| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JTable.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/JTabbedPane.ModelListener.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/JTable.AccessibleJTable.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/JTable.html)    [**NO FRAMES**](http://docs.google.com/JTable.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#26in1rg) | DETAIL: [FIELD](#2jxsxqh) | [CONSTR](#3tbugp1) | [METHOD](#3l18frh) |

## **javax.swing**

Class JTable

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.awt.Component](http://docs.google.com/java/awt/Component.html)  
 [java.awt.Container](http://docs.google.com/java/awt/Container.html)  
 [javax.swing.JComponent](http://docs.google.com/javax/swing/JComponent.html)  
 **javax.swing.JTable**

**All Implemented Interfaces:** [ImageObserver](http://docs.google.com/java/awt/image/ImageObserver.html), [MenuContainer](http://docs.google.com/java/awt/MenuContainer.html), [Serializable](http://docs.google.com/java/io/Serializable.html), [EventListener](http://docs.google.com/java/util/EventListener.html), [Accessible](http://docs.google.com/javax/accessibility/Accessible.html), [CellEditorListener](http://docs.google.com/javax/swing/event/CellEditorListener.html), [ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html), [RowSorterListener](http://docs.google.com/javax/swing/event/RowSorterListener.html), [TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html), [TableModelListener](http://docs.google.com/javax/swing/event/TableModelListener.html), [Scrollable](http://docs.google.com/javax/swing/Scrollable.html)

public class **JTable**extends [JComponent](http://docs.google.com/javax/swing/JComponent.html)implements [TableModelListener](http://docs.google.com/javax/swing/event/TableModelListener.html), [Scrollable](http://docs.google.com/javax/swing/Scrollable.html), [TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html), [ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html), [CellEditorListener](http://docs.google.com/javax/swing/event/CellEditorListener.html), [Accessible](http://docs.google.com/javax/accessibility/Accessible.html), [RowSorterListener](http://docs.google.com/javax/swing/event/RowSorterListener.html)

The JTable is used to display and edit regular two-dimensional tables of cells. See [How to Use Tables](http://java.sun.com/docs/books/tutorial/uiswing/components/table.html) in *The Java Tutorial* for task-oriented documentation and examples of using JTable.

The JTable has many facilities that make it possible to customize its rendering and editing but provides defaults for these features so that simple tables can be set up easily. For example, to set up a table with 10 rows and 10 columns of numbers:

TableModel dataModel = new AbstractTableModel() {  
 public int getColumnCount() { return 10; }  
 public int getRowCount() { return 10;}  
 public Object getValueAt(int row, int col) { return new Integer(row\*col); }  
 };  
 JTable table = new JTable(dataModel);  
 JScrollPane scrollpane = new JScrollPane(table);

Note that if you wish to use a JTable in a standalone view (outside of a JScrollPane) and want the header displayed, you can get it using [getTableHeader()](http://docs.google.com/javax/swing/JTable.html#getTableHeader()) and display it separately.

To enable sorting and filtering of rows, use a RowSorter. You can set up a row sorter in either of two ways:

* Directly set the RowSorter. For example: table.setRowSorter(new TableRowSorter(model)).
* Set the autoCreateRowSorter property to true, so that the JTable creates a RowSorter for you. For example: setAutoCreateRowSorter(true).

When designing applications that use the JTable it is worth paying close attention to the data structures that will represent the table's data. The DefaultTableModel is a model implementation that uses a Vector of Vectors of Objects to store the cell values. As well as copying the data from an application into the DefaultTableModel, it is also possible to wrap the data in the methods of the TableModel interface so that the data can be passed to the JTable directly, as in the example above. This often results in more efficient applications because the model is free to choose the internal representation that best suits the data. A good rule of thumb for deciding whether to use the AbstractTableModel or the DefaultTableModel is to use the AbstractTableModel as the base class for creating subclasses and the DefaultTableModel when subclassing is not required.

The "TableExample" directory in the demo area of the source distribution gives a number of complete examples of JTable usage, covering how the JTable can be used to provide an editable view of data taken from a database and how to modify the columns in the display to use specialized renderers and editors.

The JTable uses integers exclusively to refer to both the rows and the columns of the model that it displays. The JTable simply takes a tabular range of cells and uses getValueAt(int, int) to retrieve the values from the model during painting. It is important to remember that the column and row indexes returned by various JTable methods are in terms of the JTable (the view) and are not necessarily the same indexes used by the model.

By default, columns may be rearranged in the JTable so that the view's columns appear in a different order to the columns in the model. This does not affect the implementation of the model at all: when the columns are reordered, the JTable maintains the new order of the columns internally and converts its column indices before querying the model.

So, when writing a TableModel, it is not necessary to listen for column reordering events as the model will be queried in its own coordinate system regardless of what is happening in the view. In the examples area there is a demonstration of a sorting algorithm making use of exactly this technique to interpose yet another coordinate system where the order of the rows is changed, rather than the order of the columns.

Similarly when using the sorting and filtering functionality provided by RowSorter the underlying TableModel does not need to know how to do sorting, rather RowSorter will handle it. Coordinate conversions will be necessary when using the row based methods of JTable with the underlying TableModel. All of JTables row based methods are in terms of the RowSorter, which is not necessarily the same as that of the underlying TableModel. For example, the selection is always in terms of JTable so that when using RowSorter you will need to convert using convertRowIndexToView or convertRowIndexToModel. The following shows how to convert coordinates from JTable to that of the underlying model:

int[] selection = table.getSelectedRows();  
 for (int i = 0; i < selection.length; i++) {  
 selection[i] = table.convertRowIndexToModel(selection[i]);  
 }  
 // selection is now in terms of the underlying TableModel

By default if sorting is enabled JTable will persist the selection and variable row heights in terms of the model on sorting. For example if row 0, in terms of the underlying model, is currently selected, after the sort row 0, in terms of the underlying model will be selected. Visually the selection may change, but in terms of the underlying model it will remain the same. The one exception to that is if the model index is no longer visible or was removed. For example, if row 0 in terms of model was filtered out the selection will be empty after the sort.

J2SE 5 adds methods to JTable to provide convenient access to some common printing needs. Simple new [print()](http://docs.google.com/javax/swing/JTable.html#print()) methods allow for quick and easy addition of printing support to your application. In addition, a new [getPrintable(javax.swing.JTable.PrintMode, java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat)) method is available for more advanced printing needs.

As for all JComponent classes, you can use [InputMap](http://docs.google.com/javax/swing/InputMap.html) and [ActionMap](http://docs.google.com/javax/swing/ActionMap.html) to associate an [Action](http://docs.google.com/javax/swing/Action.html) object with a [KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) and execute the action under specified conditions.

**Warning:** Swing is not thread safe. For more information see [Swing's Threading Policy](http://docs.google.com/package-summary.html#threading).

**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:**[DefaultTableModel](http://docs.google.com/javax/swing/table/DefaultTableModel.html), [TableRowSorter](http://docs.google.com/javax/swing/table/TableRowSorter.html)

| **Nested Class Summary** | |
| --- | --- |
| protected  class | [**JTable.AccessibleJTable**](http://docs.google.com/javax/swing/JTable.AccessibleJTable.html)            This class implements accessibility support for the JTable class. |
| static class | [**JTable.DropLocation**](http://docs.google.com/javax/swing/JTable.DropLocation.html)            A subclass of TransferHandler.DropLocation representing a drop location for a JTable. |
| static class | [**JTable.PrintMode**](http://docs.google.com/javax/swing/JTable.PrintMode.html)            Printing modes, used in printing JTables. |

| **Nested classes/interfaces inherited from class javax.swing.**[**JComponent**](http://docs.google.com/javax/swing/JComponent.html) |
| --- |
| [JComponent.AccessibleJComponent](http://docs.google.com/javax/swing/JComponent.AccessibleJComponent.html) |

| **Nested classes/interfaces inherited from class java.awt.**[**Container**](http://docs.google.com/java/awt/Container.html) |
| --- |
| [Container.AccessibleAWTContainer](http://docs.google.com/java/awt/Container.AccessibleAWTContainer.html) |

| **Nested classes/interfaces inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [Component.AccessibleAWTComponent](http://docs.google.com/java/awt/Component.AccessibleAWTComponent.html), [Component.BaselineResizeBehavior](http://docs.google.com/java/awt/Component.BaselineResizeBehavior.html), [Component.BltBufferStrategy](http://docs.google.com/java/awt/Component.BltBufferStrategy.html), [Component.FlipBufferStrategy](http://docs.google.com/java/awt/Component.FlipBufferStrategy.html) |

| **Field Summary** | |
| --- | --- |
| static int | [**AUTO\_RESIZE\_ALL\_COLUMNS**](http://docs.google.com/javax/swing/JTable.html#AUTO_RESIZE_ALL_COLUMNS)            During all resize operations, proportionately resize all columns. |
| static int | [**AUTO\_RESIZE\_LAST\_COLUMN**](http://docs.google.com/javax/swing/JTable.html#AUTO_RESIZE_LAST_COLUMN)            During all resize operations, apply adjustments to the last column only. |
| static int | [**AUTO\_RESIZE\_NEXT\_COLUMN**](http://docs.google.com/javax/swing/JTable.html#AUTO_RESIZE_NEXT_COLUMN)            When a column is adjusted in the UI, adjust the next column the opposite way. |
| static int | [**AUTO\_RESIZE\_OFF**](http://docs.google.com/javax/swing/JTable.html#AUTO_RESIZE_OFF)            Do not adjust column widths automatically; use a scrollbar. |
| static int | [**AUTO\_RESIZE\_SUBSEQUENT\_COLUMNS**](http://docs.google.com/javax/swing/JTable.html#AUTO_RESIZE_SUBSEQUENT_COLUMNS)            During UI adjustment, change subsequent columns to preserve the total width; this is the default behavior. |
| protected  boolean | [**autoCreateColumnsFromModel**](http://docs.google.com/javax/swing/JTable.html#autoCreateColumnsFromModel)            The table will query the TableModel to build the default set of columns if this is true. |
| protected  int | [**autoResizeMode**](http://docs.google.com/javax/swing/JTable.html#autoResizeMode)            Determines if the table automatically resizes the width of the table's columns to take up the entire width of the table, and how it does the resizing. |
| protected  [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) | [**cellEditor**](http://docs.google.com/javax/swing/JTable.html#cellEditor)            The active cell editor object, that overwrites the screen real estate occupied by the current cell and allows the user to change its contents. |
| protected  boolean | [**cellSelectionEnabled**](http://docs.google.com/javax/swing/JTable.html#cellSelectionEnabled)            Obsolete as of Java 2 platform v1.3. |
| protected  [TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) | [**columnModel**](http://docs.google.com/javax/swing/JTable.html#columnModel)            The TableColumnModel of the table. |
| protected  [TableModel](http://docs.google.com/javax/swing/table/TableModel.html) | [**dataModel**](http://docs.google.com/javax/swing/JTable.html#dataModel)            The TableModel of the table. |
| protected  [Hashtable](http://docs.google.com/java/util/Hashtable.html) | [**defaultEditorsByColumnClass**](http://docs.google.com/javax/swing/JTable.html#defaultEditorsByColumnClass)            A table of objects that display and edit the contents of a cell, indexed by class as declared in getColumnClass in the TableModel interface. |
| protected  [Hashtable](http://docs.google.com/java/util/Hashtable.html) | [**defaultRenderersByColumnClass**](http://docs.google.com/javax/swing/JTable.html#defaultRenderersByColumnClass)            A table of objects that display the contents of a cell, indexed by class as declared in getColumnClass in the TableModel interface. |
| protected  int | [**editingColumn**](http://docs.google.com/javax/swing/JTable.html#editingColumn)            Identifies the column of the cell being edited. |
| protected  int | [**editingRow**](http://docs.google.com/javax/swing/JTable.html#editingRow)            Identifies the row of the cell being edited. |
| protected  [Component](http://docs.google.com/java/awt/Component.html) | [**editorComp**](http://docs.google.com/javax/swing/JTable.html#editorComp)            If editing, the Component that is handling the editing. |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**gridColor**](http://docs.google.com/javax/swing/JTable.html#gridColor)            The color of the grid. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**preferredViewportSize**](http://docs.google.com/javax/swing/JTable.html#preferredViewportSize)            Used by the Scrollable interface to determine the initial visible area. |
| protected  int | [**rowHeight**](http://docs.google.com/javax/swing/JTable.html#rowHeight)            The height in pixels of each row in the table. |
| protected  int | [**rowMargin**](http://docs.google.com/javax/swing/JTable.html#rowMargin)            The height in pixels of the margin between the cells in each row. |
| protected  boolean | [**rowSelectionAllowed**](http://docs.google.com/javax/swing/JTable.html#rowSelectionAllowed)            True if row selection is allowed in this table. |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**selectionBackground**](http://docs.google.com/javax/swing/JTable.html#selectionBackground)            The background color of selected cells. |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**selectionForeground**](http://docs.google.com/javax/swing/JTable.html#selectionForeground)            The foreground color of selected cells. |
| protected  [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) | [**selectionModel**](http://docs.google.com/javax/swing/JTable.html#selectionModel)            The ListSelectionModel of the table, used to keep track of row selections. |
| protected  boolean | [**showHorizontalLines**](http://docs.google.com/javax/swing/JTable.html#showHorizontalLines)            The table draws horizontal lines between cells if showHorizontalLines is true. |
| protected  boolean | [**showVerticalLines**](http://docs.google.com/javax/swing/JTable.html#showVerticalLines)            The table draws vertical lines between cells if showVerticalLines is true. |
| protected  [JTableHeader](http://docs.google.com/javax/swing/table/JTableHeader.html) | [**tableHeader**](http://docs.google.com/javax/swing/JTable.html#tableHeader)            The TableHeader working with the table. |

| **Fields inherited from class javax.swing.**[**JComponent**](http://docs.google.com/javax/swing/JComponent.html) |
| --- |
| [accessibleContext](http://docs.google.com/javax/swing/JComponent.html#accessibleContext), [listenerList](http://docs.google.com/javax/swing/JComponent.html#listenerList), [TOOL\_TIP\_TEXT\_KEY](http://docs.google.com/javax/swing/JComponent.html#TOOL_TIP_TEXT_KEY), [ui](http://docs.google.com/javax/swing/JComponent.html#ui), [UNDEFINED\_CONDITION](http://docs.google.com/javax/swing/JComponent.html#UNDEFINED_CONDITION), [WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT](http://docs.google.com/javax/swing/JComponent.html#WHEN_ANCESTOR_OF_FOCUSED_COMPONENT), [WHEN\_FOCUSED](http://docs.google.com/javax/swing/JComponent.html#WHEN_FOCUSED), [WHEN\_IN\_FOCUSED\_WINDOW](http://docs.google.com/javax/swing/JComponent.html#WHEN_IN_FOCUSED_WINDOW) |

| **Fields inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [BOTTOM\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#BOTTOM_ALIGNMENT), [CENTER\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#CENTER_ALIGNMENT), [LEFT\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#LEFT_ALIGNMENT), [RIGHT\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#RIGHT_ALIGNMENT), [TOP\_ALIGNMENT](http://docs.google.com/java/awt/Component.html#TOP_ALIGNMENT) |

| **Fields inherited from interface java.awt.image.**[**ImageObserver**](http://docs.google.com/java/awt/image/ImageObserver.html) |
| --- |
| [ABORT](http://docs.google.com/java/awt/image/ImageObserver.html#ABORT), [ALLBITS](http://docs.google.com/java/awt/image/ImageObserver.html#ALLBITS), [ERROR](http://docs.google.com/java/awt/image/ImageObserver.html#ERROR), [FRAMEBITS](http://docs.google.com/java/awt/image/ImageObserver.html#FRAMEBITS), [HEIGHT](http://docs.google.com/java/awt/image/ImageObserver.html#HEIGHT), [PROPERTIES](http://docs.google.com/java/awt/image/ImageObserver.html#PROPERTIES), [SOMEBITS](http://docs.google.com/java/awt/image/ImageObserver.html#SOMEBITS), [WIDTH](http://docs.google.com/java/awt/image/ImageObserver.html#WIDTH) |

| **Constructor Summary** | |
| --- | --- |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html#JTable())()            Constructs a default JTable that is initialized with a default data model, a default column model, and a default selection model. |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html#JTable(int,%20int))(int numRows, int numColumns)            Constructs a JTable with numRows and numColumns of empty cells using DefaultTableModel. |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html#JTable(java.lang.Object%5B%5D%5B%5D,%20java.lang.Object%5B%5D))([Object](http://docs.google.com/java/lang/Object.html)[][] rowData, [Object](http://docs.google.com/java/lang/Object.html)[] columnNames)            Constructs a JTable to display the values in the two dimensional array, rowData, with column names, columnNames. |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html#JTable(javax.swing.table.TableModel))([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dm)            Constructs a JTable that is initialized with dm as the data model, a default column model, and a default selection model. |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html#JTable(javax.swing.table.TableModel,%20javax.swing.table.TableColumnModel))([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dm, [TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) cm)            Constructs a JTable that is initialized with dm as the data model, cm as the column model, and a default selection model. |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html#JTable(javax.swing.table.TableModel,%20javax.swing.table.TableColumnModel,%20javax.swing.ListSelectionModel))([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dm, [TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) cm, [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) sm)            Constructs a JTable that is initialized with dm as the data model, cm as the column model, and sm as the selection model. |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html#JTable(java.util.Vector,%20java.util.Vector))([Vector](http://docs.google.com/java/util/Vector.html) rowData, [Vector](http://docs.google.com/java/util/Vector.html) columnNames)            Constructs a JTable to display the values in the Vector of Vectors, rowData, with column names, columnNames. |

