| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TableColumn.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/table/TableCellRenderer.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/table/TableColumnModel.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/table/TableColumn.html)    [**NO FRAMES**](http://docs.google.com/TableColumn.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#1ci93xb) | [METHOD](#1pxezwc) |

## **javax.swing.table**

Class TableColumn

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.swing.table.TableColumn**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html)

public class **TableColumn**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html)

A TableColumn represents all the attributes of a column in a JTable, such as width, resizibility, minimum and maximum width. In addition, the TableColumn provides slots for a renderer and an editor that can be used to display and edit the values in this column.

It is also possible to specify renderers and editors on a per type basis rather than a per column basis - see the setDefaultRenderer method in the JTable class. This default mechanism is only used when the renderer (or editor) in the TableColumn is null.

The TableColumn stores the link between the columns in the JTable and the columns in the TableModel. The modelIndex is the column in the TableModel, which will be queried for the data values for the cells in this column. As the column moves around in the view this modelIndex does not change.

**Note:** Some implementations may assume that all TableColumnModels are unique, therefore we would recommend that the same TableColumn instance not be added more than once to a TableColumnModel. To show TableColumns with the same column of data from the model, create a new instance with the same modelIndex.

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

**See Also:**[TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html), [DefaultTableColumnModel](http://docs.google.com/javax/swing/table/DefaultTableColumnModel.html), [JTableHeader.getDefaultRenderer()](http://docs.google.com/javax/swing/table/JTableHeader.html#getDefaultRenderer()), [JTable.getDefaultRenderer(Class)](http://docs.google.com/javax/swing/JTable.html#getDefaultRenderer(java.lang.Class)), [JTable.getDefaultEditor(Class)](http://docs.google.com/javax/swing/JTable.html#getDefaultEditor(java.lang.Class)), [JTable.getCellRenderer(int, int)](http://docs.google.com/javax/swing/JTable.html#getCellRenderer(int,%20int)), [JTable.getCellEditor(int, int)](http://docs.google.com/javax/swing/JTable.html#getCellEditor(int,%20int))

| **Field Summary** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**CELL\_RENDERER\_PROPERTY**](http://docs.google.com/javax/swing/table/TableColumn.html#CELL_RENDERER_PROPERTY)            Obsolete as of Java 2 platform v1.3. |
| protected  [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) | [**cellEditor**](http://docs.google.com/javax/swing/table/TableColumn.html#cellEditor)            The editor used to edit the data cells of the column. |
| protected  [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) | [**cellRenderer**](http://docs.google.com/javax/swing/table/TableColumn.html#cellRenderer)            The renderer used to draw the data cells of the column. |
| static [String](http://docs.google.com/java/lang/String.html) | [**COLUMN\_WIDTH\_PROPERTY**](http://docs.google.com/javax/swing/table/TableColumn.html#COLUMN_WIDTH_PROPERTY)            Obsolete as of Java 2 platform v1.3. |
| static [String](http://docs.google.com/java/lang/String.html) | [**HEADER\_RENDERER\_PROPERTY**](http://docs.google.com/javax/swing/table/TableColumn.html#HEADER_RENDERER_PROPERTY)            Obsolete as of Java 2 platform v1.3. |
| static [String](http://docs.google.com/java/lang/String.html) | [**HEADER\_VALUE\_PROPERTY**](http://docs.google.com/javax/swing/table/TableColumn.html#HEADER_VALUE_PROPERTY)            Obsolete as of Java 2 platform v1.3. |
| protected  [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) | [**headerRenderer**](http://docs.google.com/javax/swing/table/TableColumn.html#headerRenderer)            The renderer used to draw the header of the column. |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**headerValue**](http://docs.google.com/javax/swing/table/TableColumn.html#headerValue)            The header value of the column. |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**identifier**](http://docs.google.com/javax/swing/table/TableColumn.html#identifier)            This object is not used internally by the drawing machinery of the JTable; identifiers may be set in the TableColumn as as an optional way to tag and locate table columns. |
| protected  boolean | [**isResizable**](http://docs.google.com/javax/swing/table/TableColumn.html#isResizable)            If true, the user is allowed to resize the column; the default is true. |
| protected  int | [**maxWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#maxWidth)            The maximum width of the column. |
| protected  int | [**minWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#minWidth)            The minimum width of the column. |
| protected  int | [**modelIndex**](http://docs.google.com/javax/swing/table/TableColumn.html#modelIndex)            The index of the column in the model which is to be displayed by this TableColumn. |
| protected  int | [**resizedPostingDisableCount**](http://docs.google.com/javax/swing/table/TableColumn.html#resizedPostingDisableCount)  **Deprecated.** *as of Java 2 platform v1.3* |
| protected  int | [**width**](http://docs.google.com/javax/swing/table/TableColumn.html#width)            The width of the column. |

