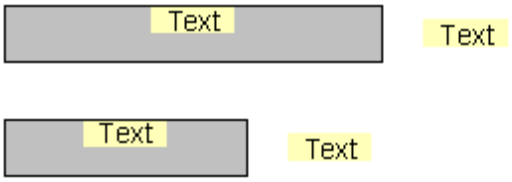
VerticalPictureFillMode

Property of NeAnnotation

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the annotation fills the designated field in vertical direction, if the annotation is used as an NeIPicture. Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the top position and equals the constant TOP; +100 describes the ultimate bottom position and equals the constant BOTTOM. If the annotation is used as an NeILabel, this property will not apply.

Possible Values	Description
AUTO	If by the property **Extent** a size of the heigth was specified, the background of the annotation (annotation rectangle) will be tiled in vertical direction. If no extend was specified, the background will be stretched in vertical direction.  Extent specified: The rectangle is tiled vertically (while the horizontal direction was set to "center"). Right of the layer there is the original size of the character string and its background.  No extent set: the rectangle is vertically stretched to the size of the layer.
BOTTOM	Within the available space of the annotation the background picture is placed in the bottom position (value: +100). At the same time, the horizontal fill mode was set to CENTER (value: 0). 
CENTER	Within the available space of the annotation the background picture is placed in the center position (value: 0). In the picture below, the horizontal direction is also CENTER. 
STRETCH	The available space of the annotation is filled in vertical direction by stretching the background

	picture. 
TILE	The available space of the annotation is tiled in vertical direction by the background picture. In the picture below, at the same time the horizontal fill mode was set to the ultimate left position (value: -100). 
TOP	Within the available space of the annotation the background picture is placed in the topmost position (value: -100). 

Accessing Methods
void setVerticalPictureFillMode (int newValue)
int getVerticalPictureFillMode ()

Also see HorizontalPictureFillMode

10.3 NeAreaStyle

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **java.awt.Color**

An NeAreaStyle is an extension of the class java.awt.Color that in addition provides a hatching pattern or a color gradient.

Properties to Handle the Elements of an Area

Pattern	Hatching pattern
PatternColor	Color of the hatching pattern

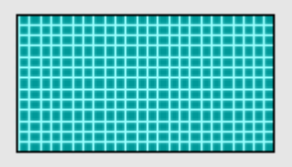
Constructors of the Class

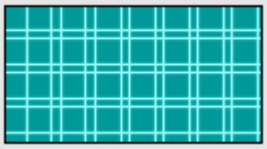
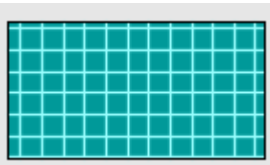
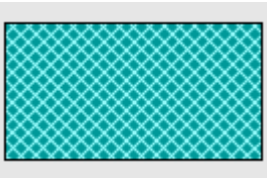
NeAreaStyle

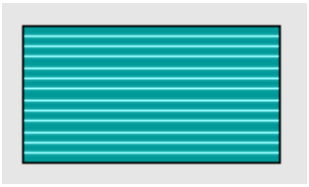
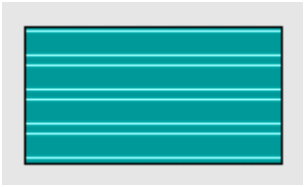
Constructor of NeAreaStyle

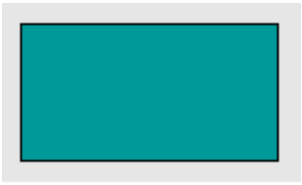
This constructor lets you generate an areaStyle object which describes the appearance of an area by its background color, its hatching pattern and its pattern color.

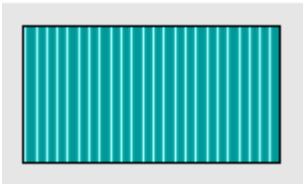
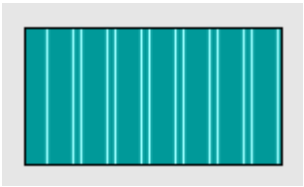
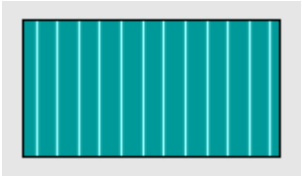
Declaration
NeAreaStyle (de.netronic.common.beanbase.NeAreaStyle style)

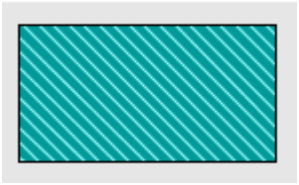
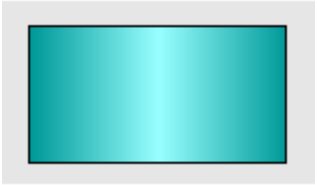
Parameter	Data Type	Description
style	de.netronic.common.beanbase.NeAreaStyle	AreaStyle object containing a background color, a hatching pattern, and a pattern color.
	Possible Values:	
	FILL_PATTERN_BACKSLASH_-DOUBLE	Hatching pattern that shows double backslash lines with spaces of intermediate width. 
	FILL_PATTERN_BACKSLASH_-WIDE	Hatching pattern that shows backslash lines with spaces of large width. 
	FILL_PATTERN_CROSS	Hatching pattern that shows horizontal and vertical cross lines with spaces of intermediate width. 

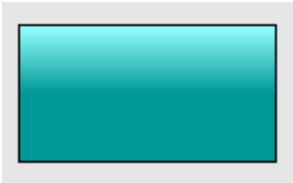
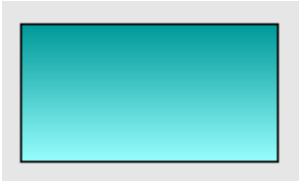
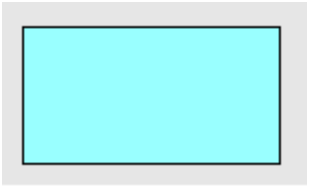
FILL_PATTERN_CROSS_DOUBLE	Hatching pattern that shows horizontal and vertical double cross lines with spaces of intermediate width. 
FILL_PATTERN_CROSS_WIDE	Hatching pattern that shows horizontal and vertical cross lines with spaces of large width. 
FILL_PATTERN_DIAGONAL	Hatching pattern that shows diagonal cross lines with spaces of intermediate width. 
FILL_PATTERN_- DIAGONAL_DOUBLE	Hatching pattern that shows diagonal double cross lines with spaces of intermediate width. 

FILL_PATTERN_DIAGONAL_WIDE	Hatching pattern that shows diagonal cross lines with spaces of large width. 
FILL_PATTERN_HORIZONTAL	Hatching pattern that shows horizontal lines with spaces of intermediate width. 
FILL_PATTERN_- HORIZONTAL_DOUBLE	Hatching pattern that shows horizontal double lines with spaces of intermediate width. 
FILL_PATTERN_- HORIZONTAL_WIDE	Hatching pattern that shows horizontal lines with spaces of large width. 

FILL_PATTERN_NONE	Pattern that has no a hatching pattern, which therefore lacks the foreground color. The backgrund color will completely cover the area. 
FILL_PATTERN_SLASH	Hatching pattern that shows slash lines with spaces of intermediate width. 
FILL_PATTERN_SLASH_DOBLE	Hatching pattern that shows double slash lines with spaces of intermediate width. 
FILL_PATTERN_SLASH_WIDE	Hatching pattern that shows slash lines with spaces of large width. 

FILL_PATTERN_TRANSPARENT	Empty pattern. It does not have a foreground color nor a backgrund color.
	
FILL_PATTERN_VERTICAL	Hatching pattern that shows vertical lines with spaces of intermediate width.
	
FILL_PATTERN_- VERTICAL_DOUBLE	Hatching pattern that shows vertical double lines with spaces of intermediate width.
	
FILL_PATTERN_VERTICAL_WIDE	Hatching pattern that shows vertical lines with spaces of large width.
	

FILL_PATTERN_BACKSLASH	Hatching pattern that shows backslash lines with spaces of intermediate width. 
FILL_PATTERN_GRADIENT_-DIAGONAL_DOWN	Vertical color gradientColor gradient directed diagonally from the top to the bottom 
FILL_PATTERN_GRADIENT_-DIAGONAL_UP	Color gradient directed diagonally from the bottom to the top 
FILL_PATTERN_GRADIENT_-HORIZONTAL	Horizontal color gradient 
FILL_PATTERN_GRADIENT_-HORIZONTAL_CONVEX	Horizontally convex color gradient 

FILL_PATTERN_GRADIENT_-LEFT_LIGHTED	Color gradient lighted on the left 
FILL_PATTERN_GRADIENT_-TOP_LIGHTED	Color gradient lighted at the top 
FILL_PATTERN_GRADIENT_-VERTICAL	Vertical color gradient 
FILL_PATTERN_GRADIENT_-VERTICAL_CONVEX	Vertically convex color gradient 
FILL_PATTERN_SOLID	The hatching pattern is a solid area. Therefore the foregrund color will completely cover the area. 

NeAreaStyle

Constructor of NeAreaStyle

This constructor lets you generate an areaStyle object which describes the appearance of an area by its background color.

Declaration
NeAreaStyle (java.awt.Color Color)

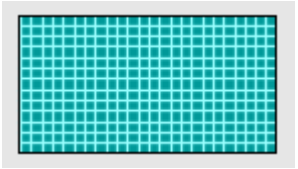
Parameter	Data Type	Description
Color	java.awt.Color	Background color of the area

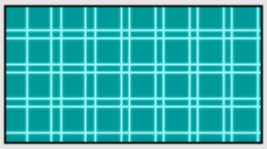
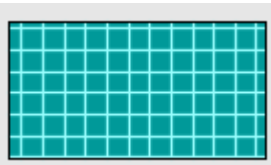
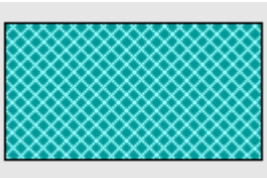
NeAreaStyle

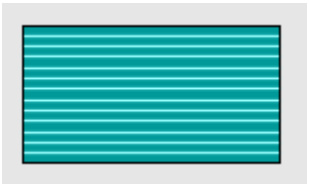
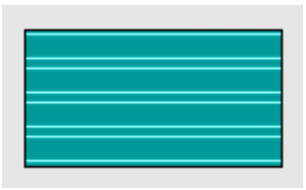
Constructor of NeAreaStyle

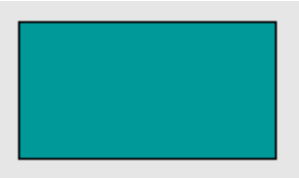
This constructor lets you generate an areaStyle object which describes the appearance of an area by its background color, its hatching pattern and its pattern color.

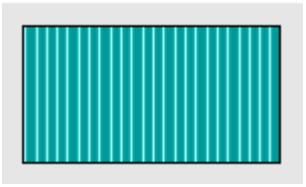
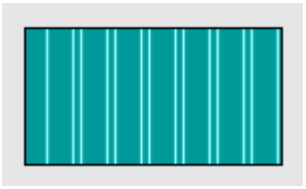
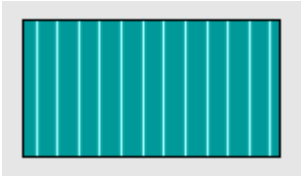
Declaration
NeAreaStyle (java.awt.Color backgroundColor, int pattern, java.awt.Color patternColor)

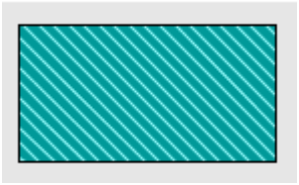
Parameter	Data Type	Description
backgroundColor	java.awt.Color	Background color of the area
pattern	int	Hatching patternd of the area
	Possible Values:	
	FILL_PATTERN_BACKSLASH_-DOUBLE	Hatching pattern that shows double backslash lines with spaces of intermediate width. 
	FILL_PATTERN_BACKSLASH_-WIDE	Hatching pattern that shows backslash lines with spaces of large width. 
	FILL_PATTERN_CROSS	Hatching pattern that shows horizontal and vertical cross lines with spaces of intermediate width. 

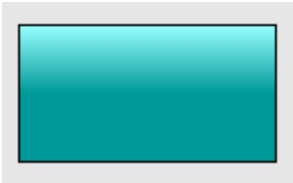
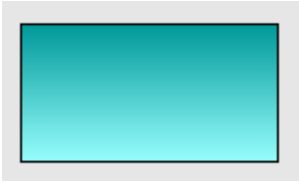
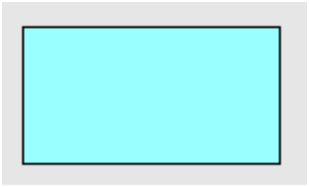
FILL_PATTERN_CROSS_DOUBLE	Hatching pattern that shows horizontal and vertical double cross lines with spaces of intermediate width. 
FILL_PATTERN_CROSS_WIDE	Hatching pattern that shows horizontal and vertical cross lines with spaces of large width. 
FILL_PATTERN_DIAGONAL	Hatching pattern that shows diagonal cross lines with spaces of intermediate width. 
FILL_PATTERN_- DIAGONAL_DOUBLE	Hatching pattern that shows diagonal double cross lines with spaces of intermediate width. 

FILL_PATTERN_DIAGONAL_WIDE	Hatching pattern that shows diagonal cross lines with spaces of large width. 
FILL_PATTERN_HORIZONTAL	Hatching pattern that shows horizontal lines with spaces of intermediate width. 
FILL_PATTERN_- HORIZONTAL_DOUBLE	Hatching pattern that shows horizontal double lines with spaces of intermediate width. 
FILL_PATTERN_- HORIZONTAL_WIDE	Hatching pattern that shows horizontal lines with spaces of large width. 

FILL_PATTERN_NONE	Pattern that has no a hatching pattern, which therefore lacks the foreground color. The backgrund color will completely cover the area. 
FILL_PATTERN_SLASH	Hatching pattern that shows slash lines with spaces of intermediate width. 
FILL_PATTERN_SLASH_DOUBLE	Hatching pattern that shows double slash lines with spaces of intermediate width. 
FILL_PATTERN_SLASH_WIDE	Hatching pattern that shows slash lines with spaces of large width. 

FILL_PATTERN_TRANSPARENT	Empty pattern. It does not have a foreground color nor a backgrund color. 
FILL_PATTERN_VERTICAL	Hatching pattern that shows vertical lines with spaces of intermediate width. 
FILL_PATTERN_- VERTICAL_DOUBLE	Hatching pattern that shows vertical double lines with spaces of intermediate width. 
FILL_PATTERN_VERTICAL_WIDE	Hatching pattern that shows vertical lines with spaces of large width. 

FILL_PATTERN_BACKSLASH	Hatching pattern that shows backslash lines with spaces of intermediate width. 
FILL_PATTERN_GRADIENT_-DIAGONAL_DOWN	Vertical color gradientColor gradient directed diagonally from the top to the bottom 
FILL_PATTERN_GRADIENT_-DIAGONAL_UP	Color gradient directed diagonally from the bottom to the top 
FILL_PATTERN_GRADIENT_-HORIZONTAL	Horizontal color gradient 
FILL_PATTERN_GRADIENT_-HORIZONTAL_CONVEX	Horizontally convex color gradient 

FILL_PATTERN_GRADIENT_-LEFT_LIGHTED	Color gradient lighted on the left 
FILL_PATTERN_GRADIENT_-TOP_LIGHTED	Color gradient lighted at the top 
FILL_PATTERN_GRADIENT_-VERTICAL	Vertical color gradient 
FILL_PATTERN_GRADIENT_-VERTICAL_CONVEX	Vertically convex color gradient 
FILL_PATTERN_SOLID	The hatching pattern is a solid area. Therefore the foregrund color will completely cover the area. 

patternColor	java.awt.Color	Color of the hatching pattern

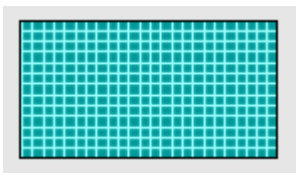
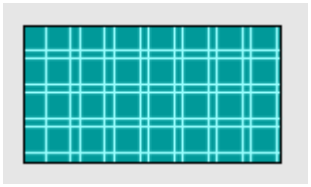
Properties of the Class

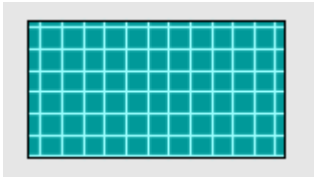
Pattern

Read Only Property of NeAreaStyle

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

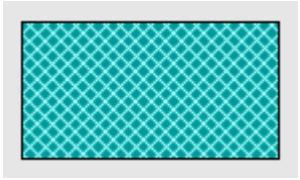
This property lets you set or retrieve a hatching pattern of the area that was set by the constructor.

Possible Values	Description
FILL_PATTERN_BACKSLASH_DOUBLE	Hatching pattern that shows double backslash lines with spaces of intermediate width. 
FILL_PATTERN_BACKSLASH_WIDE	Hatching pattern that shows backslash lines with spaces of large width. 
FILL_PATTERN_CROSS	Hatching pattern that shows horizontal and vertical cross lines with spaces of intermediate width. 
FILL_PATTERN_CROSS_DOUBLE	Hatching pattern that shows horizontal and vertical double cross lines with spaces of intermediate width. 
FILL_PATTERN_CROSS_WIDE	Hatching pattern that shows horizontal and vertical cross lines with spaces of large width.



FILL_PATTERN_DIAGONAL

Hatching pattern that shows diagonal cross lines with spaces of intermediate width.



FILL_PATTERN_DIAGONAL_DOUBLE

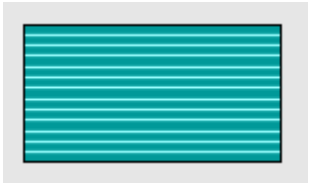
Hatching pattern that shows diagonal double cross lines with spaces of intermediate width.

FILL_PATTERN_DIAGONAL_WIDE

Hatching pattern that shows diagonal cross lines with spaces of large width.

FILL_PATTERN_HORIZONTAL

Hatching pattern that shows horizontal lines with spaces of intermediate width.



FILL_PATTERN_HORIZONTAL_DOUBLE

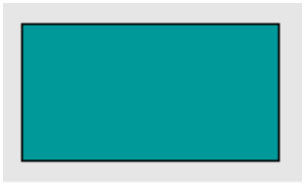
Hatching pattern that shows horizontal double lines with spaces of intermediate width.

FILL_PATTERN_HORIZONTAL_WIDE

Hatching pattern that shows horizontal lines with spaces of large width.

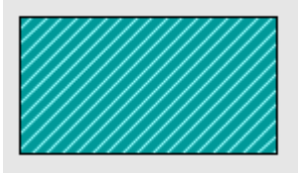
FILL_PATTERN_NONE

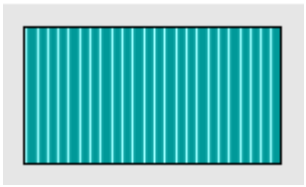
Pattern that has no a hatching pattern, which therefore lacks the foreground color. The backgrund color will completely cover the area.

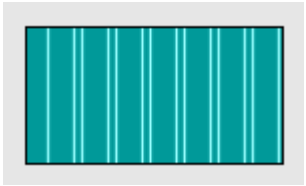


FILL_PATTERN_SLASH

Hatching pattern that shows slash lines with spaces of intermediate width.

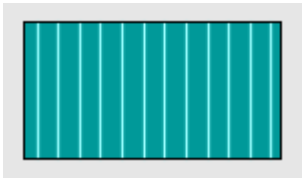


FILL_PATTERN_SLASH_DOUBLE	Hatching pattern that shows double slash lines with spaces of intermediate width.
	
FILL_PATTERN_SLASH_WIDE	Hatching pattern that shows slash lines with spaces of large width.
	
FILL_PATTERN_TRANSPARENT	Empty pattern. It does not have a foreground color nor a backgrund color.
	
FILL_PATTERN_VERTICAL	Hatching pattern that shows vertical lines with spaces of intermediate width.
	
FILL_PATTERN_VERTICAL_DOUBLE	Hatching pattern that shows vertical double lines with spaces of intermediate width.



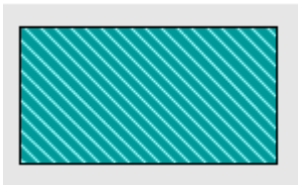
FILL_PATTERN_VERTICAL_WIDE

Hatching pattern that shows vertical lines with spaces of large width.



FILL_PATTERN_BACKSLASH

Hatching pattern that shows backslash lines with spaces of intermediate width.



FILL_PATTERN_GRADIENT_DIAGONAL_DOWN

Vertical color gradientColor gradient directed diagonally from the top to the bottom

FILL_PATTERN_GRADIENT_DIAGONAL_UP

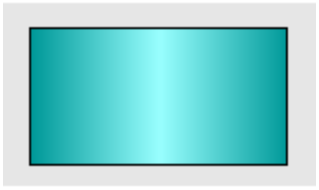
Color gradient directed diagonally from the bottom to the top

FILL_PATTERN_GRADIENT_HORIZONTAL

Horizontal color gradient

FILL_PATTERN_GRADIENT_HORIZONTAL_CONVEX

Horizontally convex color gradient



FILL_PATTERN_GRADIENT_LEFT_LIGHTED

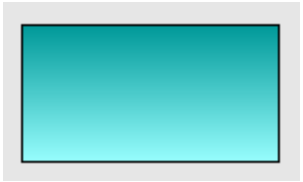
Color gradient lighted on the left

FILL_PATTERN_GRADIENT_TOP_LIGHTED

Color gradient lighted at the top

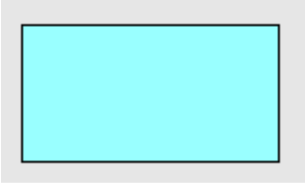
FILL_PATTERN_GRADIENT_VERTICAL

Vertical color gradient



FILL_PATTERN_GRADIENT_VERTICAL_CONVEX

Vertically convex color gradient

FILL_PATTERN_SOLID		The hatching pattern is a solid area. Therefore the foregrund color will completely cover the area.
		

Accessing Methods
int getPattern()

PatternColor

Read Only Property of NeAreaStyle

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you retrieve the color of the hatching pattern set by the constructor.

Accessing Methods
java.awt.Color getPatternColor()

10.4 NeColorMap

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**

This class offers methods to map colors to key values for the handling of dynamic colors. Two different types of key values can be used in a color map: single,

discrete values and range values. If discrete values and range values overlap, discrete values are given priority over range values.

Methods to Handle the Color Map

addEntry(...)	Adds a discrete value to the color map.
addPropertyChangeListener(...)	Adds a listener for change events in the map object.
addRange(...)	Adds a range value to the color map.
getColor(...)	Color of a given key in the map.
iterateColors()	Returns an iterator object of all colors.
removeEntry(...)	Removes a discrete value from the map.
removePropertyChangeListener(...)	Removes a listener for change events in the map object.
removeRange(...)	Removes a range value from the map.

Constructors of the Class

NeColorMap

Constructor of NeColorMap

This constructor lets you generate an empty color map. Discrete entries and range values you can add later on via the appropriate methods.

Declaration
NeColorMap ()

Methods of the Class

addEntry

Method of NeColorMap

This method lets you add to the color map a discrete value as a key.

Declaration
void addEntry (java.lang.Object key, java.awt.Color theColor)

	Data Type	Description
Parameter		
key	java.lang.Object	Object that forms the key and that is to be added to the color map. The object for example is the value of an entity attribute. It represents a discrete value.
theColor	java.awt.Color	Color object to be assigned.
Return Value	void	

Also see removeEntry

addPropertyChangeListener

Method of NeColorMap

This method adds a listener for change events in the JGantt object. The listener will be informed each time a property of the color scheme has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

addRange

Method of NeColorMap

This method lets you add a range value to the color map. The range extends from this key up to, but not including, the next range key of the same class. A range key may for example be a date. The range key needs to be comparable to other range keys, giving a result of being larger, smaller or equal. So all range keys of the same color map need to be of the same type, for example a date. This is how you can assign the same color to all nodes that hold a date which is equal or larger than the key considered but which is smaller than the key following. If there

is no key following, any value equal or larger than the key considered will belong to the range.

Declaration
void addRange (java.lang.Comparable key, java.awt.Color theColor)

	Data Type	Description
Parameter		
key	java.lang.Comparable	Object that forms the key and that is to be added to the color map. The object may for example be the value of an entity attribute representing a range of values.
theColor	java.awt.Color	Color object to be assigned.
Return Value	void	

Also see removeRange

getColor

Method of NeColorMap

By this method you can retrieve the color of a given key.

Declaration
java.awt.Color getColor (java.lang.Object key)

	Data Type	Description
Parameter		
key	java.lang.Object	Object that forms the key and that is to be searched for in the color map. The object may for example be the value of an entity attribute representing a range of values. If the object was found, the correponding color will be returned.
Return Value	java.awt.Color	Color corresponding to the key in the color map.

iterateColors

Method of NeColorMap

This method returns an iterator object of all colors.

Declaration
java.util.Iterator iterateColors ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned

removeEntry

Method of NeColorMap

This method lets you remove a discrete value from the map as well as the color object allocated.