| **Method Summary** | |
| --- | --- |
| void | [**addColumn**](http://docs.google.com/javax/swing/JTable.html#addColumn(javax.swing.table.TableColumn))([TableColumn](http://docs.google.com/javax/swing/table/TableColumn.html) aColumn)            Appends aColumn to the end of the array of columns held by this JTable's column model. |
| void | [**addColumnSelectionInterval**](http://docs.google.com/javax/swing/JTable.html#addColumnSelectionInterval(int,%20int))(int index0, int index1)            Adds the columns from index0 to index1, inclusive, to the current selection. |
| void | [**addNotify**](http://docs.google.com/javax/swing/JTable.html#addNotify())()            Calls the configureEnclosingScrollPane method. |
| void | [**addRowSelectionInterval**](http://docs.google.com/javax/swing/JTable.html#addRowSelectionInterval(int,%20int))(int index0, int index1)            Adds the rows from index0 to index1, inclusive, to the current selection. |
| void | [**changeSelection**](http://docs.google.com/javax/swing/JTable.html#changeSelection(int,%20int,%20boolean,%20boolean))(int rowIndex, int columnIndex, boolean toggle, boolean extend)            Updates the selection models of the table, depending on the state of the two flags: toggle and extend. |
| void | [**clearSelection**](http://docs.google.com/javax/swing/JTable.html#clearSelection())()            Deselects all selected columns and rows. |
| void | [**columnAdded**](http://docs.google.com/javax/swing/JTable.html#columnAdded(javax.swing.event.TableColumnModelEvent))([TableColumnModelEvent](http://docs.google.com/javax/swing/event/TableColumnModelEvent.html) e)            Invoked when a column is added to the table column model. |
| int | [**columnAtPoint**](http://docs.google.com/javax/swing/JTable.html#columnAtPoint(java.awt.Point))([Point](http://docs.google.com/java/awt/Point.html) point)            Returns the index of the column that point lies in, or -1 if the result is not in the range [0, getColumnCount()-1]. |
| void | [**columnMarginChanged**](http://docs.google.com/javax/swing/JTable.html#columnMarginChanged(javax.swing.event.ChangeEvent))([ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) e)            Invoked when a column is moved due to a margin change. |
| void | [**columnMoved**](http://docs.google.com/javax/swing/JTable.html#columnMoved(javax.swing.event.TableColumnModelEvent))([TableColumnModelEvent](http://docs.google.com/javax/swing/event/TableColumnModelEvent.html) e)            Invoked when a column is repositioned. |
| void | [**columnRemoved**](http://docs.google.com/javax/swing/JTable.html#columnRemoved(javax.swing.event.TableColumnModelEvent))([TableColumnModelEvent](http://docs.google.com/javax/swing/event/TableColumnModelEvent.html) e)            Invoked when a column is removed from the table column model. |
| void | [**columnSelectionChanged**](http://docs.google.com/javax/swing/JTable.html#columnSelectionChanged(javax.swing.event.ListSelectionEvent))([ListSelectionEvent](http://docs.google.com/javax/swing/event/ListSelectionEvent.html) e)            Invoked when the selection model of the TableColumnModel is changed. |
| protected  void | [**configureEnclosingScrollPane**](http://docs.google.com/javax/swing/JTable.html#configureEnclosingScrollPane())()            If this JTable is the viewportView of an enclosing JScrollPane (the usual situation), configure this ScrollPane by, amongst other things, installing the table's tableHeader as the columnHeaderView of the scroll pane. |
| int | [**convertColumnIndexToModel**](http://docs.google.com/javax/swing/JTable.html#convertColumnIndexToModel(int))(int viewColumnIndex)            Maps the index of the column in the view at viewColumnIndex to the index of the column in the table model. |
| int | [**convertColumnIndexToView**](http://docs.google.com/javax/swing/JTable.html#convertColumnIndexToView(int))(int modelColumnIndex)            Maps the index of the column in the table model at modelColumnIndex to the index of the column in the view. |
| int | [**convertRowIndexToModel**](http://docs.google.com/javax/swing/JTable.html#convertRowIndexToModel(int))(int viewRowIndex)            Maps the index of the row in terms of the view to the underlying TableModel. |
| int | [**convertRowIndexToView**](http://docs.google.com/javax/swing/JTable.html#convertRowIndexToView(int))(int modelRowIndex)            Maps the index of the row in terms of the TableModel to the view. |
| protected  [TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) | [**createDefaultColumnModel**](http://docs.google.com/javax/swing/JTable.html#createDefaultColumnModel())()            Returns the default column model object, which is a DefaultTableColumnModel. |
| void | [**createDefaultColumnsFromModel**](http://docs.google.com/javax/swing/JTable.html#createDefaultColumnsFromModel())()            Creates default columns for the table from the data model using the getColumnCount method defined in the TableModel interface. |
| protected  [TableModel](http://docs.google.com/javax/swing/table/TableModel.html) | [**createDefaultDataModel**](http://docs.google.com/javax/swing/JTable.html#createDefaultDataModel())()            Returns the default table model object, which is a DefaultTableModel. |
| protected  void | [**createDefaultEditors**](http://docs.google.com/javax/swing/JTable.html#createDefaultEditors())()            Creates default cell editors for objects, numbers, and boolean values. |
| protected  void | [**createDefaultRenderers**](http://docs.google.com/javax/swing/JTable.html#createDefaultRenderers())()            Creates default cell renderers for objects, numbers, doubles, dates, booleans, and icons. |
| protected  [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) | [**createDefaultSelectionModel**](http://docs.google.com/javax/swing/JTable.html#createDefaultSelectionModel())()            Returns the default selection model object, which is a DefaultListSelectionModel. |
| protected  [JTableHeader](http://docs.google.com/javax/swing/table/JTableHeader.html) | [**createDefaultTableHeader**](http://docs.google.com/javax/swing/JTable.html#createDefaultTableHeader())()            Returns the default table header object, which is a JTableHeader. |
| static [JScrollPane](http://docs.google.com/javax/swing/JScrollPane.html) | [**createScrollPaneForTable**](http://docs.google.com/javax/swing/JTable.html#createScrollPaneForTable(javax.swing.JTable))([JTable](http://docs.google.com/javax/swing/JTable.html) aTable)  **Deprecated.** *As of Swing version 1.0.2, replaced by new JScrollPane(aTable).* |
| void | [**doLayout**](http://docs.google.com/javax/swing/JTable.html#doLayout())()            Causes this table to lay out its rows and columns. |
| boolean | [**editCellAt**](http://docs.google.com/javax/swing/JTable.html#editCellAt(int,%20int))(int row, int column)            Programmatically starts editing the cell at row and column, if those indices are in the valid range, and the cell at those indices is editable. |
| boolean | [**editCellAt**](http://docs.google.com/javax/swing/JTable.html#editCellAt(int,%20int,%20java.util.EventObject))(int row, int column, [EventObject](http://docs.google.com/java/util/EventObject.html) e)            Programmatically starts editing the cell at row and column, if those indices are in the valid range, and the cell at those indices is editable. |
| void | [**editingCanceled**](http://docs.google.com/javax/swing/JTable.html#editingCanceled(javax.swing.event.ChangeEvent))([ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) e)            Invoked when editing is canceled. |
| void | [**editingStopped**](http://docs.google.com/javax/swing/JTable.html#editingStopped(javax.swing.event.ChangeEvent))([ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) e)            Invoked when editing is finished. |
| [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) | [**getAccessibleContext**](http://docs.google.com/javax/swing/JTable.html#getAccessibleContext())()            Gets the AccessibleContext associated with this JTable. |
| boolean | [**getAutoCreateColumnsFromModel**](http://docs.google.com/javax/swing/JTable.html#getAutoCreateColumnsFromModel())()            Determines whether the table will create default columns from the model. |
| boolean | [**getAutoCreateRowSorter**](http://docs.google.com/javax/swing/JTable.html#getAutoCreateRowSorter())()            Returns true if whenever the model changes, a new RowSorter should be created and installed as the table's sorter; otherwise, returns false. |
| int | [**getAutoResizeMode**](http://docs.google.com/javax/swing/JTable.html#getAutoResizeMode())()            Returns the auto resize mode of the table. |
| [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) | [**getCellEditor**](http://docs.google.com/javax/swing/JTable.html#getCellEditor())()            Returns the active cell editor, which is null if the table is not currently editing. |
| [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) | [**getCellEditor**](http://docs.google.com/javax/swing/JTable.html#getCellEditor(int,%20int))(int row, int column)            Returns an appropriate editor for the cell specified by row and column. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getCellRect**](http://docs.google.com/javax/swing/JTable.html#getCellRect(int,%20int,%20boolean))(int row, int column, boolean includeSpacing)            Returns a rectangle for the cell that lies at the intersection of row and column. |
| [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) | [**getCellRenderer**](http://docs.google.com/javax/swing/JTable.html#getCellRenderer(int,%20int))(int row, int column)            Returns an appropriate renderer for the cell specified by this row and column. |
| boolean | [**getCellSelectionEnabled**](http://docs.google.com/javax/swing/JTable.html#getCellSelectionEnabled())()            Returns true if both row and column selection models are enabled. |
| [TableColumn](http://docs.google.com/javax/swing/table/TableColumn.html) | [**getColumn**](http://docs.google.com/javax/swing/JTable.html#getColumn(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) identifier)            Returns the TableColumn object for the column in the table whose identifier is equal to identifier, when compared using equals. |
| [Class](http://docs.google.com/java/lang/Class.html)<?> | [**getColumnClass**](http://docs.google.com/javax/swing/JTable.html#getColumnClass(int))(int column)            Returns the type of the column appearing in the view at column position column. |
| int | [**getColumnCount**](http://docs.google.com/javax/swing/JTable.html#getColumnCount())()            Returns the number of columns in the column model. |
| [TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) | [**getColumnModel**](http://docs.google.com/javax/swing/JTable.html#getColumnModel())()            Returns the TableColumnModel that contains all column information of this table. |
| [String](http://docs.google.com/java/lang/String.html) | [**getColumnName**](http://docs.google.com/javax/swing/JTable.html#getColumnName(int))(int column)            Returns the name of the column appearing in the view at column position column. |
| boolean | [**getColumnSelectionAllowed**](http://docs.google.com/javax/swing/JTable.html#getColumnSelectionAllowed())()            Returns true if columns can be selected. |
| [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) | [**getDefaultEditor**](http://docs.google.com/javax/swing/JTable.html#getDefaultEditor(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass)            Returns the editor to be used when no editor has been set in a TableColumn. |
| [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) | [**getDefaultRenderer**](http://docs.google.com/javax/swing/JTable.html#getDefaultRenderer(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass)            Returns the cell renderer to be used when no renderer has been set in a TableColumn. |
| boolean | [**getDragEnabled**](http://docs.google.com/javax/swing/JTable.html#getDragEnabled())()            Returns whether or not automatic drag handling is enabled. |
| [JTable.DropLocation](http://docs.google.com/javax/swing/JTable.DropLocation.html) | [**getDropLocation**](http://docs.google.com/javax/swing/JTable.html#getDropLocation())()            Returns the location that this component should visually indicate as the drop location during a DnD operation over the component, or null if no location is to currently be shown. |
| [DropMode](http://docs.google.com/javax/swing/DropMode.html) | [**getDropMode**](http://docs.google.com/javax/swing/JTable.html#getDropMode())()            Returns the drop mode for this component. |
| int | [**getEditingColumn**](http://docs.google.com/javax/swing/JTable.html#getEditingColumn())()            Returns the index of the column that contains the cell currently being edited. |
| int | [**getEditingRow**](http://docs.google.com/javax/swing/JTable.html#getEditingRow())()            Returns the index of the row that contains the cell currently being edited. |
| [Component](http://docs.google.com/java/awt/Component.html) | [**getEditorComponent**](http://docs.google.com/javax/swing/JTable.html#getEditorComponent())()            Returns the component that is handling the editing session. |
| boolean | [**getFillsViewportHeight**](http://docs.google.com/javax/swing/JTable.html#getFillsViewportHeight())()            Returns whether or not this table is always made large enough to fill the height of an enclosing viewport. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getGridColor**](http://docs.google.com/javax/swing/JTable.html#getGridColor())()            Returns the color used to draw grid lines. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getIntercellSpacing**](http://docs.google.com/javax/swing/JTable.html#getIntercellSpacing())()            Returns the horizontal and vertical space between cells. |
| [TableModel](http://docs.google.com/javax/swing/table/TableModel.html) | [**getModel**](http://docs.google.com/javax/swing/JTable.html#getModel())()            Returns the TableModel that provides the data displayed by this JTable. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredScrollableViewportSize**](http://docs.google.com/javax/swing/JTable.html#getPreferredScrollableViewportSize())()            Returns the preferred size of the viewport for this table. |
| [Printable](http://docs.google.com/java/awt/print/Printable.html) | [**getPrintable**](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat))([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)            Return a Printable for use in printing this JTable. |
| int | [**getRowCount**](http://docs.google.com/javax/swing/JTable.html#getRowCount())()            Returns the number of rows that can be shown in the JTable, given unlimited space. |
| int | [**getRowHeight**](http://docs.google.com/javax/swing/JTable.html#getRowHeight())()            Returns the height of a table row, in pixels. |
| int | [**getRowHeight**](http://docs.google.com/javax/swing/JTable.html#getRowHeight(int))(int row)            Returns the height, in pixels, of the cells in row. |
| int | [**getRowMargin**](http://docs.google.com/javax/swing/JTable.html#getRowMargin())()            Gets the amount of empty space, in pixels, between cells. |
| boolean | [**getRowSelectionAllowed**](http://docs.google.com/javax/swing/JTable.html#getRowSelectionAllowed())()            Returns true if rows can be selected. |
| [RowSorter](http://docs.google.com/javax/swing/RowSorter.html)<? extends [TableModel](http://docs.google.com/javax/swing/table/TableModel.html)> | [**getRowSorter**](http://docs.google.com/javax/swing/JTable.html#getRowSorter())()            Returns the object responsible for sorting. |
| int | [**getScrollableBlockIncrement**](http://docs.google.com/javax/swing/JTable.html#getScrollableBlockIncrement(java.awt.Rectangle,%20int,%20int))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect, int orientation, int direction)            Returns visibleRect.height or visibleRect.width, depending on this table's orientation. |
| boolean | [**getScrollableTracksViewportHeight**](http://docs.google.com/javax/swing/JTable.html#getScrollableTracksViewportHeight())()            Returns false to indicate that the height of the viewport does not determine the height of the table, unless getFillsViewportHeight is true and the preferred height of the table is smaller than the viewport's height. |
| boolean | [**getScrollableTracksViewportWidth**](http://docs.google.com/javax/swing/JTable.html#getScrollableTracksViewportWidth())()            Returns false if autoResizeMode is set to AUTO\_RESIZE\_OFF, which indicates that the width of the viewport does not determine the width of the table. |
| int | [**getScrollableUnitIncrement**](http://docs.google.com/javax/swing/JTable.html#getScrollableUnitIncrement(java.awt.Rectangle,%20int,%20int))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect, int orientation, int direction)            Returns the scroll increment (in pixels) that completely exposes one new row or column (depending on the orientation). |
| int | [**getSelectedColumn**](http://docs.google.com/javax/swing/JTable.html#getSelectedColumn())()            Returns the index of the first selected column, -1 if no column is selected. |
| int | [**getSelectedColumnCount**](http://docs.google.com/javax/swing/JTable.html#getSelectedColumnCount())()            Returns the number of selected columns. |
| int[] | [**getSelectedColumns**](http://docs.google.com/javax/swing/JTable.html#getSelectedColumns())()            Returns the indices of all selected columns. |
| int | [**getSelectedRow**](http://docs.google.com/javax/swing/JTable.html#getSelectedRow())()            Returns the index of the first selected row, -1 if no row is selected. |
| int | [**getSelectedRowCount**](http://docs.google.com/javax/swing/JTable.html#getSelectedRowCount())()            Returns the number of selected rows. |
| int[] | [**getSelectedRows**](http://docs.google.com/javax/swing/JTable.html#getSelectedRows())()            Returns the indices of all selected rows. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectionBackground**](http://docs.google.com/javax/swing/JTable.html#getSelectionBackground())()            Returns the background color for selected cells. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectionForeground**](http://docs.google.com/javax/swing/JTable.html#getSelectionForeground())()            Returns the foreground color for selected cells. |
| [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) | [**getSelectionModel**](http://docs.google.com/javax/swing/JTable.html#getSelectionModel())()            Returns the ListSelectionModel that is used to maintain row selection state. |
| boolean | [**getShowHorizontalLines**](http://docs.google.com/javax/swing/JTable.html#getShowHorizontalLines())()            Returns true if the table draws horizontal lines between cells, false if it doesn't. |
| boolean | [**getShowVerticalLines**](http://docs.google.com/javax/swing/JTable.html#getShowVerticalLines())()            Returns true if the table draws vertical lines between cells, false if it doesn't. |
| boolean | [**getSurrendersFocusOnKeystroke**](http://docs.google.com/javax/swing/JTable.html#getSurrendersFocusOnKeystroke())()            Returns true if the editor should get the focus when keystrokes cause the editor to be activated |
| [JTableHeader](http://docs.google.com/javax/swing/table/JTableHeader.html) | [**getTableHeader**](http://docs.google.com/javax/swing/JTable.html#getTableHeader())()            Returns the tableHeader used by this JTable. |
| [String](http://docs.google.com/java/lang/String.html) | [**getToolTipText**](http://docs.google.com/javax/swing/JTable.html#getToolTipText(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)            Overrides JComponent's getToolTipText method in order to allow the renderer's tips to be used if it has text set. |
| [TableUI](http://docs.google.com/javax/swing/plaf/TableUI.html) | [**getUI**](http://docs.google.com/javax/swing/JTable.html#getUI())()            Returns the L&F object that renders this component. |
| [String](http://docs.google.com/java/lang/String.html) | [**getUIClassID**](http://docs.google.com/javax/swing/JTable.html#getUIClassID())()            Returns the suffix used to construct the name of the L&F class used to render this component. |
| boolean | [**getUpdateSelectionOnSort**](http://docs.google.com/javax/swing/JTable.html#getUpdateSelectionOnSort())()            Returns true if the selection should be updated after sorting. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getValueAt**](http://docs.google.com/javax/swing/JTable.html#getValueAt(int,%20int))(int row, int column)            Returns the cell value at row and column. |
| protected  void | [**initializeLocalVars**](http://docs.google.com/javax/swing/JTable.html#initializeLocalVars())()            Initializes table properties to their default values. |
| boolean | [**isCellEditable**](http://docs.google.com/javax/swing/JTable.html#isCellEditable(int,%20int))(int row, int column)            Returns true if the cell at row and column is editable. |
| boolean | [**isCellSelected**](http://docs.google.com/javax/swing/JTable.html#isCellSelected(int,%20int))(int row, int column)            Returns true if the specified indices are in the valid range of rows and columns and the cell at the specified position is selected. |
| boolean | [**isColumnSelected**](http://docs.google.com/javax/swing/JTable.html#isColumnSelected(int))(int column)            Returns true if the specified index is in the valid range of columns, and the column at that index is selected. |
| boolean | [**isEditing**](http://docs.google.com/javax/swing/JTable.html#isEditing())()            Returns true if a cell is being edited. |
| boolean | [**isRowSelected**](http://docs.google.com/javax/swing/JTable.html#isRowSelected(int))(int row)            Returns true if the specified index is in the valid range of rows, and the row at that index is selected. |
| void | [**moveColumn**](http://docs.google.com/javax/swing/JTable.html#moveColumn(int,%20int))(int column, int targetColumn)            Moves the column column to the position currently occupied by the column targetColumn in the view. |
| protected  [String](http://docs.google.com/java/lang/String.html) | [**paramString**](http://docs.google.com/javax/swing/JTable.html#paramString())()            Returns a string representation of this table. |
| [Component](http://docs.google.com/java/awt/Component.html) | [**prepareEditor**](http://docs.google.com/javax/swing/JTable.html#prepareEditor(javax.swing.table.TableCellEditor,%20int,%20int))([TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) editor, int row, int column)            Prepares the editor by querying the data model for the value and selection state of the cell at row, column. |
| [Component](http://docs.google.com/java/awt/Component.html) | [**prepareRenderer**](http://docs.google.com/javax/swing/JTable.html#prepareRenderer(javax.swing.table.TableCellRenderer,%20int,%20int))([TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) renderer, int row, int column)            Prepares the renderer by querying the data model for the value and selection state of the cell at row, column. |
| boolean | [**print**](http://docs.google.com/javax/swing/JTable.html#print())()            A convenience method that displays a printing dialog, and then prints this JTable in mode PrintMode.FIT\_WIDTH, with no header or footer text. |
| boolean | [**print**](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode))([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode)            A convenience method that displays a printing dialog, and then prints this JTable in the given printing mode, with no header or footer text. |
| boolean | [**print**](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat))([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)            A convenience method that displays a printing dialog, and then prints this JTable in the given printing mode, with the specified header and footer text. |
| boolean | [**print**](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean))([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat, boolean showPrintDialog, [PrintRequestAttributeSet](http://docs.google.com/javax/print/attribute/PrintRequestAttributeSet.html) attr, boolean interactive)            Prints this table, as specified by the fully featured [print](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean,%20javax.print.PrintService)) method, with the default printer specified as the print service. |
| boolean | [**print**](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean,%20javax.print.PrintService))([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat, [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat, boolean showPrintDialog, [PrintRequestAttributeSet](http://docs.google.com/javax/print/attribute/PrintRequestAttributeSet.html) attr, boolean interactive, [PrintService](http://docs.google.com/javax/print/PrintService.html) service)            Prints this JTable. |
| protected  boolean | [**processKeyBinding**](http://docs.google.com/javax/swing/JTable.html#processKeyBinding(javax.swing.KeyStroke,%20java.awt.event.KeyEvent,%20int,%20boolean))([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) ks, [KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e, int condition, boolean pressed)            Invoked to process the key bindings for ks as the result of the KeyEvent e. |
| void | [**removeColumn**](http://docs.google.com/javax/swing/JTable.html#removeColumn(javax.swing.table.TableColumn))([TableColumn](http://docs.google.com/javax/swing/table/TableColumn.html) aColumn)            Removes aColumn from this JTable's array of columns. |
| void | [**removeColumnSelectionInterval**](http://docs.google.com/javax/swing/JTable.html#removeColumnSelectionInterval(int,%20int))(int index0, int index1)            Deselects the columns from index0 to index1, inclusive. |
| void | [**removeEditor**](http://docs.google.com/javax/swing/JTable.html#removeEditor())()            Discards the editor object and frees the real estate it used for cell rendering. |
| void | [**removeNotify**](http://docs.google.com/javax/swing/JTable.html#removeNotify())()            Calls the unconfigureEnclosingScrollPane method. |
| void | [**removeRowSelectionInterval**](http://docs.google.com/javax/swing/JTable.html#removeRowSelectionInterval(int,%20int))(int index0, int index1)            Deselects the rows from index0 to index1, inclusive. |
| protected  void | [**resizeAndRepaint**](http://docs.google.com/javax/swing/JTable.html#resizeAndRepaint())()            Equivalent to revalidate followed by repaint. |
| int | [**rowAtPoint**](http://docs.google.com/javax/swing/JTable.html#rowAtPoint(java.awt.Point))([Point](http://docs.google.com/java/awt/Point.html) point)            Returns the index of the row that point lies in, or -1 if the result is not in the range [0, getRowCount()-1]. |
| void | [**selectAll**](http://docs.google.com/javax/swing/JTable.html#selectAll())()            Selects all rows, columns, and cells in the table. |
| void | [**setAutoCreateColumnsFromModel**](http://docs.google.com/javax/swing/JTable.html#setAutoCreateColumnsFromModel(boolean))(boolean autoCreateColumnsFromModel)            Sets this table's autoCreateColumnsFromModel flag. |
| void | [**setAutoCreateRowSorter**](http://docs.google.com/javax/swing/JTable.html#setAutoCreateRowSorter(boolean))(boolean autoCreateRowSorter)            Specifies whether a RowSorter should be created for the table whenever its model changes. |
| void | [**setAutoResizeMode**](http://docs.google.com/javax/swing/JTable.html#setAutoResizeMode(int))(int mode)            Sets the table's auto resize mode when the table is resized. |
| void | [**setCellEditor**](http://docs.google.com/javax/swing/JTable.html#setCellEditor(javax.swing.table.TableCellEditor))([TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) anEditor)            Sets the active cell editor. |
| void | [**setCellSelectionEnabled**](http://docs.google.com/javax/swing/JTable.html#setCellSelectionEnabled(boolean))(boolean cellSelectionEnabled)            Sets whether this table allows both a column selection and a row selection to exist simultaneously. |
| void | [**setColumnModel**](http://docs.google.com/javax/swing/JTable.html#setColumnModel(javax.swing.table.TableColumnModel))([TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) columnModel)            Sets the column model for this table to newModel and registers for listener notifications from the new column model. |
| void | [**setColumnSelectionAllowed**](http://docs.google.com/javax/swing/JTable.html#setColumnSelectionAllowed(boolean))(boolean columnSelectionAllowed)            Sets whether the columns in this model can be selected. |
| void | [**setColumnSelectionInterval**](http://docs.google.com/javax/swing/JTable.html#setColumnSelectionInterval(int,%20int))(int index0, int index1)            Selects the columns from index0 to index1, inclusive. |
| void | [**setDefaultEditor**](http://docs.google.com/javax/swing/JTable.html#setDefaultEditor(java.lang.Class,%20javax.swing.table.TableCellEditor))([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass, [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) editor)            Sets a default cell editor to be used if no editor has been set in a TableColumn. |
| void | [**setDefaultRenderer**](http://docs.google.com/javax/swing/JTable.html#setDefaultRenderer(java.lang.Class,%20javax.swing.table.TableCellRenderer))([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass, [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) renderer)            Sets a default cell renderer to be used if no renderer has been set in a TableColumn. |
| void | [**setDragEnabled**](http://docs.google.com/javax/swing/JTable.html#setDragEnabled(boolean))(boolean b)            Turns on or off automatic drag handling. |
| void | [**setDropMode**](http://docs.google.com/javax/swing/JTable.html#setDropMode(javax.swing.DropMode))([DropMode](http://docs.google.com/javax/swing/DropMode.html) dropMode)            Sets the drop mode for this component. |
| void | [**setEditingColumn**](http://docs.google.com/javax/swing/JTable.html#setEditingColumn(int))(int aColumn)            Sets the editingColumn variable. |
| void | [**setEditingRow**](http://docs.google.com/javax/swing/JTable.html#setEditingRow(int))(int aRow)            Sets the editingRow variable. |
| void | [**setFillsViewportHeight**](http://docs.google.com/javax/swing/JTable.html#setFillsViewportHeight(boolean))(boolean fillsViewportHeight)            Sets whether or not this table is always made large enough to fill the height of an enclosing viewport. |
| void | [**setGridColor**](http://docs.google.com/javax/swing/JTable.html#setGridColor(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) gridColor)            Sets the color used to draw grid lines to gridColor and redisplays. |
| void | [**setIntercellSpacing**](http://docs.google.com/javax/swing/JTable.html#setIntercellSpacing(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) intercellSpacing)            Sets the rowMargin and the columnMargin -- the height and width of the space between cells -- to intercellSpacing. |
| void | [**setModel**](http://docs.google.com/javax/swing/JTable.html#setModel(javax.swing.table.TableModel))([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dataModel)            Sets the data model for this table to newModel and registers with it for listener notifications from the new data model. |
| void | [**setPreferredScrollableViewportSize**](http://docs.google.com/javax/swing/JTable.html#setPreferredScrollableViewportSize(java.awt.Dimension))([Dimension](http://docs.google.com/java/awt/Dimension.html) size)            Sets the preferred size of the viewport for this table. |
| void | [**setRowHeight**](http://docs.google.com/javax/swing/JTable.html#setRowHeight(int))(int rowHeight)            Sets the height, in pixels, of all cells to rowHeight, revalidates, and repaints. |
| void | [**setRowHeight**](http://docs.google.com/javax/swing/JTable.html#setRowHeight(int,%20int))(int row, int rowHeight)            Sets the height for row to rowHeight, revalidates, and repaints. |
| void | [**setRowMargin**](http://docs.google.com/javax/swing/JTable.html#setRowMargin(int))(int rowMargin)            Sets the amount of empty space between cells in adjacent rows. |
| void | [**setRowSelectionAllowed**](http://docs.google.com/javax/swing/JTable.html#setRowSelectionAllowed(boolean))(boolean rowSelectionAllowed)            Sets whether the rows in this model can be selected. |
| void | [**setRowSelectionInterval**](http://docs.google.com/javax/swing/JTable.html#setRowSelectionInterval(int,%20int))(int index0, int index1)            Selects the rows from index0 to index1, inclusive. |
| void | [**setRowSorter**](http://docs.google.com/javax/swing/JTable.html#setRowSorter(javax.swing.RowSorter))([RowSorter](http://docs.google.com/javax/swing/RowSorter.html)<? extends [TableModel](http://docs.google.com/javax/swing/table/TableModel.html)> sorter)            Sets the RowSorter. |
| void | [**setSelectionBackground**](http://docs.google.com/javax/swing/JTable.html#setSelectionBackground(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) selectionBackground)            Sets the background color for selected cells. |
| void | [**setSelectionForeground**](http://docs.google.com/javax/swing/JTable.html#setSelectionForeground(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) selectionForeground)            Sets the foreground color for selected cells. |
| void | [**setSelectionMode**](http://docs.google.com/javax/swing/JTable.html#setSelectionMode(int))(int selectionMode)            Sets the table's selection mode to allow only single selections, a single contiguous interval, or multiple intervals. |
| void | [**setSelectionModel**](http://docs.google.com/javax/swing/JTable.html#setSelectionModel(javax.swing.ListSelectionModel))([ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) newModel)            Sets the row selection model for this table to newModel and registers for listener notifications from the new selection model. |
| void | [**setShowGrid**](http://docs.google.com/javax/swing/JTable.html#setShowGrid(boolean))(boolean showGrid)            Sets whether the table draws grid lines around cells. |
| void | [**setShowHorizontalLines**](http://docs.google.com/javax/swing/JTable.html#setShowHorizontalLines(boolean))(boolean showHorizontalLines)            Sets whether the table draws horizontal lines between cells. |
| void | [**setShowVerticalLines**](http://docs.google.com/javax/swing/JTable.html#setShowVerticalLines(boolean))(boolean showVerticalLines)            Sets whether the table draws vertical lines between cells. |
| void | [**setSurrendersFocusOnKeystroke**](http://docs.google.com/javax/swing/JTable.html#setSurrendersFocusOnKeystroke(boolean))(boolean surrendersFocusOnKeystroke)            Sets whether editors in this JTable get the keyboard focus when an editor is activated as a result of the JTable forwarding keyboard events for a cell. |
| void | [**setTableHeader**](http://docs.google.com/javax/swing/JTable.html#setTableHeader(javax.swing.table.JTableHeader))([JTableHeader](http://docs.google.com/javax/swing/table/JTableHeader.html) tableHeader)            Sets the tableHeader working with this JTable to newHeader. |
| void | [**setUI**](http://docs.google.com/javax/swing/JTable.html#setUI(javax.swing.plaf.TableUI))([TableUI](http://docs.google.com/javax/swing/plaf/TableUI.html) ui)            Sets the L&F object that renders this component and repaints. |
| void | [**setUpdateSelectionOnSort**](http://docs.google.com/javax/swing/JTable.html#setUpdateSelectionOnSort(boolean))(boolean update)            Specifies whether the selection should be updated after sorting. |
| void | [**setValueAt**](http://docs.google.com/javax/swing/JTable.html#setValueAt(java.lang.Object,%20int,%20int))([Object](http://docs.google.com/java/lang/Object.html) aValue, int row, int column)            Sets the value for the cell in the table model at row and column. |
| void | [**sizeColumnsToFit**](http://docs.google.com/javax/swing/JTable.html#sizeColumnsToFit(boolean))(boolean lastColumnOnly)  **Deprecated.** *As of Swing version 1.0.3, replaced by doLayout().* |
| void | [**sizeColumnsToFit**](http://docs.google.com/javax/swing/JTable.html#sizeColumnsToFit(int))(int resizingColumn)            Obsolete as of Java 2 platform v1.4. |
| void | [**sorterChanged**](http://docs.google.com/javax/swing/JTable.html#sorterChanged(javax.swing.event.RowSorterEvent))([RowSorterEvent](http://docs.google.com/javax/swing/event/RowSorterEvent.html) e)            RowSorterListener notification that the RowSorter has changed in some way. |
| void | [**tableChanged**](http://docs.google.com/javax/swing/JTable.html#tableChanged(javax.swing.event.TableModelEvent))([TableModelEvent](http://docs.google.com/javax/swing/event/TableModelEvent.html) e)            Invoked when this table's TableModel generates a TableModelEvent. |
| protected  void | [**unconfigureEnclosingScrollPane**](http://docs.google.com/javax/swing/JTable.html#unconfigureEnclosingScrollPane())()            Reverses the effect of configureEnclosingScrollPane by replacing the columnHeaderView of the enclosing scroll pane with null. |
| void | [**updateUI**](http://docs.google.com/javax/swing/JTable.html#updateUI())()            Notification from the UIManager that the L&F has changed. |
| void | [**valueChanged**](http://docs.google.com/javax/swing/JTable.html#valueChanged(javax.swing.event.ListSelectionEvent))([ListSelectionEvent](http://docs.google.com/javax/swing/event/ListSelectionEvent.html) e)            Invoked when the row selection changes -- repaints to show the new selection. |