| **Constructor Summary** | |
| --- | --- |
| [**TableColumn**](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn())()            Cover method, using a default model index of 0, default width of 75, a null renderer and a null editor. |
| [**TableColumn**](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn(int))(int modelIndex)            Cover method, using a default width of 75, a null renderer and a null editor. |
| [**TableColumn**](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn(int,%20int))(int modelIndex, int width)            Cover method, using a null renderer and a null editor. |
| [**TableColumn**](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn(int,%20int,%20javax.swing.table.TableCellRenderer,%20javax.swing.table.TableCellEditor))(int modelIndex, int width, [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) cellRenderer, [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) cellEditor)            Creates and initializes an instance of TableColumn with the specified model index, width, cell renderer, and cell editor; all TableColumn constructors delegate to this one. |

| **Method Summary** | |
| --- | --- |
| void | [**addPropertyChangeListener**](http://docs.google.com/javax/swing/table/TableColumn.html#addPropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Adds a PropertyChangeListener to the listener list. |
| protected  [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) | [**createDefaultHeaderRenderer**](http://docs.google.com/javax/swing/table/TableColumn.html#createDefaultHeaderRenderer())()            As of Java 2 platform v1.3, this method is not called by the TableColumn constructor. |
| void | [**disableResizedPosting**](http://docs.google.com/javax/swing/table/TableColumn.html#disableResizedPosting())()  **Deprecated.** *as of Java 2 platform v1.3* |
| void | [**enableResizedPosting**](http://docs.google.com/javax/swing/table/TableColumn.html#enableResizedPosting())()  **Deprecated.** *as of Java 2 platform v1.3* |
| [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) | [**getCellEditor**](http://docs.google.com/javax/swing/table/TableColumn.html#getCellEditor())()            Returns the TableCellEditor used by the JTable to edit values for this column. |
| [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) | [**getCellRenderer**](http://docs.google.com/javax/swing/table/TableColumn.html#getCellRenderer())()            Returns the TableCellRenderer used by the JTable to draw values for this column. |
| [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) | [**getHeaderRenderer**](http://docs.google.com/javax/swing/table/TableColumn.html#getHeaderRenderer())()            Returns the TableCellRenderer used to draw the header of the TableColumn. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getHeaderValue**](http://docs.google.com/javax/swing/table/TableColumn.html#getHeaderValue())()            Returns the Object used as the value for the header renderer. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getIdentifier**](http://docs.google.com/javax/swing/table/TableColumn.html#getIdentifier())()            Returns the identifier object for this column. |
| int | [**getMaxWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#getMaxWidth())()            Returns the maximum width for the TableColumn. |
| int | [**getMinWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#getMinWidth())()            Returns the minimum width for the TableColumn. |
| int | [**getModelIndex**](http://docs.google.com/javax/swing/table/TableColumn.html#getModelIndex())()            Returns the model index for this column. |
| int | [**getPreferredWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#getPreferredWidth())()            Returns the preferred width of the TableColumn. |
| [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html)[] | [**getPropertyChangeListeners**](http://docs.google.com/javax/swing/table/TableColumn.html#getPropertyChangeListeners())()            Returns an array of all the PropertyChangeListeners added to this TableColumn with addPropertyChangeListener(). |
| boolean | [**getResizable**](http://docs.google.com/javax/swing/table/TableColumn.html#getResizable())()            Returns true if the user is allowed to resize the TableColumn's width, false otherwise. |
| int | [**getWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#getWidth())()            Returns the width of the TableColumn. |