Declaration
void removeEntry (java.lang.Object key)

	Data Type	Description
Parameter		
key	java.lang.Object	Key to be removed.
Return Value	void	

Also see addEntry

removePropertyChangeListener

Method of NeColorMap

This method lets you remove a listener for change events in the map object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

removeRange

Method of NeColorMap

This method lets you remove a range value from the map as well as the color object allocated.

Declaration
void removeRange (java.lang.Comparable key)

	Data Type	Description
Parameter		
key	java.lang.Comparable	Key to be removed.
Return Value	void	

Also see addRange

10.5 NeCombinedFilter

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIFilter**

This class provides a constructor that serves to generate a filter which contains a condition composed of conditions of other filters. For simple conditions please see class **NeValueFilter**, for inverted conditions please see class **NeNotFilter**.

Constructors of the Class

NeCombinedFilter

Constructor of NeCombinedFilter

This constructor lets you generate a filter that contains a condition composed of the conditions of other filters.

Declaration
NeCombinedFilter (de.netronic.common.intface.NeIFilter theFirst, java.lang.String operator, de.netronic.common.intface.NeIFilter theSecond)

Parameter	Data Type	Description
theFirst	de.netronic.common.intface.NeIFilter	Filter, the conditions of which will be combined with the conditions of the filter in the third parameter.
operator	java.lang.String	Operator, that may consist of a logic "and" (in Java: &&) or a logic "or" (in Java: ||).
theSecond	de.netronic.common.intface.NeIFilter	Filter, the conditions of which will be combined with the conditions of the filter in the first parameter.

10.6 NeDateLine

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **de.netronic.common.beanbase.NeValueLine**

This class allows to add a data line to the Gantt graph via different constructors.

Constructor to Add a Date Line

Constructors of the Class

These constructors serve to generate a date line object. In addition to the date, on which the line is to be drawn, you can pass a line type and a line color. If you do not pass a line type or a color, the corresponding default values will be automatically set.

NeDateLine

Constructor of NeDateLine

Date line set to the date passed. The default line color is red, the default line type is a solid line.

Declaration
NeDateLine (long thisValue)

Parameter	Data Type	Description
thisValue	long	Date on which the date line will be displayed. The figures passed will be interpreted as the number of milliseconds since 1.1.1970.

NeDateLine

Constructor of NeDateLine

Date line set to the date and to the line type passed by the parameters. The default line color is red.

Declaration
NeDateLine (long thisValue, int thisLineType)

Parameter	Data Type	Description
thisValue	long	Date on which the date line will be displayed. The figures passed will be interpreted as the number of milliseconds since 1.1.1970.
thisLineType	int	Line type of the date line
	Possible Values:	
	NE_LINE_TYPE_0	Solid line
	NE_LINE_TYPE_1	Line of long dashes separated by average spaces
	NE_LINE_TYPE_10	Line of one very long dash alternating with three dots, separated by very short spaces
	NE_LINE_TYPE_11	Line of one very long dash alternating with two very short dashes, separated by very short spaces
	NE_LINE_TYPE_12	Line of one very long dash alternating with two dots, separated by very short spaces
	NE_LINE_TYPE_13	Line of one very long dash alternating with one very short dash, separated by very short spaces
	NE_LINE_TYPE_14	Line of one very long dash alternating with one dot, separated by very short spaces

NE_LINE_TYPE_15	Line of one extremely long dash alternating with one dot, separated by very short spaces
NE_LINE_TYPE_16	Line of short dashes separated by very narrow spaces
NE_LINE_TYPE_17	Line of rather long dashes separated by very short spaces
NE_LINE_TYPE_18	Line of one very long dash alternating with four dots, separated by very short spaces
NE_LINE_TYPE_2	Dotted line
NE_LINE_TYPE_3	Line of short dashes separated by very narrow spaces
NE_LINE_TYPE_4	Line of short dashes separated by narrow spaces
NE_LINE_TYPE_5	Line of long dashes separated by average spaces
NE_LINE_TYPE_6	Line of long dashes sparated by very wide spaces

NE_LINE_TYPE_7	Line of short dashes separated by wide spaces - - - - - - - - - - - - - - - - - -
NE_LINE_TYPE_8	Line of rather long dashes separated by average spaces – – – – – – – – – – – – – –
NE_LINE_TYPE_9	Line of one very long dash alternating with three very short dashes, separated by very short spaces —---—---—---—---—---—---—

NeDateLine

Constructor of NeDateLine

Date line that is assigned the date, the color and the line type passed by the parameters.

Declaration
NeDateLine (long thisValue, java.awt.Color thisLineColor, int thisLineType)

Parameter	Data Type	Description
thisValue	long	Date on which the date line will be displayed. The figures passed will be interpreted as the number of milliseconds since 1.1.1970.
thisLineColor	java.awt.Color	Color of the date line
thisLineType	int	Line type of the date line
	Possible Values:	
	NE_LINE_TYPE_0	Solid line ————————————
	NE_LINE_TYPE_1	Line of long dashes separated by average spaces – – – – – – – – – – – –
	NE_LINE_TYPE_10	Line of one very long dash alternating with three dots, separated by very short spaces —···—···—···—···—···—···—
	NE_LINE_TYPE_11	Line of one very long dash alternating with two very short dashes, separated by very short spaces —··—··—··—··—··—··—··—
	NE_LINE_TYPE_12	Line of one very long dash alternating with two dots, separated by very short spaces —··—··—··—··—··—··—··—
	NE_LINE_TYPE_13	Line of one very long dash alternating with one very short dash, separated by very short spaces —·—·—·—·—·—·—·—·—
	NE_LINE_TYPE_14	Line of one very long dash alternating with one dot, separated by very short spaces ————————————

NE_LINE_TYPE_15	Line of one extremely long dash alternating with one dot, separated by very short spaces
NE_LINE_TYPE_16	Line of short dashes separated by very narrow spaces
NE_LINE_TYPE_17	Line of rather long dashes separated by very short spaces
NE_LINE_TYPE_18	Line of one very long dash alternating with four dots, separated by very short spaces
NE_LINE_TYPE_2	Dotted line
NE_LINE_TYPE_3	Line of short dashes separated by very narrow spaces
NE_LINE_TYPE_4	Line of short dashes separated by narrow spaces
NE_LINE_TYPE_5	Line of long dashes separated by average spaces
NE_LINE_TYPE_6	Line of long dashes sparated by very wide spaces

	NE_LINE_TYPE_7	Line of short dashes separated by wide spaces - - - - - - - - - - - - - - - - - -
	NE_LINE_TYPE_8	Line of rather long dashes separated by average spaces – – – – – – – – – – – – – –
	NE_LINE_TYPE_9	Line of one very long dash alternating with three very short dashes, separated by very short spaces —---—---—---—---—---—---—

NeDateLine

Constructor of NeDateLine

Date line set to the date and to the color passed by the parameters. The default line type is a solid line.

Declaration
NeDateLine (long thisValue, java.awt.Color thisLineColor)

Parameter	Data Type	Description
thisValue	long	Date on which the date line will be displayed. The figures passed will be interpreted as the number of milliseconds since 1.1.1970.
thisLineColor	java.awt.Color	Color of the date line

NeDateLine

Constructor of NeDateLine

Date line set to the date passed. The graphical attributes of the line are taken from the lNeLineStyle object.

Declaration
NeDateLine (NeLineStyle thisLineStyle, long thisValue)

Parameter	Data Type	Description
thisLineStyle	NeLineStyle	Line style for the date line.
thisValue	long	Date on which the date line will be displayed. The figures passed will be interpreted as the number of milliseconds since 1.1.1970.

10.7 NeDynamicLabel

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIDynamicLabel**

This abstract class defines a dynamic NeILabel object, which can be of the type **NeAnnotation** or **NeSymbol**. Please implement the method **getActualLabel** in a derived class. This way, you can - depending on the parameter **refObject** - specify the NeILabel object returned.

Methods to handle the dynamic label

getActualLabel(...)	Abstract method to be implemented that returns the actual NeILabel object.

Methods of the Class

getActualLabel

Method of NeDynamicLabel

Abstract method to be implemented that returns the actual NeILabel object.

Declaration
NeILabel getActualLabel (java.lang.Object refObject)

	Data Type	Description
Parameter		
refObject	java.lang.Object	Reference object (usually an NeIEntity object)
Return Value	NeILabel	Actual NeILabel

Example Code

```
public NeILabel getActualLabel(Object refObject)
  {
  NeIEntity entity = (NeIEntity) refObject;
  String symbolString = entity.getValueAsString("symbol");

  NeSymbol symbol;

  if (symbolString.equals("Triangle"))
    symbol = new NeSymbol (74, NeSymbol.SYMBOL_TABLE_GR_BAR);
  else
    symbol = new NeSymbol (75, NeSymbol.SYMBOL_TABLE_GR_BAR);

  symbol.setExtent(new Dimension(600,600));
  symbol.setAreaStyle(Color.blue);

  return symbol;
  }
```

10.8 NeDynamicPicture

Belongs to VariousClasses

Package name	**de.netronic.common.beanbase**
Extends	**de.netronic.common.beanbase.NeDynamicLabel**
Implements	**de.netronic.common.intface.NeIDynamicPicture**

This abstract class defines a dynamic NePicture object. Please implement the method **getActualPicture** in a derived class. This way, you can - depending on the parameter **refObject** - define the NePicture object returned.

Methods of the Class

getActualPicture

Method of NeDynamicPicture

Abstract method to be implemented that returns the actual NePicture object.

Declaration
NePicture getActualPicture (java.lang.Object refObject)

	Data Type	Description
Parameter		
refObject	java.lang.Object	Reference object (usually an NeIEntity object)
Return Value	NePicture	Actual NePicture object

Example Code

```
public NePicture getActualPicture(Object refObject)
  {
  NeIEntity entity = (NeIEntity) refObject;
  String hasAction = enttity.getValueAsString("hasAction");

  if (hasAction!=null && hasAction.equals("1"))
    return myHasActionPicture;
  else
    return null;
  }
```

10.9 NeEntityAttributeColor

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **java.awt.Color**

This class allows to use values of **color** type AppData attributes as colors in VARCHART JGantt.

Methods to handle the color of an entity attribute

getActualColor(...)	Returns the color from the color value attribute of the **refObject** entity.

Constructors of the Class

NeEntityAttributeColor

Constructor of NeEntityAttributeColor

Generates a color of the type **NeEntityAttributeColor** by using the default color and the color value attribute name.

Declaration
NeEntityAttributeColor (java.awt.Color theColor, java.lang.String theAttributeName)

Parameter	Data Type	Description
theColor	java.awt.Color	Default color
theAttributeName	java.lang.String	Name of the attribute from which the color is to be retrieved

Methods of the Class

getActualColor

Method of NeEntityAttributeColor

Returns the color from the color value attribute of the **refObject** entity. In case it equals zero or is no valid color, the default color will be returned.

Declaration
java.awt.Color getActualColor (java.lang.Object refObject)

	Data Type	Description
Parameter		
refObject	java.lang.Object	Entity from which the contents of the color attribute are to be retrieved.
Return Value	java.awt.Color	Returned color

10.10 NeEntityComparator

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **java.lang.Object**

Implements	**java.io.Serializable**
I	**java.util.Comparator**

This class lets you to compare entities.

Properties to Handle the Comparison of Entities

AttributeNames	List of names of the compared attributes of an entity
Reversed	Inverted flags of the comparison

Methods to Handle the Comparison of Entities

compare(...)	Compares the values of attributes of two entities.

Constructors of the Class

By the constructors you can generate comparators that compare entities on the base of their attribute values. A second parameter allows to invert the result.

NeEntityComparator

Constructor of NeEntityComparator

This constructor lets you generate a comparator object to compare entities. You can create a paramater to set whether or not the result of the comparison is to be reversed. The missing attribute names to hold the entities you can create later on via the property set/getAttributeNames.

Declaration
NeEntityComparator (boolean compareReversed)

Parameter	Data Type	Description
compareReversed	boolean	Boolean value that sets, whether or not the result is to be inverted.

NeEntityComparator

Constructor of NeEntityComparator

This constructor lets you generate a comparator object to compare entities. The parameters missing now can be generated later on by the properties set/getAttributeNames and set/getReversed.

Declaration
NeEntityComparator ()

NeEntityComparator

Constructor of NeEntityComparator

This constructor lets you generate a comparator, that compares entities via an array of attribute values. The list of corresponding attribute names is passed by the first parameter. Beside, for each attribute you can set whether or not the result of the comparison is to be reversed.

Declaration
NeEntityComparator (java.lang.String theAttributeNames, boolean compareReversed)

Parameter	Data Type	Description
theAttributeNames	java.lang.String	String holding the name of the attribute, the value of which is to be compared.
compareReversed	boolean	Boolean value that sets, whether or not the result is to be inverted.

NeEntityComparator

Constructor of NeEntityComparator

This constructor lets you generate a comparator, that compares entities via an array of attribute values. The list of corresponding attribute names is passed by the first parameter. Beside, for each attribute you can set whether or not the result of the comparison is to be reversed.

Declaration
NeEntityComparator (java.lang.String[] theAttributeNames, boolean compareReversed)

Parameter	Data Type	Description
theAttributeNames	java.lang.String[]	Array of strings holding the names of the attributes, the values of which are to be compared.
compareReversed	boolean	Boolean value, that sets for all attributes, whether or not the result is to be inverted.

NeEntityComparator

Constructor of NeEntityComparator

This constructor lets you generate a comparator, that compares entities via an array of attribute values. The list of corresponding attribute names is passed by the first parameter. Beside, for each attribute you can set whether or not the result of the comparison is to be reversed.

Declaration
NeEntityComparator (java.lang.String[] theAttributeNames, boolean [] compareReversed)

Parameter	Data Type	Description
theAttributeNames	java.lang.String[]	Array of strings holding the names of the attributes, the values of which are to be compared.
compareReversed	boolean []	Array of boolean values that sets for each attribute, whether of not its result is to be reversed.

Properties of the Class

AttributeNames

Read Only Property of NeEntityComparator

Typ	**java.lang.String[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve an array of strings that hold the names of attributes of the entity that have been compared.

Accessing Methods
java.lang.String[] getAttributeNames()

Reversed

Read Only Property of NeEntityComparator

Typ	**boolean []**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve an array of boolean values that act as flags. The flags indicate whether the results of the comparison of the attributes are inverted. They can be used to sort entities after a value of their attributes in descending order.

Accessing Methods
boolean [] getReversed()

Methods of the Class

compare

Method of NeEntityComparator

This method lets you compare the values of selected of attributes of two entities.

Declaration
int compare (java.lang.Object entity1, java.lang.Object entity2)

	Data Type	Description
Parameter		
entity1	java.lang.Object	First entitiy object, to be compared to the second one.
entity2	java.lang.Object	Second entitiy object, to be compared to the first one.
Return Value	int	Returns a negative integer, zero, or a positive integer if the first argument is less than, equal to, or greater than the second one, respectively. The sign of the result is inverted if **compareReversed** is true. Returns zero if one of the objects is not an entity.

10.11 NeEntityEditorDialog

Belongs to VariousClasses

Package name **de.netronic.bean.entityeditor**
Extends **javax.swing.JDialog**

An NeEntityEditorDialog serves for simple editing or for displaying attribute values of one or several entities. The selection of attributes to be edited can be customized. An attribute can be defined to be editable. This setting can be valid for all entities or they can be adapted via NeIFilter objects to the entity just displayed.
There are two different ways to open an NeEntityEditorDialog. Either directly from the application via the method editEntity or editEntities, or from JGantt by selecting the option "Edit" from the pop-up menu or by creating an entity.

Constructors of the Class

NeEntityEditorDialog

Constructor of NeEntityEditorDialog

Generates an NeEntityEditorDialog that has the parent "frame" and the title "title"

Declaration
NeEntityEditorDialog (java.awt.Frame frame, java.lang.String title)

Parameter	Data Type	Description
frame	java.awt.Frame	Owner of the dialog
title	java.lang.String	The title of the dialog. If set to **null** the title will be "Edit data".

Properties of the Class

Buffered

Property of NeEntityEditorDialog

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

If the editor works in the "buffered" mode all values modified are kept internally in the editor. They are written to the entity only after the user has pressed the "OK" button. Before, the modifications entered can be undone by pressing the "Cancel" button.
If this property is set to **false**, the values modified are written to the entity directly after the field has been edited. In this case, the dialog will not show a "Cancel" button.

Accessing Methods
void setBuffered (boolean newValue)
boolean isBuffered ()

ContentsDefinition

Property of NeEntityEditorDialog

Typ	**de.netronic.common.contentsdefinition.NeContentsDefinition**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**null**

Defines the attributes to be displayed by the editor. This definition can be adapted to an entity by various objects of the class de.netronic.common.contentsdefinition.NeRowDefinition. The field styles specified in the contents definition or its row definition will not be considered. If you do not set this property, all attributes of the entity set will be displayed and can be edited.

Accessing Methods
void setContentsDefinition (de.netronic.common.contentsdefinition.NeContentsDefinition newValue)
de.netronic.common.contentsdefinition.NeContentsDefinition getContentsDefinition ()

Example Code

```
// use contentsDefinition from table for entytiEditor as well
NeContentsDefinition cd = jGantt1.getTable().getContentsDefinition();
jGantt1.getEntityEditorDialog().setContentsDefinition(cd);
```

IdentifyBy

Property of NeEntityEditorDialog

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**"<userID>"**

Name of the attribute used to identify the entity. The contents of this attribute will be displayed in the top left corner of the EntityEditor. Via the pre-defined value "<userID>" the user identification of the entity can be used for identification.

Accessing Methods
void setIdentifyBy (java.lang.String newValue)
java.lang.String getIdentifyBy ()

Methods of the Class

editEntities

Method of NeEntityEditorDialog

Opens the editor as a modal dialog. The entities provided by the iterator can be edited consecutively.

Declaration
boolean editEntities (java.util.Iterator entities)

	Data Type	Description
Parameter		
entities	java.util.Iterator	Iterator to provide the entities to be edited.
Return Value	boolean	Returns **true**, after the user has clicked "OK" in the dialog. Returns **false**, after the user has clicked "Cancel" in the dialog.

Also see editEntity

editEntity

Method of NeEntityEditorDialog

Opens the editor as a modal dialog for the entity "entity".

Declaration
boolean editEntity (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity to be edited.
Return Value	boolean	Returns **true**, after the user has clicked "OK" in the dialog. Returns **false**, after the user has clicked "Cancel" in the dialog

Also see editEntities

10.12 NeGroupComparator

Belongs to VariousClasses

Package name **de.netronic.bean.layouter**
Implements **de.netronic.common.intface.NeIGroupComparator**

The class NeGroupComparator compares groups after their group entity.
If group entities do not exist, groups are compared after their grouping codes.

Properties to Handle Groups

Dynamic	Dynamic sorting of groups

Methods to Handle Groups

dependsOn()	Does sorting depend on the modification of data?

Constructors of the Class

NeGroupComparator

Constructor of NeGroupComparator

Generates an NeGroupComparator that sorts groups after the value of several attributes of their group entity in ascending or descending order. Sorting can be chosen to be performed dynamically or statically.

Declaration
NeGroupComparator (java.lang.String[] theAttributeNames, boolean [] compareReversed, boolean dynamic)

Parameter	Data Type	Description
theAttributeNames	java.lang.String[]	List of attribute names, after which the sorting is performed.
compareReversed	boolean []	If this parameter equals the value "true", groups are sorted in descending order.
dynamic	boolean	If this parameter equals the value "true", groups are sorted dynamically (sorting is automatically repearted after a modification of data).

NeGroupComparator

Constructor of NeGroupComparator

Generates an NeGroupComparator that sorts groups after the value of several attributes of their group entity in ascending or descending order. Sorting is automatically repearted after a modification of data (dynamic sorting).

Declaration
NeGroupComparator (java.lang.String[] theAttributeNames, boolean [] compareReversed)

Parameter	Data Type	Description
theAttributeNames	java.lang.String[]	List of attribute names, after which the sorting is performed.
compareReversed	boolean []	If this parameter equals the value "true", groups are sorted in descending order.

NeGroupComparator

Constructor of NeGroupComparator

Generates an NeGroupComparator that sorts groups after the value of several attributes of their group entity in ascending or descending order. Sorting is automatically repeated after a modification of data (dynamic sorting).

Declaration
NeGroupComparator (java.lang.String[] theAttributeNames, boolean compareReversed)

Parameter	Data Type	Description
theAttributeNames	java.lang.String[]	List of attribute names, after which the sorting is performed.
compareReversed	boolean	If this parameter equals the value "true", groups are sorted in descending order.

NeGroupComparator

Constructor of NeGroupComparator

Generates an NeGroupComparator that sorts groups after the value of an attribute of their group entity in ascending or descending order. Sorting can be chosen to be performed dynamically or statically.

Declaration
NeGroupComparator (java.lang.String theAttributeName, boolean compareReversed, boolean dynamic)

Parameter	Data Type	Description
theAttributeName	java.lang.String	Name of the attribute, after which the sorting is performed.
compareReversed	boolean	If this parameter equals the value "true", groups are sorted in descending order.
dynamic	boolean	If this parameter equals the value "true", groups are sorted dynamically (sorting is automatically repearted after a modification of data).

NeGroupComparator

Constructor of NeGroupComparator

Generates an NeGroupComparator that sorts groups after the value of an attribute of their group entity in ascending or descending order. Sorting is automatically repeated after a modification of data (dynamic sorting).

Declaration
NeGroupComparator (java.lang.String theAttributeName, boolean compareReversed)

Parameter	Data Type	Description
theAttributeName	java.lang.String	Name of the attribute, after which the sorting is performed.
compareReversed	boolean	If this parameter equals the value "true", groups are sorted in descending order.

NeGroupComparator

Constructor of NeGroupComparator

Generates an NeGroupComparator that sorts groups after the ID of their group entity in ascending or descending order. Sorting is automatically repeated after a modification of data (dynamic sorting).

Declaration
NeGroupComparator (boolean compareReversed)

Parameter	Data Type	Description
compareReversed	boolean	If this parameter equals the value "true", groups will be sorted in descending order.

NeGroupComparator

Constructor of NeGroupComparator

Generates an NeGroupComparator that sorts groups after the ID of their group entity. Sorting is performed dynamically.

Declaration
NeGroupComparator ()

Properties of the Class

Dynamic

Property of NeGroupComparator

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

If set to true, groups are re-sorted automatically on changes of data.

Accessing Methods
void setDynamic (boolean newValue)
boolean isDynamic ()

Methods of the Class

dependsOn

Method of NeGroupComparator

If dynamic sorting is not activated (depending on the type of constructor used) the method will return **false**. If dynamic sorting is activated, the method will return **true** if the entity "entity" is the groupEntity of the group "group" and if "attribute" is one of the attributNames which are used for sorting the groups.

Declaration
boolean dependsOn ()

	Data Type	Description
Return Value	boolean	

10.13 NeIDrawingConstants

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface offers constants that define different ways of drawing elements.

10.14 NeIDrawingElement

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface holds properties and methods to handle DrawingElement objects, mainly for internal administrative purposes.

Properties to Handle Drawing Elements

Opaque Opaque drawing element

RefObject	Retrieves the reference object.

Methods to Handle Drawing Elements

getBoundingBox(...)	Minimum size rectangle surrounding the label.
identify(...)	Locates a point
identifyObjects(...)	Locates drawing elements in a rectangle
paintElements(...)	Takes the drawing element to the Gantt graph.

Properties of the Interface

Opaque

Property of NeIDrawingElement

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

A drawing element is opaque if it fills completely the bounding box by which it is limited. Since beneath opaque objects drawing is not needed, rerieving this property may be used for optimized drawing.

Accessing Methods
void setOpaque (boolean newValue)
boolean isOpaque ()

RefObject

Read Only Property of NeIDrawingElement

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This Property retrieves the reference object of the drawing element.

Accessing Methods
java.lang.Object getRefObject()

Methods of the Interface

getBoundingBox

Method of NeIDrawingElement

This method calculates the minimum rectangle that can be drawn around a drawing element.

Declaration
java.awt.Rectangle getBoundingBox (java.awt.Rectangle oldRect)

	Data Type	Description
Parameter		
oldRect	java.awt.Rectangle	This parameter serves to pass an existing rectangle object, that receives the values of the bounding box. For reasons of performance, it makes sense to re-use existing rectangles. If you leave this parameter empty, a new rectangle object will automatically be generated and returned.
Return Value	java.awt.Rectangle	Minimum rectangle retrieved.

identify

Method of NeIDrawingElement

This method retrieves, whether or not the point passed is situated within the drawing element.

Declaration
boolean identify (java.awt.Point thePoint)

	Data Type	Description
Parameter		
thePoint	java.awt.Point	Position in coordinate system of the drawing element.
Return Value	boolean	**True**: the point passed has been located within the drawing element, **false**: the point passed could not be located within the drawing element.

identifyObjects

Method of NeIDrawingElement

This method retrieves, whether in the rectangle passed drawing elements exist.