| **Methods inherited from class javax.swing.**[**JComponent**](http://docs.google.com/javax/swing/JComponent.html) |
| --- |
| [addAncestorListener](http://docs.google.com/javax/swing/JComponent.html#addAncestorListener(javax.swing.event.AncestorListener)), [addVetoableChangeListener](http://docs.google.com/javax/swing/JComponent.html#addVetoableChangeListener(java.beans.VetoableChangeListener)), [computeVisibleRect](http://docs.google.com/javax/swing/JComponent.html#computeVisibleRect(java.awt.Rectangle)), [contains](http://docs.google.com/javax/swing/JComponent.html#contains(int,%20int)), [createToolTip](http://docs.google.com/javax/swing/JComponent.html#createToolTip()), [disable](http://docs.google.com/javax/swing/JComponent.html#disable()), [enable](http://docs.google.com/javax/swing/JComponent.html#enable()), [firePropertyChange](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20boolean,%20boolean)), [firePropertyChange](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20char,%20char)), [firePropertyChange](http://docs.google.com/javax/swing/JComponent.html#firePropertyChange(java.lang.String,%20int,%20int)), [fireVetoableChange](http://docs.google.com/javax/swing/JComponent.html#fireVetoableChange(java.lang.String,%20java.lang.Object,%20java.lang.Object)), [getActionForKeyStroke](http://docs.google.com/javax/swing/JComponent.html#getActionForKeyStroke(javax.swing.KeyStroke)), [getActionMap](http://docs.google.com/javax/swing/JComponent.html#getActionMap()), [getAlignmentX](http://docs.google.com/javax/swing/JComponent.html#getAlignmentX()), [getAlignmentY](http://docs.google.com/javax/swing/JComponent.html#getAlignmentY()), [getAncestorListeners](http://docs.google.com/javax/swing/JComponent.html#getAncestorListeners()), [getAutoscrolls](http://docs.google.com/javax/swing/JComponent.html#getAutoscrolls()), [getBaseline](http://docs.google.com/javax/swing/JComponent.html#getBaseline(int,%20int)), [getBaselineResizeBehavior](http://docs.google.com/javax/swing/JComponent.html#getBaselineResizeBehavior()), [getBorder](http://docs.google.com/javax/swing/JComponent.html#getBorder()), [getBounds](http://docs.google.com/javax/swing/JComponent.html#getBounds(java.awt.Rectangle)), [getClientProperty](http://docs.google.com/javax/swing/JComponent.html#getClientProperty(java.lang.Object)), [getComponentGraphics](http://docs.google.com/javax/swing/JComponent.html#getComponentGraphics(java.awt.Graphics)), [getComponentPopupMenu](http://docs.google.com/javax/swing/JComponent.html#getComponentPopupMenu()), [getConditionForKeyStroke](http://docs.google.com/javax/swing/JComponent.html#getConditionForKeyStroke(javax.swing.KeyStroke)), [getDebugGraphicsOptions](http://docs.google.com/javax/swing/JComponent.html#getDebugGraphicsOptions()), [getDefaultLocale](http://docs.google.com/javax/swing/JComponent.html#getDefaultLocale()), [getFontMetrics](http://docs.google.com/javax/swing/JComponent.html#getFontMetrics(java.awt.Font)), [getGraphics](http://docs.google.com/javax/swing/JComponent.html#getGraphics()), [getHeight](http://docs.google.com/javax/swing/JComponent.html#getHeight()), [getInheritsPopupMenu](http://docs.google.com/javax/swing/JComponent.html#getInheritsPopupMenu()), [getInputMap](http://docs.google.com/javax/swing/JComponent.html#getInputMap()), [getInputMap](http://docs.google.com/javax/swing/JComponent.html#getInputMap(int)), [getInputVerifier](http://docs.google.com/javax/swing/JComponent.html#getInputVerifier()), [getInsets](http://docs.google.com/javax/swing/JComponent.html#getInsets()), [getInsets](http://docs.google.com/javax/swing/JComponent.html#getInsets(java.awt.Insets)), [getListeners](http://docs.google.com/javax/swing/JComponent.html#getListeners(java.lang.Class)), [getLocation](http://docs.google.com/javax/swing/JComponent.html#getLocation(java.awt.Point)), [getMaximumSize](http://docs.google.com/javax/swing/JComponent.html#getMaximumSize()), [getMinimumSize](http://docs.google.com/javax/swing/JComponent.html#getMinimumSize()), [getNextFocusableComponent](http://docs.google.com/javax/swing/JComponent.html#getNextFocusableComponent()), [getPopupLocation](http://docs.google.com/javax/swing/JComponent.html#getPopupLocation(java.awt.event.MouseEvent)), [getPreferredSize](http://docs.google.com/javax/swing/JComponent.html#getPreferredSize()), [getRegisteredKeyStrokes](http://docs.google.com/javax/swing/JComponent.html#getRegisteredKeyStrokes()), [getRootPane](http://docs.google.com/javax/swing/JComponent.html#getRootPane()), [getSize](http://docs.google.com/javax/swing/JComponent.html#getSize(java.awt.Dimension)), [getToolTipLocation](http://docs.google.com/javax/swing/JComponent.html#getToolTipLocation(java.awt.event.MouseEvent)), [getToolTipText](http://docs.google.com/javax/swing/JComponent.html#getToolTipText()), [getTopLevelAncestor](http://docs.google.com/javax/swing/JComponent.html#getTopLevelAncestor()), [getTransferHandler](http://docs.google.com/javax/swing/JComponent.html#getTransferHandler()), [getVerifyInputWhenFocusTarget](http://docs.google.com/javax/swing/JComponent.html#getVerifyInputWhenFocusTarget()), [getVetoableChangeListeners](http://docs.google.com/javax/swing/JComponent.html#getVetoableChangeListeners()), [getVisibleRect](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect()), [getWidth](http://docs.google.com/javax/swing/JComponent.html#getWidth()), [getX](http://docs.google.com/javax/swing/JComponent.html#getX()), [getY](http://docs.google.com/javax/swing/JComponent.html#getY()), [grabFocus](http://docs.google.com/javax/swing/JComponent.html#grabFocus()), [isDoubleBuffered](http://docs.google.com/javax/swing/JComponent.html#isDoubleBuffered()), [isLightweightComponent](http://docs.google.com/javax/swing/JComponent.html#isLightweightComponent(java.awt.Component)), [isManagingFocus](http://docs.google.com/javax/swing/JComponent.html#isManagingFocus()), [isOpaque](http://docs.google.com/javax/swing/JComponent.html#isOpaque()), [isOptimizedDrawingEnabled](http://docs.google.com/javax/swing/JComponent.html#isOptimizedDrawingEnabled()), [isPaintingForPrint](http://docs.google.com/javax/swing/JComponent.html#isPaintingForPrint()), [isPaintingTile](http://docs.google.com/javax/swing/JComponent.html#isPaintingTile()), [isRequestFocusEnabled](http://docs.google.com/javax/swing/JComponent.html#isRequestFocusEnabled()), [isValidateRoot](http://docs.google.com/javax/swing/JComponent.html#isValidateRoot()), [paint](http://docs.google.com/javax/swing/JComponent.html#paint(java.awt.Graphics)), [paintBorder](http://docs.google.com/javax/swing/JComponent.html#paintBorder(java.awt.Graphics)), [paintChildren](http://docs.google.com/javax/swing/JComponent.html#paintChildren(java.awt.Graphics)), [paintComponent](http://docs.google.com/javax/swing/JComponent.html#paintComponent(java.awt.Graphics)), [paintImmediately](http://docs.google.com/javax/swing/JComponent.html#paintImmediately(int,%20int,%20int,%20int)), [paintImmediately](http://docs.google.com/javax/swing/JComponent.html#paintImmediately(java.awt.Rectangle)), [print](http://docs.google.com/javax/swing/JComponent.html#print(java.awt.Graphics)), [printAll](http://docs.google.com/javax/swing/JComponent.html#printAll(java.awt.Graphics)), [printBorder](http://docs.google.com/javax/swing/JComponent.html#printBorder(java.awt.Graphics)), [printChildren](http://docs.google.com/javax/swing/JComponent.html#printChildren(java.awt.Graphics)), [printComponent](http://docs.google.com/javax/swing/JComponent.html#printComponent(java.awt.Graphics)), [processComponentKeyEvent](http://docs.google.com/javax/swing/JComponent.html#processComponentKeyEvent(java.awt.event.KeyEvent)), [processKeyEvent](http://docs.google.com/javax/swing/JComponent.html#processKeyEvent(java.awt.event.KeyEvent)), [processMouseEvent](http://docs.google.com/javax/swing/JComponent.html#processMouseEvent(java.awt.event.MouseEvent)), [processMouseMotionEvent](http://docs.google.com/javax/swing/JComponent.html#processMouseMotionEvent(java.awt.event.MouseEvent)), [putClientProperty](http://docs.google.com/javax/swing/JComponent.html#putClientProperty(java.lang.Object,%20java.lang.Object)), [registerKeyboardAction](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20javax.swing.KeyStroke,%20int)), [registerKeyboardAction](http://docs.google.com/javax/swing/JComponent.html#registerKeyboardAction(java.awt.event.ActionListener,%20java.lang.String,%20javax.swing.KeyStroke,%20int)), [removeAncestorListener](http://docs.google.com/javax/swing/JComponent.html#removeAncestorListener(javax.swing.event.AncestorListener)), [removeVetoableChangeListener](http://docs.google.com/javax/swing/JComponent.html#removeVetoableChangeListener(java.beans.VetoableChangeListener)), [repaint](http://docs.google.com/javax/swing/JComponent.html#repaint(long,%20int,%20int,%20int,%20int)), [repaint](http://docs.google.com/javax/swing/JComponent.html#repaint(java.awt.Rectangle)), [requestDefaultFocus](http://docs.google.com/javax/swing/JComponent.html#requestDefaultFocus()), [requestFocus](http://docs.google.com/javax/swing/JComponent.html#requestFocus()), [requestFocus](http://docs.google.com/javax/swing/JComponent.html#requestFocus(boolean)), [requestFocusInWindow](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow()), [requestFocusInWindow](http://docs.google.com/javax/swing/JComponent.html#requestFocusInWindow(boolean)), [resetKeyboardActions](http://docs.google.com/javax/swing/JComponent.html#resetKeyboardActions()), [reshape](http://docs.google.com/javax/swing/JComponent.html#reshape(int,%20int,%20int,%20int)), [revalidate](http://docs.google.com/javax/swing/JComponent.html#revalidate()), [scrollRectToVisible](http://docs.google.com/javax/swing/JComponent.html#scrollRectToVisible(java.awt.Rectangle)), [setActionMap](http://docs.google.com/javax/swing/JComponent.html#setActionMap(javax.swing.ActionMap)), [setAlignmentX](http://docs.google.com/javax/swing/JComponent.html#setAlignmentX(float)), [setAlignmentY](http://docs.google.com/javax/swing/JComponent.html#setAlignmentY(float)), [setAutoscrolls](http://docs.google.com/javax/swing/JComponent.html#setAutoscrolls(boolean)), [setBackground](http://docs.google.com/javax/swing/JComponent.html#setBackground(java.awt.Color)), [setBorder](http://docs.google.com/javax/swing/JComponent.html#setBorder(javax.swing.border.Border)), [setComponentPopupMenu](http://docs.google.com/javax/swing/JComponent.html#setComponentPopupMenu(javax.swing.JPopupMenu)), [setDebugGraphicsOptions](http://docs.google.com/javax/swing/JComponent.html#setDebugGraphicsOptions(int)), [setDefaultLocale](http://docs.google.com/javax/swing/JComponent.html#setDefaultLocale(java.util.Locale)), [setDoubleBuffered](http://docs.google.com/javax/swing/JComponent.html#setDoubleBuffered(boolean)), [setEnabled](http://docs.google.com/javax/swing/JComponent.html#setEnabled(boolean)), [setFocusTraversalKeys](http://docs.google.com/javax/swing/JComponent.html#setFocusTraversalKeys(int,%20java.util.Set)), [setFont](http://docs.google.com/javax/swing/JComponent.html#setFont(java.awt.Font)), [setForeground](http://docs.google.com/javax/swing/JComponent.html#setForeground(java.awt.Color)), [setInheritsPopupMenu](http://docs.google.com/javax/swing/JComponent.html#setInheritsPopupMenu(boolean)), [setInputMap](http://docs.google.com/javax/swing/JComponent.html#setInputMap(int,%20javax.swing.InputMap)), [setInputVerifier](http://docs.google.com/javax/swing/JComponent.html#setInputVerifier(javax.swing.InputVerifier)), [setMaximumSize](http://docs.google.com/javax/swing/JComponent.html#setMaximumSize(java.awt.Dimension)), [setMinimumSize](http://docs.google.com/javax/swing/JComponent.html#setMinimumSize(java.awt.Dimension)), [setNextFocusableComponent](http://docs.google.com/javax/swing/JComponent.html#setNextFocusableComponent(java.awt.Component)), [setOpaque](http://docs.google.com/javax/swing/JComponent.html#setOpaque(boolean)), [setPreferredSize](http://docs.google.com/javax/swing/JComponent.html#setPreferredSize(java.awt.Dimension)), [setRequestFocusEnabled](http://docs.google.com/javax/swing/JComponent.html#setRequestFocusEnabled(boolean)), [setToolTipText](http://docs.google.com/javax/swing/JComponent.html#setToolTipText(java.lang.String)), [setTransferHandler](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler)), [setUI](http://docs.google.com/javax/swing/JComponent.html#setUI(javax.swing.plaf.ComponentUI)), [setVerifyInputWhenFocusTarget](http://docs.google.com/javax/swing/JComponent.html#setVerifyInputWhenFocusTarget(boolean)), [setVisible](http://docs.google.com/javax/swing/JComponent.html#setVisible(boolean)), [unregisterKeyboardAction](http://docs.google.com/javax/swing/JComponent.html#unregisterKeyboardAction(javax.swing.KeyStroke)), [update](http://docs.google.com/javax/swing/JComponent.html#update(java.awt.Graphics)) |