| void | [**removePropertyChangeListener**](http://docs.google.com/javax/swing/table/TableColumn.html#removePropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Removes a PropertyChangeListener from the listener list. |
| void | [**setCellEditor**](http://docs.google.com/javax/swing/table/TableColumn.html#setCellEditor(javax.swing.table.TableCellEditor))([TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) cellEditor)            Sets the editor to used by when a cell in this column is edited. |
| void | [**setCellRenderer**](http://docs.google.com/javax/swing/table/TableColumn.html#setCellRenderer(javax.swing.table.TableCellRenderer))([TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) cellRenderer)            Sets the TableCellRenderer used by JTable to draw individual values for this column. |
| void | [**setHeaderRenderer**](http://docs.google.com/javax/swing/table/TableColumn.html#setHeaderRenderer(javax.swing.table.TableCellRenderer))([TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) headerRenderer)            Sets the TableCellRenderer used to draw the TableColumn's header to headerRenderer. |
| void | [**setHeaderValue**](http://docs.google.com/javax/swing/table/TableColumn.html#setHeaderValue(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) headerValue)            Sets the Object whose string representation will be used as the value for the headerRenderer. |
| void | [**setIdentifier**](http://docs.google.com/javax/swing/table/TableColumn.html#setIdentifier(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) identifier)            Sets the TableColumn's identifier to anIdentifier. |
| void | [**setMaxWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#setMaxWidth(int))(int maxWidth)            Sets the TableColumn's maximum width to maxWidth or, if maxWidth is less than the minimum width, to the minimum width. |
| void | [**setMinWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#setMinWidth(int))(int minWidth)            Sets the TableColumn's minimum width to minWidth, adjusting the new minimum width if necessary to ensure that 0 <= minWidth <= maxWidth. |
| void | [**setModelIndex**](http://docs.google.com/javax/swing/table/TableColumn.html#setModelIndex(int))(int modelIndex)            Sets the model index for this column. |
| void | [**setPreferredWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#setPreferredWidth(int))(int preferredWidth)            Sets this column's preferred width to preferredWidth. |
| void | [**setResizable**](http://docs.google.com/javax/swing/table/TableColumn.html#setResizable(boolean))(boolean isResizable)            Sets whether this column can be resized. |
| void | [**setWidth**](http://docs.google.com/javax/swing/table/TableColumn.html#setWidth(int))(int width)            This method should not be used to set the widths of columns in the JTable, use setPreferredWidth instead. |
| void | [**sizeWidthToFit**](http://docs.google.com/javax/swing/table/TableColumn.html#sizeWidthToFit())()            Resizes the TableColumn to fit the width of its header cell. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### COLUMN\_WIDTH\_PROPERTY

public static final [String](http://docs.google.com/java/lang/String.html) **COLUMN\_WIDTH\_PROPERTY**

Obsolete as of Java 2 platform v1.3. Please use string literals to identify properties.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.table.TableColumn.COLUMN_WIDTH_PROPERTY)

### HEADER\_VALUE\_PROPERTY

public static final [String](http://docs.google.com/java/lang/String.html) **HEADER\_VALUE\_PROPERTY**

Obsolete as of Java 2 platform v1.3. Please use string literals to identify properties.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.table.TableColumn.HEADER_VALUE_PROPERTY)

### HEADER\_RENDERER\_PROPERTY

public static final [String](http://docs.google.com/java/lang/String.html) **HEADER\_RENDERER\_PROPERTY**

Obsolete as of Java 2 platform v1.3. Please use string literals to identify properties.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.table.TableColumn.HEADER_RENDERER_PROPERTY)

### CELL\_RENDERER\_PROPERTY

public static final [String](http://docs.google.com/java/lang/String.html) **CELL\_RENDERER\_PROPERTY**

Obsolete as of Java 2 platform v1.3. Please use string literals to identify properties.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.table.TableColumn.CELL_RENDERER_PROPERTY)

### modelIndex

protected int **modelIndex**

The index of the column in the model which is to be displayed by this TableColumn. As columns are moved around in the view modelIndex remains constant.