Declaration
boolean identifyObjects (java.awt.Rectangle rectangle, java.util.Collection foundObjects, boolean opaque)

	Data Type	Description
Parameter		
rectangle	java.awt.Rectangle	Rectangle, in which drawing elements are located.
foundObjects	java.util.Collection	Collection of drawing elements found in the rectangle passed.
opaque	boolean	This parameter defines, whether uncolored areas in a dynamic object are opaque or transparent. In transparent objects an uncolored area will not be identified as a part of an object. If only the transparen part of an object is located in the rectangle, it cannot be identified.
Return Value	boolean	**True**: drawing elements have been located in the rectangle passed, **false**: drawing elements could not be located in the rectangle passed.

paintElements

Method of NeIDrawingElement

This method takes the drawing element to the Gantt graph.

Declaration
int paintElements (NeIPainter painter)

	Data Type	Description
Parameter		
painter	NeIPainter	
Return Value	int	Painter object to be drawn.

10.15 NeIDynamicColor

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface defines a color that may change at runtime, for example in dependence of the value of an attribute.

Methods to Handle Dynamic Colors

getActualColor(...)	Retrieves the actual color of the reference object.

Methods of the Interface

getActualColor

Method of NeIDynamicColor

Retrieves the color which has been assigned to the reference object, and which depends on the data of the reference object.

Declaration
java.awt.Color getActualColor (java.lang.Object object)

	Data Type	Description
Parameter		
object	java.lang.Object	Object, the color of which is to be retrieved. The type of reference object can either be of NeIEntity (to assign colors to nodes or group nodes via the corresponding color schemes) or of NeITimeSpan (to assign colors to a calender grid or the time scale).
Return Value	java.awt.Color	Color object returned

10.16 NeIDynamicLabel

Belongs to VariousClasses

Package name **de.netronic.common.intface**
Extends **de.netronic.common.intface.NeILabel**

This interface offers methods to handle dynamic labels.

Methods to Handle Dynamic Labels

getActualLabel(...) Retrieves the actual label of the reference object.

Methods of the Interface

getActualLabel

Method of NeIDynamicLabel

Retrieves the label which has been assigned to the reference object, and which depends on the data of the reference object.

Declaration
NeILabel getActualLabel (java.lang.Object object)

	Data Type	Description
Parameter		
object	java.lang.Object	Object, the label of which is to be retrieved.
Return Value	NeILabel	Label object returned

10.17 NeIDynamicPicture

Belongs to VariousClasses

Package name **de.netronic.common.intface**
Extends **de.netronic.common.intface.NeIPicture**

This interface defines a picture object, the actual contents of which is determined dynamically at run time.

Methods to Handle Dynamic Pictures

getActualPicture(...)	Retrieves the actual picture of the reference object.

Methods of the Interface

getActualPicture

Method of NeIDynamicPicture

Retrieves the picture which is allocated to the reference object (mre exactly: to an attribute of the entity), and which depends on the data of the reference object.

Declaration
NeIPicture getActualPicture (java.lang.Object object)

	Data Type	Description
Parameter		
object	java.lang.Object	Object, the picture of which is to be retrieved. Most times, the object is an entity.
Return Value	NeIPicture	Picture object returned

10.18 NeIFilter

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface defines a filter that applies to entites. Via the method **apply** you can verify whether or not an entity fulfills the condition in the filter.

Methods to Handle Filter Objects

addPropertyChangeListener(...)	Adds a listener for change events in the filter object.
apply(...)	Verifies if the conditions of the filter apply to an entity
removePropertyChangeListener(...)	Removes a listener for change events.

Methods of the Interface

addPropertyChangeListener

Method of NeIFilter

This method adds a listener for change events in the filter object. The listener will be informed each time a property was changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

apply

Method of NeIFilter

When JGantt invokes this method, the filter object verifies whether the entity passed complies with the filter conditions.

Declaration
boolean apply (NeIEntity entity)

	Data Type	Description
Parameter		
entity	NeIEntity	Entity that the filter is to apply to.
Return Value	boolean	**true**: the conditions apply to the entity, **false**: the conditions do not apply to the entity.

removePropertyChangeListener

Method of NeIFilter

Removes an existing listener for change events in the filter object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

10.19 NeIGroupComparator

Belongs to VariousClasses

Package name **de.netronic.common.intface**
Extends **java.util.Comparator**

When for sorting groups (see JGantt.GroupsSortedBy) a comparator is used that implements the interface NeIGroupComparator, JGantt will be enabled to adapt the sorting of groups dynamically to the modifications of the data that the nodes are based upon.

Methods of the Interface

dependsOn

Method of NeIGroupComparator

The method will return "true", if the sorting of the group called "group" depends of the value of the attribute "attribute" in the entity "entity".
This method is invoked by the layouter object after the data of an activity in a group have changed. The layouter of the group affected can then decide, whether - due to the modification - sorting of the sub-groups has become necessary.
If the comaparator always returns "false", sorting is static and does not automatically react to data modification. If it always returns true, the sequence is checked on each modification and may probably be sorted anew. For reasons of performance, the latter case cannot be recommended.

Declaration
boolean dependsOn (NeILayouterGroup group, NeIEntity entity, java.lang.String attributeName)

	Data Type	Description
Parameter		
group	NeILayouterGroup	Group, of which the sub-groups are due to be sorted.
entity	NeIEntity	Entity to which the below attribute name belongs.
attributeName	java.lang.String	Name of the attribute, the value of which determines whether or not sorting is performed.
Return Value	boolean	

10.20 NeIGroupValueUpdater

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface offers methods properties to generate group nodes (summary bars) and to calculate dates of summary bars.

Properties of the Interface

Active

Property of NeIGroupValueUpdater

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

By this property you can set or retrieve wether the group value updater object is enabled (true) or disabled (false).

Accessing Methods
void setActive (boolean newValue)
boolean isActive ()

CreateMissingGroupNodes

Property of NeIGroupValueUpdater

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

By this property you can set or retrieve, wether nodes that are missing in the group are automatically created (true) or if they are not (false).

Accessing Methods
void setCreateMissingGroupNodes (boolean newValue)
boolean isCreateMissingGroupNodes ()

Methods of the Interface

addReference

Method of NeIGroupValueUpdater

This method lets you add values of reference nodes to an existing list of values that is used to calculate the extent and position of the group node (summary bar).

Declaration
void addReference (java.lang.String destAttributeName, java.lang.String srcAttributeName, int mode)

	Data Type	Description
Parameter		
destAttributeName	java.lang.String	Name of the attribute, to which the calculated result is to be stored.
srcAttributeName	java.lang.String	Name of the attribute, from which the value to be added is to be taken.
mode	int	The mode defines, whether the reference value added should contribute to establish the start date or the end date of the group node.
	Possible Values:	
	MAXIMUM	The value will be used to establisch the end value of the group node.
	MINIMUM	The value will be used to establisch the start value of the group node.
Return Value	void	

Also see removeReference

removeReference

Method of NeIGroupValueUpdater

This method lets you remove values of reference nodes from an existing list of values that is used to calculate the extent and position of the group node (summary bar).

Declaration
void removeReference (java.lang.String destAttributeName)

	Data Type	Description
Parameter		
destAttributeName	java.lang.String	Name of the attribute from which the value is not to be taken any more.
Return Value	void	

Also see addReference

10.21 NeILabel

Belongs to VariousClasses

Package name **de.netronic.common.intface**
Extends **java.io.Serializable**

This interface represents a label object. A label object is a "decorative element" that can be placed onto a point of an object. In contrast to an NeIPictureObject the size of an NeILabel object is defined by the implementing class.

The methods of this class are used by the Gantt graph to position and draw the label.

Methods to Handle the Label Object

addPropertyChangeListener(...)	Adds a listener for change events in the label object.
removePropertyChangeListener(...)	Removes a listener for change events.

Methods to Handle the Positioning

calculateBottom(...)	Bottom of the Label
calculateBoundingBox(...)	Minimum size rectangle surrounding the label
calculateTop(...)	Top of the label
createDrawingElement(...)	Element to draw manually generated labels.

Methods of the Interface

addPropertyChangeListener

Method of NeILabel

This method adds a listener for change events in the label object. The listener will be informed each time a property has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

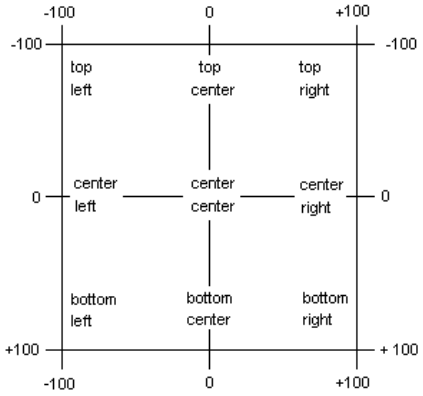
Also see removePropertyChangeListener

calculateBottom

Method of NeILabel

This method calculates the bottom border of the label, that is, the y value of its coordinates. It will be returned as an offset from the top margin of the diagram, measured by 1/100 mm.

Declaration
int calculateBottom (java.awt.Point refPoint, NeIEntity entity)

	Data Type	Description
Parameter		
refPoint	java.awt.Point	Position in the coordinate system of the label. You can select one of the below constants, to place the symbol in defined places within a positioning square. Alternatively, you may enter coordinates from -100 to +100 to place the symbol inbetween the pre-defined positions. In addition, you can enter coordinates beyond -100 and +100. 
entity	NeIEntity	Entity of the node that the label belongs to.
Return Value	int	Calculated y value of the bottom border.

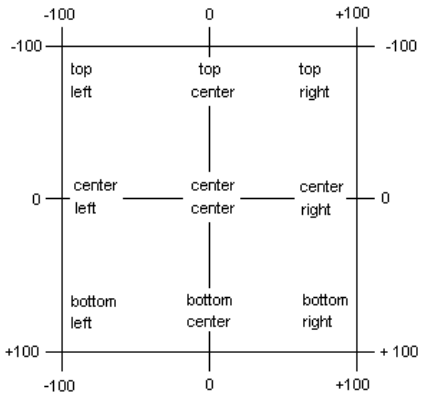
Also see calculateBoundingBox
calculateTop

calculateBoundingBox

Method of NeILabel

This methods calculates the minimum rectangle that can be drawn around a label.

Declaration
java.awt.Rectangle calculateBoundingBox (java.awt.Point refPoint, NeIEntity entity, java.awt.Rectangle oldRect)

	Data Type	Description
Parameter		
refPoint	java.awt.Point	Position in coordinate system of the symbol. You can select one of the below constants, to place the symbol at defined places within a positioning square. Alternatively, you may enter coordinates from -100 to +100 to place the symbol inbetween the pre-defined positions. In addition, you can enter coordinates beyond -100 and +100. 
entity	NeIEntity	Entity of the node that the label belongs to.
oldRect	java.awt.Rectangle	This parameter serves to pass an existing rectangle object that receives the values of the bounding box. For reasons of performance, it makes sense to re-use existing rectangles. If you leave this parameter empty, a new rectangle object will automatically be generated and returned.
Return Value	java.awt.Rectangle	Minimum rectangle calculated.

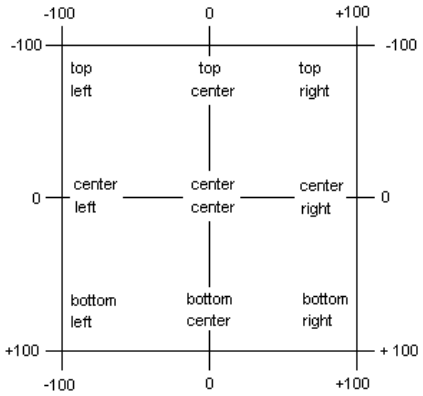
Also see calculateBottom
calculateTop

calculateTop

Method of NeILabel

This method calculates the top border of the label, that is, the y value of its coordinates. It will be returned as an offset from the top margin of the diagram, measured by 1/100 mm.

Declaration
int calculateTop (java.awt.Point refPoint, NeIEntity entity)

	Data Type	Description
Parameter		
refPoint	java.awt.Point	Position in coordinate system of the label. You can select one of the below constants, to place the symbol at defined places within a positioning square. Alternatively, you may enter coordinates from -100 to +100 to place the symbol inbetween the pre-defined positions. In addition, you can enter coordinates beyond -100 and +100. 
entity	NeIEntity	Entity of the node that the label belongs to.
Return Value	int	Calculated y value of the top border.

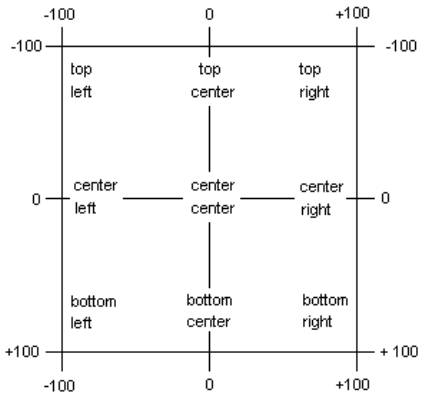
Also see calculateBottom
calculateBoundingBox

createDrawingElement

Method of NeILabel

This method lets you generate a drawing element, that is used to draw manually generated labels.

Declaration
NeIDrawingElement createDrawingElement (java.awt.Point refPoint, NeIEntity entity)

	Data Type	Description
Parameter		
refPoint	java.awt.Point	Position in coordinate system of the symbol. You can select one of the below constants, to place the symbol at defined places within a positioning square. Alternatively, you may enter coordinates from -100 to +100 to place the symbol inbetween the pre-defined positions. In addition, you can enter coordinates beyond -100 and +100. 
entity	NeIEntity	Entity
Return Value	NeIDrawingElement	

removePropertyChangeListener

Method of NeILabel

Removes an existing listener for change events in the label object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

10.22 NeILabelAttachment

Belongs to VariousClasses

Package name **de.netronic.common.intface**
Extends **java.io.Serializable**

This interface represents the placement of a label at an object..

Properties to Handle Label Attachments

Label	Retrieves the symbol that the attachment belongs to.
Position	Retrieves the position in the coordinate system of the link.
Position	Retrieves the position in the coordinate system of the layer.
RefPosition	Retrieves the position in the coordinate system of the label.
RefPosition	Retrieves the position in the coordinate system of the label.

Properties of the Interface

Label

Read Only Property of NeILabelAttachment

Typ	**NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the label that belongs to the label attachment.

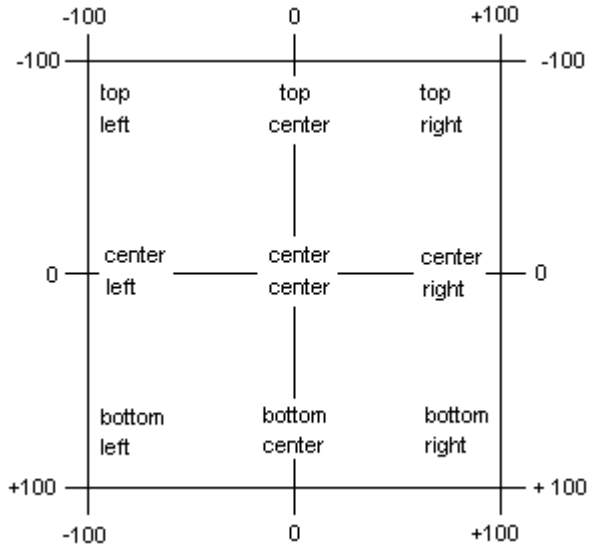
Accessing Methods
NeILabel getLabel()

Position

Read Only Property of NeILabelAttachment

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the position of the label in the coordinate system of the object that the label was attached to. Possible range of values: -100...+100.



Possible Values	Description
POINT_BOTTOM_CENTER	Bottom center position. Coordinates: 0/-100.
POINT_BOTTOM_LEFT	Bottom left position. Coordinates: -100/100.
POINT_BOTTOM_RIGHT	Bottom right position. Coordinates: 100/100.
POINT_CENTER_CENTER	Center center position. Coordinates: 0/0.
POINT_CENTER_LEFT	Center left position. Coordinates: -100/0.
POINT_CENTER_RIGHT	Center right position. Coordinates: 100/0.
POINT_TOP_CENTER	Top center position. Coordinates: 0/-100.
POINT_TOP_LEFT	Top left position. Coordinates: -100/-100.
POINT_TOP_RIGHT	Top right position. Coordinates: 100/-100.

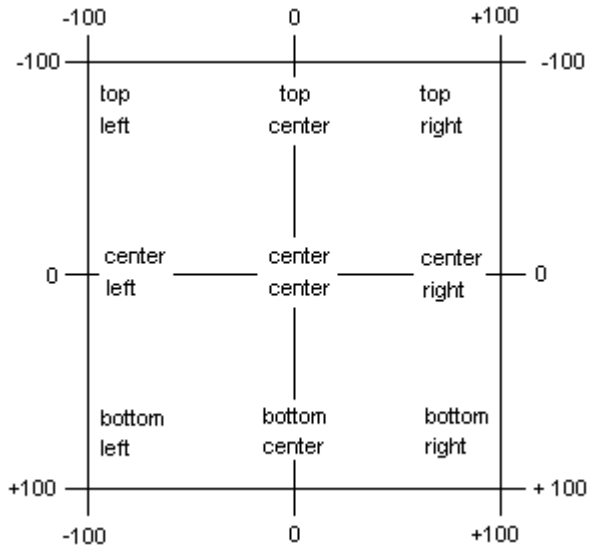
Accessing Methods
int getPosition()

RefPosition

Read Only Property of NeILabelAttachment

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property the position in the coordinate system of the label. Possible range of values: -100...+100.



Possible Values	Description
POINT_BOTTOM_CENTER	Bottom center position. Coordinates: 0/-100.
POINT_BOTTOM_LEFT	Bottom left position. Coordinates: -100/100.
POINT_BOTTOM_RIGHT	Bottom right position. Coordinates: 100/100.
POINT_CENTER_CENTER	Center center position. Coordinates: 0/0.
POINT_CENTER_LEFT	Center left position. Coordinates: -100/0.
POINT_CENTER_RIGHT	Center right position. Coordinates: 100/0.
POINT_TOP_CENTER	Top center position. Coordinates: 0/-100.
POINT_TOP_LEFT	Top left position. Coordinates: -100/-100.
POINT_TOP_RIGHT	Top right position. Coordinates: 100/-100.

Accessing Methods
int getRefPosition()

10.23 NeILineAttributes

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface lets you define the properties of a grid line.

Properties of the Grid Line

DrawingPriority	For internal use only.
LineColor	Sets/retrieves the color of the line.
LineThickness	Sets/retrieves the thickness of the line.
LineType	Sets/retrieves the line type.

Properties of the Interface

DrawingPriority

Property of NeILineAttributes

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**0**

For internal use only.

Accessing Methods
void setDrawingPriority (int newValue)
int getDrawingPriority ()

LineColor

Property of NeILineAttributes

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**Color.BLACK**

This property lets you set or retrieve the color of a line.

Accessing Methods
void setLineColor (java.awt.Color newValue)
java.awt.Color getLineColor ()

Example Code

```
NeIHorLineGrid horGroupGridLevel1 = horLineGrids.getGridForGroups(1);
horGroupGridLevel1.setLineColor (myLevel1Color);
```

LineThickness

Property of NeILineAttributes

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**expert**
Default Value	**1**

This property lets you set or retrieve the thickness of a line.

Accessing Methods
void setLineThickness (int newValue)
int getLineThickness ()

Example Code

```
NeIHorLineGrid horGroupGridLevel1 = horLineGrids.getGridForGroups(1);
horGroupGridLevel1.setLineThickness(130);
```

LineType

Property of NeILineAttributes

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**LINE_TYPE_0**

This property lets you set or retrieve the type of a line.

Possible Values	Description
LINE_TYPE_0	Solid line
LINE_TYPE_1	Line of long dashes separated by average spaces
LINE_TYPE_10	Line of one very long dash alternating with three dots, separated by very short spaces
LINE_TYPE_11	Line of one very long dash alternating with two very short dashes, separated by very short spaces
LINE_TYPE_12	Line of one very long dash alternating with two dots, separated by very short spaces
LINE_TYPE_13	Line of one very long dash alternating with one very short dash, separated by very short spaces
LINE_TYPE_14	Line of one very long dash alternating with one dot, separated by very short spaces
LINE_TYPE_15	Line of one extremely long dash alternating with one dot, separated by very short spaces
LINE_TYPE_16	Line of short dashes separated by very narrow spaces
LINE_TYPE_17	Line of rather long dashes separated by very short spaces

LINE_TYPE_18	Line of one very long dash alternating with four dots, separated by very short spaces
LINE_TYPE_2	Dotted line
LINE_TYPE_3	Line of short dashes separated by very narrow spaces
LINE_TYPE_4	Line of short dashes separated by narrow spaces
LINE_TYPE_5	Line of long dashes separated by average spaces
LINE_TYPE_6	Line of long dashes sparated by very wide spaces
LINE_TYPE_7	Line of short dashes separated by wide spaces
LINE_TYPE_8	Line of rather long dashes separated by average spaces
LINE_TYPE_9	Line of one very long dash alternating with three very short dashes, separated by very short spaces

Accessing Methods
void setLineType (int newValue)
int getLineType ()

10.24 NeILinkLabelAttachment

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface represents the placement of a label at a link..

Properties of the Interface

Label

Read Only Property of NeILinkLabelAttachment

Typ	**NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the label that belongs to the label attachment.

Accessing Methods
NeILabel getLabel()

Position

Read Only Property of NeILinkLabelAttachment

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property retrieves the position of the label in the coordinate system of the object that the label was attached to. Possible range of values: -100...+100.

Possible Values	Description
CENTER	Position at the center of the horizontal line. value: 0
SOURCE	Positioin at the beginning of the horizontal line. value: -100
TARGET	Position at the end of the horizontal line. value: 100

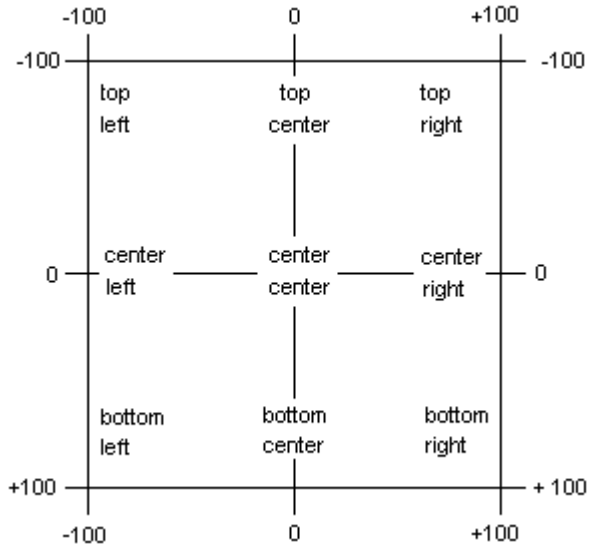
Accessing Methods
int getPosition()

RefPosition

Read Only Property of NeILinkLabelAttachment

Typ	**java.awt.Point**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property the position in the coordinate system of the label. Possible range of values: -100...+100.



Possible Values	Description
POINT_BOTTOM_CENTER	Bottom center position. Coordinates: 0/-100.
POINT_BOTTOM_LEFT	Bottom left position. Coordinates: -100/100.
POINT_BOTTOM_RIGHT	Bottom right position. Coordinates: 100/100.
POINT_CENTER_CENTER	Center center position. Coordinates: 0/0.
POINT_CENTER_LEFT	Center left position. Coordinates: -100/0.
POINT_CENTER_RIGHT	Center right position. Coordinates: 100/0.
POINT_TOP_CENTER	Top center position. Coordinates: 0/-100.
POINT_TOP_LEFT	Top left position. Coordinates: -100/-100.
POINT_TOP_RIGHT	Top right position. Coordinates: 100/-100.

Accessing Methods
java.awt.Point getRefPosition()

10.25 NeIPainter

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface offers properties to paint drawing elements. All properties are for internal use only.

Properties for Internal Use only

DefaultHints	Hints for graphical components
DrawOption	Zeichnungsoptionen
Graphics	Java-Zeichenfläche
SymbolHints	Hints for drawing symbols.

Properties of the Interface

DefaultHints

Read Only Property of NelPainter

Typ	**java.awt.RenderingHints**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Property for internal use only.

Accessing Methods
java.awt.RenderingHints getDefaultHints()

DrawOption

Read Only Property of NelPainter

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Property for internal use only.

Possible Values	Description
DRAW_PICTURES_ASYNCHRONOUSLY	For internal use only
PRINT_DRAW	For internal use only
SCREEN_DRAW_CUMULATIVELY	For internal use only
SCREEN_DRAW_SYNCHRONOUSLY	For internal use only

Accessing Methods
int getDrawOption()

Graphics

Read Only Property of NeIPainter

Typ	**java.awt.Graphics2D**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Property for internal use only.

Accessing Methods
java.awt.Graphics2D getGraphics()

SymbolHints

Read Only Property of NeIPainter

Typ	**java.awt.RenderingHints**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Property for internal use only.

Accessing Methods
java.awt.RenderingHints getSymbolHints()

10.26 NeIPicture

Belongs to VariousClasses

Package name	**de.netronic.common.intface**
Extends	**java.io.Serializable**

This interface represents a picture object. A picture object is a "decorative element" that fills a space in another object. In contrast to a NeILabel object the size of an NeIPicture object is defined by the object that it is allocated to.

The methods of this class are used by a component to position and draw the label.

Methods to Handle the Picture Object

addPropertyChangeListener(...)	Adds a listener for change events in the picture object.

createDrawingElement(...)	Element to draw manually generated pictures.
removePropertyChangeListener(...)	Removes a listener for change events in the picture object.