| **Methods inherited from class java.awt.**[**Container**](http://docs.google.com/java/awt/Container.html) |
| --- |
| [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20int)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20java.lang.Object)), [add](http://docs.google.com/java/awt/Container.html#add(java.awt.Component,%20java.lang.Object,%20int)), [add](http://docs.google.com/java/awt/Container.html#add(java.lang.String,%20java.awt.Component)), [addContainerListener](http://docs.google.com/java/awt/Container.html#addContainerListener(java.awt.event.ContainerListener)), [addImpl](http://docs.google.com/java/awt/Container.html#addImpl(java.awt.Component,%20java.lang.Object,%20int)), [addPropertyChangeListener](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.beans.PropertyChangeListener)), [addPropertyChangeListener](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener)), [applyComponentOrientation](http://docs.google.com/java/awt/Container.html#applyComponentOrientation(java.awt.ComponentOrientation)), [areFocusTraversalKeysSet](http://docs.google.com/java/awt/Container.html#areFocusTraversalKeysSet(int)), [countComponents](http://docs.google.com/java/awt/Container.html#countComponents()), [deliverEvent](http://docs.google.com/java/awt/Container.html#deliverEvent(java.awt.Event)), [findComponentAt](http://docs.google.com/java/awt/Container.html#findComponentAt(int,%20int)), [findComponentAt](http://docs.google.com/java/awt/Container.html#findComponentAt(java.awt.Point)), [getComponent](http://docs.google.com/java/awt/Container.html#getComponent(int)), [getComponentAt](http://docs.google.com/java/awt/Container.html#getComponentAt(int,%20int)), [getComponentAt](http://docs.google.com/java/awt/Container.html#getComponentAt(java.awt.Point)), [getComponentCount](http://docs.google.com/java/awt/Container.html#getComponentCount()), [getComponents](http://docs.google.com/java/awt/Container.html#getComponents()), [getComponentZOrder](http://docs.google.com/java/awt/Container.html#getComponentZOrder(java.awt.Component)), [getContainerListeners](http://docs.google.com/java/awt/Container.html#getContainerListeners()), [getFocusTraversalKeys](http://docs.google.com/java/awt/Container.html#getFocusTraversalKeys(int)), [getFocusTraversalPolicy](http://docs.google.com/java/awt/Container.html#getFocusTraversalPolicy()), [getLayout](http://docs.google.com/java/awt/Container.html#getLayout()), [getMousePosition](http://docs.google.com/java/awt/Container.html#getMousePosition(boolean)), [insets](http://docs.google.com/java/awt/Container.html#insets()), [invalidate](http://docs.google.com/java/awt/Container.html#invalidate()), [isAncestorOf](http://docs.google.com/java/awt/Container.html#isAncestorOf(java.awt.Component)), [isFocusCycleRoot](http://docs.google.com/java/awt/Container.html#isFocusCycleRoot()), [isFocusCycleRoot](http://docs.google.com/java/awt/Container.html#isFocusCycleRoot(java.awt.Container)), [isFocusTraversalPolicyProvider](http://docs.google.com/java/awt/Container.html#isFocusTraversalPolicyProvider()), [isFocusTraversalPolicySet](http://docs.google.com/java/awt/Container.html#isFocusTraversalPolicySet()), [layout](http://docs.google.com/java/awt/Container.html#layout()), [list](http://docs.google.com/java/awt/Container.html#list(java.io.PrintStream,%20int)), [list](http://docs.google.com/java/awt/Container.html#list(java.io.PrintWriter,%20int)), [locate](http://docs.google.com/java/awt/Container.html#locate(int,%20int)), [minimumSize](http://docs.google.com/java/awt/Container.html#minimumSize()), [paintComponents](http://docs.google.com/java/awt/Container.html#paintComponents(java.awt.Graphics)), [preferredSize](http://docs.google.com/java/awt/Container.html#preferredSize()), [printComponents](http://docs.google.com/java/awt/Container.html#printComponents(java.awt.Graphics)), [processContainerEvent](http://docs.google.com/java/awt/Container.html#processContainerEvent(java.awt.event.ContainerEvent)), [processEvent](http://docs.google.com/java/awt/Container.html#processEvent(java.awt.AWTEvent)), [remove](http://docs.google.com/java/awt/Container.html#remove(java.awt.Component)), [remove](http://docs.google.com/java/awt/Container.html#remove(int)), [removeAll](http://docs.google.com/java/awt/Container.html#removeAll()), [removeContainerListener](http://docs.google.com/java/awt/Container.html#removeContainerListener(java.awt.event.ContainerListener)), [setComponentZOrder](http://docs.google.com/java/awt/Container.html#setComponentZOrder(java.awt.Component,%20int)), [setFocusCycleRoot](http://docs.google.com/java/awt/Container.html#setFocusCycleRoot(boolean)), [setFocusTraversalPolicy](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicy(java.awt.FocusTraversalPolicy)), [setFocusTraversalPolicyProvider](http://docs.google.com/java/awt/Container.html#setFocusTraversalPolicyProvider(boolean)), [setLayout](http://docs.google.com/java/awt/Container.html#setLayout(java.awt.LayoutManager)), [transferFocusBackward](http://docs.google.com/java/awt/Container.html#transferFocusBackward()), [transferFocusDownCycle](http://docs.google.com/java/awt/Container.html#transferFocusDownCycle()), [validate](http://docs.google.com/java/awt/Container.html#validate()), [validateTree](http://docs.google.com/java/awt/Container.html#validateTree()) |

| **Methods inherited from class java.awt.**[**Component**](http://docs.google.com/java/awt/Component.html) |
| --- |
| [action](http://docs.google.com/java/awt/Component.html#action(java.awt.Event,%20java.lang.Object)), [add](http://docs.google.com/java/awt/Component.html#add(java.awt.PopupMenu)), [addComponentListener](http://docs.google.com/java/awt/Component.html#addComponentListener(java.awt.event.ComponentListener)), [addFocusListener](http://docs.google.com/java/awt/Component.html#addFocusListener(java.awt.event.FocusListener)), [addHierarchyBoundsListener](http://docs.google.com/java/awt/Component.html#addHierarchyBoundsListener(java.awt.event.HierarchyBoundsListener)), [addHierarchyListener](http://docs.google.com/java/awt/Component.html#addHierarchyListener(java.awt.event.HierarchyListener)), [addInputMethodListener](http://docs.google.com/java/awt/Component.html#addInputMethodListener(java.awt.event.InputMethodListener)), [addKeyListener](http://docs.google.com/java/awt/Component.html#addKeyListener(java.awt.event.KeyListener)), [addMouseListener](http://docs.google.com/java/awt/Component.html#addMouseListener(java.awt.event.MouseListener)), [addMouseMotionListener](http://docs.google.com/java/awt/Component.html#addMouseMotionListener(java.awt.event.MouseMotionListener)), [addMouseWheelListener](http://docs.google.com/java/awt/Component.html#addMouseWheelListener(java.awt.event.MouseWheelListener)), [bounds](http://docs.google.com/java/awt/Component.html#bounds()), [checkImage](http://docs.google.com/java/awt/Component.html#checkImage(java.awt.Image,%20java.awt.image.ImageObserver)), [checkImage](http://docs.google.com/java/awt/Component.html#checkImage(java.awt.Image,%20int,%20int,%20java.awt.image.ImageObserver)), [coalesceEvents](http://docs.google.com/java/awt/Component.html#coalesceEvents(java.awt.AWTEvent,%20java.awt.AWTEvent)), [contains](http://docs.google.com/java/awt/Component.html#contains(java.awt.Point)), [createImage](http://docs.google.com/java/awt/Component.html#createImage(java.awt.image.ImageProducer)), [createImage](http://docs.google.com/java/awt/Component.html#createImage(int,%20int)), [createVolatileImage](http://docs.google.com/java/awt/Component.html#createVolatileImage(int,%20int)), [createVolatileImage](http://docs.google.com/java/awt/Component.html#createVolatileImage(int,%20int,%20java.awt.ImageCapabilities)), [disableEvents](http://docs.google.com/java/awt/Component.html#disableEvents(long)), [dispatchEvent](http://docs.google.com/java/awt/Component.html#dispatchEvent(java.awt.AWTEvent)), [enable](http://docs.google.com/java/awt/Component.html#enable(boolean)), [enableEvents](http://docs.google.com/java/awt/Component.html#enableEvents(long)), [enableInputMethods](http://docs.google.com/java/awt/Component.html#enableInputMethods(boolean)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20byte,%20byte)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20double,%20double)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20float,%20float)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20long,%20long)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20java.lang.Object,%20java.lang.Object)), [firePropertyChange](http://docs.google.com/java/awt/Component.html#firePropertyChange(java.lang.String,%20short,%20short)), [getBackground](http://docs.google.com/java/awt/Component.html#getBackground()), [getBounds](http://docs.google.com/java/awt/Component.html#getBounds()), [getColorModel](http://docs.google.com/java/awt/Component.html#getColorModel()), [getComponentListeners](http://docs.google.com/java/awt/Component.html#getComponentListeners()), [getComponentOrientation](http://docs.google.com/java/awt/Component.html#getComponentOrientation()), [getCursor](http://docs.google.com/java/awt/Component.html#getCursor()), [getDropTarget](http://docs.google.com/java/awt/Component.html#getDropTarget()), [getFocusCycleRootAncestor](http://docs.google.com/java/awt/Component.html#getFocusCycleRootAncestor()), [getFocusListeners](http://docs.google.com/java/awt/Component.html#getFocusListeners()), [getFocusTraversalKeysEnabled](http://docs.google.com/java/awt/Component.html#getFocusTraversalKeysEnabled()), [getFont](http://docs.google.com/java/awt/Component.html#getFont()), [getForeground](http://docs.google.com/java/awt/Component.html#getForeground()), [getGraphicsConfiguration](http://docs.google.com/java/awt/Component.html#getGraphicsConfiguration()), [getHierarchyBoundsListeners](http://docs.google.com/java/awt/Component.html#getHierarchyBoundsListeners()), [getHierarchyListeners](http://docs.google.com/java/awt/Component.html#getHierarchyListeners()), [getIgnoreRepaint](http://docs.google.com/java/awt/Component.html#getIgnoreRepaint()), [getInputContext](http://docs.google.com/java/awt/Component.html#getInputContext()), [getInputMethodListeners](http://docs.google.com/java/awt/Component.html#getInputMethodListeners()), [getInputMethodRequests](http://docs.google.com/java/awt/Component.html#getInputMethodRequests()), [getKeyListeners](http://docs.google.com/java/awt/Component.html#getKeyListeners()), [getLocale](http://docs.google.com/java/awt/Component.html#getLocale()), [getLocation](http://docs.google.com/java/awt/Component.html#getLocation()), [getLocationOnScreen](http://docs.google.com/java/awt/Component.html#getLocationOnScreen()), [getMouseListeners](http://docs.google.com/java/awt/Component.html#getMouseListeners()), [getMouseMotionListeners](http://docs.google.com/java/awt/Component.html#getMouseMotionListeners()), [getMousePosition](http://docs.google.com/java/awt/Component.html#getMousePosition()), [getMouseWheelListeners](http://docs.google.com/java/awt/Component.html#getMouseWheelListeners()), [getName](http://docs.google.com/java/awt/Component.html#getName()), [getParent](http://docs.google.com/java/awt/Component.html#getParent()), [getPeer](http://docs.google.com/java/awt/Component.html#getPeer()), [getPropertyChangeListeners](http://docs.google.com/java/awt/Component.html#getPropertyChangeListeners()), [getPropertyChangeListeners](http://docs.google.com/java/awt/Component.html#getPropertyChangeListeners(java.lang.String)), [getSize](http://docs.google.com/java/awt/Component.html#getSize()), [getToolkit](http://docs.google.com/java/awt/Component.html#getToolkit()), [getTreeLock](http://docs.google.com/java/awt/Component.html#getTreeLock()), [gotFocus](http://docs.google.com/java/awt/Component.html#gotFocus(java.awt.Event,%20java.lang.Object)), [handleEvent](http://docs.google.com/java/awt/Component.html#handleEvent(java.awt.Event)), [hasFocus](http://docs.google.com/java/awt/Component.html#hasFocus()), [hide](http://docs.google.com/java/awt/Component.html#hide()), [imageUpdate](http://docs.google.com/java/awt/Component.html#imageUpdate(java.awt.Image,%20int,%20int,%20int,%20int,%20int)), [inside](http://docs.google.com/java/awt/Component.html#inside(int,%20int)), [isBackgroundSet](http://docs.google.com/java/awt/Component.html#isBackgroundSet()), [isCursorSet](http://docs.google.com/java/awt/Component.html#isCursorSet()), [isDisplayable](http://docs.google.com/java/awt/Component.html#isDisplayable()), [isEnabled](http://docs.google.com/java/awt/Component.html#isEnabled()), [isFocusable](http://docs.google.com/java/awt/Component.html#isFocusable()), [isFocusOwner](http://docs.google.com/java/awt/Component.html#isFocusOwner()), [isFocusTraversable](http://docs.google.com/java/awt/Component.html#isFocusTraversable()), [isFontSet](http://docs.google.com/java/awt/Component.html#isFontSet()), [isForegroundSet](http://docs.google.com/java/awt/Component.html#isForegroundSet()), [isLightweight](http://docs.google.com/java/awt/Component.html#isLightweight()), [isMaximumSizeSet](http://docs.google.com/java/awt/Component.html#isMaximumSizeSet()), [isMinimumSizeSet](http://docs.google.com/java/awt/Component.html#isMinimumSizeSet()), [isPreferredSizeSet](http://docs.google.com/java/awt/Component.html#isPreferredSizeSet()), [isShowing](http://docs.google.com/java/awt/Component.html#isShowing()), [isValid](http://docs.google.com/java/awt/Component.html#isValid()), [isVisible](http://docs.google.com/java/awt/Component.html#isVisible()), [keyDown](http://docs.google.com/java/awt/Component.html#keyDown(java.awt.Event,%20int)), [keyUp](http://docs.google.com/java/awt/Component.html#keyUp(java.awt.Event,%20int)), [list](http://docs.google.com/java/awt/Component.html#list()), [list](http://docs.google.com/java/awt/Component.html#list(java.io.PrintStream)), [list](http://docs.google.com/java/awt/Component.html#list(java.io.PrintWriter)), [location](http://docs.google.com/java/awt/Component.html#location()), [lostFocus](http://docs.google.com/java/awt/Component.html#lostFocus(java.awt.Event,%20java.lang.Object)), [mouseDown](http://docs.google.com/java/awt/Component.html#mouseDown(java.awt.Event,%20int,%20int)), [mouseDrag](http://docs.google.com/java/awt/Component.html#mouseDrag(java.awt.Event,%20int,%20int)), [mouseEnter](http://docs.google.com/java/awt/Component.html#mouseEnter(java.awt.Event,%20int,%20int)), [mouseExit](http://docs.google.com/java/awt/Component.html#mouseExit(java.awt.Event,%20int,%20int)), [mouseMove](http://docs.google.com/java/awt/Component.html#mouseMove(java.awt.Event,%20int,%20int)), [mouseUp](http://docs.google.com/java/awt/Component.html#mouseUp(java.awt.Event,%20int,%20int)), [move](http://docs.google.com/java/awt/Component.html#move(int,%20int)), [nextFocus](http://docs.google.com/java/awt/Component.html#nextFocus()), [paintAll](http://docs.google.com/java/awt/Component.html#paintAll(java.awt.Graphics)), [postEvent](http://docs.google.com/java/awt/Component.html#postEvent(java.awt.Event)), [prepareImage](http://docs.google.com/java/awt/Component.html#prepareImage(java.awt.Image,%20java.awt.image.ImageObserver)), [prepareImage](http://docs.google.com/java/awt/Component.html#prepareImage(java.awt.Image,%20int,%20int,%20java.awt.image.ImageObserver)), [processComponentEvent](http://docs.google.com/java/awt/Component.html#processComponentEvent(java.awt.event.ComponentEvent)), [processFocusEvent](http://docs.google.com/java/awt/Component.html#processFocusEvent(java.awt.event.FocusEvent)), [processHierarchyBoundsEvent](http://docs.google.com/java/awt/Component.html#processHierarchyBoundsEvent(java.awt.event.HierarchyEvent)), [processHierarchyEvent](http://docs.google.com/java/awt/Component.html#processHierarchyEvent(java.awt.event.HierarchyEvent)), [processInputMethodEvent](http://docs.google.com/java/awt/Component.html#processInputMethodEvent(java.awt.event.InputMethodEvent)), [processMouseWheelEvent](http://docs.google.com/java/awt/Component.html#processMouseWheelEvent(java.awt.event.MouseWheelEvent)), [remove](http://docs.google.com/java/awt/Component.html#remove(java.awt.MenuComponent)), [removeComponentListener](http://docs.google.com/java/awt/Component.html#removeComponentListener(java.awt.event.ComponentListener)), [removeFocusListener](http://docs.google.com/java/awt/Component.html#removeFocusListener(java.awt.event.FocusListener)), [removeHierarchyBoundsListener](http://docs.google.com/java/awt/Component.html#removeHierarchyBoundsListener(java.awt.event.HierarchyBoundsListener)), [removeHierarchyListener](http://docs.google.com/java/awt/Component.html#removeHierarchyListener(java.awt.event.HierarchyListener)), [removeInputMethodListener](http://docs.google.com/java/awt/Component.html#removeInputMethodListener(java.awt.event.InputMethodListener)), [removeKeyListener](http://docs.google.com/java/awt/Component.html#removeKeyListener(java.awt.event.KeyListener)), [removeMouseListener](http://docs.google.com/java/awt/Component.html#removeMouseListener(java.awt.event.MouseListener)), [removeMouseMotionListener](http://docs.google.com/java/awt/Component.html#removeMouseMotionListener(java.awt.event.MouseMotionListener)), [removeMouseWheelListener](http://docs.google.com/java/awt/Component.html#removeMouseWheelListener(java.awt.event.MouseWheelListener)), [removePropertyChangeListener](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.beans.PropertyChangeListener)), [removePropertyChangeListener](http://docs.google.com/java/awt/Component.html#removePropertyChangeListener(java.lang.String,%20java.beans.PropertyChangeListener)), [repaint](http://docs.google.com/java/awt/Component.html#repaint()), [repaint](http://docs.google.com/java/awt/Component.html#repaint(int,%20int,%20int,%20int)), [repaint](http://docs.google.com/java/awt/Component.html#repaint(long)), [resize](http://docs.google.com/java/awt/Component.html#resize(java.awt.Dimension)), [resize](http://docs.google.com/java/awt/Component.html#resize(int,%20int)), [setBounds](http://docs.google.com/java/awt/Component.html#setBounds(int,%20int,%20int,%20int)), [setBounds](http://docs.google.com/java/awt/Component.html#setBounds(java.awt.Rectangle)), [setComponentOrientation](http://docs.google.com/java/awt/Component.html#setComponentOrientation(java.awt.ComponentOrientation)), [setCursor](http://docs.google.com/java/awt/Component.html#setCursor(java.awt.Cursor)), [setDropTarget](http://docs.google.com/java/awt/Component.html#setDropTarget(java.awt.dnd.DropTarget)), [setFocusable](http://docs.google.com/java/awt/Component.html#setFocusable(boolean)), [setFocusTraversalKeysEnabled](http://docs.google.com/java/awt/Component.html#setFocusTraversalKeysEnabled(boolean)), [setIgnoreRepaint](http://docs.google.com/java/awt/Component.html#setIgnoreRepaint(boolean)), [setLocale](http://docs.google.com/java/awt/Component.html#setLocale(java.util.Locale)), [setLocation](http://docs.google.com/java/awt/Component.html#setLocation(int,%20int)), [setLocation](http://docs.google.com/java/awt/Component.html#setLocation(java.awt.Point)), [setName](http://docs.google.com/java/awt/Component.html#setName(java.lang.String)), [setSize](http://docs.google.com/java/awt/Component.html#setSize(java.awt.Dimension)), [setSize](http://docs.google.com/java/awt/Component.html#setSize(int,%20int)), [show](http://docs.google.com/java/awt/Component.html#show()), [show](http://docs.google.com/java/awt/Component.html#show(boolean)), [size](http://docs.google.com/java/awt/Component.html#size()), [toString](http://docs.google.com/java/awt/Component.html#toString()), [transferFocus](http://docs.google.com/java/awt/Component.html#transferFocus()), [transferFocusUpCycle](http://docs.google.com/java/awt/Component.html#transferFocusUpCycle()) |