### identifier

protected [Object](http://docs.google.com/java/lang/Object.html) **identifier**

This object is not used internally by the drawing machinery of the JTable; identifiers may be set in the TableColumn as as an optional way to tag and locate table columns. The table package does not modify or invoke any methods in these identifier objects other than the equals method which is used in the getColumnIndex() method in the DefaultTableColumnModel.

### width

protected int **width**

The width of the column.

### minWidth

protected int **minWidth**

The minimum width of the column.

### maxWidth

protected int **maxWidth**

The maximum width of the column.

### headerRenderer

protected [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) **headerRenderer**

The renderer used to draw the header of the column.

### headerValue

protected [Object](http://docs.google.com/java/lang/Object.html) **headerValue**

The header value of the column.

### cellRenderer

protected [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) **cellRenderer**

The renderer used to draw the data cells of the column.

### cellEditor

protected [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) **cellEditor**

The editor used to edit the data cells of the column.

### isResizable

protected boolean **isResizable**

If true, the user is allowed to resize the column; the default is true.

### resizedPostingDisableCount

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
protected transient int **resizedPostingDisableCount**

**Deprecated.** *as of Java 2 platform v1.3*This field was not used in previous releases and there are currently no plans to support it in the future.

| **Constructor Detail** |
| --- |

### TableColumn

public **TableColumn**()

Cover method, using a default model index of 0, default width of 75, a null renderer and a null editor. This method is intended for serialization.

**See Also:**[TableColumn(int, int, TableCellRenderer, TableCellEditor)](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn(int,%20int,%20javax.swing.table.TableCellRenderer,%20javax.swing.table.TableCellEditor))

### TableColumn

public **TableColumn**(int modelIndex)

Cover method, using a default width of 75, a null renderer and a null editor.

**See Also:**[TableColumn(int, int, TableCellRenderer, TableCellEditor)](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn(int,%20int,%20javax.swing.table.TableCellRenderer,%20javax.swing.table.TableCellEditor))

### TableColumn

public **TableColumn**(int modelIndex,  
 int width)

Cover method, using a null renderer and a null editor.

**See Also:**[TableColumn(int, int, TableCellRenderer, TableCellEditor)](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn(int,%20int,%20javax.swing.table.TableCellRenderer,%20javax.swing.table.TableCellEditor))

### TableColumn

public **TableColumn**(int modelIndex,  
 int width,  
 [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) cellRenderer,  
 [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) cellEditor)

Creates and initializes an instance of TableColumn with the specified model index, width, cell renderer, and cell editor; all TableColumn constructors delegate to this one. The value of width is used for both the initial and preferred width; if width is negative, they're set to 0. The minimum width is set to 15 unless the initial width is less, in which case the minimum width is set to the initial width.

When the cellRenderer or cellEditor parameter is null, a default value provided by the JTable getDefaultRenderer or getDefaultEditor method, respectively, is used to provide defaults based on the type of the data in this column. This column-centric rendering strategy can be circumvented by overriding the getCellRenderer methods in JTable.

**Parameters:**modelIndex - the index of the column in the model that supplies the data for this column in the table; the model index remains the same even when columns are reordered in the viewwidth - this column's preferred width and initial widthcellRenderer - the object used to render values in this columncellEditor - the object used to edit values in this column**See Also:**[getMinWidth()](http://docs.google.com/javax/swing/table/TableColumn.html#getMinWidth()), [JTable.getDefaultRenderer(Class)](http://docs.google.com/javax/swing/JTable.html#getDefaultRenderer(java.lang.Class)), [JTable.getDefaultEditor(Class)](http://docs.google.com/javax/swing/JTable.html#getDefaultEditor(java.lang.Class)), [JTable.getCellRenderer(int, int)](http://docs.google.com/javax/swing/JTable.html#getCellRenderer(int,%20int)), [JTable.getCellEditor(int, int)](http://docs.google.com/javax/swing/JTable.html#getCellEditor(int,%20int))

| **Method Detail** |
| --- |

### setModelIndex

public void **setModelIndex**(int modelIndex)

Sets the model index for this column. The model index is the index of the column in the model that will be displayed by this TableColumn. As the TableColumn is moved around in the view the model index remains constant.