Methods to Handle the Size of Pictures

getPreferredHeight(...)	Preferred height of the image object to be included.
getPreferredSize(...)	Preferred size of the image object to be included
getPreferredWidth(...)	Preferred width of the image object to be included

Methods of the Interface

addPropertyChangeListener

Method of NeIPicture

This method adds a listener for change events in the picure object. The listener will be informed each time a property has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

createDrawingElement

Method of NeIPicture

This method lets you generate a drawing element that is used to draw manually generated picture objects.

Declaration
NeIDrawingElement createDrawingElement (java.awt.Rectangle boundingRect, NeIEntity entity)

	Data Type	Description
Parameter		
boundingRect	java.awt.Rectangle	Rectangle that the drawing element is to be placed in.
entity	NeIEntity	Entity, in which dependencies of the kind of drawing are defined.
Return Value	NeIDrawingElement	

getPreferredHeight

Method of NeIPicture

This method retrieves the height preferred by the image object to be included.

Declaration
int getPreferredHeight (NeIEntity entity)

	Data Type	Description
Parameter		
entity	NeIEntity	Node or link entity, that this picture object belongs to.
Return Value	int	Height value returned (Unit: 1/100 mm)

Also see getPreferredSize
getPreferredWidth

getPreferredSize

Method of NeIPicture

This method retrieves the size preferred by the image object to be included.

Declaration
java.awt.Dimension getPreferredSize (NeIEntity entity)

	Data Type	Description
Parameter		
entity	NeIEntity	Node or link entity, that this picture object belongs to.
Return Value	java.awt.Dimension	Size returned

Also see getPreferredHeight
getPreferredWidth

getPreferredWidth

Method of NeIPicture

This method retrieves the width preferred by the image object to be included.

Declaration
int getPreferredWidth (NeIEntity entity)

	Data Type	Description
Parameter		
entity	NeIEntity	Node or link entity, that this picture object belongs to.
Return Value	int	Width value returned (Unit: 1/100 mm)

Also see getPreferredHeight
getPreferredSize

removePropertyChangeListener

Method of NeIPicture

Removes an existing listener for change events in the picture object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

10.27 NeITransaction

Belongs to VariousClasses

Package name **de.netronic.common.intface**

Transactions summarize methods concerning the applicationData into a comprehensive operation. This interface offers methods and properties to run and to report on transactions. Some of them are invoked by the NeITransactionHandler interface.

Properties to Handle Transactions

Methods to Handle Transactions

commit()

handleException(...)

rollback() Roll back of a transaction

Properties of the Interface

State

Read Only Property of NeITransaction

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**NEW**

Retrieves the value of an exception.

Possible Values	Description
COMMIT	NeITransaction.commit() is being performed
COMPLETED	NeITransaction.commit() was successfully finished.
FAILED	NeITransaction.commit() and NeITransaction.rollback() have failed.
NEW	New transaction
ROLLBACK	The NeITransaction.rollback() is being performed.
ROLLED_BACK	The(NeITransaction.rollback() was successfully finished.

Accessing Methods
int getState()

Methods of the Interface

commit

Method of NeITransaction

This method is invoked by the interface NeITransactionHandler to perform a transaction. The method needs to be implemented by the application programmer and contains the code that finally represents the transaction (such as createEntity() etc).

Declaration
boolean commit ()

	Data Type	Description
Return Value	boolean	If the method returns **false**, a rollback is invoked by the interface NeITransactionHandler.

Also see rollback

handleException

Method of NeITransaction

This method is invoked by the interface NeITransactionHandler, if during the method **commit** or the method **rollback** an exception occurs. The first paramter contains the exception, the second indicates whether the exeption was thrown during **commit** (isCommitting = true) or during **rollback** (isCommitting = false).

Declaration
boolean handleException (java.lang.Exception e, boolean isCommitting)

	Data Type	Description
Parameter		
e	java.lang.Exception	Indicates the exception.
isCommitting	boolean	If the value equals **true**, the exception was thrown during the performance of the **commit** method; if it equals **false**, the exception was thrown during the **rollback**.
Return Value	boolean	

rollback

Method of NeITransaction

This method is invoked by the interface NeITransactionHandler to provide roll back of a transaction.

Declaration
boolean rollback ()

	Data Type	Description
Return Value	boolean	If the method returns **true**, the rollback was successfull, in case of **false** it was not.

Also see commit

10.28 NeITransactionHandler

Belongs to VariousClasses

Package name **de.netronic.common.intface**

Transactions summarize methods concerning the applicationData into a comprehensive operation. This interface offers methods and properties to run and to report on transactions.

Properties to Handle Transactions

NestedTransactionEvents	Generation of nested transaction events

Methods to Handle Transactions

addTransactionListener(...)	Adds a listener for transactions.
commitTransaction(...)	Performs a transaction.
removeTransactionListener(...)	Removes a listener for transactions.
rollbackTransaction(...)	Reverses a transaction performed.

Properties of the Interface

NestedTransactionEvents

Property of NeITransactionHandler

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property sets/retrieves, whether or not nested transaction events are sent. Nested transactions occur, if within the method **commitTransaction** one or more further transactions are invoked. If the property returns **false**, for the transactions enclosed no events will be generated for their start and finish; only the transaction on the outmost level will trigger a start and finish event. If the property returns **true**, events will be generated for all transactions.

Accessing Methods
void setNestedTransactionEvents (boolean newValue)
boolean hasNestedTransactionEvents ()

Methods of the Interface

addTransactionListener

Method of NeITransactionHandler

This method adds a listener for transactions. The listener will be informed each time a transaction was started or finished.

Declaration
java.beans.PropertyChangeListener addTransactionListener (de.netronic.common.event.-NeTransactionListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeTransactionListener	Listener to be added.
Return Value	java.beans.PropertyChangeListener	

commitTransaction

Method of NeITransactionHandler

This method lets you perform a transaction. It invokes the method **NeITransaction.commit** for the transaction passed. Before and after the call registered NeTransactionListeners are informed about the start and the end of the transaction.
An exception will be thrown when the transaction is already performing, that is, if the state of the transaction is NeITransaction.commit or NeITransaction.rollback.

Declaration
void commitTransaction (de.netronic.common.intface.NeITransaction transaction) throws NeIllegalStateException

	Data Type	Description
Parameter		
transaction	de.netronic.common.intface.-NeITransaction	Transaction to be performed.
Return Value	void	

Also see rollbackTransaction

removeTransactionListener

Method of NeITransactionHandler

This method removes a listener for transactions.

Declaration
void removeTransactionListener (de.netronic.common.event.NeTransactionListener l)

	Data Type	Description
Parameter		
l	de.netronic.common.event.-NeTransactionListener	Listener to be removed.
Return Value	void	

rollbackTransaction

Method of NeITransactionHandler

This method reverses a transaction being performed. If the method **commitTransaction** returns the value **false**, a reversion is automatically triggered by the transaction handler.

Declaration
de.netronic.common.intface.NeITransaction rollbackTransaction (de.netronic.common.intface.-NeITransaction transaction) throws NeIllegalStateException

	Data Type	Description
Parameter		
transaction	de.netronic.common.intface.-NeITransaction	Transaction to be performed.
Return Value	de.netronic.common.intface.-NeITransaction	

Also see commitTransaction

10.29 NeIUserActionSource

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface offers methods to handle action listeners.

Methods for Internal Use

fireOnUserAction(...)	Internal method
fireOnUserActionTriggerShow(...)	Internal method

Methods to Handle the UserActionSource Object

addUserActionListener(...)	Adds a NeUserActionListener
removeUserActionListener(...)	Removes an NeUserActionListener

Methods of the Interface

addUserActionListener

Method of NeIUserActionSource

This method lets you add a NeUserActionListener to the listener list.

Declaration
void addUserActionListener (de.netronic.common.event.NeUserActionListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeUserActionListener	Listener to be added
Return Value	void	

Also see removeUserActionListener

fireOnUserAction

Method of NeIUserActionSource

For internal use only.

Declaration
void fireOnUserAction (de.netronic.common.event.NeUserActionEvent e)

	Data Type	Description
Parameter		
e	de.netronic.common.event.-NeUserActionEvent	For internal use only.
Return Value	void	

fireOnUserActionTriggerShow

Method of NeIUserActionSource

For internal use only.

Declaration
void fireOnUserActionTriggerShow (de.netronic.common.event.NeUserActionEvent e)

	Data Type	Description
Parameter		
e	de.netronic.common.event.-NeUserActionEvent	For internal use only.
Return Value	void	

removeUserActionListener

Method of NeIUserActionSource

This method lets you remove a NeUserActionListener from the listener list.

Declaration
void removeUserActionListener (de.netronic.common.event.NeUserActionListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeUserActionListener	Listener to be removed
Return Value	void	

Also see addUserActionListener

10.30 NeIValueReference

Belongs to VariousClasses

Package name **de.netronic.common.intface**

This interface offers properties and methods to handle the NeIValueReference object. It serves to comfortably handle date attributes, especially in case of a composed date that for example consists of a basic date plus a duration.

Properties to Handle Date Attributes

AttributeName	Name of the attribute that the value reference object refers to.

Methods to Transform Dates

getValueAsDouble(...)	Determines a date in an entity.
getValueAsLong(...)	Determines a date in an entity.
isDependentOnAttribute(...)	Does date depend on attribute?
makeAttributeFromValue(...)	Makes an object from a value to be stored to an attribute.
makeAttributeFromValue(...)	Makes an object from a value to be stored to an attribute.

Properties of the Interface

AttributeName

Read Only Property of NeIValueReference

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property handles the names of attribute that the value reference object refers to.

Accessing Methods
java.lang.String getAttributeName()

Methods of the Interface

getValueAsDouble

Method of NeIValueReference

This method retrieves a value, for example a curve value, from an entity. The entity is to be passed by the parameter. The value to be retrieved is defined by the value reference object.

Declaration
double getValueAsDouble (de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity in which a date is to be retrieved.
Return Value	double	Date in the form of the number of seconds since 1.1.1970.

getValueAsLong

Method of NeIValueReference

This method retrieves a value from an entity. The entity and the profile valid for it are to be passed by parameters. The value to be retrieved is defined by the value reference object.

Declaration
long getValueAsLong (de.netronic.common.intface.NeIEntity entity, de.netronic.common.intface.NeIProfile profile)

	Data Type	Description
Parameter		
entity	de.netronic.common.intface.NeIEntity	Entity in which a date is to be retrieved.
profile	de.netronic.common.intface.NeIProfile	Profile valid for the entity.
Return Value	long	Date in the form of the number of seconds since 1.1.1970.

isDependentOnAttribute

Method of NeIValueReference

This method retrieves, whether or not the value depends on the attribute passed. This may be true in the case of a date composed by another date and a duration.

Declaration
boolean isDependentOnAttribute (java.lang.String name)

	Data Type	Description
Parameter		
name	java.lang.String	Name of the attribute on which the dependence is to be verified.
Return Value	boolean	**True**: the date depends on the attribute passed, **false**: the date does not depend on the attribute passed.

makeAttributeFromValue

Method of NeIValueReference

This method makes an object from a value to be stored to an attribute.

Declaration
java.lang.Object makeAttributeFromValue (double value, de.netronic.common.intface.NeIEntity entity)

	Data Type	Description
Parameter		
value	double	Value, the attribute of which is to be retrieved.
entity	de.netronic.common.intface.NeIEntity	Entity that the attribute belongs to.
Return Value	java.lang.Object	Object returned

makeAttributeFromValue

Method of NeIValueReference

This method makes an object from a value to be stored to an attribute.

Declaration
java.lang.Object makeAttributeFromValue (long value, de.netronic.common.intface.NeIEntity entity, de.netronic.common.intface.NeIProfile profile)

	Data Type	Description
Parameter		
value	long	Value, the attribute of which is to be retrieved.
entity	de.netronic.common.intface.NeIEntity	Entity that the attribute belongs to.
profile	de.netronic.common.intface.NeIProfile	Profile valid for the entity.
Return Value	java.lang.Object	Object returned

10.31 NeLabelMap

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **de.netronic.common.beanbase.NeRangeMap**

A labelMap object handles allocations of values (or range values) and label objects. It is used mainly for dynamic labels.

Methods to Handle the Label Map

addEntry(...)	Adds a discrete value to the label map.
addPropertyChangeListener(...)	Adds a listener for change events in the map object.
addRange(...)	Adds a range value to the label map.
getLabel(...)	Retrieves the label allocated to the key.
iterateLabels()	Returns an iterator object of all labels.
removeEntry(...)	Removes a discrete value from the map.
removePropertyChangeListener(...)	Removes a listener for change events in the map object.
removePropertyChangeListener(...)	Removes a listener for change events in the map object.
removeRange(...)	Removes a range value from the map.
removeRange(...)	Removes a range value from the map.

Constructors of the Class

NeLabelMap

Constructor of NeLabelMap

This constuctor lets you generate an empty label map. Discrete entries and range keys can be added by the corresponding methods.

Declaration
NeLabelMap ()

Methods of the Class

addEntry

Method of NeLabelMap

This method lets you add an entry to the label map. The entry is interpreted as a single, discrete value. This is how you can assign the same label, for example a symbol, to all nodes that have a feature in common.

Declaration
void addEntry (java.lang.Object key, de.netronic.common.intface.NeILabel theLabel)

	Data Type	Description
Parameter		
key	java.lang.Object	Object to provide the key.
theLabel	de.netronic.common.intface.NeILabel	Label object to be allocated to the key.
Return Value	void	

Also see removeEntry

addPropertyChangeListener

Method of NeLabelMap

This method adds a listener for change events in the map object. The listener is informed each time a property was changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

addRange

Method of NeLabelMap

This method lets you add an entry to the label map. The entry is interpreted as a range value. The range extends from the key passed up to, but not including, the next range key of the same class. A range key may for example be a date. The range key needs to be comparable to other range keys, giving a result of being larger, smaller or equal. So all range keys of the same label map need to be of the same type, for example a date. This is how you can assign the same label to all nodes that hold a date which is equal or larger than the one in the key passed but which is smaller than the one in the key following.
If there is no key following, any value equal or larger than the key passed will belong to the range.

Declaration
void addRange (java.lang.Comparable key, de.netronic.common.intface.NeILabel value)

	Data Type	Description
Parameter		
key	java.lang.Comparable	Object to hold the key.
value	de.netronic.common.intface.NeILabel	Label object to be allocated to the range key.
Return Value	void	

Also see removeRange

getLabel

Method of NeLabelMap

This method lets you retrieve the label allocated to the key. Single entries are given priority over ranges.

Declaration
de.netronic.common.intface.NeILabel getLabel (java.lang.Object key)

	Data Type	Description
Parameter		
key	java.lang.Object	Key the label object of which is to be retrieved.
Return Value	de.netronic.common.intface.NeILabel	Label returned

iterateLabels

Method of NeLabelMap

This method returns an iterator object of all labels.

Declaration
java.util.Iterator iterateLabels ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned

removeEntry

Method of NeLabelMap

This method lets you remove a discrete value from the map, as well as the label allocated.

Declaration
void removeEntry (java.lang.Object key)

	Data Type	Description
Parameter		
key	java.lang.Object	Key to be removed.
Return Value	void	

Also see addEntry

removePropertyChangeListener

Method of NeLabelMap

This method lets you remove a listener for change events in the map object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

removeRange

Method of NeLabelMap

This method lets you remove a range value from the map.

Declaration
void removeRange (java.lang.Comparable key)

	Data Type	Description
Parameter		
key	java.lang.Comparable	Key to be removed.
Return Value	void	

Also see addRange

10.32 NeLineStyle

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **java.awt.Color**

An NeLineStyle is an extension of the class java.awt.Color that provides a color, a width and a type to a line.

Properties to Handle the LineStyle Elements

Type	Line type
Width	Line width

Constructors of the Class

The constructors of this class let you generate a lineStyle object with different elements assigned.

NeLineStyle

Constructor of NeLineStyle

This constructor lets you generate a lineStyle object with a color, a line thickness and a line type assigned.

Declaration
NeLineStyle (de.netronic.common.beanbase.NeLineStyle style)

Parameter	Data Type	Description
style	de.netronic.common.beanbase.NeLineStyle	LineStyle object with a color, a line thickness and a line type.

NeLineStyle

Constructor of NeLineStyle

This constructor lets you generate a lineStyle object with a color, a line thickness and a line type assigned.

Declaration
NeLineStyle (java.awt.Color color, int width, int type)

Parameter	Data Type	Description
color	java.awt.Color	Color to be assigned to the line style object.
width	int	Line thickness of the curve. Unit: 1/100 mm.
type	int	Line type to be assigned to the line style object.
	Possible Values:	
	LINE_TYPE_0	Solid line
	LINE_TYPE_1	Line of long dashes separated by average spaces
	LINE_TYPE_10	Line of one very long dash alternating with three dots, separated by very short spaces
	LINE_TYPE_11	Line of one very long dash alternating with two very short dashes, separated by very short spaces
	LINE_TYPE_12	Line of one very long dash alternating with two dots, separated by very short spaces
	LINE_TYPE_13	Line of one very long dash alternating with one very short dash, separated by very short spaces
	LINE_TYPE_14	Line of one very long dash alternating with one dot, separated by very short spaces

LINE_TYPE_15	Line of one extremely long dash alternating with one dot, separated by very short spaces
LINE_TYPE_16	Line of short dashes separated by very narrow spaces
LINE_TYPE_17	Line of rather long dashes separated by very short spaces
LINE_TYPE_18	Line of one very long dash alternating with four dots, separated by very short spaces
LINE_TYPE_2	Dotted line
LINE_TYPE_3	Line of short dashes separated by very narrow spaces
LINE_TYPE_4	Line of short dashes separated by narrow spaces
LINE_TYPE_5	Line of long dashes separated by average spaces
LINE_TYPE_6	Line of long dashes sparated by very wide spaces

	LINE_TYPE_7	Line of short dashes separated by wide spaces
	LINE_TYPE_8	Line of rather long dashes separated by average spaces
	LINE_TYPE_9	Line of one very long dash alternating with three very short dashes, separated by very short spaces 

NeLineStyle

Constructor of NeLineStyle

This constructor lets you generate a lineStyle object with a color assigned.

Declaration
NeLineStyle (java.awt.Color color)

Parameter	Data Type	Description
color	java.awt.Color	Color to be assigned to the line style object.

Properties of the Class

Type

Read Only Property of NeLineStyle

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**LINE_TYPE_0**

This property lets you retrieve the line type.

Possible Values	Description
LINE_TYPE_0	Solid line
LINE_TYPE_1	Line of long dashes separated by average spaces
LINE_TYPE_10	Line of one very long dash alternating with three dots, separated by very short spaces
LINE_TYPE_11	Line of one very long dash alternating with two very short dashes, separated by very short spaces
LINE_TYPE_12	Line of one very long dash alternating with two dots, separated by very short spaces
LINE_TYPE_13	Line of one very long dash alternating with one very short dash, separated by very short spaces
LINE_TYPE_14	Line of one very long dash alternating with one dot, separated by very short spaces
LINE_TYPE_15	Line of one extremely long dash alternating with one dot, separated by very short spaces
LINE_TYPE_16	Line of short dashes separated by very narrow spaces
LINE_TYPE_17	Line of rather long dashes separated by very short spaces

LINE_TYPE_18	Line of one very long dash alternating with four dots, separated by very short spaces
LINE_TYPE_2	Dotted line
LINE_TYPE_3	Line of short dashes separated by very narrow spaces
LINE_TYPE_4	Line of short dashes separated by narrow spaces
LINE_TYPE_5	Line of long dashes separated by average spaces
LINE_TYPE_6	Line of long dashes sparated by very wide spaces
LINE_TYPE_7	Line of short dashes separated by wide spaces
LINE_TYPE_8	Line of rather long dashes separated by average spaces
LINE_TYPE_9	Line of one very long dash alternating with three very short dashes, separated by very short spaces

Accessing Methods
int getType()

Width

Read Only Property of NeLineStyle

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1**

This property lets you set or retrieve the line width. Unit: 1/100 mm

Accessing Methods
int getWidth()

10.33 NeMappedColor

Belongs to VariousClasses

Package name	**de.netronic.common.beanbase**
Extends	**java.awt.Color**
Implements	**de.netronic.common.intface.NeIDynamicColor**

This class represents a color object, the actual color of which is retrieved from a color map. A color map consists of colors and keys that the colors are allocated to.

Methods To Handle Mapped Colors

addPropertyChangeListener(...)	Adds a listener for change events in the NeMappedColor object.
removePropertyChangeListener(...)	Removes a listener for change events.

Constructors of the Class

This constructor lets you generate the color object, the actual color of which is retrieved from a color map. A color map consists of colors and keys that the colors are allocated to.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, the RGB values and the alpha value are passed (for details please see Java class Color), as well as the map, from which the color is retrieved plus the name of the attribute that holds the key to which the color is allocated. RGB values are passed as integer types.

Declaration
NeMappedColor (int r, int g, int b, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
r	int	Value of the red portion of the color that is to be retrieved from the color map (see Java class **Color**).
g	int	Value of the green portion of the color that is to be retrieved from the color map (see Java class **Color**).
b	int	Value of the blue portion of the color that is to retreived from the color map (please see Java class **Color**).
map	NeColorMap	Map, that the color is to be retrieved from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, the RGB value (alpha value included) of the color object and the map are passed, in addition the name of the attribute that holds the key of the color map.

Declaration
NeMappedColor (int rgba, boolean hasAlpha, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
rgba	int	RGB and alpha value of the color that is to be registered to the color map (please see Java class **Color**).
hasAlpha	boolean	**True**: the figures in the parameter rgba include an alpha value, **false**: the figures in the parameter rgba do not include an alpha value.
map	NeColorMap	Map, that the color is to be retrieved from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, the RGB value of the color object and the map are passed, in addition the name of the attribute that holds the key of the color map.

Declaration
NeMappedColor (int rgb, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
rgb	int	RGB value of the color that is to be registered to the color map (please see Java class **Color**).
map	NeColorMap	Map, that the color is to be retrieved from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, the RGB values and the alpha value are passed (for details please see Java class Color), as well as the map, from which the color is retrieved plus the name of the attribute that holds the key, to which the color is allocated. RGB values are passed as integer types.

Declaration
NeMappedColor (int r, int g, int b, int alpha, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
r	int	Value of the red portion of the color that is to be retrieved from the color map (see Java class **Color**).
g	int	Value of the green portion of the color that is to be retrieved from the color map (see Java class **Color**).
b	int	Value of the blue portion of the color that is to retreived from the color map (please see Java class **Color**).
alpha	int	Alpha value of the color (please see Java class **Color**) that is to be retrieved from the color map.
map	NeColorMap	Map, that the color is to be retrieved from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, the RGB values are passed; the map, from which the color is to be retrieved as well as the name of the attribute, that holds the key, to which the color is allocated. RGB values are passed as float types.

Declaration
NeMappedColor (float r, float g, float b, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
r	float	Value of the red portion of the color that is to be retrieved from the color map (see Java class **Color**).
g	float	Value of the green portion of the color that is to be retrieved from the color map (see Java class **Color**).
b	float	Value of the blue portion of the color that is to retreived from the color map (please see Java class **Color**).
map	NeColorMap	Map, that the color is to be retrieved from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, the RGB values and the alpha value are passed (for details please see Java class Color), as well as the map, from which the color is retrieved. The RGB values are passed as float types.

Declaration
NeMappedColor (float r, float g, float b, float alpha, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
r	float	Value of the red portion of the color that is to be retrieved from the color map (see Java class **Color**).
g	float	Value of the green portion of the color that is to be retrieved from the color map (see Java class **Color**).
b	float	Value of the blue portion of the color that is to retrieved from the color map (please see Java class **Color**).
alpha	float	Alpha value of the color (please see Java class **Color**) that is to be retrieved from the color map.
map	NeColorMap	Map, that the color is to be retrieved from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, the color object and the map are passed; in addition the name of the attribute is also passed that holds the key of the color map.

Declaration
NeMappedColor (java.awt.Color color, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
color	java.awt.Color	Color object the color of which is to be retrieved from the map.
map	NeColorMap	Color map, that the color is to be retrieved from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

NeMappedColor

Constructor of NeMappedColor

This constructor generates a color object, the color of which is retrieved from a color map. As parameters, a color space, the components and the alpha value are passed (for details please see Java class Color); in addition, the map and the name of an attribute that holds the key to which the color is allocated are also passed.

Declaration

NeMappedColor (java.awt.color.ColorSpace cSpace, float[] components, float alpha, NeColorMap map, java.lang.String attributeName)

Parameter	Data Type	Description
cSpace	java.awt.color.ColorSpace	Color space value of the color (please see Java class **Color**).
components	float[]	Components value of the color (please see Java class **Color**).
alpha	float	Alpha value of the color (please see Java class **Color**).
map	NeColorMap	Map, that the color is to be derived from.
attributeName	java.lang.String	Name of the entity attribute, that holds the key for the color in the color map.