| **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()), [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** |
| --- |

### AUTO\_RESIZE\_OFF

public static final int **AUTO\_RESIZE\_OFF**

Do not adjust column widths automatically; use a scrollbar.

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

### AUTO\_RESIZE\_NEXT\_COLUMN

public static final int **AUTO\_RESIZE\_NEXT\_COLUMN**

When a column is adjusted in the UI, adjust the next column the opposite way.

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

### AUTO\_RESIZE\_SUBSEQUENT\_COLUMNS

public static final int **AUTO\_RESIZE\_SUBSEQUENT\_COLUMNS**

During UI adjustment, change subsequent columns to preserve the total width; this is the default behavior.

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

### AUTO\_RESIZE\_LAST\_COLUMN

public static final int **AUTO\_RESIZE\_LAST\_COLUMN**

During all resize operations, apply adjustments to the last column only.

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

### AUTO\_RESIZE\_ALL\_COLUMNS

public static final int **AUTO\_RESIZE\_ALL\_COLUMNS**

During all resize operations, proportionately resize all columns.

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

### dataModel

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

The TableModel of the table.

### columnModel

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

The TableColumnModel of the table.

### selectionModel

protected [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) **selectionModel**

The ListSelectionModel of the table, used to keep track of row selections.

### tableHeader

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

The TableHeader working with the table.

### rowHeight

protected int **rowHeight**

The height in pixels of each row in the table.

### rowMargin

protected int **rowMargin**

The height in pixels of the margin between the cells in each row.

### gridColor

protected [Color](http://docs.google.com/java/awt/Color.html) **gridColor**

The color of the grid.

### showHorizontalLines

protected boolean **showHorizontalLines**

The table draws horizontal lines between cells if showHorizontalLines is true.

### showVerticalLines

protected boolean **showVerticalLines**

The table draws vertical lines between cells if showVerticalLines is true.

### autoResizeMode

protected int **autoResizeMode**

Determines if the table automatically resizes the width of the table's columns to take up the entire width of the table, and how it does the resizing.

### autoCreateColumnsFromModel

protected boolean **autoCreateColumnsFromModel**

The table will query the TableModel to build the default set of columns if this is true.

### preferredViewportSize

protected [Dimension](http://docs.google.com/java/awt/Dimension.html) **preferredViewportSize**

Used by the Scrollable interface to determine the initial visible area.

### rowSelectionAllowed

protected boolean **rowSelectionAllowed**

True if row selection is allowed in this table.

### cellSelectionEnabled

protected boolean **cellSelectionEnabled**

Obsolete as of Java 2 platform v1.3. Please use the rowSelectionAllowed property and the columnSelectionAllowed property of the columnModel instead. Or use the method getCellSelectionEnabled.

### editorComp

protected transient [Component](http://docs.google.com/java/awt/Component.html) **editorComp**

If editing, the Component that is handling the editing.

### cellEditor

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

The active cell editor object, that overwrites the screen real estate occupied by the current cell and allows the user to change its contents. null if the table isn't currently editing.

### editingColumn

protected transient int **editingColumn**

Identifies the column of the cell being edited.

### editingRow

protected transient int **editingRow**

Identifies the row of the cell being edited.

### defaultRenderersByColumnClass

protected transient [Hashtable](http://docs.google.com/java/util/Hashtable.html) **defaultRenderersByColumnClass**

A table of objects that display the contents of a cell, indexed by class as declared in getColumnClass in the TableModel interface.

### defaultEditorsByColumnClass

protected transient [Hashtable](http://docs.google.com/java/util/Hashtable.html) **defaultEditorsByColumnClass**

A table of objects that display and edit the contents of a cell, indexed by class as declared in getColumnClass in the TableModel interface.

### selectionForeground

protected [Color](http://docs.google.com/java/awt/Color.html) **selectionForeground**

The foreground color of selected cells.

### selectionBackground

protected [Color](http://docs.google.com/java/awt/Color.html) **selectionBackground**

The background color of selected cells.

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

### JTable

public **JTable**()

Constructs a default JTable that is initialized with a default data model, a default column model, and a default selection model.

**See Also:**[createDefaultDataModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultDataModel()), [createDefaultColumnModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultColumnModel()), [createDefaultSelectionModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultSelectionModel())

### JTable

public **JTable**([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dm)

Constructs a JTable that is initialized with dm as the data model, a default column model, and a default selection model.

**Parameters:**dm - the data model for the table**See Also:**[createDefaultColumnModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultColumnModel()), [createDefaultSelectionModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultSelectionModel())

### JTable

public **JTable**([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dm,  
 [TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) cm)

Constructs a JTable that is initialized with dm as the data model, cm as the column model, and a default selection model.

**Parameters:**dm - the data model for the tablecm - the column model for the table**See Also:**[createDefaultSelectionModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultSelectionModel())

### JTable

public **JTable**([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dm,  
 [TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) cm,  
 [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) sm)

Constructs a JTable that is initialized with dm as the data model, cm as the column model, and sm as the selection model. If any of the parameters are null this method will initialize the table with the corresponding default model. The autoCreateColumnsFromModel flag is set to false if cm is non-null, otherwise it is set to true and the column model is populated with suitable TableColumns for the columns in dm.

**Parameters:**dm - the data model for the tablecm - the column model for the tablesm - the row selection model for the table**See Also:**[createDefaultDataModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultDataModel()), [createDefaultColumnModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultColumnModel()), [createDefaultSelectionModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultSelectionModel())

### JTable

public **JTable**(int numRows,  
 int numColumns)

Constructs a JTable with numRows and numColumns of empty cells using DefaultTableModel. The columns will have names of the form "A", "B", "C", etc.

**Parameters:**numRows - the number of rows the table holdsnumColumns - the number of columns the table holds**See Also:**[DefaultTableModel](http://docs.google.com/javax/swing/table/DefaultTableModel.html)

### JTable

public **JTable**([Vector](http://docs.google.com/java/util/Vector.html) rowData,  
 [Vector](http://docs.google.com/java/util/Vector.html) columnNames)

Constructs a JTable to display the values in the Vector of Vectors, rowData, with column names, columnNames. The Vectors contained in rowData should contain the values for that row. In other words, the value of the cell at row 1, column 5 can be obtained with the following code:

((Vector)rowData.elementAt(1)).elementAt(5);

**Parameters:**rowData - the data for the new tablecolumnNames - names of each column

### JTable

public **JTable**([Object](http://docs.google.com/java/lang/Object.html)[][] rowData,  
 [Object](http://docs.google.com/java/lang/Object.html)[] columnNames)

Constructs a JTable to display the values in the two dimensional array, rowData, with column names, columnNames. rowData is an array of rows, so the value of the cell at row 1, column 5 can be obtained with the following code:

rowData[1][5];

All rows must be of the same length as columnNames.

**Parameters:**rowData - the data for the new tablecolumnNames - names of each column

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

### addNotify

public void **addNotify**()

Calls the configureEnclosingScrollPane method.

**Overrides:**[addNotify](http://docs.google.com/javax/swing/JComponent.html#addNotify()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **See Also:**[configureEnclosingScrollPane()](http://docs.google.com/javax/swing/JTable.html#configureEnclosingScrollPane())

### configureEnclosingScrollPane

protected void **configureEnclosingScrollPane**()

If this JTable is the viewportView of an enclosing JScrollPane (the usual situation), configure this ScrollPane by, amongst other things, installing the table's tableHeader as the columnHeaderView of the scroll pane. When a JTable is added to a JScrollPane in the usual way, using new JScrollPane(myTable), addNotify is called in the JTable (when the table is added to the viewport). JTable's addNotify method in turn calls this method, which is protected so that this default installation procedure can be overridden by a subclass.

**See Also:**[addNotify()](http://docs.google.com/javax/swing/JTable.html#addNotify())

### removeNotify

public void **removeNotify**()

Calls the unconfigureEnclosingScrollPane method.

**Overrides:**[removeNotify](http://docs.google.com/javax/swing/JComponent.html#removeNotify()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **See Also:**[unconfigureEnclosingScrollPane()](http://docs.google.com/javax/swing/JTable.html#unconfigureEnclosingScrollPane())

### unconfigureEnclosingScrollPane

protected void **unconfigureEnclosingScrollPane**()

Reverses the effect of configureEnclosingScrollPane by replacing the columnHeaderView of the enclosing scroll pane with null. JTable's removeNotify method calls this method, which is protected so that this default uninstallation procedure can be overridden by a subclass.

**Since:** 1.3 **See Also:**[removeNotify()](http://docs.google.com/javax/swing/JTable.html#removeNotify()), [configureEnclosingScrollPane()](http://docs.google.com/javax/swing/JTable.html#configureEnclosingScrollPane())

### createScrollPaneForTable

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public static [JScrollPane](http://docs.google.com/javax/swing/JScrollPane.html) **createScrollPaneForTable**([JTable](http://docs.google.com/javax/swing/JTable.html) aTable)

**Deprecated.** *As of Swing version 1.0.2, replaced by new JScrollPane(aTable).*

Equivalent to new JScrollPane(aTable).

### setTableHeader

public void **setTableHeader**([JTableHeader](http://docs.google.com/javax/swing/table/JTableHeader.html) tableHeader)

Sets the tableHeader working with this JTable to newHeader. It is legal to have a null tableHeader.

**Parameters:**tableHeader - new tableHeader**See Also:**[getTableHeader()](http://docs.google.com/javax/swing/JTable.html#getTableHeader())

### getTableHeader

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

Returns the tableHeader used by this JTable.

**Returns:**the tableHeader used by this table**See Also:**[setTableHeader(javax.swing.table.JTableHeader)](http://docs.google.com/javax/swing/JTable.html#setTableHeader(javax.swing.table.JTableHeader))

### setRowHeight

public void **setRowHeight**(int rowHeight)

Sets the height, in pixels, of all cells to rowHeight, revalidates, and repaints. The height of the cells will be equal to the row height minus the row margin.

**Parameters:**rowHeight - new row height **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if rowHeight is less than 1**See Also:**[getRowHeight()](http://docs.google.com/javax/swing/JTable.html#getRowHeight())

### getRowHeight

public int **getRowHeight**()

Returns the height of a table row, in pixels. The default row height is 16.0.

**Returns:**the height in pixels of a table row**See Also:**[setRowHeight(int)](http://docs.google.com/javax/swing/JTable.html#setRowHeight(int))

### setRowHeight

public void **setRowHeight**(int row,  
 int rowHeight)

Sets the height for row to rowHeight, revalidates, and repaints. The height of the cells in this row will be equal to the row height minus the row margin.

**Parameters:**row - the row whose height is being changedrowHeight - new row height, in pixels **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if rowHeight is less than 1**Since:** 1.3

### getRowHeight

public int **getRowHeight**(int row)

Returns the height, in pixels, of the cells in row.

**Parameters:**row - the row whose height is to be returned **Returns:**the height, in pixels, of the cells in the row**Since:** 1.3

### setRowMargin

public void **setRowMargin**(int rowMargin)

Sets the amount of empty space between cells in adjacent rows.

**Parameters:**rowMargin - the number of pixels between cells in a row**See Also:**[getRowMargin()](http://docs.google.com/javax/swing/JTable.html#getRowMargin())

### getRowMargin

public int **getRowMargin**()

Gets the amount of empty space, in pixels, between cells. Equivalent to: getIntercellSpacing().height.

**Returns:**the number of pixels between cells in a row**See Also:**[setRowMargin(int)](http://docs.google.com/javax/swing/JTable.html#setRowMargin(int))

### setIntercellSpacing

public void **setIntercellSpacing**([Dimension](http://docs.google.com/java/awt/Dimension.html) intercellSpacing)

Sets the rowMargin and the columnMargin -- the height and width of the space between cells -- to intercellSpacing.

**Parameters:**intercellSpacing - a Dimension specifying the new width and height between cells**See Also:**[getIntercellSpacing()](http://docs.google.com/javax/swing/JTable.html#getIntercellSpacing())

### getIntercellSpacing

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getIntercellSpacing**()

Returns the horizontal and vertical space between cells. The default spacing is (1, 1), which provides room to draw the grid.

**Returns:**the horizontal and vertical spacing between cells**See Also:**[setIntercellSpacing(java.awt.Dimension)](http://docs.google.com/javax/swing/JTable.html#setIntercellSpacing(java.awt.Dimension))

### setGridColor

public void **setGridColor**([Color](http://docs.google.com/java/awt/Color.html) gridColor)

Sets the color used to draw grid lines to gridColor and redisplays. The default color is look and feel dependent.

**Parameters:**gridColor - the new color of the grid lines **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if gridColor is null**See Also:**[getGridColor()](http://docs.google.com/javax/swing/JTable.html#getGridColor())

### getGridColor

public [Color](http://docs.google.com/java/awt/Color.html) **getGridColor**()

Returns the color used to draw grid lines. The default color is look and feel dependent.

**Returns:**the color used to draw grid lines**See Also:**[setGridColor(java.awt.Color)](http://docs.google.com/javax/swing/JTable.html#setGridColor(java.awt.Color))

### setShowGrid

public void **setShowGrid**(boolean showGrid)

Sets whether the table draws grid lines around cells. If showGrid is true it does; if it is false it doesn't. There is no getShowGrid method as this state is held in two variables -- showHorizontalLines and showVerticalLines -- each of which can be queried independently.

**Parameters:**showGrid - true if table view should draw grid lines**See Also:**[setShowVerticalLines(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowVerticalLines(boolean)), [setShowHorizontalLines(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowHorizontalLines(boolean))

### setShowHorizontalLines

public void **setShowHorizontalLines**(boolean showHorizontalLines)

Sets whether the table draws horizontal lines between cells. If showHorizontalLines is true it does; if it is false it doesn't.

**Parameters:**showHorizontalLines - true if table view should draw horizontal lines**See Also:**[getShowHorizontalLines()](http://docs.google.com/javax/swing/JTable.html#getShowHorizontalLines()), [setShowGrid(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowGrid(boolean)), [setShowVerticalLines(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowVerticalLines(boolean))

### setShowVerticalLines

public void **setShowVerticalLines**(boolean showVerticalLines)

Sets whether the table draws vertical lines between cells. If showVerticalLines is true it does; if it is false it doesn't.

**Parameters:**showVerticalLines - true if table view should draw vertical lines**See Also:**[getShowVerticalLines()](http://docs.google.com/javax/swing/JTable.html#getShowVerticalLines()), [setShowGrid(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowGrid(boolean)), [setShowHorizontalLines(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowHorizontalLines(boolean))

### getShowHorizontalLines

public boolean **getShowHorizontalLines**()

Returns true if the table draws horizontal lines between cells, false if it doesn't. The default is true.