**Parameters:**modelIndex - the new modelIndex

### getModelIndex

public int **getModelIndex**()

Returns the model index for this column.

**Returns:**the modelIndex property

### setIdentifier

public void **setIdentifier**([Object](http://docs.google.com/java/lang/Object.html) identifier)

Sets the TableColumn's identifier to anIdentifier.

Note: identifiers are not used by the JTable, they are purely a convenience for the external tagging and location of columns.

**Parameters:**identifier - an identifier for this column**See Also:**[getIdentifier()](http://docs.google.com/javax/swing/table/TableColumn.html#getIdentifier())

### getIdentifier

public [Object](http://docs.google.com/java/lang/Object.html) **getIdentifier**()

Returns the identifier object for this column. Note identifiers are not used by JTable, they are purely a convenience for external use. If the identifier is null, getIdentifier() returns getHeaderValue as a default.

**Returns:**the identifier property**See Also:**[setIdentifier(java.lang.Object)](http://docs.google.com/javax/swing/table/TableColumn.html#setIdentifier(java.lang.Object))

### setHeaderValue

public void **setHeaderValue**([Object](http://docs.google.com/java/lang/Object.html) headerValue)

Sets the Object whose string representation will be used as the value for the headerRenderer. When the TableColumn is created, the default headerValue is null.

**Parameters:**headerValue - the new headerValue**See Also:**[getHeaderValue()](http://docs.google.com/javax/swing/table/TableColumn.html#getHeaderValue())

### getHeaderValue

public [Object](http://docs.google.com/java/lang/Object.html) **getHeaderValue**()

Returns the Object used as the value for the header renderer.

**Returns:**the headerValue property**See Also:**[setHeaderValue(java.lang.Object)](http://docs.google.com/javax/swing/table/TableColumn.html#setHeaderValue(java.lang.Object))

### setHeaderRenderer

public void **setHeaderRenderer**([TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) headerRenderer)

Sets the TableCellRenderer used to draw the TableColumn's header to headerRenderer.

It is the header renderers responsibility to render the sorting indicator. If you are using sorting and specify a renderer your renderer must render the sorting indication.

**Parameters:**headerRenderer - the new headerRenderer**See Also:**[getHeaderRenderer()](http://docs.google.com/javax/swing/table/TableColumn.html#getHeaderRenderer())

### getHeaderRenderer

public [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) **getHeaderRenderer**()

Returns the TableCellRenderer used to draw the header of the TableColumn. When the headerRenderer is null, the JTableHeader uses its defaultRenderer. The default value for a headerRenderer is null.

**Returns:**the headerRenderer property**See Also:**[setHeaderRenderer(javax.swing.table.TableCellRenderer)](http://docs.google.com/javax/swing/table/TableColumn.html#setHeaderRenderer(javax.swing.table.TableCellRenderer)), [setHeaderValue(java.lang.Object)](http://docs.google.com/javax/swing/table/TableColumn.html#setHeaderValue(java.lang.Object)), [JTableHeader.getDefaultRenderer()](http://docs.google.com/javax/swing/table/JTableHeader.html#getDefaultRenderer())

### setCellRenderer

public void **setCellRenderer**([TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) cellRenderer)

Sets the TableCellRenderer used by JTable to draw individual values for this column.

**Parameters:**cellRenderer - the new cellRenderer**See Also:**[getCellRenderer()](http://docs.google.com/javax/swing/table/TableColumn.html#getCellRenderer())

### getCellRenderer

public [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) **getCellRenderer**()

Returns the TableCellRenderer used by the JTable to draw values for this column. The cellRenderer of the column not only controls the visual look for the column, but is also used to interpret the value object supplied by the TableModel. When the cellRenderer is null, the JTable uses a default renderer based on the class of the cells in that column. The default value for a cellRenderer is null.