Methods of the Class

addPropertyChangeListener

Method of NeMappedColor

This method adds a listener for change events in the NeMappedColor object. The listener will be informed each time a property has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

removePropertyChangeListener

Method of NeMappedColor

Removes an existing listener for change events in the NeMappedColor object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

10.34 NeMappedLabel

Belongs to VariousClasses

Package name	**de.netronic.common.beanbase**
Extends	**java.lang.Object**
Implements	**de.netronic.common.intface.NeIDynamicLabel**
I	**java.io.Serializable**

A mappedLabel object is an implementation of NeIDynamicLabel, where the NeILabel object displayed is selected from a label map (NeLabelMap). The selection is made via the content of an attribute of the associated entity.

Properties to Handle Mapped Labels

DefaultLabel	Default label

Methods to Handle Mapped Labels

addPropertyChangeListener(...)	Adds a listener for change events in the NeMappedPicture object.
addPropertyChangeListener(...)	Adds a listener for change events in the NeMappedLabel object.
iterateLabels()	Iterates over the labels existing.
iteratePictures()	Iterates over the existing picture objects.
removePropertyChangeListener(...)	Removes a listener for change events.
removePropertyChangeListener(...)	Removes a listener for change events.

Constructors of the Class

NeMappedLabel

Constructor of NeMappedLabel

This constructor lets you generate an NeMappedLabel object. The label, the map, and the name of an attribute that holds the key value, are passed.

Declaration
NeMappedLabel (de.netronic.common.intface.NeILabel defaultLabel, NeLabelMap map, java.lang.String attributeName)

Parameter	Data Type	Description
defaultLabel	de.netronic.common.intface.NeILabel	Label that is used in case the table does not hold a label that corresponds to the key.
map	NeLabelMap	Label map that holds keys and allocated labels.
attributeName	java.lang.String	Name of the entity attribute that holds the key for the label in the label map.

Properties of the Class

DefaultLabel

Property of NeMappedLabel

Typ	**de.netronic.common.intface.NeILabel**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can set the default label or retrieve the one set. The default label is assigned to a node if none of the keys of the label map is true.

Accessing Methods
void setDefaultLabel (de.netronic.common.intface.NeILabel newValue)
de.netronic.common.intface.NeILabel getDefaultLabel ()

Methods of the Class

addPropertyChangeListener

Method of NeMappedLabel

This method adds a listener for change events in the NeMappedLabel object. The listener will be informed each time a property of the NeMappedLabel has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

iterateLabels

Method of NeMappedLabel

This method lets you iterate over the existing labels. The iterator delivers objects of the type NeILabel.

Declaration
java.util.Iterator iterateLabels ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned

removePropertyChangeListener

Method of NeMappedLabel

Removes an existing listener for change events in the NeMappedLabel object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

10.35 NeMappedPicture

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **java.lang.Object**

A mappedPicture object is an implementation of NeIDynamicPicture, where the NeIPicture object displayed is selected from a picture map (NePictureMap). The selection is made via the content of an attribute of the associated entity.

Constructors of the Class

NeMappedPicture

Constructor of NeMappedPicture

This constructor lets you generate an NeMappedPicture object. The picture, the map, and the name of an attribute that holds the key value of the picture, are passed.

Declaration
NeMappedPicture (de.netronic.common.intface.NeIPicture defaultPicture, NePictureMap map, java.lang.String attributeName)

Parameter	Data Type	Description
defaultPicture	de.netronic.common.intface.NeIPictur e	Picture that is used in case the table does not hold a picture that corresponds to the key.
map	NePictureMap	Picture map that holds keys and allocated pictures.
attributeName	java.lang.String	Name of the entity attribute that holds the key for the picture in the picture map.

Methods of the Class

addPropertyChangeListener

Method of NeMappedPicture

This method adds a listener for change events in the NeMappedPicture object. The listener is informed each time a property of the NeMappedPicture was changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

iteratePictures

Method of NeMappedPicture

This method lets you iterate over the existing picture objects. The iterator delivers objects of the type NeIPicture.

Declaration
void iteratePictures ()

	Data Type	Description
Return Value	void	Iterator object returned

removePropertyChangeListener

Method of NeMappedPicture

Removes an existing listener for change events on the properties of the NeMappedPicture object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

10.36 NeNotFilter

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIFilter**

This class offers a constructor that serves to generate a filter which contains an inverted condition. For simple conditions please see class **NeValueFilter**, for combined conditions please see class **NeCombinedFilter**.

Constructors of the Class

NeNotFilter

Constructor of NeNotFilter

This constructor lets you generate a filter that contains an inverted condition.

Declaration
NeNotFilter (de.netronic.common.intface.NeIFilter toBeNegated)

Parameter	Data Type	Description
toBeNegated	de.netronic.common.intface.NeIFilter	Filter, the condition of which is inverted by negation.

10.37 NeObjectChangeAdapter

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **java.lang.Object**
Implements **de.netronic.common.event.NeObjectChangeListener**

This class is an adapter class for the interface NeObjectChangeListener. Its methods are empty. This class exists as convenience for creating listener objects.

Methods to Handle Events

objectCreated(...)	Is triggered after an internal object has been created and delivers an event object.
objectDeleted(...)	Is triggered after an internal object has been deleted and delivers an event object.
objectModified(...)	Is triggered after an object has been modified and delivers an event object.
objectSelectionModified(...)	Is triggered after the selection state of an object has been modified and delivers an event object.
onObjectCreate(...)	Is triggered when an internal object is being created and delivers an event object that allows for a veto exception.
onObjectDelete(...)	Is triggered when an internal object is being deleted and delivers an event object that allows for a veto exception.
onObjectModify(...)	Is triggered when an object is being modified and delivers an event object that allows for a veto exception.
onObjectSelectionModify(...)	Is triggered while the selection state of an object is being modified and delivers an event that allows for a veto exception.
onPhantomModify(...)	Is triggered when a phantom is about to change and delivers an event object that allows for a veto exception.
prepareCreateInteraction(...)	Is triggered when the creation of an object is about to happen and delivers an event object that allows for a veto exception.

prepareModifyInteraction(...)	Is triggered when the modification of an object is about to happen and delivers an event object that allows for a veto exception.

Methods of the Interface

objectCreated

Method of NeObjectChangeListener

This method is triggered after an object has been created. The object value of the event passed holds the new object. No map of values is passed.

Declaration
void objectCreated (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectDeleted
objectModified
onObjectCreate

objectDeleted

Method of NeObjectChangeListener

This method is triggered after an object has been deleted. The object value of the event passed holds the object deleted. No map of values is passed.

Declaration
void objectDeleted (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectCreated
objectModified
onObjectDelete

objectModified

Method of NeObjectChangeListener

This method is triggered after an object has been modified. The object value of the event passed holds the modified object, the map of values contains the original values of the object that have been modified. The semantics of the key/value pairs in the map depend on the type of object modified.

Declaration
void objectModified (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectCreated
objectDeleted
onObjectModify

objectSelectionModified

Method of NeObjectChangeListener

This method is triggered after the selection state of an object has been modified. The object value of the event passed holds the object the selection state of which was modified, the map of values contains a boolean which is **true** if the object has been selected, but **false** if the object has been deselected.

Declaration
void objectSelectionModified (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see onObjectSelectionModify

onObjectCreate

Method of NeObjectChangeListener

This method is triggered when an internal object is being created and delivers an event object that allows for a veto exception. An object value is not passed by the event. The map of values contains the new values for the object to be created. The semantics of the key-value pairs in the map depend on the type of object created. The listener can throw an NeVetoException to suppress the creation of the object.

Declaration
void onObjectCreate (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectCreated
onObjectDelete
onObjectModify

onObjectDelete

Method of NeObjectChangeListener

This method is triggered when an internal object is being deleted and delivers an event object that allows for a veto exception. The object value of the event holds the object to be deleted. A map of values is not passed. The listener can throw an NeVetoException to suppress the deletion of the object.

Declaration
void onObjectDelete (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectDeleted
onObjectCreate
onObjectModify

onObjectModify

Method of NeObjectChangeListener

This method is triggered when an object is being modified and delivers an event that allows for a veto exception. The object value of the event represents the object to be modified. The map of values contains the modified values for the object. The semantics of the key-value pairs in the map depend on the type of object created. The listener can throw an NeVetoException to suppress the modification of the object.

Declaration
void onObjectModify (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectModified
onObjectCreate
onObjectDelete

onObjectSelectionModify

Method of NeObjectChangeListener

This method is triggered while the selection state of an object is being modified and delivers an event object that allows for a veto exception. The object value of the event represents the object to be modified. The map of values contains a boolean which is **true** if the object is about to be selected, **false** if the object is about to be deselected. The listener can throw an NeVetoException to suppress the modification of the selection state.

Declaration
void onObjectSelectionModify (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectSelectionModified

onPhantomModify

Method of NeObjectChangeListener

This method is triggered when a phantom is about to change and delivers an event that allows for a veto exception. The object value of the event represents the object that the phantom refers to (for example the node of a node phantom), that is, the object that is going to be modified. The listener can throw an NeVetoException to suppress the modification of the phantom.

Declaration
void onPhantomModify (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

prepareCreateInteraction

Method of NeObjectChangeListener

This method is triggered when an object is about to be created, that is, when the cursor is in the **create** mode has changed its position. The method delivers an event that allows for a veto exception. The object value of the event represents the object that is going to be created. The listener can throw an NeVetoException to suppress the creation of the object.

Declaration
void prepareCreateInteraction (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

prepareModifyInteraction

Method of NeObjectChangeListener

This method is triggered when the modification of an object is about to happen, that is, when the cursor has changed its shape, and delivers an event that allows for a veto exception. The object value of the event represents the object that is going to be modified. The listener can throw an NeVetoException to suppress the modification.

Declaration
void prepareModifyInteraction (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

10.38 NeObjectChangeEvent

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **java.util.EventObject**

This class offers events for changements in objects.

Properties to Handle Change Events

Entity — Entity associated with the object displayed.

Object	Object that is about to change or has just changed
ObjectChangeInfo	Information on the a performed or forthcoming modification
ValueMap	Map of values for change
Values	Former or future values of a modification

Methods to Handle Change Events

setObjectChangeInfo(...)	For internal use only

Constructors of the Class

NeObjectChangeEvent

Constructor of NeObjectChangeEvent

By the constructor of this class you can generate an event that transmits information on the object modified, on the cause of the modifications and on the values modified.

Declaration
NeObjectChangeEvent (java.lang.Object values, java.lang.Object source, java.lang.Object theObject)

Parameter	Data Type	Description
values	java.lang.Object	Object containing the values changed.
source	java.lang.Object	Object that causes the modification.
theObject	java.lang.Object	Object modified.

Properties of the Class

Entity

Read Only Property of NeObjectChangeEvent

Typ	**de.netronic.common.intface.NeIEntity**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the entity associated with the object. The return value may be null if an object is about to be created or if the affected object does not have an associated entity.

Accessing Methods
de.netronic.common.intface.NeIEntity getEntity()

Object

Read Only Property of NeObjectChangeEvent

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the object that is about to change or has just changed. The return value may be null if the object is about to be created.

Accessing Methods
java.lang.Object getObject()

ObjectChangeInfo

Read Only Property of NeObjectChangeEvent

Typ	**NeObjectChangeInfo**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve information on a performed or a forthcoming modification.

Accessing Methods
NeObjectChangeInfo getObjectChangeInfo()

ValueMap

Read Only Property of NeObjectChangeEvent

Typ	**java.util.Map**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the a map of the changed values, if available, otherwise the return value will be null. The semantics of the key/value pairs in the map depend on the object type and the context of the modification.

Accessing Methods
java.util.Map getValueMap()

Values

Read Only Property of NeObjectChangeEvent

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the former values after a performed modification, or the future values of a forthcoming modification, if available, otherwise the return value will be null. The semantics of the values in the values object depend on the type of object modified and the context of the modification.

Accessing Methods
java.lang.Object getValues()

Methods of the Class

setObjectChangeInfo

Method of NeObjectChangeEvent

For internal use only

Declaration
void setObjectChangeInfo (NeObjectChangeInfo newValue)

	Data Type	Description
Parameter		
newValue	NeObjectChangeInfo	For internal use only.
Return Value	void	

10.39 NeObjectChangeInfo

Belongs to VariousClasses

Package name **de.netronic.common.event**

This class offers properties to handle information on changes in objects.

Properties to Handle Information on Changes in Objects

AttributeName	Name of the attribute about to be changed during table editing
BarChange	Modification of a bar or a node
ChangeEnd	State of modification of the final date of a bar or a node
ChangeStart	State of modification of the start date of a bar or a node
CreateLink	Creation of a link
CreateNode	Creation of a node
Date	Start date of the node just created
LinkChange	Modification of a link
LinkSourceChange	Modification of the source node of a link
LinkTargetChange	Modification of the targete node of a link
MoveHorizontal	Horizontal movement of a node
MoveVertical	Vertical movement of a node
NodeChange	Modification of a node
TableEditing	Editing of the table

Properties of the Class

AttributeName

Read Only Property of NeObjectChangeInfo

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the name of an attribute that is about to be changed during table editing.

Accessing Methods
java.lang.String getAttributeName()

BarChange

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) a bar or a node is about to be modified.

Accessing Methods
boolean isBarChange()

ChangeEnd

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the final date of a bar or of a node is about to be changed.

Accessing Methods
boolean isChangeEnd()

Also see ChangeStart

ChangeStart

Read Only Property of NeObjectChangeInfo

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the start date of a bar or of a node is about to be changed.

Accessing Methods
java.lang.String isChangeStart()

Also see ChangeEnd

CreateLink

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) a link is about to be created.

Accessing Methods
boolean isCreateLink()

CreateNode

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) a node is about to be created.

Accessing Methods
boolean isCreateNode()

Date

Read Only Property of NeObjectChangeInfo

Typ	**long**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the start date of the node just created.

Accessing Methods
long getDate()

LinkChange

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) a link is about to be modified.

Accessing Methods
boolean isLinkChange()

LinkSourceChange

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the source node of a link is about to be modified.

Accessing Methods
boolean isLinkSourceChange()

LinkTargetChange

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the target node of a link is about to be modified.

Accessing Methods
boolean isLinkTargetChange()

MoveHorizontal

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) a node is about to be moved horizontally.

Accessing Methods
boolean isMoveHorizontal()

MoveVertical

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) a node is about to be moved vertically.

Accessing Methods
boolean isMoveVertical()

NodeChange

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) a node is about to be modified.

Accessing Methods
boolean isNodeChange()

TableEditing

Read Only Property of NeObjectChangeInfo

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve whether (**true**) or not (**false**) the table is to be edited.

Accessing Methods
boolean isTableEditing()

10.40 NeObjectChangeListener

Belongs to VariousClasses

Package name	**de.netronic.common.event**
Extends	**java.util.EventListener**

This is the listener interface for receiving object change events. The class that is interested in processing an object change event either implements this interface, (including the methods it contains), or extends the ObjectChangeAdapter class (overriding only the methods of interest). The listener object is then registered with

the object using the object's addListener method. An object change event is generated after creating, modifying or deleting an object.

Methods to Handle Events

objectCreated(...)	Is triggered after an internal object has been created and delivers an event object.
objectDeleted(...)	Is triggered after an internal object has been deleted and delivers an event object.
objectModified(...)	Is triggered after an object has been modified and delivers an event object.
objectSelectionModified(...)	Is triggered after the selection state of an object has been modified and delivers an event object.
onObjectCreate(...)	Is triggered when an internal object is being created and delivers an event object that allows for a veto exception.
onObjectDelete(...)	Is triggered when an internal object is being deleted and delivers an event object that allows for a veto exception.
onObjectModify(...)	Is triggered when an object is being modified and delivers an event object that allows for a veto exception.
onObjectSelectionModify(...)	Is triggered while the selection state of an object is being modified and delivers an event that allows for a veto exception.
onPhantomModify(...)	Is triggered when a phantom is about to change and delivers an event object that allows for a veto exception.
prepareCreateInteraction(...)	Is triggered when the creation of an object is about to happen and delivers an event object that allows for a veto exception.
prepareModifyInteraction(...)	Is triggered when the modification of an object is about to happen and delivers an event object that allows for a veto exception.

Methods of the Interface

objectCreated

Method of NeObjectChangeListener

This method is triggered after an object has been created. The object value of the event passed holds the new object. No map of values is passed.

Declaration
void objectCreated (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectDeleted
objectModified
onObjectCreate

objectDeleted

Method of NeObjectChangeListener

This method is triggered after an object has been deleted. The object value of the event passed holds the object deleted. No map of values is passed.

Declaration
void objectDeleted (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectCreated
objectModified
onObjectDelete

objectModified

Method of NeObjectChangeListener

This method is triggered after an object has been modified. The object value of the event passed holds the modified object, the map of values contains the original values of the object that have been modified. The semantics of the key/value pairs in the map depend on the type of object modified.

Declaration
void objectModified (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectCreated
objectDeleted
onObjectModify

objectSelectionModified

Method of NeObjectChangeListener

This method is triggered after the selection state of an object has been modified. The object value of the event passed holds the object the selection state of which was modified, the map of values contains a boolean which is **true** if the object has been selected, but **false** if the object has been deselected.

Declaration
void objectSelectionModified (NeObjectChangeEvent theEvent)

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see onObjectSelectionModify

onObjectCreate

Method of NeObjectChangeListener

This method is triggered when an internal object is being created and delivers an event object that allows for a veto exception. An object value is not passed by the event. The map of values contains the new values for the object to be created. The semantics of the key-value pairs in the map depend on the type of object created. The listener can throw an NeVetoException to suppress the creation of the object.

Declaration
void onObjectCreate (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectCreated
onObjectDelete
onObjectModify

onObjectDelete

Method of NeObjectChangeListener

This method is triggered when an internal object is being deleted and delivers an event object that allows for a veto exception. The object value of the event holds the object to be deleted. A map of values is not passed. The listener can throw an NeVetoException to suppress the deletion of the object.

Declaration
void onObjectDelete (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectDeleted
onObjectCreate
onObjectModify

onObjectModify

Method of NeObjectChangeListener

This method is triggered when an object is being modified and delivers an event that allows for a veto exception. The object value of the event represents the object to be modified. The map of values contains the modified values for the object. The semantics of the key-value pairs in the map depend on the type of object created. The listener can throw an NeVetoException to suppress the modification of the object.

Declaration
void onObjectModify (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectModified
onObjectCreate
onObjectDelete

onObjectSelectionModify

Method of NeObjectChangeListener

This method is triggered while the selection state of an object is being modified and delivers an event object that allows for a veto exception. The object value of the event represents the object to be modified. The map of values contains a boolean which is **true** if the object is about to be selected, **false** if the object is about to be deselected. The listener can throw an NeVetoException to suppress the modification of the selection state.

Declaration
void onObjectSelectionModify (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

Also see objectSelectionModified

onPhantomModify

Method of NeObjectChangeListener

This method is triggered when a phantom is about to change and delivers an event that allows for a veto exception. The object value of the event represents the object that the phantom refers to (for example the node of a node phantom), that is, the object that is going to be modified. The listener can throw an NeVetoException to suppress the modification of the phantom.

Declaration
void onPhantomModify (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

prepareCreateInteraction

Method of NeObjectChangeListener

This method is triggered when an object is about to be created, that is, when the cursor is in the **create** mode has changed its position. The method delivers an event that allows for a veto exception. The object value of the event represents the object that is going to be created. The listener can throw an NeVetoException to suppress the creation of the object.

Declaration
void prepareCreateInteraction (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

prepareModifyInteraction

Method of NeObjectChangeListener

This method is triggered when the modification of an object is about to happen, that is, when the cursor has changed its shape, and delivers an event that allows for a veto exception. The object value of the event represents the object that is going to be modified. The listener can throw an NeVetoException to suppress the modification.

Declaration
void prepareModifyInteraction (NeObjectChangeEvent theEvent) throws NeVetoException

	Data Type	Description
Parameter		
theEvent	NeObjectChangeEvent	Event to be triggered
Return Value	void	

10.41 NePicture

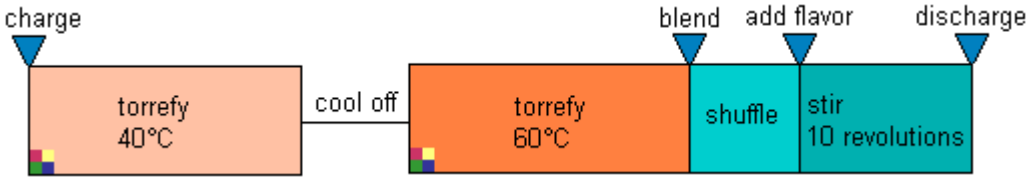
Belongs to VariousClasses

Package name	**de.netronic.common.beanbase**
Extends	**NeSymbol**
Implements	**de.netronic.common.intface.NeIPicture**
I	**de.netronic.common.intface.NeILabel**

Objects of the class **NePicture** are decorative elements, the appearence of which is defined by a bit map.

Picture objects for example are used in table fields and nodes. Nodes may consist of two different types of graphical elements: "layers" and "decorations". The latter

can be positioned on layers. Decorations are represented by annotations, by symbols or by picture objects. There are two different types of decorations, the NeIPicture type and the NeILabel type. So picture objects can be used as both, as an NeIPicture type or an NeILabel type. Below, two picture objects are placed in the bottom left corners of two layers.



Please also see the classes NeAnnotation and NeSymbol.

Properties to Design the Behavior of the Picture

Alignment	Alignment of the foreground theme on its background
BackgroundStyle	Background color of the picture object
Extent	Extent of the background rectangle of the picture object
HorizontalPictureFillMode	Fill mode for the horizontal direction of the background rectangle
Offset	Offset of the postition when used as a n NeILabel
RefPointPosition	Reference point of the picture object
VerticalPictureFillMode	Fill mode for the vertical direction of the background of the annotation

Methods to Design the Behavior of the Picture

setPicture(...)	Bit map from a file
setPicture(...)	Bit map from a URL

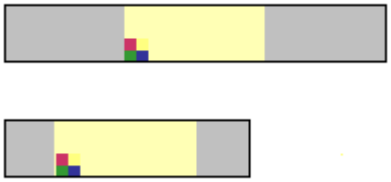
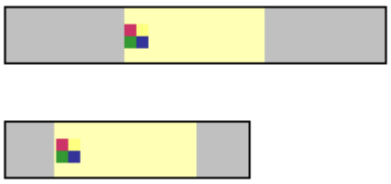
Properties of the Class

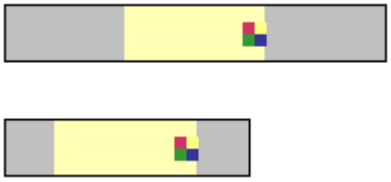
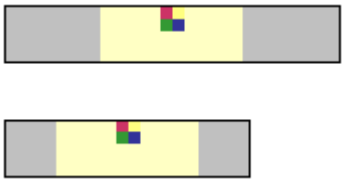
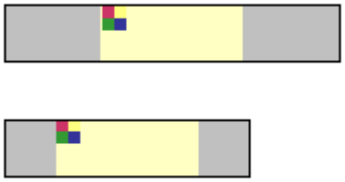
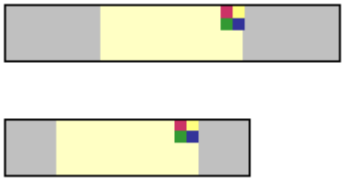
Alignment

Property of NePicture

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**ALIGNMENT_CENTER_CENTER**

This property lets you set or retrieve the allignment of the foreground theme on its background. To reproduce the examples below, the property **Extent** needs to be set in addition, and the horzontal and vertical fill mode was set to **CENTER**.

Possible Values	Description
ALIGNMENT_BOTTOM_CENTER	The foreground theme is placed at the bottom in the center. 
ALIGNMENT_BOTTOM_LEFT	The foreground theme is aligned at the bottom on the left hand side. 
ALIGNMENT_BOTTOM_RIGHT	The foreground theme is aligned at the bottom on the right hand side. 
ALIGNMENT_CENTER_CENTER	The foreground theme is placed in the vertical and horizontal center. 
ALIGNMENT_CENTER_LEFT	The foreground theme is placed in the vertical center on the left hand side. 

ALIGNMENT_CENTER_RIGHT	The foreground theme is placed in the vertical center on the right hand side. 
ALIGNMENT_TOP_CENTER	The foreground theme is placed at the top in the center. 
ALIGNMENT_TOP_LEFT	The foreground theme is placed at the top on the left hand side. 
ALIGNMENT_TOP_RIGHT	The foreground theme is placed at the top on the right hand side. 

Accessing Methods
void setAlignment (int newValue)
int getAlignment ()

BackgroundStyle

Property of NePicture

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the color of the background rectangle of the picture object. You can use simple colors or an NeAreaStyle.

Accessing Methods
void setBackgroundStyle (java.awt.Color newValue)
java.awt.Color getBackgroundStyle ()

Extent

Property of NePicture

Typ	**java.awt.Dimension**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the size of the background rectangle of the picture object. Unit: 1/100 mm. If you do not set a value here, the background defaults to the "natural" size of the bit map. It derives from the size in pixels * 2540 / 72.