**Returns:**true if the table draws horizontal lines between cells, false if it doesn't**See Also:**[setShowHorizontalLines(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowHorizontalLines(boolean))

### getShowVerticalLines

public boolean **getShowVerticalLines**()

Returns true if the table draws vertical lines between cells, false if it doesn't. The default is true.

**Returns:**true if the table draws vertical lines between cells, false if it doesn't**See Also:**[setShowVerticalLines(boolean)](http://docs.google.com/javax/swing/JTable.html#setShowVerticalLines(boolean))

### setAutoResizeMode

public void **setAutoResizeMode**(int mode)

Sets the table's auto resize mode when the table is resized.

**Parameters:**mode - One of 5 legal values: AUTO\_RESIZE\_OFF, AUTO\_RESIZE\_NEXT\_COLUMN, AUTO\_RESIZE\_SUBSEQUENT\_COLUMNS, AUTO\_RESIZE\_LAST\_COLUMN, AUTO\_RESIZE\_ALL\_COLUMNS**See Also:**[getAutoResizeMode()](http://docs.google.com/javax/swing/JTable.html#getAutoResizeMode()), [doLayout()](http://docs.google.com/javax/swing/JTable.html#doLayout())

### getAutoResizeMode

public int **getAutoResizeMode**()

Returns the auto resize mode of the table. The default mode is AUTO\_RESIZE\_SUBSEQUENT\_COLUMNS.

**Returns:**the autoResizeMode of the table**See Also:**[setAutoResizeMode(int)](http://docs.google.com/javax/swing/JTable.html#setAutoResizeMode(int)), [doLayout()](http://docs.google.com/javax/swing/JTable.html#doLayout())

### setAutoCreateColumnsFromModel

public void **setAutoCreateColumnsFromModel**(boolean autoCreateColumnsFromModel)

Sets this table's autoCreateColumnsFromModel flag. This method calls createDefaultColumnsFromModel if autoCreateColumnsFromModel changes from false to true.

**Parameters:**autoCreateColumnsFromModel - true if JTable should automatically create columns**See Also:**[getAutoCreateColumnsFromModel()](http://docs.google.com/javax/swing/JTable.html#getAutoCreateColumnsFromModel()), [createDefaultColumnsFromModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultColumnsFromModel())

### getAutoCreateColumnsFromModel

public boolean **getAutoCreateColumnsFromModel**()

Determines whether the table will create default columns from the model. If true, setModel will clear any existing columns and create new columns from the new model. Also, if the event in the tableChanged notification specifies that the entire table changed, then the columns will be rebuilt. The default is true.

**Returns:**the autoCreateColumnsFromModel of the table**See Also:**[setAutoCreateColumnsFromModel(boolean)](http://docs.google.com/javax/swing/JTable.html#setAutoCreateColumnsFromModel(boolean)), [createDefaultColumnsFromModel()](http://docs.google.com/javax/swing/JTable.html#createDefaultColumnsFromModel())

### createDefaultColumnsFromModel

public void **createDefaultColumnsFromModel**()

Creates default columns for the table from the data model using the getColumnCount method defined in the TableModel interface.

Clears any existing columns before creating the new columns based on information from the model.

**See Also:**[getAutoCreateColumnsFromModel()](http://docs.google.com/javax/swing/JTable.html#getAutoCreateColumnsFromModel())

### setDefaultRenderer

public void **setDefaultRenderer**([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass,  
 [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) renderer)

Sets a default cell renderer to be used if no renderer has been set in a TableColumn. If renderer is null, removes the default renderer for this column class.

**Parameters:**columnClass - set the default cell renderer for this columnClassrenderer - default cell renderer to be used for this columnClass**See Also:**[getDefaultRenderer(java.lang.Class)](http://docs.google.com/javax/swing/JTable.html#getDefaultRenderer(java.lang.Class)), [setDefaultEditor(java.lang.Class, javax.swing.table.TableCellEditor)](http://docs.google.com/javax/swing/JTable.html#setDefaultEditor(java.lang.Class,%20javax.swing.table.TableCellEditor))

### getDefaultRenderer

public [TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) **getDefaultRenderer**([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass)

Returns the cell renderer to be used when no renderer has been set in a TableColumn. During the rendering of cells the renderer is fetched from a Hashtable of entries according to the class of the cells in the column. If there is no entry for this columnClass the method returns the entry for the most specific superclass. The JTable installs entries for Object, Number, and Boolean, all of which can be modified or replaced.

**Parameters:**columnClass - return the default cell renderer for this columnClass **Returns:**the renderer for this columnClass**See Also:**[setDefaultRenderer(java.lang.Class, javax.swing.table.TableCellRenderer)](http://docs.google.com/javax/swing/JTable.html#setDefaultRenderer(java.lang.Class,%20javax.swing.table.TableCellRenderer)), [getColumnClass(int)](http://docs.google.com/javax/swing/JTable.html#getColumnClass(int))

### setDefaultEditor

public void **setDefaultEditor**([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass,  
 [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) editor)

Sets a default cell editor to be used if no editor has been set in a TableColumn. If no editing is required in a table, or a particular column in a table, uses the isCellEditable method in the TableModel interface to ensure that this JTable will not start an editor in these columns. If editor is null, removes the default editor for this column class.

**Parameters:**columnClass - set the default cell editor for this columnClasseditor - default cell editor to be used for this columnClass**See Also:**[TableModel.isCellEditable(int, int)](http://docs.google.com/javax/swing/table/TableModel.html#isCellEditable(int,%20int)), [getDefaultEditor(java.lang.Class)](http://docs.google.com/javax/swing/JTable.html#getDefaultEditor(java.lang.Class)), [setDefaultRenderer(java.lang.Class, javax.swing.table.TableCellRenderer)](http://docs.google.com/javax/swing/JTable.html#setDefaultRenderer(java.lang.Class,%20javax.swing.table.TableCellRenderer))

### getDefaultEditor

public [TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) **getDefaultEditor**([Class](http://docs.google.com/java/lang/Class.html)<?> columnClass)

Returns the editor to be used when no editor has been set in a TableColumn. During the editing of cells the editor is fetched from a Hashtable of entries according to the class of the cells in the column. If there is no entry for this columnClass the method returns the entry for the most specific superclass. The JTable installs entries for Object, Number, and Boolean, all of which can be modified or replaced.

**Parameters:**columnClass - return the default cell editor for this columnClass **Returns:**the default cell editor to be used for this columnClass**See Also:**[setDefaultEditor(java.lang.Class, javax.swing.table.TableCellEditor)](http://docs.google.com/javax/swing/JTable.html#setDefaultEditor(java.lang.Class,%20javax.swing.table.TableCellEditor)), [getColumnClass(int)](http://docs.google.com/javax/swing/JTable.html#getColumnClass(int))

### setDragEnabled

public void **setDragEnabled**(boolean b)

Turns on or off automatic drag handling. In order to enable automatic drag handling, this property should be set to true, and the table's TransferHandler needs to be non-null. The default value of the dragEnabled property is false.

The job of honoring this property, and recognizing a user drag gesture, lies with the look and feel implementation, and in particular, the table's TableUI. When automatic drag handling is enabled, most look and feels (including those that subclass BasicLookAndFeel) begin a drag and drop operation whenever the user presses the mouse button over an item (in single selection mode) or a selection (in other selection modes) and then moves the mouse a few pixels. Setting this property to true can therefore have a subtle effect on how selections behave.

If a look and feel is used that ignores this property, you can still begin a drag and drop operation by calling exportAsDrag on the table's TransferHandler.

**Parameters:**b - whether or not to enable automatic drag handling **Throws:** [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html) - if b is true and GraphicsEnvironment.isHeadless() returns true**Since:** 1.4 **See Also:**[GraphicsEnvironment.isHeadless()](http://docs.google.com/java/awt/GraphicsEnvironment.html#isHeadless()), [getDragEnabled()](http://docs.google.com/javax/swing/JTable.html#getDragEnabled()), [JComponent.setTransferHandler(javax.swing.TransferHandler)](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler)), [TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html)

### getDragEnabled

public boolean **getDragEnabled**()

Returns whether or not automatic drag handling is enabled.

**Returns:**the value of the dragEnabled property**Since:** 1.4 **See Also:**[setDragEnabled(boolean)](http://docs.google.com/javax/swing/JTable.html#setDragEnabled(boolean))

### setDropMode

public final void **setDropMode**([DropMode](http://docs.google.com/javax/swing/DropMode.html) dropMode)

Sets the drop mode for this component. For backward compatibility, the default for this property is DropMode.USE\_SELECTION. Usage of one of the other modes is recommended, however, for an improved user experience. DropMode.ON, for instance, offers similar behavior of showing items as selected, but does so without affecting the actual selection in the table.

JTable supports the following drop modes:

* DropMode.USE\_SELECTION
* DropMode.ON
* DropMode.INSERT
* DropMode.INSERT\_ROWS
* DropMode.INSERT\_COLS
* DropMode.ON\_OR\_INSERT
* DropMode.ON\_OR\_INSERT\_ROWS
* DropMode.ON\_OR\_INSERT\_COLS

The drop mode is only meaningful if this component has a TransferHandler that accepts drops.

**Parameters:**dropMode - the drop mode to use **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the drop mode is unsupported or null**Since:** 1.6 **See Also:**[getDropMode()](http://docs.google.com/javax/swing/JTable.html#getDropMode()), [getDropLocation()](http://docs.google.com/javax/swing/JTable.html#getDropLocation()), [JComponent.setTransferHandler(javax.swing.TransferHandler)](http://docs.google.com/javax/swing/JComponent.html#setTransferHandler(javax.swing.TransferHandler)), [TransferHandler](http://docs.google.com/javax/swing/TransferHandler.html)

### getDropMode

public final [DropMode](http://docs.google.com/javax/swing/DropMode.html) **getDropMode**()

Returns the drop mode for this component.

**Returns:**the drop mode for this component**Since:** 1.6 **See Also:**[setDropMode(javax.swing.DropMode)](http://docs.google.com/javax/swing/JTable.html#setDropMode(javax.swing.DropMode))

### getDropLocation

public final [JTable.DropLocation](http://docs.google.com/javax/swing/JTable.DropLocation.html) **getDropLocation**()

Returns the location that this component should visually indicate as the drop location during a DnD operation over the component, or null if no location is to currently be shown.

This method is not meant for querying the drop location from a TransferHandler, as the drop location is only set after the TransferHandler's canImport has returned and has allowed for the location to be shown.

When this property changes, a property change event with name "dropLocation" is fired by the component.

**Returns:**the drop location**Since:** 1.6 **See Also:**[setDropMode(javax.swing.DropMode)](http://docs.google.com/javax/swing/JTable.html#setDropMode(javax.swing.DropMode)), [TransferHandler.canImport(TransferHandler.TransferSupport)](http://docs.google.com/javax/swing/TransferHandler.html#canImport(javax.swing.TransferHandler.TransferSupport))

### setAutoCreateRowSorter

public void **setAutoCreateRowSorter**(boolean autoCreateRowSorter)

Specifies whether a RowSorter should be created for the table whenever its model changes.

When setAutoCreateRowSorter(true) is invoked, a TableRowSorter is immediately created and installed on the table. While the autoCreateRowSorter property remains true, every time the model is changed, a new TableRowSorter is created and set as the table's row sorter.

**Parameters:**autoCreateRowSorter - whether or not a RowSorter should be automatically created**Since:** 1.6 **See Also:**[TableRowSorter](http://docs.google.com/javax/swing/table/TableRowSorter.html)

### getAutoCreateRowSorter

public boolean **getAutoCreateRowSorter**()

Returns true if whenever the model changes, a new RowSorter should be created and installed as the table's sorter; otherwise, returns false.

**Returns:**true if a RowSorter should be created when the model changes**Since:** 1.6

### setUpdateSelectionOnSort

public void **setUpdateSelectionOnSort**(boolean update)

Specifies whether the selection should be updated after sorting. If true, on sorting the selection is reset such that the same rows, in terms of the model, remain selected. The default is true.

**Parameters:**update - whether or not to update the selection on sorting**Since:** 1.6

### getUpdateSelectionOnSort

public boolean **getUpdateSelectionOnSort**()

Returns true if the selection should be updated after sorting.

**Returns:**whether to update the selection on a sort**Since:** 1.6

### setRowSorter

public void **setRowSorter**([RowSorter](http://docs.google.com/javax/swing/RowSorter.html)<? extends [TableModel](http://docs.google.com/javax/swing/table/TableModel.html)> sorter)

Sets the RowSorter. RowSorter is used to provide sorting and filtering to a JTable.

This method clears the selection and resets any variable row heights.

If the underlying model of the RowSorter differs from that of this JTable undefined behavior will result.

**Parameters:**sorter - the RowSorter; null turns sorting off**Since:** 1.6 **See Also:**[TableRowSorter](http://docs.google.com/javax/swing/table/TableRowSorter.html)

### getRowSorter

public [RowSorter](http://docs.google.com/javax/swing/RowSorter.html)<? extends [TableModel](http://docs.google.com/javax/swing/table/TableModel.html)> **getRowSorter**()

Returns the object responsible for sorting.

**Returns:**the object responsible for sorting**Since:** 1.6

### setSelectionMode

public void **setSelectionMode**(int selectionMode)

Sets the table's selection mode to allow only single selections, a single contiguous interval, or multiple intervals.

Note: JTable provides all the methods for handling column and row selection. When setting states, such as setSelectionMode, it not only updates the mode for the row selection model but also sets similar values in the selection model of the columnModel. If you want to have the row and column selection models operating in different modes, set them both directly.

Both the row and column selection models for JTable default to using a DefaultListSelectionModel so that JTable works the same way as the JList. See the setSelectionMode method in JList for details about the modes.

**See Also:**[JList.setSelectionMode(int)](http://docs.google.com/javax/swing/JList.html#setSelectionMode(int))

### setRowSelectionAllowed

public void **setRowSelectionAllowed**(boolean rowSelectionAllowed)

Sets whether the rows in this model can be selected.

**Parameters:**rowSelectionAllowed - true if this model will allow row selection**See Also:**[getRowSelectionAllowed()](http://docs.google.com/javax/swing/JTable.html#getRowSelectionAllowed())

### getRowSelectionAllowed

public boolean **getRowSelectionAllowed**()

Returns true if rows can be selected.

**Returns:**true if rows can be selected, otherwise false**See Also:**[setRowSelectionAllowed(boolean)](http://docs.google.com/javax/swing/JTable.html#setRowSelectionAllowed(boolean))

### setColumnSelectionAllowed

public void **setColumnSelectionAllowed**(boolean columnSelectionAllowed)

Sets whether the columns in this model can be selected.

**Parameters:**columnSelectionAllowed - true if this model will allow column selection**See Also:**[getColumnSelectionAllowed()](http://docs.google.com/javax/swing/JTable.html#getColumnSelectionAllowed())

### getColumnSelectionAllowed

public boolean **getColumnSelectionAllowed**()

Returns true if columns can be selected.

**Returns:**true if columns can be selected, otherwise false**See Also:**[setColumnSelectionAllowed(boolean)](http://docs.google.com/javax/swing/JTable.html#setColumnSelectionAllowed(boolean))

### setCellSelectionEnabled

public void **setCellSelectionEnabled**(boolean cellSelectionEnabled)

Sets whether this table allows both a column selection and a row selection to exist simultaneously. When set, the table treats the intersection of the row and column selection models as the selected cells. Override isCellSelected to change this default behavior. This method is equivalent to setting both the rowSelectionAllowed property and columnSelectionAllowed property of the columnModel to the supplied value.

**Parameters:**cellSelectionEnabled - true if simultaneous row and column selection is allowed**See Also:**[getCellSelectionEnabled()](http://docs.google.com/javax/swing/JTable.html#getCellSelectionEnabled()), [isCellSelected(int, int)](http://docs.google.com/javax/swing/JTable.html#isCellSelected(int,%20int))

### getCellSelectionEnabled

public boolean **getCellSelectionEnabled**()

Returns true if both row and column selection models are enabled. Equivalent to getRowSelectionAllowed() && getColumnSelectionAllowed().

**Returns:**true if both row and column selection models are enabled**See Also:**[setCellSelectionEnabled(boolean)](http://docs.google.com/javax/swing/JTable.html#setCellSelectionEnabled(boolean))

### selectAll

public void **selectAll**()

Selects all rows, columns, and cells in the table.

### clearSelection

public void **clearSelection**()

Deselects all selected columns and rows.

### setRowSelectionInterval

public void **setRowSelectionInterval**(int index0,  
 int index1)

Selects the rows from index0 to index1, inclusive.

**Parameters:**index0 - one end of the intervalindex1 - the other end of the interval **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if index0 or index1 lie outside [0, getRowCount()-1]

### setColumnSelectionInterval

public void **setColumnSelectionInterval**(int index0,  
 int index1)

Selects the columns from index0 to index1, inclusive.

**Parameters:**index0 - one end of the intervalindex1 - the other end of the interval **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if index0 or index1 lie outside [0, getColumnCount()-1]

### addRowSelectionInterval

public void **addRowSelectionInterval**(int index0,  
 int index1)

Adds the rows from index0 to index1, inclusive, to the current selection.

**Parameters:**index0 - one end of the intervalindex1 - the other end of the interval **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if index0 or index1 lie outside [0, getRowCount()-1]

### addColumnSelectionInterval

public void **addColumnSelectionInterval**(int index0,  
 int index1)

Adds the columns from index0 to index1, inclusive, to the current selection.

**Parameters:**index0 - one end of the intervalindex1 - the other end of the interval **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if index0 or index1 lie outside [0, getColumnCount()-1]

### removeRowSelectionInterval

public void **removeRowSelectionInterval**(int index0,  
 int index1)

Deselects the rows from index0 to index1, inclusive.

**Parameters:**index0 - one end of the intervalindex1 - the other end of the interval **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if index0 or index1 lie outside [0, getRowCount()-1]

### removeColumnSelectionInterval

public void **removeColumnSelectionInterval**(int index0,  
 int index1)

Deselects the columns from index0 to index1, inclusive.

**Parameters:**index0 - one end of the intervalindex1 - the other end of the interval **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if index0 or index1 lie outside [0, getColumnCount()-1]

### getSelectedRow

public int **getSelectedRow**()

Returns the index of the first selected row, -1 if no row is selected.

**Returns:**the index of the first selected row

### getSelectedColumn

public int **getSelectedColumn**()

Returns the index of the first selected column, -1 if no column is selected.

**Returns:**the index of the first selected column

### getSelectedRows

public int[] **getSelectedRows**()

Returns the indices of all selected rows.

**Returns:**an array of integers containing the indices of all selected rows, or an empty array if no row is selected**See Also:**[getSelectedRow()](http://docs.google.com/javax/swing/JTable.html#getSelectedRow())

### getSelectedColumns

public int[] **getSelectedColumns**()

Returns the indices of all selected columns.

**Returns:**an array of integers containing the indices of all selected columns, or an empty array if no column is selected**See Also:**[getSelectedColumn()](http://docs.google.com/javax/swing/JTable.html#getSelectedColumn())

### getSelectedRowCount

public int **getSelectedRowCount**()

Returns the number of selected rows.

**Returns:**the number of selected rows, 0 if no rows are selected

### getSelectedColumnCount

public int **getSelectedColumnCount**()

Returns the number of selected columns.

**Returns:**the number of selected columns, 0 if no columns are selected

### isRowSelected

public boolean **isRowSelected**(int row)

Returns true if the specified index is in the valid range of rows, and the row at that index is selected.

**Returns:**true if row is a valid index and the row at that index is selected (where 0 is the first row)

### isColumnSelected

public boolean **isColumnSelected**(int column)

Returns true if the specified index is in the valid range of columns, and the column at that index is selected.

**Parameters:**column - the column in the column model **Returns:**true if column is a valid index and the column at that index is selected (where 0 is the first column)

### isCellSelected

public boolean **isCellSelected**(int row,  
 int column)

Returns true if the specified indices are in the valid range of rows and columns and the cell at the specified position is selected.