**Returns:**the cellRenderer property**See Also:**[setCellRenderer(javax.swing.table.TableCellRenderer)](http://docs.google.com/javax/swing/table/TableColumn.html#setCellRenderer(javax.swing.table.TableCellRenderer)), [JTable.setDefaultRenderer(java.lang.Class, javax.swing.table.TableCellRenderer)](http://docs.google.com/javax/swing/JTable.html#setDefaultRenderer(java.lang.Class,%20javax.swing.table.TableCellRenderer))

### setCellEditor

public void **setCellEditor**([TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) cellEditor)

Sets the editor to used by when a cell in this column is edited.

**Parameters:**cellEditor - the new cellEditor**See Also:**[getCellEditor()](http://docs.google.com/javax/swing/table/TableColumn.html#getCellEditor())

### getCellEditor

public [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) **getCellEditor**()

Returns the TableCellEditor used by the JTable to edit values for this column. When the cellEditor is null, the JTable uses a default editor based on the class of the cells in that column. The default value for a cellEditor is null.

**Returns:**the cellEditor property**See Also:**[setCellEditor(javax.swing.table.TableCellEditor)](http://docs.google.com/javax/swing/table/TableColumn.html#setCellEditor(javax.swing.table.TableCellEditor)), [JTable.setDefaultEditor(java.lang.Class, javax.swing.table.TableCellEditor)](http://docs.google.com/javax/swing/JTable.html#setDefaultEditor(java.lang.Class,%20javax.swing.table.TableCellEditor))

### setWidth

public void **setWidth**(int width)

This method should not be used to set the widths of columns in the JTable, use setPreferredWidth instead. Like a layout manager in the AWT, the JTable adjusts a column's width automatically whenever the table itself changes size, or a column's preferred width is changed. Setting widths programmatically therefore has no long term effect.

This method sets this column's width to width. If width exceeds the minimum or maximum width, it is adjusted to the appropriate limiting value.

**Parameters:**width - the new width**See Also:**[getWidth()](http://docs.google.com/javax/swing/table/TableColumn.html#getWidth()), [setMinWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setMinWidth(int)), [setMaxWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setMaxWidth(int)), [setPreferredWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setPreferredWidth(int)), [JTable.doLayout()](http://docs.google.com/javax/swing/JTable.html#doLayout())

### getWidth

public int **getWidth**()

Returns the width of the TableColumn. The default width is 75.

**Returns:**the width property**See Also:**[setWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setWidth(int))

### setPreferredWidth

public void **setPreferredWidth**(int preferredWidth)

Sets this column's preferred width to preferredWidth. If preferredWidth exceeds the minimum or maximum width, it is adjusted to the appropriate limiting value.

For details on how the widths of columns in the JTable (and JTableHeader) are calculated from the preferredWidth, see the doLayout method in JTable.

**Parameters:**preferredWidth - the new preferred width**See Also:**[getPreferredWidth()](http://docs.google.com/javax/swing/table/TableColumn.html#getPreferredWidth()), [JTable.doLayout()](http://docs.google.com/javax/swing/JTable.html#doLayout())

### getPreferredWidth

public int **getPreferredWidth**()

Returns the preferred width of the TableColumn. The default preferred width is 75.

**Returns:**the preferredWidth property**See Also:**[setPreferredWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setPreferredWidth(int))

### setMinWidth

public void **setMinWidth**(int minWidth)

Sets the TableColumn's minimum width to minWidth, adjusting the new minimum width if necessary to ensure that 0 <= minWidth <= maxWidth. For example, if the minWidth argument is negative, this method sets the minWidth property to 0.

If the value of the width or preferredWidth property is less than the new minimum width, this method sets that property to the new minimum width.

**Parameters:**minWidth - the new minimum width**See Also:**[getMinWidth()](http://docs.google.com/javax/swing/table/TableColumn.html#getMinWidth()), [setPreferredWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setPreferredWidth(int)), [setMaxWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setMaxWidth(int))

### getMinWidth

public int **getMinWidth**()

Returns the minimum width for the TableColumn. The TableColumn's width can't be made less than this either by the user or programmatically.

**Returns:**the minWidth property**See Also:**[setMinWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setMinWidth(int)), [TableColumn(int, int, TableCellRenderer, TableCellEditor)](http://docs.google.com/javax/swing/table/TableColumn.html#TableColumn(int,%20int,%20javax.swing.table.TableCellRenderer,%20javax.swing.table.TableCellEditor))

### setMaxWidth

public void **setMaxWidth**(int maxWidth)

Sets the TableColumn's maximum width to maxWidth or, if maxWidth is less than the minimum width, to the minimum width.