Accessing Methods
void setExtent (java.awt.Dimension newValue)
java.awt.Dimension getExtent ()

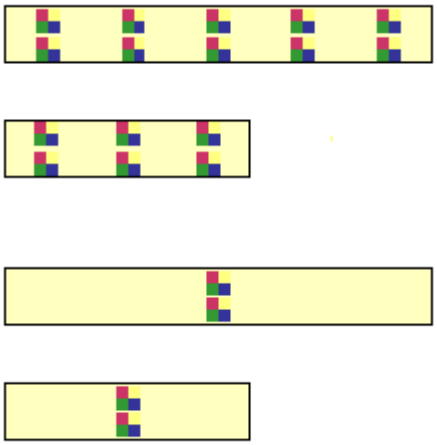
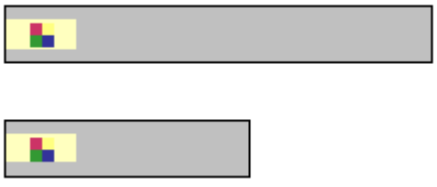
HorizontalPictureFillMode

Property of NePicture

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the background rectangle of the bitmap fills the available area in horizontal direction, if the picture

is used as an NeIPicture. Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the ultimate left position and equals the constant LEFT; +100 describes the ultimate right position and equals the constant RIGHT. If the bitmap is used as an NeILabel, this property will not apply.

Possible Values	Description
AUTO	If by the property **Extent** a width was specified, the background rectangle of the picture tile the available space in horizontal direction until the space is filled. If no horizontal extent was specified, the symbol will be stretched in horizontal direction.  In the upper layers the size of the picture was set to 500x200 (unit: 1/100 mm); in the bottom layers no width was set. The vertical fill mode was set to **TILE**.
CENTER	Within the available space of the layer the picture is placed in the horizontal center position (value: 0).  The vertical fill mode was also set to CENTER, therefore the picture appears centered in both directions.
LEFT	Within the available space the picture is placed in the ultmate left position (value: -100). 
RIGHT	Within the available space the picture is placed in the ultimate right position (value: +100).

STRETCH

The available space is filled in horizontal direction by stretching the background rectangle.

TILE

The available space is tiled in horizontal direction by the background rectangle, if a width was set by the property **Extent**. If no width was set, the foreground will be used for tiling; remaining gaps that were possibly produced by only complete objects being tiled, are filled by the background.

Top picture: Extent specified for background rectangle; bottom picture: No extent specified. The vertical fill mode was also set to "tile".

Accessing Methods

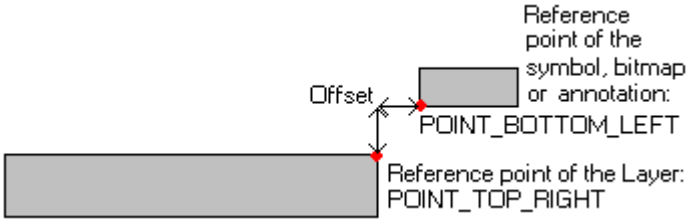
void setHorizontalPictureFillMode (int newValue)
int getHorizontalPictureFillMode ()

Offset

Property of NePicture

Typ	**java.awt.Point**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

When using the picture object as an NeILabel, the object can be positioned on the layer by superimposing the reference point of the reference object and the reference point of the background rectangle of the picture. This property lets you add an offset by which the position of the reference points will differ, or retrieve the offset assigned.



Accessing Methods
void setOffset (java.awt.Point newValue)
java.awt.Point getOffset ()

Also see RefPointPosition

RefPointPosition

Property of NePicture

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**POSITION_CENTER_CENTER**

When the bitmap is positioned on an object, the reference point of the object and the reference point of the bitmap are superimposed. This property lets you set or retrieve the reference point of the bitmap.

Possible Values	Description
POSITION_BOTTOM_CENTER	The reference point is situated in the center of the bottom border.
POSITION_BOTTOM_LEFT	The reference point is situated at the bottom left corner.
POSITION_BOTTOM_RIGHT	The reference point is situated at the bottom right corner.
POSITION_CENTER_CENTER	The reference point is situated in the vertical and horizontal center.
POSITION_CENTER_LEFT	The reference point is situated in the vertical center on the left hand side.
POSITION_CENTER_RIGHT	The refernce point is situated in the vertical center on the right hand side.
POSITION_TOP_CENTER	The reference point is situated in the center of the top border.
POSITION_TOP_LEFT	The refrence point is situated at the top left corner.
POSITION_TOP_RIGHT	The refernce point is situated at the top right corner.

Accessing Methods
void setRefPointPosition (int newValue)
int getRefPointPosition ()

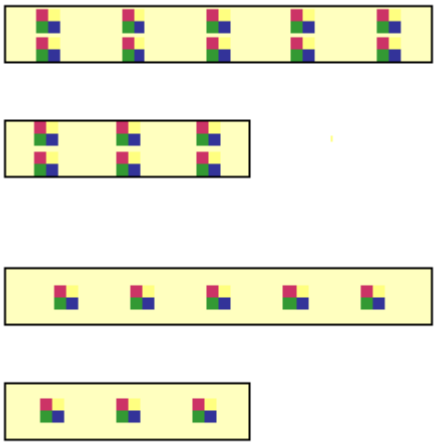
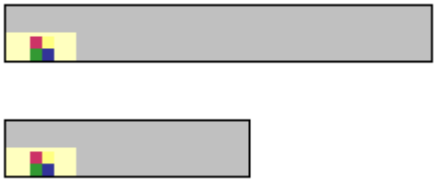
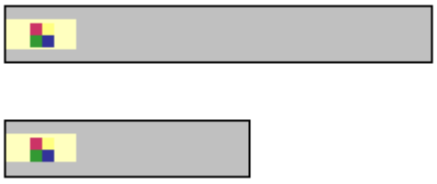
Also see Offset

VerticalPictureFillMode

Property of NePicture

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the background rectangle of a bitmap fills the available area in vertical direction, if the picture is used as an NeIPicture. Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the ultimate top position and equals the constant TOP; +100 describes the ultimate bottom position and equals the constant BOTTOM. If the bitmap is used as an NeILabel, this property will not apply.

Possible Values	Description
AUTO	If by the property **Extent** a height was specified, the background rectangle of the picture will be tiled in vertical direction until the space is filled. If no vertical extend was specified, the symbol will be stretched in horizontal direction.  In the upper layers the height of the picture was set to 500x200 (unit: 1/100 mm); in the bottom layers no height was set. The horizontal fill mode was set to **TILE**.
BOTTOM	Within the available space the picture is placed in the bottom position (value: +100). 
CENTER	Within the available space the picture is placed in the center position (value: 0). 
STRETCH	The available space is filled in vertical direction by stretching the background rectangle.

TILE

The available space is tiled in vertical direction by the background rectangle, if a width was set by the property **Extent**. If no width was set, the foreground is used for tiling; remaining gaps that were possibly produced by only complete objects being tiled, are filled by the background.

Top picture: Extent specified for background rectangle; bottom picture: No extent specified. The horizontal fill mode was also set to "tile".

TOP

Within the available space the picture is placed in the ultimate top position (value: -100).

Accessing Methods
void setVerticalPictureFillMode (int newValue)
int getVerticalPictureFillMode ()

Methods of the Class

setPicture

Method of NePicture

This method allows to load a bit map from a file.

Declaration
void setPicture (java.lang.String fileName)

	Data Type	Description
Parameter		
fileName	java.lang.String	File name (including path), from which the bit map is retrieved.
Return Value	void	

setPicture

Method of NePicture

This method allows to load a bit map from a URL

Declaration
void setPicture (java.net.URL url)

	Data Type	Description
Parameter		
url	java.net.URL	URL from which the bit map is retrieved.
Return Value	void	

10.42 NePictureMap

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **de.netronic.common.beanbase.NeRangeMap**

A picureMap object handles allocations of values (or range values) and picure objects. It is mainly used for dynamic picture objects.

Methods to Handle the Picture Map

addEntry(...)	Adds a discrete value to the picture map.
addPropertyChangeListener(...)	Adds a listener for change events in the map object.
addRange(...)	Adds a range value to the picture map.
getPicture(...)	Retrieves the picture object allocated to the key.
iteratePictures()	Returns an iterator object of all picture objects.
removeEntry(...)	Removes a discrete value from the map.

Constructors of the Class

NePictureMap

Constructor of NePictureMap

This constuctor lets you generate an empty picture map. Discrete entries and range keys can be added by the corresponding methods.

Declaration
NePictureMap ()

Methods of the Class

addEntry

Method of NePictureMap

This method lets you add an entry to the picture map. The entry is interpreted as a single, discrete value. This is how you can assign the same label, for example a symbol, to all nodes that have a feature in common.

Declaration
void addEntry (java.lang.Object key, de.netronic.common.intface.NeILabel thePicture)

	Data Type	Description
Parameter		
key	java.lang.Object	Object to provide the key.
thePicture	de.netronic.common.intface.NeILabel	Picture object to be allocated to the key.
Return Value	void	

addPropertyChangeListener

Method of NePictureMap

This method adds a listener for change events in the JGantt object. The listener is informed each time a property was changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

addRange

Method of NePictureMap

This method lets you add an entry to the picture map. The entry is interpreted as a range value. The range extends from the key passed up to, but not including, the next range key of the same class. A range key may for example be a date. The range key needs to be comparable to other range keys, giving a result of being larger, smaller or equal. So all range keys of the same picture map need to be of the same type, for example a date. This is how you can assign the same picture object to all nodes that hold a date which is equal or larger than the one in the key passed but which is smaller than the one in the key succeeding.
If there is no key succeeding, any value equal or larger than the key passed will belong to the range.

Declaration
void addRange (java.lang.Comparable key, de.netronic.common.intface.NeIPicture value)

	Data Type	Description
Parameter		
key	java.lang.Comparable	Object to hold the key.
value	de.netronic.common.intface.NeIPicture	Picture to be allocated to the range key.
Return Value	void	

getPicture

Method of NePictureMap

This method lets you retrieve the picture object allocated to the key. Single entries are given priority over ranges.

Declaration
de.netronic.common.intface.NeIPicture getPicture (java.lang.Object key)

	Data Type	Description
Parameter		
key	java.lang.Object	Key the picture object of which is to be retrieved.
Return Value	de.netronic.common.intface.NeIPicture	Picture object returned

iteratePictures

Method of NePictureMap

This method returns an iterator object of all picture objects.

Declaration
java.util.Iterator iteratePictures ()

	Data Type	Description
Return Value	java.util.Iterator	Iterator object returned

removeEntry

Method of NePictureMap

This method lets you remove a discrete value from the map, as well as the picture object allocated.

Declaration
void removeEntry (java.lang.Object key)

	Data Type	Description
Parameter		
key	java.lang.Object	Key to be removed.
Return Value	void	

removePropertyChangeListener

Method of NePictureMap

This method lets you remove a listener for change events in the map object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener listener)

	Data Type	Description
Parameter		
listener	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

removeRange

Method of NePictureMap

This method lets you remove a range value from the map.

Declaration
void removeRange (java.lang.Comparable key)

	Data Type	Description
Parameter		
key	java.lang.Comparable	Key to be removed.
Return Value	void	

10.43 NePictureStack

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIPicture**

A picture stack (class NePictureStack) is an impementation of NeIPicture, by which you can pile picture objects in the third (spacial) dimension. The picture objects may hide one another fully or partially. A picture stack has a vertical and horizontal size that derives from the largest horizontal and vertical extent present among the pictures.

Properties to Handle the Picture Stack

HorizontalPictureFillMode	Fill mode for the horizontal direction of the picture stack
VerticalPictureFillMode	Fill mode for the vertical direction of the picture stack

Methods to Handle the Picture Stack

addPicture(...)	Adds a picture to the existing picture stack.
getPicture(...)	Retrieves the picture at the index specified
removePicture(...)	Removes the specified picture from the picture stack.

Constructors of the Class

NePictureStack

Constructor of NePictureStack

Generates an empty picture stack.

Declaration
NePictureStack ()

Properties of the Class

HorizontalPictureFillMode

Read Only Property of NePictureStack

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the picture stack fills the available area in horizontal direction. The picture stack in this case acts like a picture object (class NePicture). Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the ultimate left position and equals the constant LEFT; +100 describes the ultimate right position and equals the constant RIGHT.

Possible Values	Description
AUTO	The stack will tile the available space in horizontal direction until the space is filled.
CENTER	Within the available space of the layer the picture stack is placed in the horizontal center position (value: 0).
LEFT	Within the available space the picture stack is placed in the ultmate left position (value: -100).
RIGHT	Within the available space the picture stack is placed in the ultimate right position (value: +100).
STRETCH	The available space is filled in horizontal direction by stretching the picture stack.
TILE	The available space is tiled in horizontal direction by the pictures stack.

Accessing Methods
int getHorizontalPictureFillMode()

VerticalPictureFillMode

Read Only Property of NePictureStack

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the pictures stack fills the available area in vertical direction. The picture stack in this case acts like a picture object (class NePicture). Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the ultimate top position and equals the constant TOP; +100 describes the ultimate bottom position and equals the constant BOTTOM.

Possible Values	Description
AUTO	The stack will tile the available space in horizontal direction until the space is filled.
BOTTOM	Within the available space the picture stack is placed in the bottom position (value: +100).
CENTER	Within the available space the picture stack is placed in the center position (value: 0).
STRETCH	The available space is filled in vertical direction by stretching the picture stack.
TILE	The available space is tiled in vertical direction by the picture stack.
TOP	Within the available space the picture stack is placed in the ultimate position (value: -100).

Accessing Methods
int getVerticalPictureFillMode()

Methods of the Class

addPicture

Method of NePictureStack

This method adds another picture object to the existing list of pictures.

Declaration
void addPicture (NeIPicture thePicture)

	Data Type	Description
Parameter		
thePicture	NeIPicture	Picture object to be added.
Return Value	void	

Also see removePicture

getPicture

Method of NePictureStack

This method retrieves the picture object at the specified index.

Declaration
NeIPicture getPicture (int index)

	Data Type	Description
Parameter		
index	int	Index of the list of picture objects in the stack
Return Value	NeIPicture	Picture object retrieved

removePicture

Method of NePictureStack

This method removes a picture object from the list of pictures in the stack.

Declaration
void removePicture (NeIPicture thePicture)

	Data Type	Description
Parameter		
thePicture	NeIPicture	Picture object to be removed.
Return Value	void	

Also see addPicture

10.44 NePictureStripe

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIPicture**

A picture stripe is an implementation of NeIPicture, by which you can combine several NeIPicture objects into a horizontal or vertical picture stripe.
The stripe as a whole can be used as a NeIPicture object, for example to be inserted in a layer or a table column. Since a picture stripe itself is an NeIPicture object, picture stripes can be nested.
The features of a picture stripe result from the features of the pictures that it contains: The preferred height of a horizontally arranged picture stripe equals the preferred height of the tallest picture; the preferredWidth results from the total width of all pictures contained. In picture stripes of vertical arrangement, the peferred height results from the total height of all pictures contained; the preferred width equals the preferred width of the widest picture.

Properties to Design the Picture Stripe

ScalingDownProportional	Proportional downscaling of the pictures in case of space deficiency in the stripe
SeparationLines	Separation lines between the pictures of a picture stripe

Methods to Design the Picture Stripe

addPicture(...)	Adds a picture to the picture stripe and allows for excess space rating.
addPicture(...)	Adds a picture to the picture stripe. No space rating.

hasSeparationLines(...)	Separation lines between the pictures of a picture stripe
setSeparationLines(...)	Separation lines between the pictures of a picture stripe

Constructors of the Class

NePictureStripe

Constructor of NePictureStripe

The constructors let you create a picture stripe object vertically or horizontally composed from other pictures.

Declaration
NePictureStripe (boolean vertical)

Parameter	Data Type	Description
vertical	boolean	If this parameter is set to **true**, the picture stripe will be composed vertically; on **false**, pictures will be added in horizontal direction.

Properties of the Class

ScalingDownProportional

Property of NePictureStripe

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**false**

This property lets you downscale the pictures in a picture stripe in case of space deficiency. You can choose proportional downscaling (**true**), which is calculated from the ratio between the available space and the total of all preferred sizes. Alternatively (**false**), the picture is diminished by the space calculated from the size of missing space and the weight (see **addPicture**) of the picture. So setting the weight to 0.0 prevents a picture from being displayed in a size smaller than its preferred size, if pictures are present that have weights of higher values.

Accessing Methods
void setScalingDownProportional (boolean newValue)
boolean isScalingDownProportional ()

SeparationLines

Property of NePictureStripe

Typ	**boolean[]**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Values	**false**

This property lets you draw a line between the picture objects of a picture stripe. If more pictures exist than fields of the array of this property, the value at the highest index will be used for excess pictures. This way, by setting an array of a single value, i.e. separationLines[0] to **true**, you can very easily set a separation line for all pictures in the picture stripe.

Accessing Methods
void setSeparationLines (integer index, boolean newValues)
void setSeparationLines (boolean[] newValue)
boolean hasSeparationLines (integer index)
boolean[] hasSeparationLines ()

Methods of the Class

addPicture

Method of NePictureStripe

This method lets you add a picture to the picture stripe and add a weight parameter. It distributes excess space in the picture stripe among the pictures by adding all weight parameters and allocating a proportional share of space to each picture.

Declaration
void addPicture (float weight, NeIPicture thePicture)

	Data Type	Description
Parameter		
weight	float	Weight or rating factor for the distribution of excess or deficient space among the pictures. Excess or deficient space is distributed proportionally acording to the value of this factor to the picture objects.
thePicture	NeIPicture	Picture object to be added.
Return Value	void	

Also see addPicture

addPicture

Method of NePictureStripe

This method lets you add a picture to the picture stripe. When excess space in the stripe is distributed, the picture added will be considered by a weight factor of 1.

Declaration
void addPicture (NeIPicture thePicture)

	Data Type	Description
Parameter		
thePicture	NeIPicture	Picture object to be added.
Return Value	void	

Also see addPicture

hasSeparationLines

Method of NePictureStripe

This method lets you retrieve, whether a separation line was set after a picture object at the index specified.

Declaration
boolean hasSeparationLines (int index)

	Data Type	Description
Parameter		
index	int	Index of the picture, of which the separation line is retrieved.
Return Value	boolean	**False**: no separation line exists; **true**: a separation line exists.

Also see setSeparationLines

setSeparationLines

Method of NePictureStripe

This method lets you set a separation line after a picture object in the picture stripe.

Declaration
boolean setSeparationLines (int index)

	Data Type	Description
Parameter		
index	int	Index of the picture, after which a separation line shall be drawn.
Return Value	boolean	**False**: no separation line will be drawn; **true**: a seapration line will be drawn.

Also see hasSeparationLines

10.45 NeRelativeValueReference

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**

Implements **de.netronic.common.intface.NeIValueReference**

This class offers methods to handle a relative date attribute.

Properties to Handle the RelativeReferenceValue Object

AttributeName	Name of the attribute that holds the relative value.
BaseValueReference	ValueReference object that holds the basic date.
Factor	Factor, by which the duration is multipied.
Offset	Value to be added
ProfileBased	Date based on a profile?
ReferenceValueReference	ValueReference object that holds the reference date.
UnitString	Unit of the relative value.

Methods to Handle the RelativeReferenceValue Object

addPropertyChangeListener(...)	Adds a listener for change events in the NeValueReference object.
removePropertyChangeListener(...)	Removes a listener for change events in the NeSumValueReference object.

Constructors of the Class

NeRelativeValueReference

Constructor of NeRelativeValueReference

This constructor lets you generate an object to handle a relative date attribute. It derives from of a referring date (referenceValueReference) which relates to a base date (baseValueReference) and results in, for example. 20%.

Declaration
NeRelativeValueReference (java.lang.String attributeName, double factor, de.netronic.common.intface.NeIValueReference baseRef, de.netronic.common.intface.-NeIValueReference refValueRef)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute that holds the relation value.
factor	double	Factor, by which the value of the attribute is multiplied.
baseRef	de.netronic.common.intface.-NeIValueReference	ValueReference object that holds the basic date. Its value will be set against the value of **refValueRef** by using the relation defined in the attribute **attributeName**.
refValueRef	de.netronic.common.intface.-NeIValueReference	ValueReference object that holds the reference date.

NeRelativeValueReference

Constructor of NeRelativeValueReference

This constructor lets you generate an object to handle a relative date attribute. It derives from of a referring date (referenceValueReference) which relates to a base date (baseValueReference) and results in, for example. 20%.

Declaration
NeRelativeValueReference (java.lang.String attributeName)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute that holds the relation value.

Properties of the Class

AttributeName

Property of NeRelativeValueReference

Typ **java.lang.String**
Bound **no**
Vetoable **no**
Exposure Level **regular**

Name of an attribute that holds the relative value. The value of this attribute (for example 30) represents the relation (30 %) between the two values of BaseValueReference and ReferenceValueReference.

Accessing Methods
void setAttributeName (java.lang.String newValue)
java.lang.String getAttributeName ()

Also see BaseValueReference
ReferenceValueReference

BaseValueReference

Property of NeRelativeValueReference

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

ValueReference object that holds the basic date. Its value will be set against the value of **ReferenceValueReference** by using the relation defined in the attribute **AttributeName**.

Accessing Methods
void setBaseValueReference (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getBaseValueReference ()

Also see AttributeName
ReferenceValueReference

Factor

Property of NeRelativeValueReference

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1.0**

By this property you can set or retrieve the factor that the duration is multiplied by. It serves for example to transform the duration into a percentage.

Accessing Methods
void setFactor (double newValue)
double getFactor ()

Offset

Property of NeRelativeValueReference

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0.0**

By this property you can set or retrieve a value to be added to NeRelativeValueReference.

Accessing Methods
void setOffset (double newValue)
double getOffset ()

ProfileBased

Property of NeRelativeValueReference

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

By this property you can set or retrieve whether the result should consider the profile object assigned to the entity.

Accessing Methods
void setProfileBased (boolean newValue)
boolean isProfileBased ()

ReferenceValueReference

Property of NeRelativeValueReference

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

ValueReference object that holds the reference date.

Accessing Methods
void setReferenceValueReference (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getReferenceValueReference ()

UnitString

Read Only Property of NeRelativeValueReference

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the unit of the relative value resulting.

Accessing Methods
java.lang.String getUnitString()

Methods of the Class

addPropertyChangeListener

Method of NeRelativeValueReference

This method adds a listener for change events in the NeValueReference object. The listener will be informed each time a property has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

removePropertyChangeListener

Method of NeRelativeValueReference

Removes an existing listener for change events in the NeSumValueReference object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

10.46 NeRowInteractionAdapter

Belongs to VariousClasses

Package name **de.netronic.common.event**
Implements **de.netronic.common.event.NeRowInteractionListener**

This class is an adapter class for the interface NeRowInteractionListener. Its methods are empty. This class exists as convenience for creating listener objects.

10.47 NeRowInteractionEvent

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **java.util.EventObject**

Event which is created when moving table rows.

Methods of the Class

getDraggedEntity

Method of NeRowInteractionEvent

Retrieves the NeIEntity, which has been dragged.

Declaration
de.netronic.common.intface.NeIEntity getDraggedEntity ()

	Data Type	Description
Return Value	de.netronic.common.intface.NeIEntity	The NeIEntity, which has been dragged.

Example Code

```
myDraggedEntity = theEvent.getDraggedEntity();
```

getDropBefore

Method of NeRowInteractionEvent

The mouse pointer resides in the upper (true) or lower (false) half of the current table row.

Declaration
boolean getDropBefore ()

	Data Type	Description
Return Value	boolean	The mouse pointer resides in the upper (true) or lower (false) half of the current table row.

Example Code

```
boolean myDropBefore = theEvent.getDropBefore();
```

getHoveredEntity

Method of NeRowInteractionEvent

Retrieves the entity which is being hovered by the mouse.

Declaration
de.netronic.common.intface.NeIEntity getHoveredEntity ()

	Data Type	Description
Return Value	de.netronic.common.intface.NeIEntity	The entity which is being hovered by the mouse.

Example Code

```
myHoveredEntity = theEvent.getHoveredEntity();
```

10.48 NeRowInteractionListener

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **java.util.EventListener**

This is the listener interface for receiving row interaction events. The class that is interested in processing an row interaction event either implements this interface, (including the methods it contains), or extends the NeRowInteractionAdapter class (overriding only the methods of interest). The listener object is then registered with

the object using the object's addListener method. A row interaction event is generated after moving a table row.

Methods of the Interface

onRowMove

Method of NeRowInteractionListener

This method is triggered when an object is being modified and delivers an event that allows for a veto exception. The object value of the event represents the object to be modified. The map of values contains the modified values for the object. The semantics of the key-value pairs in the map depend on the type of object created. The listener can throw an NeVetoException to suppress the modification of the object.

Declaration
void onRowMove ()

	Data Type	Description
Return Value	void	

onRowPhantomModify

Method of NeRowInteractionListener

This method is triggered when a phantom is about to change and delivers an event that allows for a veto exception. The listener can throw an NeVetoException to suppress the modification of the phantom.

Declaration
void onRowPhantomModify ()

	Data Type	Description
Return Value	void	

prepareRowMoveInteraction

Method of NeRowInteractionListener

This method is triggered when the modification of an object is about to happen, that is, when the cursor has changed its shape, and delivers an event that allows for a veto exception. The listener can throw an NeVetoException to suppress the modification.