**Parameters:**row - the row being queriedcolumn - the column being queried **Returns:**true if row and column are valid indices and the cell at index (row, column) is selected, where the first row and first column are at index 0

### changeSelection

public void **changeSelection**(int rowIndex,  
 int columnIndex,  
 boolean toggle,  
 boolean extend)

Updates the selection models of the table, depending on the state of the two flags: toggle and extend. Most changes to the selection that are the result of keyboard or mouse events received by the UI are channeled through this method so that the behavior may be overridden by a subclass. Some UIs may need more functionality than this method provides, such as when manipulating the lead for discontiguous selection, and may not call into this method for some selection changes.

This implementation uses the following conventions:

* toggle: *false*, extend: *false*. Clear the previous selection and ensure the new cell is selected.
* toggle: *false*, extend: *true*. Extend the previous selection from the anchor to the specified cell, clearing all other selections.
* toggle: *true*, extend: *false*. If the specified cell is selected, deselect it. If it is not selected, select it.
* toggle: *true*, extend: *true*. Apply the selection state of the anchor to all cells between it and the specified cell.

**Parameters:**rowIndex - affects the selection at rowcolumnIndex - affects the selection at columntoggle - see description aboveextend - if true, extend the current selection**Since:** 1.3

### getSelectionForeground

public [Color](http://docs.google.com/java/awt/Color.html) **getSelectionForeground**()

Returns the foreground color for selected cells.

**Returns:**the Color object for the foreground property**See Also:**[setSelectionForeground(java.awt.Color)](http://docs.google.com/javax/swing/JTable.html#setSelectionForeground(java.awt.Color)), [setSelectionBackground(java.awt.Color)](http://docs.google.com/javax/swing/JTable.html#setSelectionBackground(java.awt.Color))

### setSelectionForeground

public void **setSelectionForeground**([Color](http://docs.google.com/java/awt/Color.html) selectionForeground)

Sets the foreground color for selected cells. Cell renderers can use this color to render text and graphics for selected cells.

The default value of this property is defined by the look and feel implementation.

This is a [JavaBeans](http://java.sun.com/docs/books/tutorial/javabeans/whatis/beanDefinition.html) bound property.

**Parameters:**selectionForeground - the Color to use in the foreground for selected list items**See Also:**[getSelectionForeground()](http://docs.google.com/javax/swing/JTable.html#getSelectionForeground()), [setSelectionBackground(java.awt.Color)](http://docs.google.com/javax/swing/JTable.html#setSelectionBackground(java.awt.Color)), [JComponent.setForeground(java.awt.Color)](http://docs.google.com/javax/swing/JComponent.html#setForeground(java.awt.Color)), [JComponent.setBackground(java.awt.Color)](http://docs.google.com/javax/swing/JComponent.html#setBackground(java.awt.Color)), [JComponent.setFont(java.awt.Font)](http://docs.google.com/javax/swing/JComponent.html#setFont(java.awt.Font))

### getSelectionBackground

public [Color](http://docs.google.com/java/awt/Color.html) **getSelectionBackground**()

Returns the background color for selected cells.

**Returns:**the Color used for the background of selected list items**See Also:**[setSelectionBackground(java.awt.Color)](http://docs.google.com/javax/swing/JTable.html#setSelectionBackground(java.awt.Color)), [setSelectionForeground(java.awt.Color)](http://docs.google.com/javax/swing/JTable.html#setSelectionForeground(java.awt.Color))

### setSelectionBackground

public void **setSelectionBackground**([Color](http://docs.google.com/java/awt/Color.html) selectionBackground)

Sets the background color for selected cells. Cell renderers can use this color to the fill selected cells.

The default value of this property is defined by the look and feel implementation.

This is a [JavaBeans](http://java.sun.com/docs/books/tutorial/javabeans/whatis/beanDefinition.html) bound property.

**Parameters:**selectionBackground - the Color to use for the background of selected cells**See Also:**[getSelectionBackground()](http://docs.google.com/javax/swing/JTable.html#getSelectionBackground()), [setSelectionForeground(java.awt.Color)](http://docs.google.com/javax/swing/JTable.html#setSelectionForeground(java.awt.Color)), [JComponent.setForeground(java.awt.Color)](http://docs.google.com/javax/swing/JComponent.html#setForeground(java.awt.Color)), [JComponent.setBackground(java.awt.Color)](http://docs.google.com/javax/swing/JComponent.html#setBackground(java.awt.Color)), [JComponent.setFont(java.awt.Font)](http://docs.google.com/javax/swing/JComponent.html#setFont(java.awt.Font))

### getColumn

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

Returns the TableColumn object for the column in the table whose identifier is equal to identifier, when compared using equals.

**Parameters:**identifier - the identifier object **Returns:**the TableColumn object that matches the identifier **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if identifier is null or no TableColumn has this identifier

### convertColumnIndexToModel

public int **convertColumnIndexToModel**(int viewColumnIndex)

Maps the index of the column in the view at viewColumnIndex to the index of the column in the table model. Returns the index of the corresponding column in the model. If viewColumnIndex is less than zero, returns viewColumnIndex.

**Parameters:**viewColumnIndex - the index of the column in the view **Returns:**the index of the corresponding column in the model**See Also:**[convertColumnIndexToView(int)](http://docs.google.com/javax/swing/JTable.html#convertColumnIndexToView(int))

### convertColumnIndexToView

public int **convertColumnIndexToView**(int modelColumnIndex)

Maps the index of the column in the table model at modelColumnIndex to the index of the column in the view. Returns the index of the corresponding column in the view; returns -1 if this column is not being displayed. If modelColumnIndex is less than zero, returns modelColumnIndex.

**Parameters:**modelColumnIndex - the index of the column in the model **Returns:**the index of the corresponding column in the view**See Also:**[convertColumnIndexToModel(int)](http://docs.google.com/javax/swing/JTable.html#convertColumnIndexToModel(int))

### convertRowIndexToView

public int **convertRowIndexToView**(int modelRowIndex)

Maps the index of the row in terms of the TableModel to the view. If the contents of the model are not sorted the model and view indices are the same.

**Parameters:**modelRowIndex - the index of the row in terms of the model **Returns:**the index of the corresponding row in the view, or -1 if the row isn't visible **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if sorting is enabled and passed an index outside the number of rows of the TableModel**Since:** 1.6 **See Also:**[TableRowSorter](http://docs.google.com/javax/swing/table/TableRowSorter.html)

### convertRowIndexToModel

public int **convertRowIndexToModel**(int viewRowIndex)

Maps the index of the row in terms of the view to the underlying TableModel. If the contents of the model are not sorted the model and view indices are the same.

**Parameters:**viewRowIndex - the index of the row in the view **Returns:**the index of the corresponding row in the model **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if sorting is enabled and passed an index outside the range of the JTable as determined by the method getRowCount**Since:** 1.6 **See Also:**[TableRowSorter](http://docs.google.com/javax/swing/table/TableRowSorter.html), [getRowCount()](http://docs.google.com/javax/swing/JTable.html#getRowCount())

### getRowCount

public int **getRowCount**()

Returns the number of rows that can be shown in the JTable, given unlimited space. If a RowSorter with a filter has been specified, the number of rows returned may differ from that of the underlying TableModel.

**Returns:**the number of rows shown in the JTable**See Also:**[getColumnCount()](http://docs.google.com/javax/swing/JTable.html#getColumnCount())

### getColumnCount

public int **getColumnCount**()

Returns the number of columns in the column model. Note that this may be different from the number of columns in the table model.

**Returns:**the number of columns in the table**See Also:**[getRowCount()](http://docs.google.com/javax/swing/JTable.html#getRowCount()), [removeColumn(javax.swing.table.TableColumn)](http://docs.google.com/javax/swing/JTable.html#removeColumn(javax.swing.table.TableColumn))

### getColumnName

public [String](http://docs.google.com/java/lang/String.html) **getColumnName**(int column)

Returns the name of the column appearing in the view at column position column.

**Parameters:**column - the column in the view being queried **Returns:**the name of the column at position column in the view where the first column is column 0

### getColumnClass

public [Class](http://docs.google.com/java/lang/Class.html)<?> **getColumnClass**(int column)

Returns the type of the column appearing in the view at column position column.

**Parameters:**column - the column in the view being queried **Returns:**the type of the column at position column in the view where the first column is column 0

### getValueAt

public [Object](http://docs.google.com/java/lang/Object.html) **getValueAt**(int row,  
 int column)

Returns the cell value at row and column.

**Note**: The column is specified in the table view's display order, and not in the TableModel's column order. This is an important distinction because as the user rearranges the columns in the table, the column at a given index in the view will change. Meanwhile the user's actions never affect the model's column ordering.

**Parameters:**row - the row whose value is to be queriedcolumn - the column whose value is to be queried **Returns:**the Object at the specified cell

### setValueAt

public void **setValueAt**([Object](http://docs.google.com/java/lang/Object.html) aValue,  
 int row,  
 int column)

Sets the value for the cell in the table model at row and column.

**Note**: The column is specified in the table view's display order, and not in the TableModel's column order. This is an important distinction because as the user rearranges the columns in the table, the column at a given index in the view will change. Meanwhile the user's actions never affect the model's column ordering. aValue is the new value.

**Parameters:**aValue - the new valuerow - the row of the cell to be changedcolumn - the column of the cell to be changed**See Also:**[getValueAt(int, int)](http://docs.google.com/javax/swing/JTable.html#getValueAt(int,%20int))

### isCellEditable

public boolean **isCellEditable**(int row,  
 int column)

Returns true if the cell at row and column is editable. Otherwise, invoking setValueAt on the cell will have no effect.

**Note**: The column is specified in the table view's display order, and not in the TableModel's column order. This is an important distinction because as the user rearranges the columns in the table, the column at a given index in the view will change. Meanwhile the user's actions never affect the model's column ordering.

**Parameters:**row - the row whose value is to be queriedcolumn - the column whose value is to be queried **Returns:**true if the cell is editable**See Also:**[setValueAt(java.lang.Object, int, int)](http://docs.google.com/javax/swing/JTable.html#setValueAt(java.lang.Object,%20int,%20int))

### addColumn

public void **addColumn**([TableColumn](http://docs.google.com/javax/swing/table/TableColumn.html) aColumn)

Appends aColumn to the end of the array of columns held by this JTable's column model. If the column name of aColumn is null, sets the column name of aColumn to the name returned by getModel().getColumnName().

To add a column to this JTable to display the modelColumn'th column of data in the model with a given width, cellRenderer, and cellEditor you can use:

addColumn(new TableColumn(modelColumn, width, cellRenderer, cellEditor));

[Any of the TableColumn constructors can be used instead of this one.] The model column number is stored inside the TableColumn and is used during rendering and editing to locate the appropriates data values in the model. The model column number does not change when columns are reordered in the view.

**Parameters:**aColumn - the TableColumn to be added**See Also:**[removeColumn(javax.swing.table.TableColumn)](http://docs.google.com/javax/swing/JTable.html#removeColumn(javax.swing.table.TableColumn))

### removeColumn

public void **removeColumn**([TableColumn](http://docs.google.com/javax/swing/table/TableColumn.html) aColumn)

Removes aColumn from this JTable's array of columns. Note: this method does not remove the column of data from the model; it just removes the TableColumn that was responsible for displaying it.

**Parameters:**aColumn - the TableColumn to be removed**See Also:**[addColumn(javax.swing.table.TableColumn)](http://docs.google.com/javax/swing/JTable.html#addColumn(javax.swing.table.TableColumn))

### moveColumn

public void **moveColumn**(int column,  
 int targetColumn)

Moves the column column to the position currently occupied by the column targetColumn in the view. The old column at targetColumn is shifted left or right to make room.

**Parameters:**column - the index of column to be movedtargetColumn - the new index of the column

### columnAtPoint

public int **columnAtPoint**([Point](http://docs.google.com/java/awt/Point.html) point)

Returns the index of the column that point lies in, or -1 if the result is not in the range [0, getColumnCount()-1].

**Parameters:**point - the location of interest **Returns:**the index of the column that point lies in, or -1 if the result is not in the range [0, getColumnCount()-1]**See Also:**[rowAtPoint(java.awt.Point)](http://docs.google.com/javax/swing/JTable.html#rowAtPoint(java.awt.Point))

### rowAtPoint

public int **rowAtPoint**([Point](http://docs.google.com/java/awt/Point.html) point)

Returns the index of the row that point lies in, or -1 if the result is not in the range [0, getRowCount()-1].

**Parameters:**point - the location of interest **Returns:**the index of the row that point lies in, or -1 if the result is not in the range [0, getRowCount()-1]**See Also:**[columnAtPoint(java.awt.Point)](http://docs.google.com/javax/swing/JTable.html#columnAtPoint(java.awt.Point))

### getCellRect

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getCellRect**(int row,  
 int column,  
 boolean includeSpacing)

Returns a rectangle for the cell that lies at the intersection of row and column. If includeSpacing is true then the value returned has the full height and width of the row and column specified. If it is false, the returned rectangle is inset by the intercell spacing to return the true bounds of the rendering or editing component as it will be set during rendering.

If the column index is valid but the row index is less than zero the method returns a rectangle with the y and height values set appropriately and the x and width values both set to zero. In general, when either the row or column indices indicate a cell outside the appropriate range, the method returns a rectangle depicting the closest edge of the closest cell that is within the table's range. When both row and column indices are out of range the returned rectangle covers the closest point of the closest cell.

In all cases, calculations that use this method to calculate results along one axis will not fail because of anomalies in calculations along the other axis. When the cell is not valid the includeSpacing parameter is ignored.

**Parameters:**row - the row index where the desired cell is locatedcolumn - the column index where the desired cell is located in the display; this is not necessarily the same as the column index in the data model for the table; the [convertColumnIndexToView(int)](http://docs.google.com/javax/swing/JTable.html#convertColumnIndexToView(int)) method may be used to convert a data model column index to a display column indexincludeSpacing - if false, return the true cell bounds - computed by subtracting the intercell spacing from the height and widths of the column and row models **Returns:**the rectangle containing the cell at location row,column**See Also:**[getIntercellSpacing()](http://docs.google.com/javax/swing/JTable.html#getIntercellSpacing())

### doLayout

public void **doLayout**()

Causes this table to lay out its rows and columns. Overridden so that columns can be resized to accomodate a change in the size of a containing parent. Resizes one or more of the columns in the table so that the total width of all of this JTable's columns is equal to the width of the table.

Before the layout begins the method gets the resizingColumn of the tableHeader. When the method is called as a result of the resizing of an enclosing window, the resizingColumn is null. This means that resizing has taken place "outside" the JTable and the change - or "delta" - should be distributed to all of the columns regardless of this JTable's automatic resize mode.

If the resizingColumn is not null, it is one of the columns in the table that has changed size rather than the table itself. In this case the auto-resize modes govern the way the extra (or deficit) space is distributed amongst the available columns.

The modes are:

* AUTO\_RESIZE\_OFF: Don't automatically adjust the column's widths at all. Use a horizontal scrollbar to accomodate the columns when their sum exceeds the width of the Viewport. If the JTable is not enclosed in a JScrollPane this may leave parts of the table invisible.
* AUTO\_RESIZE\_NEXT\_COLUMN: Use just the column after the resizing column. This results in the "boundary" or divider between adjacent cells being independently adjustable.
* AUTO\_RESIZE\_SUBSEQUENT\_COLUMNS: Use all columns after the one being adjusted to absorb the changes. This is the default behavior.
* AUTO\_RESIZE\_LAST\_COLUMN: Automatically adjust the size of the last column only. If the bounds of the last column prevent the desired size from being allocated, set the width of the last column to the appropriate limit and make no further adjustments.
* AUTO\_RESIZE\_ALL\_COLUMNS: Spread the delta amongst all the columns in the JTable, including the one that is being adjusted.

Note: When a JTable makes adjustments to the widths of the columns it respects their minimum and maximum values absolutely. It is therefore possible that, even after this method is called, the total width of the columns is still not equal to the width of the table. When this happens the JTable does not put itself in AUTO\_RESIZE\_OFF mode to bring up a scroll bar, or break other commitments of its current auto-resize mode -- instead it allows its bounds to be set larger (or smaller) than the total of the column minimum or maximum, meaning, either that there will not be enough room to display all of the columns, or that the columns will not fill the JTable's bounds. These respectively, result in the clipping of some columns or an area being painted in the JTable's background color during painting.

The mechanism for distributing the delta amongst the available columns is provided in a private method in the JTable class:

adjustSizes(long targetSize, final Resizable3 r, boolean inverse)

an explanation of which is provided in the following section. Resizable3 is a private interface that allows any data structure containing a collection of elements with a size, preferred size, maximum size and minimum size to have its elements manipulated by the algorithm.

### Distributing the delta

#### Overview

Call "DELTA" the difference between the target size and the sum of the preferred sizes of the elements in r. The individual sizes are calculated by taking the original preferred sizes and adding a share of the DELTA - that share being based on how far each preferred size is from its limiting bound (minimum or maximum).

#### Definition

Call the individual constraints min[i], max[i], and pref[i].

Call their respective sums: MIN, MAX, and PREF.

Each new size will be calculated using:

size[i] = pref[i] + delta[i]

where each individual delta[i] is calculated according to:

If (DELTA < 0) we are in shrink mode where:

DELTA  
 delta[i] = ------------ \* (pref[i] - min[i])  
 (PREF - MIN)

If (DELTA > 0) we are in expand mode where:

DELTA  
 delta[i] = ------------ \* (max[i] - pref[i])  
 (MAX - PREF)

The overall effect is that the total size moves that same percentage, k, towards the total minimum or maximum and that percentage guarantees accomodation of the required space, DELTA.

#### Details

Naive evaluation of the formulae presented here would be subject to the aggregated rounding errors caused by doing this operation in finite precision (using ints). To deal with this, the multiplying factor above, is constantly recalculated and this takes account of the rounding errors in the previous iterations. The result is an algorithm that produces a set of integers whose values exactly sum to the supplied targetSize, and does so by spreading the rounding errors evenly over the given elements.

#### When the MAX and MIN bounds are hit

When targetSize is outside the [MIN, MAX] range, the algorithm sets all sizes to their appropriate limiting value (maximum or minimum).

**Overrides:**[doLayout](http://docs.google.com/java/awt/Container.html#doLayout()) in class [Container](http://docs.google.com/java/awt/Container.html) **See Also:**[LayoutManager.layoutContainer(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#layoutContainer(java.awt.Container)), [Container.setLayout(java.awt.LayoutManager)](http://docs.google.com/java/awt/Container.html#setLayout(java.awt.LayoutManager)), [Container.validate()](http://docs.google.com/java/awt/Container.html#validate())

### sizeColumnsToFit

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

**Deprecated.** *As of Swing version 1.0.3, replaced by doLayout().*

Sizes the table columns to fit the available space.

**See Also:**[doLayout()](http://docs.google.com/javax/swing/JTable.html#doLayout())

### sizeColumnsToFit

public void **sizeColumnsToFit**(int resizingColumn)

Obsolete as of Java 2 platform v1.4. Please use the doLayout() method instead.

**Parameters:**resizingColumn - the column whose resizing made this adjustment necessary or -1 if there is no such column**See Also:**[doLayout()](http://docs.google.com/javax/swing/JTable.html#doLayout())

### getToolTipText

public [String](http://docs.google.com/java/lang/String.html) **getToolTipText**([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)

Overrides JComponent's getToolTipText method in order to allow the renderer's tips to be used if it has text set.

Note: For JTable to properly display tooltips of its renderers JTable must be a registered component with the ToolTipManager. This is done automatically in initializeLocalVars, but if at a later point JTable is told setToolTipText(null) it will unregister the table component, and no tips from renderers will display anymore.

**Overrides:**[getToolTipText](http://docs.google.com/javax/swing/JComponent.html#getToolTipText(java.awt.event.MouseEvent)) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **See Also:**[JComponent.getToolTipText()](http://docs.google.com/javax/swing/JComponent.html#getToolTipText())

### setSurrendersFocusOnKeystroke

public void **setSurrendersFocusOnKeystroke**(boolean surrendersFocusOnKeystroke)

Sets whether editors in this JTable get the keyboard focus when an editor is activated as a result of the JTable forwarding keyboard events for a cell. By default, this property is false, and the JTable retains the focus unless the cell is clicked.