If the value of the width or preferredWidth property is more than the new maximum width, this method sets that property to the new maximum width.

**Parameters:**maxWidth - the new maximum width**See Also:**[getMaxWidth()](http://docs.google.com/javax/swing/table/TableColumn.html#getMaxWidth()), [setPreferredWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setPreferredWidth(int)), [setMinWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setMinWidth(int))

### getMaxWidth

public int **getMaxWidth**()

Returns the maximum width for the TableColumn. The TableColumn's width can't be made larger than this either by the user or programmatically. The default maxWidth is Integer.MAX\_VALUE.

**Returns:**the maxWidth property**See Also:**[setMaxWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setMaxWidth(int))

### setResizable

public void **setResizable**(boolean isResizable)

Sets whether this column can be resized.

**Parameters:**isResizable - if true, resizing is allowed; otherwise false**See Also:**[getResizable()](http://docs.google.com/javax/swing/table/TableColumn.html#getResizable())

### getResizable

public boolean **getResizable**()

Returns true if the user is allowed to resize the TableColumn's width, false otherwise. You can change the width programmatically regardless of this setting. The default is true.

**Returns:**the isResizable property**See Also:**[setResizable(boolean)](http://docs.google.com/javax/swing/table/TableColumn.html#setResizable(boolean))

### sizeWidthToFit

public void **sizeWidthToFit**()

Resizes the TableColumn to fit the width of its header cell. This method does nothing if the header renderer is null (the default case). Otherwise, it sets the minimum, maximum and preferred widths of this column to the widths of the minimum, maximum and preferred sizes of the Component delivered by the header renderer. The transient "width" property of this TableColumn is also set to the preferred width. Note this method is not used internally by the table package.

**See Also:**[setPreferredWidth(int)](http://docs.google.com/javax/swing/table/TableColumn.html#setPreferredWidth(int))

### disableResizedPosting

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **disableResizedPosting**()

**Deprecated.** *as of Java 2 platform v1.3*

This field was not used in previous releases and there are currently no plans to support it in the future.

### enableResizedPosting

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public void **enableResizedPosting**()

**Deprecated.** *as of Java 2 platform v1.3*

This field was not used in previous releases and there are currently no plans to support it in the future.

### addPropertyChangeListener

public void **addPropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)

Adds a PropertyChangeListener to the listener list. The listener is registered for all properties.

A PropertyChangeEvent will get fired in response to an explicit call to setFont, setBackground, or setForeground on the current component. Note that if the current component is inheriting its foreground, background, or font from its container, then no event will be fired in response to a change in the inherited property.

**Parameters:**listener - the listener to be added

### removePropertyChangeListener

public void **removePropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)

Removes a PropertyChangeListener from the listener list. The PropertyChangeListener to be removed was registered for all properties.

**Parameters:**listener - the listener to be removed

### getPropertyChangeListeners

public [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html)[] **getPropertyChangeListeners**()

Returns an array of all the PropertyChangeListeners added to this TableColumn with addPropertyChangeListener().

**Returns:**all of the PropertyChangeListeners added or an empty array if no listeners have been added**Since:** 1.4

### createDefaultHeaderRenderer

protected [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) **createDefaultHeaderRenderer**()

As of Java 2 platform v1.3, this method is not called by the TableColumn constructor. Previously this method was used by the TableColumn to create a default header renderer. As of Java 2 platform v1.3, the default header renderer is null. JTableHeader now provides its own shared default renderer, just as the JTable does for its cell renderers.

**Returns:**the default header renderer**See Also:**[JTableHeader.createDefaultRenderer()](http://docs.google.com/javax/swing/table/JTableHeader.html#createDefaultRenderer())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TableColumn.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/table/TableCellRenderer.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/table/TableColumnModel.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/table/TableColumn.html)    [**NO FRAMES**](http://docs.google.com/TableColumn.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#1ci93xb) | [METHOD](#1pxezwc) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).