Declaration
void prepareRowMoveInteraction ()

	Data Type	Description
Return Value	void	

rowMoved

Method of NeRowInteractionListener

This method is triggered after an object has been modified. The object value of the event passed holds the modified object, the map of values contains the original values of the object that have been modified. The semantics of the key/value pairs in the map depend on the type of object modified.

Declaration
void rowMoved ()

	Data Type	Description
Return Value	void	

10.49 NeSimpleDateFormat

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **java.text.SimpleDateFormat**

Beyond the features of class java.text.SimpleDateFormat this class offers a way to format and parse labels for quarters.

Constructors of the Class

NeSimpleDateFormat

Constructor of NeSimpleDateFormat

Constructs a NeSimpleDateFormat using the given pattern.

Declaration
NeSimpleDateFormat (java.lang.String pattern)

Parameter	Data Type	Description
pattern	java.lang.String	A pattern similar to the one used in java.text.SimpleDateFormat. In addition to the wellknown letters the letters below are handled: 'Q' - number of quarter as Arabic numerals 'R' - number of quarter as Roman numerals

10.50 NeSumValueReference

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIValueReference**

This class offers methods to handle a composed date attribute of an entity.

Properties to Handle the SumValueReference Object

AttributeName	Name of the attribute that holds the duration to be added.
BaseValueReference	ValueReference object that holds the basic date.
Factor	Factor, by which the duration is multipied.
Offset	Value to be added
ProfileBased	Date based on a profile?

UnitString	Unit of the duration to be added.

Methods to Handle the SumValueReference Object

addPropertyChangeListener(...)	Adds a listener for change events in the NeValueReference object.
removePropertyChangeListener(...)	Removes a listener for change events in the NeSumValueReference object.

Constructors of the Class

NeSumValueReference

Constructor of NeSumValueReference

This constructor lets you generate an object to handle a composed date attribute. The composed date consists of a basic date delivered by the NeValueReference object, to which a duration multiplied by a factor is added.

Declaration
NeSumValueReference (java.lang.String attributeName, double factor, de.netronic.common.intface.NeIValueReference baseRef)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute that holds the duration to be added.
factor	double	Factor, by which the value of the attribute is multiplied.
baseRef	de.netronic.common.intface.NeIValue Reference	ValueReference object that holds the basic date. Its value will be set against the value of **refValueRef** by using the relation defined in the attribute **attributeName**.

NeSumValueReference

Constructor of NeSumValueReference

This constructor lets you generate an object to handle a composed date attribute. The composed date consists of a basic date delivered by the NeValueReference object, to which a duration multiplied by a factor is added.

Declaration
NeSumValueReference (java.lang.String attributeName)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute that holds the duration to be added.

Properties of the Class

AttributeName

Property of NeSumValueReference

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of an attribute that holds the duration to be added.

Accessing Methods
void setAttributeName (java.lang.String newValue)
java.lang.String getAttributeName ()

Also see UnitString

BaseValueReference

Property of NeSumValueReference

Typ	**de.netronic.common.intface.NeIValueReference**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

ValueReference object that holds the basic date.

Accessing Methods
void setBaseValueReference (de.netronic.common.intface.NeIValueReference newValue)
de.netronic.common.intface.NeIValueReference getBaseValueReference ()

Factor

Property of NeSumValueReference

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1.0**

Factor that the duration is multiplied by. It serves for example to transform the duration into a percentage.

Accessing Methods
void setFactor (double newValue)
double getFactor ()

Offset

Property of NeSumValueReference

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0.0**

By this property you can set or retrieve a value to be added to NeSumValueReference.

Accessing Methods
void setOffset (double newValue)
double getOffset ()

ProfileBased

Property of NeSumValueReference

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**true**

By this property you can set or retrieve whether the composition of the date should consider the profile object assigned to the entity.

Accessing Methods
void setProfileBased (boolean newValue)
boolean isProfileBased ()

UnitString

Read Only Property of NeSumValueReference

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

By this property you can retrieve the unit of the duration that is to be added to the basic date.

Accessing Methods
java.lang.String getUnitString()

Also see AttributeName

Methods of the Class

addPropertyChangeListener

Method of NeSumValueReference

This method adds a listener for change events in the NeValueReference object. The listener will be informed each time a property has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

removePropertyChangeListener

Method of NeSumValueReference

Removes an existing listener for change events in the NeSumValueReference object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

10.51 NeSymbol

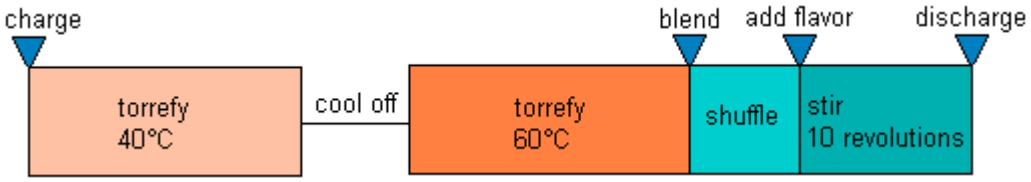
Belongs to VariousClasses

Package name **de.netronic.common.beanbase**

Implements	**de.netronic.common.intface.NeIPicture**
I	**de.netronic.common.intface.NeILabel**

Objects of the NeSymbol class are vector based symbols that for example are used in table fields and in nodes.

Nodes may consist of two different types of graphical elements: "layers" and "decorations". The latter can be positioned on layers. Decorations are represented by annotations, by symbols or by picture objects. There are two different types of decorations, the NeIPicture type and the NeILabel type. So symbols can be used as both, as an NeIPicture type or an NeILabel type. Below, symbols are placed above the layers.



Please also see the classes NeAnnotation and NePicture

Properties to Influence the Appearance of the Symbol

AreaStyle	Area style of the symbol
BackgroundStyle	Background color of the symbol
BorderStyle	Line type of the surrounding border line of the symbol
Extent	Extent of the positioning rectangle of the symbol
HorizontalPictureFillMode	Fill mode for the horizontal direction of the symbol
LineStyle	Line type of the inner lines of the symbol
Offset	Offset between the reference points
RefPointPosition	Reference point of the symbol
SymbolNumber	Number of the symbol
SymbolTable	Number of the table
VerticalPictureFillMode	Fill mode for the vertical direction of the symbol

Constructors of the Class

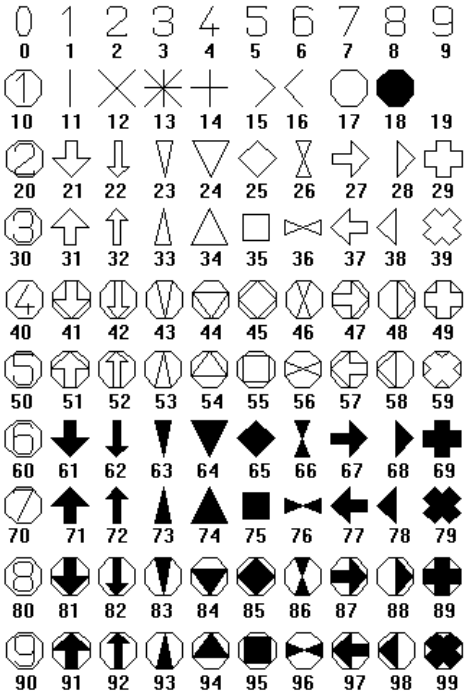
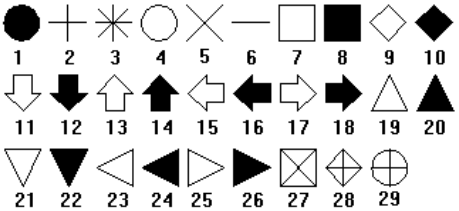
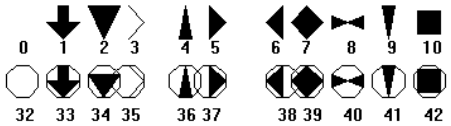
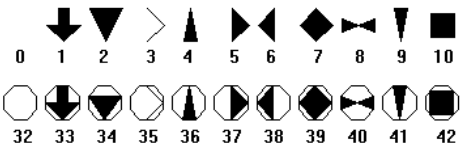
The constructors let you create a symbol object or an empty object that needs to be filled with a symbol later on.

NeSymbol

Constructor of NeSymbol

This constructor lets you generate a symbol.

Declaration
NeSymbol (int symbolNumber, int symbolTable)

Parameter	Data Type	Description
symbolNumber	int	Number of the symbol
symbolTable	int	The default value of the symbol table is SYMBOL_TABLE_VC_WITHOUT_OFFSET .
	Possible Values:	
	SYMBOL_TABLE_GR_BAR	
	SYMBOL_TABLE_GR_MMG	
	SYMBOL_TABLE_VC_- WITH_OFFSET	
	SYMBOL_TABLE_VC_- WITHOUT_OFFSET	

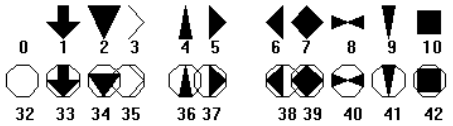
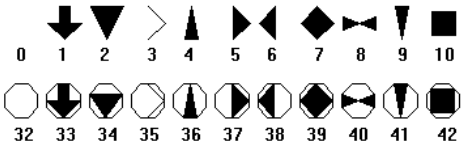
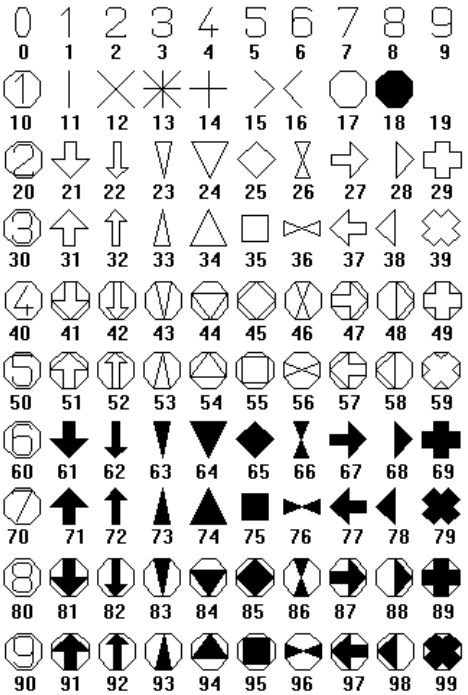
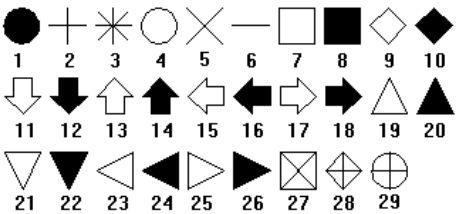
NeSymbol

Constructor of NeSymbol

This constructor lets you generate a symbol.

Declaration
NeSymbol (int symbolNumber)

Parameter	Data Type	Description

symbolNumber	int	Number of the symbol: Table SYMBOL_TABLE_VC_WITH_OFFSET:  Table SYMBOL_TABLE_VC_WITHOUT_OFFSET:  Table SYMBOL_TABLE_GR_BAR:  Table SYMBOL_TABLE_GR_MMG:  	

NeSymbol

Constructor of NeSymbol

This constructor lets you generate an empty symbol object.

Declaration
NeSymbol ()

Properties of the Class

AreaStyle

Property of NeSymbol

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**color of the LineStyle**

Area style of the symbol

Accessing Methods
void setAreaStyle (java.awt.Color newValue)
java.awt.Color getAreaStyle ()

BackgroundStyle

Property of NeSymbol

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Background color of the symbol. You can use a simple color object or a n object of the class NeAreaStyle.

Accessing Methods
void setBackgroundStyle (java.awt.Color newValue)
java.awt.Color getBackgroundStyle ()

BorderStyle

Property of NeSymbol

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Line type of the surrounding border line of the symbol

Accessing Methods
void setBorderStyle (java.awt.Color newValue)
java.awt.Color getBorderStyle ()

Extent

Property of NeSymbol

Typ	**java.awt.Dimension**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

If the symbol is used as an NeILabel object, you can define or retrieve the size of the symbol via this property.
If the symbol is used as a NeIPcture object, this property assigns it a favorite size. If the fill mode was set to **AUTO** or **TILE**, the symbol will be copied as many times in vertical and in horizontal direction as is needed to fill the positioning rectangle, the size of which is controlled by the layer. Only complete copies of the symbol are drawn.
If you do not assign a value here, the symbol will be stretched to fit into the rectangle.
Unit: 1/100 mm.

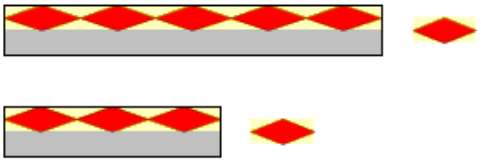
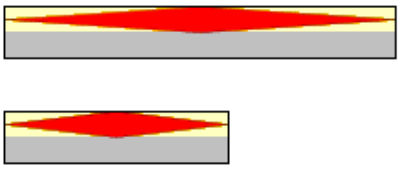
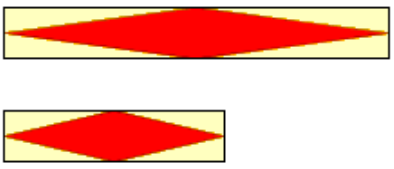
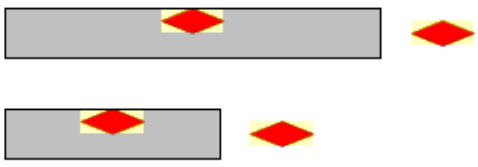
Accessing Methods
void setExtent (java.awt.Dimension newValue)
java.awt.Dimension getExtent ()

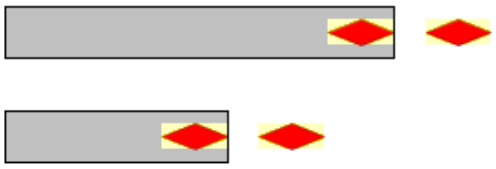
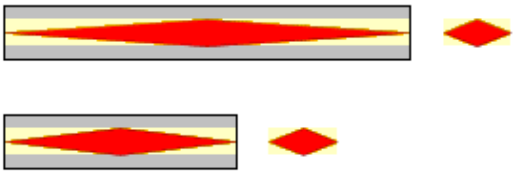
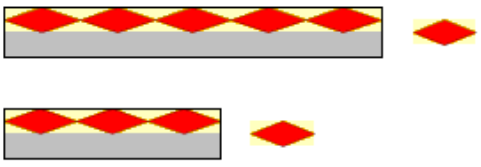
HorizontalPictureFillMode

Property of NeSymbol

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the symbol fills the available area in horizontal direction, if the symbol is used as an NeIPicture. Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the ultimate left position and equals the constant LEFT; +100 describes the ultimate right position and equals the constant RIGHT. In contrast to an annotation and a picture object, in a symbol the background and the foreground are combined and do not behave separately. If the symbol is used as an NeILabel, this property will not apply.

Possible Values	Description
AUTO	If by the property **Extent** a width was specified, the symbol will be tiled in horizontal direction until the space is filled. If no horizontal extend was specified, the symbol will be stretched in horizontal direction.  In the picture above a width and a heigth were specified, which allows the symbol to be displayed also outside the layer. Within the layer, the available area is filled completely with copies of the size specified in horizontal direction. The vertical fill mode was set to **TOP**.  In the picture above only a height was specified; the missing dimension does not allow the symbol to be displayed outside the layer, so it is not displayed beyond its right border. Within the layer the symbol is stretched in the missing dimension until it covers the width of the layer completely.  If for no dimension a size is specified, the symbol is stretched until it fills the available space completely.
CENTER	Within the available space of the annotation the symbol is placed in the center position (value: 0).  The vertical fill mode was set to **TOP**.
LEFT	Within the available space the symbol is placed in

	the leftmost position (value: -100).  The vertical fill mode was set to BOTTOM.
RIGHT	Within the available space the symbol is placed in the rightmost position (value: +100).  The vertical fill mode was set to **CENTER**.
STRETCH	The available space is filled in horizontal direction by stretching the symbol.  The vertical fill mode was set to **CENTER**.
TILE	The available space is tiled in horizontal direction by the symbol, if a width was set by the property **Extent**.  The vertical fill mode was set to **TOP**.

Accessing Methods
void setHorizontalPictureFillMode (int newValue)
int getHorizontalPictureFillMode ()

Also see VerticalPictureFillMode

LineStyle

Property of NeSymbol

Typ	**java.awt.Color**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**Color.black**

Line type of the inner lines of the symbol. The inner lines of a symbol comprise all lines except for the surrounding border line.

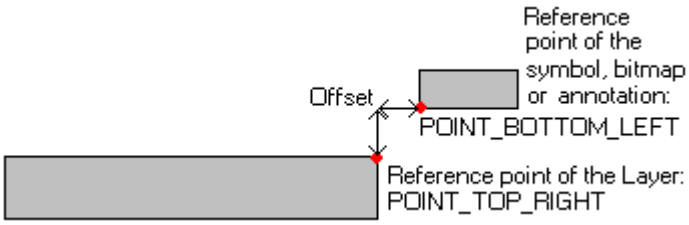
Accessing Methods
void setLineStyle (java.awt.Color newValue)
java.awt.Color getLineStyle ()

Offset

Property of NeSymbol

Typ	**java.awt.Point**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

When a symbol is used as an NeILabel, and for example is positioned at a layer, the reference point of the layer and the reference point of the symbol are superimposed. This property lets you add an offset by which their position differs, or lets you retrieve the offset assigned.



Accessing Methods
void setOffset (java.awt.Point newValue)
java.awt.Point getOffset ()

Also see RefPointPosition

RefPointPosition

Property of NeSymbol

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**POSITION_CENTER_CENTER**

When a symbol is positioned on an object, the reference point of the object and the reference point of the symbol are superimposed. This property lets you set or retrieve the reference point of the symbol.

Possible Values	Description
POSITION_BOTTOM_CENTER	The reference point is situated in the center of the bottom border.
POSITION_BOTTOM_LEFT	The reference point is situated at the bottom left corner.
POSITION_BOTTOM_RIGHT	The reference point is situated at the bottom right corner.
POSITION_CENTER_CENTER	The reference point is situated in the vertical and horizontal center.
POSITION_CENTER_LEFT	The reference point is situated in the vertical center on the left hand side.
POSITION_CENTER_RIGHT	The refernce point is situated in the vertical center on the right hand side.
POSITION_TOP_CENTER	The reference point is situated in the center of the top border.
POSITION_TOP_LEFT	The refrence point is situated at the top left corner.
POSITION_TOP_RIGHT	The refernce point is situated at the top right corner.

Accessing Methods
void setRefPointPosition (int newValue)
int getRefPointPosition ()

Also see Offset

SymbolNumber

Property of NeSymbol

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Symbols are kept in four different tables, where they are identified by a number. Therefore, for their indentification, symbols apart from their number also need a table number.

Table SYMBOL_TABLE_VC_WITH_OFFSET:

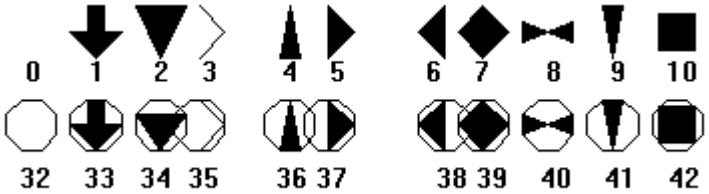


Table SYMBOL_TABLE_VC_WITHOUT_OFFSET:

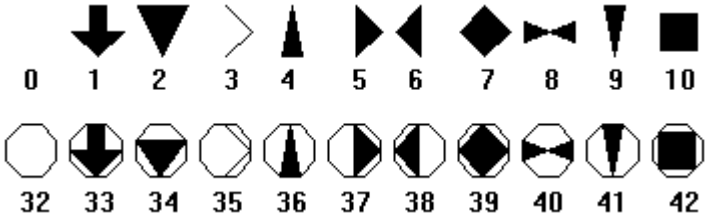


Table SYMBOL_TABLE_GR_BAR:

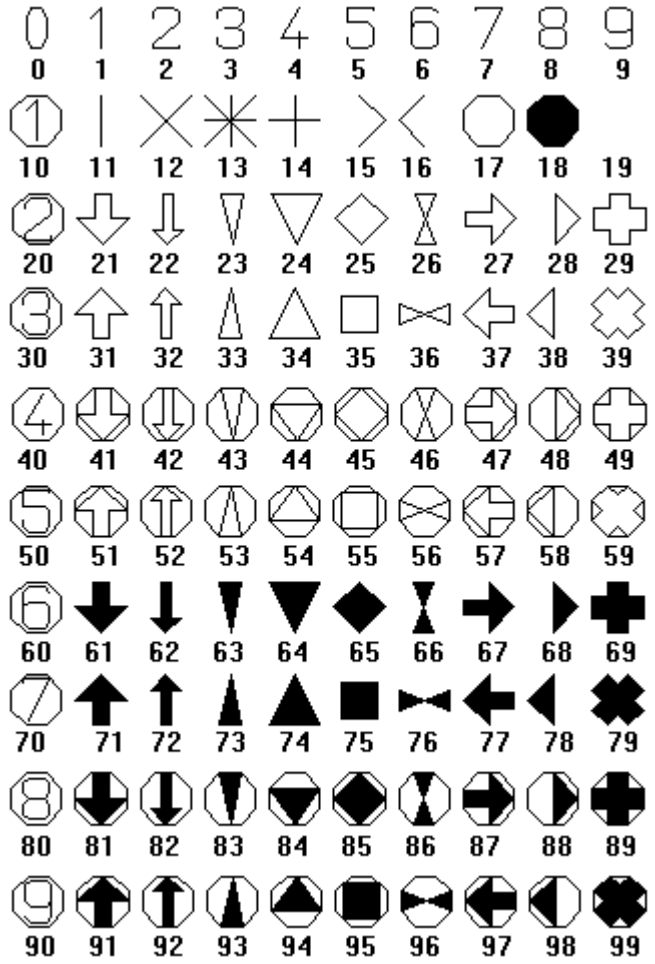
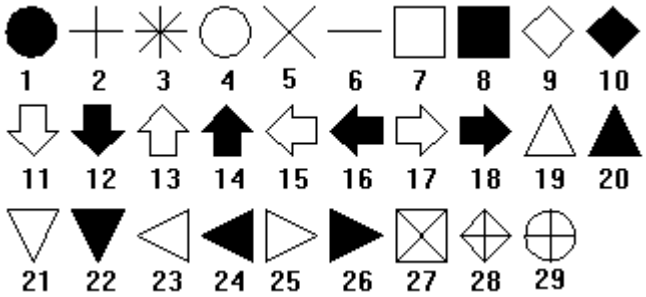


Table SYMBOL_TABLE_GR_MMG:



Accessing Methods
void setSymbolNumber (int newValue)
int getSymbolNumber ()

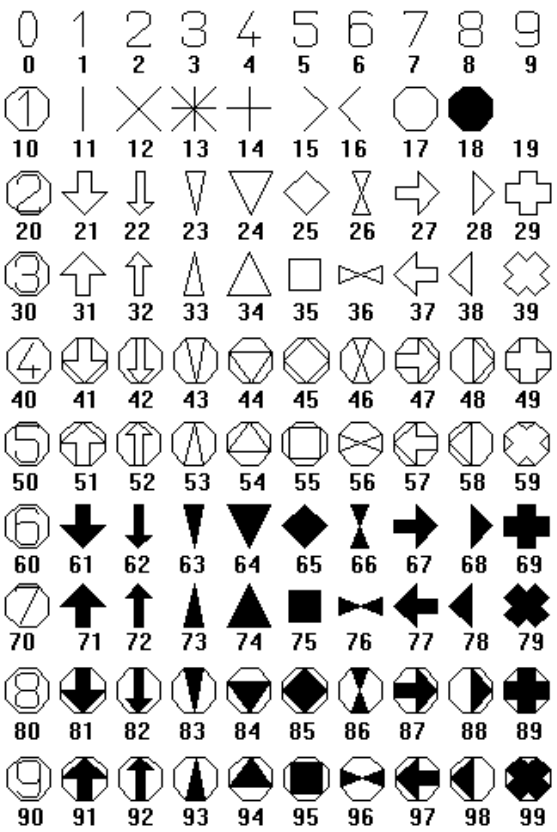
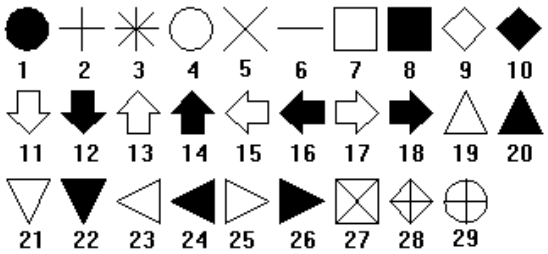
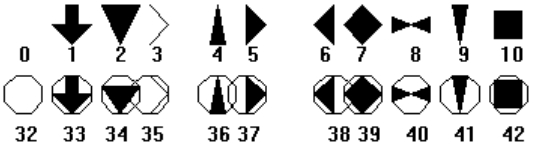
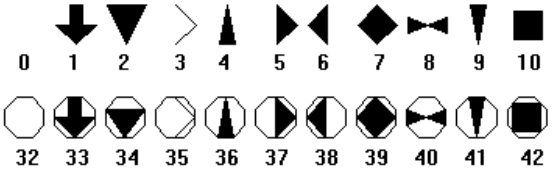
Also see SymbolTable

SymbolTable

Property of NeSymbol

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Symbols are kept in four different tables, where they are identified by a number. Therefore, for their indentification apart from their number, symbols also need the name of the table that they are to be taken from.

Possible Values	Description
SYMBOL_TABLE_GR_BAR	
SYMBOL_TABLE_GR_MMG	
SYMBOL_TABLE_VC_WITH_OFFSET	
SYMBOL_TABLE_VC_WITHOUT_OFFSET	

Accessing Methods
void setSymbolTable (int newValue)
int getSymbolTable ()

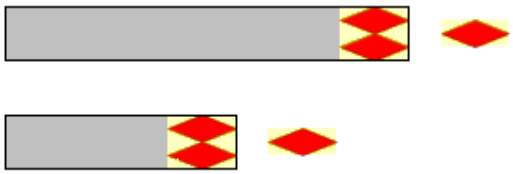
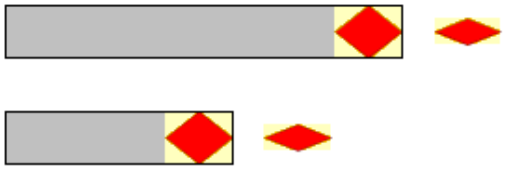
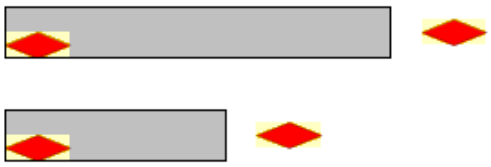
Also see SymbolNumber

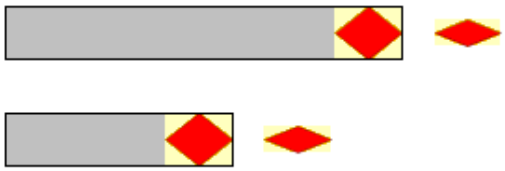
VerticalPictureFillMode

Property of NeSymbol

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**AUTO**

This property lets you set or retrieve the mode according to which the symbol fills the available area in vertical direction, if the symbol is used as an NeIPicture. Apart from the constants listed below, any integer value from {-100...+100} is allowed. -100 describes the ultimate top position and equals the constant TOP; +100 describes the ultimate bottom position and equals the constant BOTTOM. In contrast to an annotation and a picture object, in a symbol the background and the foreground are combined and do not behave separately. If the bitmap is used as an NeILabel, this property will not apply.

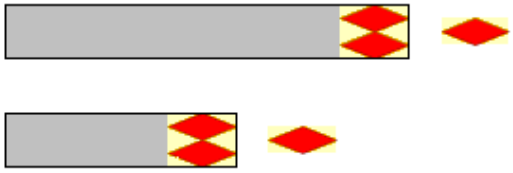
Possible Values	Description
AUTO	If by the property **Extent** a height was specified, the symbol will be tiled in vertical direction. If no height was specified, the symbol will be stretched in vertical direction.  Above: A height was specified, therefore the symbol is tiled.  Above: No height was specified, therefore the symbol was stretched to match the height of the layer.
BOTTOM	Within the available space the symbol is placed in the bottom position (value: +100).  The horizontal fill mode was set to **LEFT**.
CENTER	Within the available space the symbol is placed in the center position (value: 0).  The horizonatl fill mode was set to **RIGHT**.
STRETCH	The available space is filled in vertical direction by stretching the symbol.



The horizontal fill mode was set to **RIGHT**.

TILE

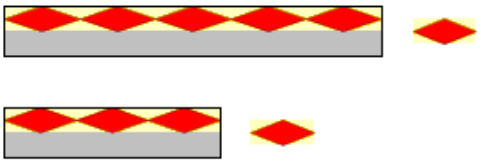
The available space is tiled in vertical direction by the symbol, if a height was set by the property **Extent**.



The horizontal fill mode was set to **RIGHT**.

TOP

Within the available space the symbol is placed in the topmost position (value: -100).



The horizontal fill mode was set to **TILE**.

Accessing Methods
void setVerticalPictureFillMode (int newValue)
int getVerticalPictureFillMode ()

Also see HorizontalPictureFillMode

10.52 NeTransaction

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**

Implements **de.netronic.common.intface.NelTransaction**

Abstract class containing methods that can be overwritten if they should do jobs different to their default jobs. The method **commit** also is abstract; in addition, it is empty. In order to make it work, it has to be overwritten.

Properties to Handle a Transaction

Id	Character string to identify the transaction
State	State of the transaction

Methods to Handle a Transaction

commit()	Empty method to be overwritten.
handleException(...)	Default implementation, may be overwritten.
rollback()	Default implementation, may be overwritten.
toString()	Default implementation, may be overwritten.

Constructors of the Class

NeTransaction

Constructor of NeTransaction

Constructor that lets you generate a transaction object that has an identification tag.

Declaration
NeTransaction (java.lang.String id)

Parameter	Data Type	Description
id	java.lang.String	Character string to identify the transaction

NeTransaction

Constructor of NeTransaction

Constructor that lets you generate an empty transaction object. Missing things such as the identification tag you can assign later on by methods and properties.

Declaration
NeTransaction ()

Properties of the Class

Id

Property of NeTransaction

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve a character string to identify the transaction.

Accessing Methods
void setId (java.lang.String newValue)
java.lang.String getId ()

State

Property of NeTransaction

Typ	**int**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

This property lets you set or retrieve the state of the transaction.

Possible Values	Description
COMMIT	NeITransaction.commit() is being performed
COMPLETED	NeITransaction.commit() was successfully finished.
FAILED	NeITransaction.commit() and NeITransaction.rollback() have failed.
NEW	New transaction
ROLLBACK	The NeITransaction.rollback() is being performed.
ROLLED_BACK	The(NeITransaction.rollback() was successfully finished.

Accessing Methods
void setState (int newValue)
int getState ()

Methods of the Class

commit

Method of NeTransaction

This method is an abstract method and it is empty. The method needs to be overwritten and implemented by the application programmer.

Declaration
boolean commit ()

	Data Type	Description
Return Value	boolean	

handleException

Method of NeTransaction

The default implementation of this method prints a stack trace and returns **false** if the methods **commit** or **rollback** have failed.

Declaration
void handleException (java.lang.Exception e, boolean committing)

	Data Type	Description
Parameter		
e	java.lang.Exception	Indicates the exception.
	Possible Values:	
	COMMIT	NeITransaction.commit() is being performed
	COMPLETED	NeITransaction.commit() was successfully finished.
	FAILED	NeITransaction.commit() and NeITransaction.rollback() have failed.
	NEW	New transaction
	ROLLBACK	The NeITransaction.rollback() is being performed.
	ROLLED_BACK	The(NeITransaction.rollback() was successfully finished.
committing	boolean	If the value equals **true**, the exception was thrown during the performance of the **commit** method; if it equals **false**, the exception was thrown during the **rollback**.
Return Value	void	

rollback

Method of NeTransaction

The default implementation of this method is invoked by the interface NeITransactionHandler to provide a roll back of the transaction if the commit method returns **false**, or if **rollbackTransaction** was called.

Declaration
boolean rollback ()

	Data Type	Description
Return Value	boolean	If the method returns **true**, the rollback was successfull, in case of **false** it was not.

toString

Method of NeTransaction

The default implementation of this method returns the transaction, its identification tag and its state as a character string.

Declaration
java.lang.String toString ()

	Data Type	Description
Return Value	java.lang.String	Character string returned

10.53 NeTransactionAdapter

Belongs to VariousClasses

Package name	**de.netronic.common.event**
Extends	**java.lang.Object**
Implements	**de.netronic.common.event.NeTransactionListener**

This class is an adapter class for the interface NeTransactionListener. Its methods are empty. This class exists as convenience for creating listener objects.

10.54 NeTransactionEvent

Belongs to VariousClasses

Package name	**de.netronic.common.event**
Extends	**java.util.EventObject**

This class contains methods to handle the events of transactions.

Properties to Handle Transaction Events

Source	Source by which the event was triggered
Transaction	Transaction started or finished

Constructors of the Class

By the constructors of this class you can generate events of transactions that transmit information on the source and the transaction itself.

NeTransactionEvent

Constructor of NeTransactionEvent

This constructor lets you generate an event that transmits information on the triggering source of the transaction and on the transaction started or finished.

Declaration
NeTransactionEvent (java.lang.Object source, de.netronic.common.intface.NeITransaction transaction)

Parameter	Data Type	Description
source	java.lang.Object	Source, by which the transaction was started or finished.
transaction	de.netronic.common.intface.NeITransaction	Transaction that was started or finished.

Properties of the Class

Source

Property of NeTransactionEvent

Typ	**java.lang.Object**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Source by which the event was triggered

Accessing Methods
void setSource (java.lang.Object newValue)
java.lang.Object getSource ()

Transaction

Property of NeTransactionEvent

Typ	**de.netronic.common.intface.NeITransaction**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Transaction started or finished

Accessing Methods
void setTransaction (de.netronic.common.intface.NeITransaction newValue)
de.netronic.common.intface.NeITransaction getTransaction ()

10.55 NeTransactionListener

Belongs to VariousClasses

Package name	**de.netronic.common.event**
Extends	**java.util.EventListener**

This is the listener interface for receiving transaction events. The class that is interested in processing a transaction event either implements this interface, (including the methods it contains), or extends the NeTransactionAdapter class (overriding only the methods of interest). The listener object will then be registered with the transaction using the transactionhandler's addListener method. A transaction event is generated at the start and at the finish of a transaction.

Event Methods for Transactions

transactionEnd(...)	Event on the finish of a transaction
transactionStart(...)	Event on the start of a transaction

Methods of the Interface

transactionEnd

Method of NeTransactionListener

This method is invoked on the finish of a transaction.

Declaration
void transactionEnd (de.netronic.common.event.NeTransactionEvent event)

	Data Type	Description
Parameter		
event	de.netronic.common.event.-NeTransactionEvent	Event received on the finish of a transaction.
Return Value	void	

transactionStart

Method of NeTransactionListener

This method is invoked on the start of a transaction.

Declaration
void transactionStart (de.netronic.common.event.NeTransactionEvent event)

	Data Type	Description
Parameter		
event	de.netronic.common.event.-NeTransactionEvent	Event triggered on the start of a transaction.
Return Value	void	

10.56 NeUserAction

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Extends **javax.swing.AbstractAction**
Implements **de.netronic.common.intface.NeIUserActionSource**

Contains the implementations of the interface NeIUserActionSource.

Properties for Internal Use

Suspended For internal use only

Methods for Internal Use

actionPerformed(...)	Internal method
fireOnUserAction(...)	Internal method
fireOnUserActionTriggerShow(...)	Internal method
internalActionPerformed(...)	Internal method
prepareTrigger(...)	Internal method
updateSuspendState()	

Methods to Handle the Listener

addUserActionListener(...)	Adds an NeUserActionListener.
removeUserActionListener(...)	Removes an NeUserActionListener

Constructors of the Class

NeUserAction

Constructor of NeUserAction

This constructor generates a user action, its name and a symbol for the button by which it is triggered.

Declaration
NeUserAction (java.lang.String commandKey, java.lang.String name, javax.swing.Icon icon)

Parameter	Data Type	Description
commandKey	java.lang.String	Name of the user action
name	java.lang.String	Name of the user action to be displayed on the button that triggers the user action.
icon	javax.swing.Icon	Icon of the user action to be displayed on the button that triggers the user action.

Properties of the Class

Suspended

Property of NeUserAction

Typ	**boolean**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

For internal use only

Accessing Methods
void setSuspended (boolean newValue)
boolean isSuspended ()

Methods of the Class

actionPerformed

Method of NeUserAction

For internal use only.

Declaration
void actionPerformed (java.awt.event.ActionEvent e)

	Data Type	Description
Parameter		
e	java.awt.event.ActionEvent	For internal use only.
Return Value	void	

addUserActionListener

Method of NeUserAction

This method lets you add a NeUserActionListener to the listener list.

Declaration
void addUserActionListener (de.netronic.common.event.NeUserActionListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeUserActionListener	Listener to be added.
Return Value	void	

Also see removeUserActionListener

fireOnUserAction

Method of NeUserAction

For internal use only.

Declaration
void fireOnUserAction (de.netronic.common.event.NeUserActionEvent e)

	Data Type	Description
Parameter		
e	de.netronic.common.event.-NeUserActionEvent	For internal use only.
Return Value	void	

fireOnUserActionTriggerShow

Method of NeUserAction

For internal use only.

Declaration
void fireOnUserActionTriggerShow (de.netronic.common.event.NeUserActionEvent e)

	Data Type	Description
Parameter		
e	de.netronic.common.event.-NeUserActionEvent	For internal use only.
Return Value	void	

internalActionPerformed

Method of NeUserAction

For internal use only.

Declaration
void internalActionPerformed (java.awt.event.ActionEvent e)

	Data Type	Description
Parameter		
e	java.awt.event.ActionEvent	For internal use only.
Return Value	void	

prepareTrigger

Method of NeUserAction

For internal use only.

Declaration
boolean prepareTrigger (javax.swing.AbstractButton trigger)

	Data Type	Description
Parameter		
trigger	javax.swing.AbstractButton	For internal use only.
Return Value	boolean	

removeUserActionListener

Method of NeUserAction

This method lets you remove a NeUserActionListener from the listener list.

Declaration
void removeUserActionListener (de.netronic.common.event.NeUserActionListener listener)

	Data Type	Description
Parameter		
listener	de.netronic.common.event.-NeUserActionListener	Listener to be removed.
Return Value	void	

Also see addUserActionListener

updateSuspendState

Method of NeUserAction

For internal use only

Declaration
boolean updateSuspendState ()

	Data Type	Description
Return Value	boolean	

10.57 NeUserActionAdapter

Belongs to VariousClasses

Package name **de.netronic.common.event**
Implements **de.netronic.common.event.NeUserActionExtendedListener**

This class is an adapter class for the interface NeUserActionExtendedListener. Its methods are empty. This class exists as convenience for creating listener objects.

Methods to Handle Events

onUserAction(...)	Method invoked on user action.
onUserActionTriggerShow(...)	Method invoked on display of a button or menu item for a user action.

Methods of the Interface

onUserAction

Method of NeUserActionListener

This method is invoked when a user action is triggered.

Declaration
void onUserAction (NeUserActionEvent e) throws NeVetoException

	Data Type	Description
Parameter		
e	NeUserActionEvent	Event triggered on the user's action
Return Value	void	

onUserActionTriggerShow

Method of NeUserActionListener

This method ist invoked when a menu item or button is displayed by which a user interaction can be triggered.

Declaration
void onUserActionTriggerShow (NeUserActionEvent e) throws NeVetoException

	Data Type	Description
Parameter		
e	NeUserActionEvent	Event triggered on the user's action
Return Value	void	

10.58 NeUserActionEvent

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **java.util.EventObject**

This class contains methods to handle the events of user actions.

Methods to Handle Action Events

getAction()	Retrieves the associated action object
getActionEvent()	Retrieves the action event
getButton()	Retrieves the button or menu item
getTarget()	Retrieves the target object of the action

Constructors of the Class

By the constructors of this class you can generate events of user actions, that transmit information on the source, the user action itself and the target of the user action.

NeUserActionEvent

Constructor of NeUserActionEvent

This constructor lets you generate an event that transmits information on the triggering source of the user action, on the action triggered and on the button by which the action was triggered.

Declaration
NeUserActionEvent (java.lang.Object source, javax.swing.Action theAction, javax.swing.AbstractButton theButton)

Parameter	Data Type	Description
source	java.lang.Object	Source by which the user action was triggered.
theAction	javax.swing.Action	Action that was triggered.
theButton	javax.swing.AbstractButton	Button by which the user action was triggered.

NeUserActionEvent

Constructor of NeUserActionEvent

This constructor lets you generate an event that transmits information on the triggering source of the user action, the action triggered, the button, by which the action was triggered plus the target that the user action is directed to.

Declaration

NeUserActionEvent (java.lang.Object source, javax.swing.Action theAction, javax.swing.AbstractButton theButton, java.lang.Object target)

Parameter	Data Type	Description
source	java.lang.Object	Source by which the user action was triggered.
theAction	javax.swing.Action	Action that was triggered.
theButton	javax.swing.AbstractButton	Button by which the user action was triggered.
target	java.lang.Object	Target of the user action.

NeUserActionEvent

Constructor of NeUserActionEvent

This constructor lets you generate an event that transmits information on the triggering source of the user action, on the action triggered, on the button, by which the action was triggered, on the target that the user action is directed to and on the event itself.

Declaration
NeUserActionEvent (java.lang.Object source, javax.swing.Action theAction, javax.swing.AbstractButton theButton, java.lang.Object target, java.awt.event.ActionEvent theEvent)

Parameter	Data Type	Description
source	java.lang.Object	Source by which the user action was triggered.
theAction	javax.swing.Action	Action that was triggered.
theButton	javax.swing.AbstractButton	Button by which the user action was triggered.
target	java.lang.Object	Target of the user action.
theEvent	java.awt.event.ActionEvent	Event to be generated.

Methods of the Class

getAction

Method of NeUserActionEvent

This method lets you retrieve the action object associated with the event.

Declaration
javax.swing.Action getAction ()

	Data Type	Description
Return Value	javax.swing.Action	Action returned

getActionEvent

Method of NeUserActionEvent

This method lets you retrieve the action event. For events of onUserActionTriggerShow null will be returned, for events of onUserAction null may be returned as well.

Declaration
java.awt.event.ActionEvent getActionEvent ()

	Data Type	Description
Return Value	java.awt.event.ActionEvent	Action event returned

getButton

Method of NeUserActionEvent

This method lets you retrieve the button or menu item which caused the event. If no button or menu item was involved, null will be returned.

Declaration
javax.swing.AbstractButton getButton ()

	Data Type	Description
Return Value	javax.swing.AbstractButton	Button or menu item returned

getTarget

Method of NeUserActionEvent

This method lets you retrieve the target of the action.

Declaration
java.lang.Object getTarget ()

	Data Type	Description
Return Value	java.lang.Object	Target object returned

10.59 NeUserActionExtendedListener

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **NeUserActionListener**

This listener interface extends the interface NeUserActionListener by the method onUserActionPerformed, which is called after a user action has been performed.

Methods of the Interface

onUserActionPerformed

Method of NeUserActionExtendedListener

This method is invoked after a user action was performed.

Declaration
void onUserActionPerformed ()

	Data Type	Description
Return Value	void	

10.60 NeUserActionListener

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **java.util.EventListener**

This is the listener interface for receiving userAction events. The class that is interested in processing a userAction event either implements this interface, (including the methods it contains), or extends the NeUserActionAdapter class

(overriding only the methods of interest). The listener object will then be registered with the userAction using the userAction's addListener method. A userAction event is generated when a userAction was performed.

Methods to Handle Events

onUserAction(...)	Method invoked on user action.
onUserActionTriggerShow(...)	Method invoked on display of a button or menu item for a user action.

Methods of the Interface

onUserAction

Method of NeUserActionListener

This method is invoked when a user action is triggered.

Declaration
void onUserAction (NeUserActionEvent e) throws NeVetoException

	Data Type	Description
Parameter		
e	NeUserActionEvent	Event triggered on the user's action
Return Value	void	

onUserActionTriggerShow

Method of NeUserActionListener

This method ist invoked when a menu item or button is displayed by which a user interaction can be triggered.

Declaration
void onUserActionTriggerShow (NeUserActionEvent e) throws NeVetoException

	Data Type	Description
Parameter		
e	NeUserActionEvent	Event triggered on the user's action
Return Value	void	

10.61 NeValueFilter

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIFilter**

This filter compares the value of an attribute to a pre-defined value. Different types of comparison can be derived from five different operators that are available.

=, <, >
The three operators above are comparison operators. They compare values and can be combined by "or".

!, #
The two operators above are control operators. ! will invert the result. By using '#' you can control the behavor of empty attributes. If set, a comparison with an empty attribute will result in **false**, except if the comparison value equals **null** and the operator is '='.
If '#' is not set, empty attributes will be turned into an empty or '0' object of the same type as the comparison value. For example, if the comaprison value is an integer object, an empty attribute value will be replaced by an Integer (0) value.

Examples: The given comparison value equals Integer 0. With the operators specified in the top row and the attribute value in the left column, the results in the below table will be obtained.

Operator / Attribute value	=	=#	!=	<	>=	< >
Integer (-10)	F	F	T	T	F	T
Integer (0)	T	T	F	F	T	F
String "10"	F	F	F	F	T	T
Empty attribute	F	T	F	F	F	F

For inverted conditions please use class **NeNotFilter**, for combined conditions please see class **NeCombinedFilter**.

Constructors of the Class

NeValueFilter

Constructor of NeValueFilter

This constructor lets you generate a filter that compares the value of an attribute to a comparison value of the type **object**. The operators < and > can only work if the object complies with the interface **Comparable**.

Declaration
NeValueFilter (java.lang.String attributeName, java.lang.String operator, java.lang.Object comparisonValue)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute, the content of which will be compared to the comparisonValue.
operator	java.lang.String	Operator to specify the type of comparison.
comparisonValue	java.lang.Object	Value compared to the content of the attribute.

NeValueFilter

Constructor of NeValueFilter

This constructor lets you generate a filter that compares the value of an attribute to a comparison value of the data type **double**.

Declaration
NeValueFilter (java.lang.String attributeName, java.lang.String operator, double comparisonValue)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute, the content of which will be compared to the comparisonValue.
operator	java.lang.String	Operator to specify the type of comparison.
comparisonValue	double	Value compared to the content of the attribute.

NeValueFilter

Constructor of NeValueFilter

This constructor lets you generate a filter that compares the value of an attribute to a comparison value of the data type **long**.

Declaration
NeValueFilter (java.lang.String attributeName, java.lang.String operator, long comparisonValue)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the attribute, the content of which will be compared to the comparisonValue.
operator	java.lang.String	Operator to specify the type of comparison.
comparisonValue	long	Value compared to the content of the attribute.

10.62 NeValueReference

Belongs to VariousClasses

Package name **de.netronic.common.beanbase**
Implements **de.netronic.common.intface.NeIValueReference**

ValueReference objects transmit dates between layers and attributes of entities. They allow to comfortably handle dates in layers.

This class offers methods to handle a basic date attribute. It represents the base to handle composed dates - please see class **NeSumValueReference**.

Properties to Handle the Date Attribute

AttributeName	Name of the date attribute
Factor	Factor, by which the duration is multipied.
Offset	Value to be added

Methods to Handle the ValueReference Object

addPropertyChangeListener(...)	Adds a listener for change events in the NeValueReference object.
removePropertyChangeListener(...)	Removes a listener for change events in the NeValueReference object.

Constructors of the Class

NeValueReference

Constructor of NeValueReference

This constructor lets you generate an object to handle a basic date attribute of an entity.

Declaration
NeValueReference (java.lang.String attributeName, double factor)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the date attribute
factor	double	Factor, by which the value of the attribute is multiplied.

NeValueReference

Constructor of NeValueReference

This constructor lets you generate an object to handle a basic date attribute of an entity.

Declaration
NeValueReference (java.lang.String attributeName)

Parameter	Data Type	Description
attributeName	java.lang.String	Name of the date attribute

Properties of the Class

AttributeName

Property of NeValueReference

Typ	**java.lang.String**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**

Name of the attribute that holds the date.

Accessing Methods
void setAttributeName (java.lang.String newValue)
java.lang.String getAttributeName ()

Factor

Property of NeValueReference

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**1.0**

By this property you can set or retrieve the factor by which the value is multiplied that is specified by the property set/getAttribute. It serves for example to transform the duration into a percentage.

Accessing Methods
void setFactor (double newValue)
double getFactor ()

Offset

Property of NeValueReference

Typ	**double**
Bound	**no**
Vetoable	**no**
Exposure Level	**regular**
Default Value	**0.0**

By this property you can set or retrieve a value to be added to NeValueReference.

Accessing Methods
void setOffset (double newValue)
double getOffset ()

Methods of the Class

addPropertyChangeListener

Method of NeValueReference

This method adds a listener for change events in the NeValueReference object. The listener will be informed each time a property has been changed.

Declaration
void addPropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be added.
Return Value	void	

Also see removePropertyChangeListener

removePropertyChangeListener

Method of NeValueReference

Removes an existing listener for change events in the NeValueReference object.

Declaration
void removePropertyChangeListener (java.beans.PropertyChangeListener l)

	Data Type	Description
Parameter		
l	java.beans.PropertyChangeListener	Listener to be removed.
Return Value	void	

Also see addPropertyChangeListener

10.63 NeVetoException

Belongs to VariousClasses

Package name **de.netronic.common.event**
Extends **java.lang.Exception**

An NeVetoException can be thrown by an application during the handling of an interaction notification event (for example onObjectChanged ()) sent by a JComponent in order to cancel the interaction.

Constructors of the Class

NeVetoException

Constructor of NeVetoException

Via this constructor, you can generate an exception.

Declaration
NeVetoException ()

11 Index

D

G

H

I

M

N

O

S

T