**Parameters:**surrendersFocusOnKeystroke - true if the editor should get the focus when keystrokes cause the editor to be activated**Since:** 1.4 **See Also:**[getSurrendersFocusOnKeystroke()](http://docs.google.com/javax/swing/JTable.html#getSurrendersFocusOnKeystroke())

### getSurrendersFocusOnKeystroke

public boolean **getSurrendersFocusOnKeystroke**()

Returns true if the editor should get the focus when keystrokes cause the editor to be activated

**Returns:**true if the editor should get the focus when keystrokes cause the editor to be activated**Since:** 1.4 **See Also:**[setSurrendersFocusOnKeystroke(boolean)](http://docs.google.com/javax/swing/JTable.html#setSurrendersFocusOnKeystroke(boolean))

### editCellAt

public boolean **editCellAt**(int row,  
 int column)

Programmatically starts editing the cell at row and column, if those indices are in the valid range, and the cell at those indices is editable. Note that this is a convenience method for editCellAt(int, int, null).

**Parameters:**row - the row to be editedcolumn - the column to be edited **Returns:**false if for any reason the cell cannot be edited, or if the indices are invalid

### editCellAt

public boolean **editCellAt**(int row,  
 int column,  
 [EventObject](http://docs.google.com/java/util/EventObject.html) e)

Programmatically starts editing the cell at row and column, if those indices are in the valid range, and the cell at those indices is editable. To prevent the JTable from editing a particular table, column or cell value, return false from the isCellEditable method in the TableModel interface.

**Parameters:**row - the row to be editedcolumn - the column to be editede - event to pass into shouldSelectCell; note that as of Java 2 platform v1.2, the call to shouldSelectCell is no longer made **Returns:**false if for any reason the cell cannot be edited, or if the indices are invalid

### isEditing

public boolean **isEditing**()

Returns true if a cell is being edited.

**Returns:**true if the table is editing a cell**See Also:**[editingColumn](http://docs.google.com/javax/swing/JTable.html#editingColumn), [editingRow](http://docs.google.com/javax/swing/JTable.html#editingRow)

### getEditorComponent

public [Component](http://docs.google.com/java/awt/Component.html) **getEditorComponent**()

Returns the component that is handling the editing session. If nothing is being edited, returns null.

**Returns:**Component handling editing session

### getEditingColumn

public int **getEditingColumn**()

Returns the index of the column that contains the cell currently being edited. If nothing is being edited, returns -1.

**Returns:**the index of the column that contains the cell currently being edited; returns -1 if nothing being edited**See Also:**[editingRow](http://docs.google.com/javax/swing/JTable.html#editingRow)

### getEditingRow

public int **getEditingRow**()

Returns the index of the row that contains the cell currently being edited. If nothing is being edited, returns -1.

**Returns:**the index of the row that contains the cell currently being edited; returns -1 if nothing being edited**See Also:**[editingColumn](http://docs.google.com/javax/swing/JTable.html#editingColumn)

### getUI

public [TableUI](http://docs.google.com/javax/swing/plaf/TableUI.html) **getUI**()

Returns the L&F object that renders this component.

**Returns:**the TableUI object that renders this component

### setUI

public void **setUI**([TableUI](http://docs.google.com/javax/swing/plaf/TableUI.html) ui)

Sets the L&F object that renders this component and repaints.

**Parameters:**ui - the TableUI L&F object**See Also:**[UIDefaults.getUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/UIDefaults.html#getUI(javax.swing.JComponent))

### updateUI

public void **updateUI**()

Notification from the UIManager that the L&F has changed. Replaces the current UI object with the latest version from the UIManager.

**Overrides:**[updateUI](http://docs.google.com/javax/swing/JComponent.html#updateUI()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **See Also:**[JComponent.updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI())

### getUIClassID

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

Returns the suffix used to construct the name of the L&F class used to render this component.

**Overrides:**[getUIClassID](http://docs.google.com/javax/swing/JComponent.html#getUIClassID()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **Returns:**the string "TableUI"**See Also:**[JComponent.getUIClassID()](http://docs.google.com/javax/swing/JComponent.html#getUIClassID()), [UIDefaults.getUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/UIDefaults.html#getUI(javax.swing.JComponent))

### setModel

public void **setModel**([TableModel](http://docs.google.com/javax/swing/table/TableModel.html) dataModel)

Sets the data model for this table to newModel and registers with it for listener notifications from the new data model.

**Parameters:**dataModel - the new data source for this table **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if newModel is null**See Also:**[getModel()](http://docs.google.com/javax/swing/JTable.html#getModel())

### getModel

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

Returns the TableModel that provides the data displayed by this JTable.

**Returns:**the TableModel that provides the data displayed by this JTable**See Also:**[setModel(javax.swing.table.TableModel)](http://docs.google.com/javax/swing/JTable.html#setModel(javax.swing.table.TableModel))

### setColumnModel

public void **setColumnModel**([TableColumnModel](http://docs.google.com/javax/swing/table/TableColumnModel.html) columnModel)

Sets the column model for this table to newModel and registers for listener notifications from the new column model. Also sets the column model of the JTableHeader to columnModel.

**Parameters:**columnModel - the new data source for this table **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if columnModel is null**See Also:**[getColumnModel()](http://docs.google.com/javax/swing/JTable.html#getColumnModel())

### getColumnModel

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

Returns the TableColumnModel that contains all column information of this table.

**Returns:**the object that provides the column state of the table**See Also:**[setColumnModel(javax.swing.table.TableColumnModel)](http://docs.google.com/javax/swing/JTable.html#setColumnModel(javax.swing.table.TableColumnModel))

### setSelectionModel

public void **setSelectionModel**([ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) newModel)

Sets the row selection model for this table to newModel and registers for listener notifications from the new selection model.

**Parameters:**newModel - the new selection model **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if newModel is null**See Also:**[getSelectionModel()](http://docs.google.com/javax/swing/JTable.html#getSelectionModel())

### getSelectionModel

public [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) **getSelectionModel**()

Returns the ListSelectionModel that is used to maintain row selection state.

**Returns:**the object that provides row selection state, null if row selection is not allowed**See Also:**[setSelectionModel(javax.swing.ListSelectionModel)](http://docs.google.com/javax/swing/JTable.html#setSelectionModel(javax.swing.ListSelectionModel))

### sorterChanged

public void **sorterChanged**([RowSorterEvent](http://docs.google.com/javax/swing/event/RowSorterEvent.html) e)

RowSorterListener notification that the RowSorter has changed in some way.

**Specified by:**[sorterChanged](http://docs.google.com/javax/swing/event/RowSorterListener.html#sorterChanged(javax.swing.event.RowSorterEvent)) in interface [RowSorterListener](http://docs.google.com/javax/swing/event/RowSorterListener.html) **Parameters:**e - the RowSorterEvent describing the change **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if e is null**Since:** 1.6

### tableChanged

public void **tableChanged**([TableModelEvent](http://docs.google.com/javax/swing/event/TableModelEvent.html) e)

Invoked when this table's TableModel generates a TableModelEvent. The TableModelEvent should be constructed in the coordinate system of the model; the appropriate mapping to the view coordinate system is performed by this JTable when it receives the event.

Application code will not use these methods explicitly, they are used internally by JTable.

Note that as of 1.3, this method clears the selection, if any.

**Specified by:**[tableChanged](http://docs.google.com/javax/swing/event/TableModelListener.html#tableChanged(javax.swing.event.TableModelEvent)) in interface [TableModelListener](http://docs.google.com/javax/swing/event/TableModelListener.html)

### columnAdded

public void **columnAdded**([TableColumnModelEvent](http://docs.google.com/javax/swing/event/TableColumnModelEvent.html) e)

Invoked when a column is added to the table column model.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[columnAdded](http://docs.google.com/javax/swing/event/TableColumnModelListener.html#columnAdded(javax.swing.event.TableColumnModelEvent)) in interface [TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html) **See Also:**[TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html)

### columnRemoved

public void **columnRemoved**([TableColumnModelEvent](http://docs.google.com/javax/swing/event/TableColumnModelEvent.html) e)

Invoked when a column is removed from the table column model.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[columnRemoved](http://docs.google.com/javax/swing/event/TableColumnModelListener.html#columnRemoved(javax.swing.event.TableColumnModelEvent)) in interface [TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html) **See Also:**[TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html)

### columnMoved

public void **columnMoved**([TableColumnModelEvent](http://docs.google.com/javax/swing/event/TableColumnModelEvent.html) e)

Invoked when a column is repositioned. If a cell is being edited, then editing is stopped and the cell is redrawn.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[columnMoved](http://docs.google.com/javax/swing/event/TableColumnModelListener.html#columnMoved(javax.swing.event.TableColumnModelEvent)) in interface [TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html) **Parameters:**e - the event received**See Also:**[TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html)

### columnMarginChanged

public void **columnMarginChanged**([ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) e)

Invoked when a column is moved due to a margin change. If a cell is being edited, then editing is stopped and the cell is redrawn.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[columnMarginChanged](http://docs.google.com/javax/swing/event/TableColumnModelListener.html#columnMarginChanged(javax.swing.event.ChangeEvent)) in interface [TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html) **Parameters:**e - the event received**See Also:**[TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html)

### columnSelectionChanged

public void **columnSelectionChanged**([ListSelectionEvent](http://docs.google.com/javax/swing/event/ListSelectionEvent.html) e)

Invoked when the selection model of the TableColumnModel is changed.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[columnSelectionChanged](http://docs.google.com/javax/swing/event/TableColumnModelListener.html#columnSelectionChanged(javax.swing.event.ListSelectionEvent)) in interface [TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html) **Parameters:**e - the event received**See Also:**[TableColumnModelListener](http://docs.google.com/javax/swing/event/TableColumnModelListener.html)

### valueChanged

public void **valueChanged**([ListSelectionEvent](http://docs.google.com/javax/swing/event/ListSelectionEvent.html) e)

Invoked when the row selection changes -- repaints to show the new selection.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[valueChanged](http://docs.google.com/javax/swing/event/ListSelectionListener.html#valueChanged(javax.swing.event.ListSelectionEvent)) in interface [ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html) **Parameters:**e - the event received**See Also:**[ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html)

### editingStopped

public void **editingStopped**([ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) e)

Invoked when editing is finished. The changes are saved and the editor is discarded.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[editingStopped](http://docs.google.com/javax/swing/event/CellEditorListener.html#editingStopped(javax.swing.event.ChangeEvent)) in interface [CellEditorListener](http://docs.google.com/javax/swing/event/CellEditorListener.html) **Parameters:**e - the event received**See Also:**[CellEditorListener](http://docs.google.com/javax/swing/event/CellEditorListener.html)

### editingCanceled

public void **editingCanceled**([ChangeEvent](http://docs.google.com/javax/swing/event/ChangeEvent.html) e)

Invoked when editing is canceled. The editor object is discarded and the cell is rendered once again.

Application code will not use these methods explicitly, they are used internally by JTable.

**Specified by:**[editingCanceled](http://docs.google.com/javax/swing/event/CellEditorListener.html#editingCanceled(javax.swing.event.ChangeEvent)) in interface [CellEditorListener](http://docs.google.com/javax/swing/event/CellEditorListener.html) **Parameters:**e - the event received**See Also:**[CellEditorListener](http://docs.google.com/javax/swing/event/CellEditorListener.html)

### setPreferredScrollableViewportSize

public void **setPreferredScrollableViewportSize**([Dimension](http://docs.google.com/java/awt/Dimension.html) size)

Sets the preferred size of the viewport for this table.

**Parameters:**size - a Dimension object specifying the preferredSize of a JViewport whose view is this table**See Also:**[Scrollable.getPreferredScrollableViewportSize()](http://docs.google.com/javax/swing/Scrollable.html#getPreferredScrollableViewportSize())

### getPreferredScrollableViewportSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getPreferredScrollableViewportSize**()

Returns the preferred size of the viewport for this table.

**Specified by:**[getPreferredScrollableViewportSize](http://docs.google.com/javax/swing/Scrollable.html#getPreferredScrollableViewportSize()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**a Dimension object containing the preferredSize of the JViewport which displays this table**See Also:**[Scrollable.getPreferredScrollableViewportSize()](http://docs.google.com/javax/swing/Scrollable.html#getPreferredScrollableViewportSize())

### getScrollableUnitIncrement

public int **getScrollableUnitIncrement**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect,  
 int orientation,  
 int direction)

Returns the scroll increment (in pixels) that completely exposes one new row or column (depending on the orientation).

This method is called each time the user requests a unit scroll.

**Specified by:**[getScrollableUnitIncrement](http://docs.google.com/javax/swing/Scrollable.html#getScrollableUnitIncrement(java.awt.Rectangle,%20int,%20int)) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Parameters:**visibleRect - the view area visible within the viewportorientation - either SwingConstants.VERTICAL or SwingConstants.HORIZONTALdirection - less than zero to scroll up/left, greater than zero for down/right **Returns:**the "unit" increment for scrolling in the specified direction**See Also:**[Scrollable.getScrollableUnitIncrement(java.awt.Rectangle, int, int)](http://docs.google.com/javax/swing/Scrollable.html#getScrollableUnitIncrement(java.awt.Rectangle,%20int,%20int))

### getScrollableBlockIncrement

public int **getScrollableBlockIncrement**([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect,  
 int orientation,  
 int direction)

Returns visibleRect.height or visibleRect.width, depending on this table's orientation. Note that as of Swing 1.1.1 (Java 2 v 1.2.2) the value returned will ensure that the viewport is cleanly aligned on a row boundary.

**Specified by:**[getScrollableBlockIncrement](http://docs.google.com/javax/swing/Scrollable.html#getScrollableBlockIncrement(java.awt.Rectangle,%20int,%20int)) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Parameters:**visibleRect - The view area visible within the viewportorientation - Either SwingConstants.VERTICAL or SwingConstants.HORIZONTAL.direction - Less than zero to scroll up/left, greater than zero for down/right. **Returns:**visibleRect.height or visibleRect.width per the orientation**See Also:**[Scrollable.getScrollableBlockIncrement(java.awt.Rectangle, int, int)](http://docs.google.com/javax/swing/Scrollable.html#getScrollableBlockIncrement(java.awt.Rectangle,%20int,%20int))

### getScrollableTracksViewportWidth

public boolean **getScrollableTracksViewportWidth**()

Returns false if autoResizeMode is set to AUTO\_RESIZE\_OFF, which indicates that the width of the viewport does not determine the width of the table. Otherwise returns true.

**Specified by:**[getScrollableTracksViewportWidth](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportWidth()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**false if autoResizeMode is set to AUTO\_RESIZE\_OFF, otherwise returns true**See Also:**[Scrollable.getScrollableTracksViewportWidth()](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportWidth())

### getScrollableTracksViewportHeight

public boolean **getScrollableTracksViewportHeight**()

Returns false to indicate that the height of the viewport does not determine the height of the table, unless getFillsViewportHeight is true and the preferred height of the table is smaller than the viewport's height.

**Specified by:**[getScrollableTracksViewportHeight](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportHeight()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**false unless getFillsViewportHeight is true and the table needs to be stretched to fill the viewport**See Also:**[Scrollable.getScrollableTracksViewportHeight()](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportHeight()), [setFillsViewportHeight(boolean)](http://docs.google.com/javax/swing/JTable.html#setFillsViewportHeight(boolean)), [getFillsViewportHeight()](http://docs.google.com/javax/swing/JTable.html#getFillsViewportHeight())

### setFillsViewportHeight

public void **setFillsViewportHeight**(boolean fillsViewportHeight)

Sets whether or not this table is always made large enough to fill the height of an enclosing viewport. If the preferred height of the table is smaller than the viewport, then the table will be stretched to fill the viewport. In other words, this ensures the table is never smaller than the viewport. The default for this property is false.

**Parameters:**fillsViewportHeight - whether or not this table is always made large enough to fill the height of an enclosing viewport**Since:** 1.6 **See Also:**[getFillsViewportHeight()](http://docs.google.com/javax/swing/JTable.html#getFillsViewportHeight()), [getScrollableTracksViewportHeight()](http://docs.google.com/javax/swing/JTable.html#getScrollableTracksViewportHeight())

### getFillsViewportHeight

public boolean **getFillsViewportHeight**()

Returns whether or not this table is always made large enough to fill the height of an enclosing viewport.

**Returns:**whether or not this table is always made large enough to fill the height of an enclosing viewport**Since:** 1.6 **See Also:**[setFillsViewportHeight(boolean)](http://docs.google.com/javax/swing/JTable.html#setFillsViewportHeight(boolean))

### processKeyBinding

protected boolean **processKeyBinding**([KeyStroke](http://docs.google.com/javax/swing/KeyStroke.html) ks,  
 [KeyEvent](http://docs.google.com/java/awt/event/KeyEvent.html) e,  
 int condition,  
 boolean pressed)

**Description copied from class:** [**JComponent**](http://docs.google.com/javax/swing/JComponent.html#processKeyBinding(javax.swing.KeyStroke,%20java.awt.event.KeyEvent,%20int,%20boolean)) Invoked to process the key bindings for ks as the result of the KeyEvent e. This obtains the appropriate InputMap, gets the binding, gets the action from the ActionMap, and then (if the action is found and the component is enabled) invokes notifyAction to notify the action.

**Overrides:**[processKeyBinding](http://docs.google.com/javax/swing/JComponent.html#processKeyBinding(javax.swing.KeyStroke,%20java.awt.event.KeyEvent,%20int,%20boolean)) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **Parameters:**ks - the KeyStroke queriede - the KeyEventcondition - one of the following values:

* JComponent.WHEN\_FOCUSED
* JComponent.WHEN\_ANCESTOR\_OF\_FOCUSED\_COMPONENT
* JComponent.WHEN\_IN\_FOCUSED\_WINDOW

pressed - true if the key is pressed **Returns:**true if there was a binding to an action, and the action was enabled

### createDefaultRenderers

protected void **createDefaultRenderers**()

Creates default cell renderers for objects, numbers, doubles, dates, booleans, and icons.

**See Also:**[DefaultTableCellRenderer](http://docs.google.com/javax/swing/table/DefaultTableCellRenderer.html)

### createDefaultEditors

protected void **createDefaultEditors**()

Creates default cell editors for objects, numbers, and boolean values.

**See Also:**[DefaultCellEditor](http://docs.google.com/javax/swing/DefaultCellEditor.html)

### initializeLocalVars

protected void **initializeLocalVars**()

Initializes table properties to their default values.

### createDefaultDataModel

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

Returns the default table model object, which is a DefaultTableModel. A subclass can override this method to return a different table model object.

**Returns:**the default table model object**See Also:**[DefaultTableModel](http://docs.google.com/javax/swing/table/DefaultTableModel.html)

### createDefaultColumnModel

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

Returns the default column model object, which is a DefaultTableColumnModel. A subclass can override this method to return a different column model object.

**Returns:**the default column model object**See Also:**[DefaultTableColumnModel](http://docs.google.com/javax/swing/table/DefaultTableColumnModel.html)

### createDefaultSelectionModel

protected [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) **createDefaultSelectionModel**()

Returns the default selection model object, which is a DefaultListSelectionModel. A subclass can override this method to return a different selection model object.

**Returns:**the default selection model object**See Also:**[DefaultListSelectionModel](http://docs.google.com/javax/swing/DefaultListSelectionModel.html)

### createDefaultTableHeader

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

Returns the default table header object, which is a JTableHeader. A subclass can override this method to return a different table header object.

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

### resizeAndRepaint

protected void **resizeAndRepaint**()

Equivalent to revalidate followed by repaint.

### getCellEditor

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

Returns the active cell editor, which is null if the table is not currently editing.

**Returns:**the TableCellEditor that does the editing, or null if the table is not currently editing.**See Also:**[cellEditor](http://docs.google.com/javax/swing/JTable.html#cellEditor), [getCellEditor(int, int)](http://docs.google.com/javax/swing/JTable.html#getCellEditor(int,%20int))

### setCellEditor

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

Sets the active cell editor.

**Parameters:**anEditor - the active cell editor**See Also:**[cellEditor](http://docs.google.com/javax/swing/JTable.html#cellEditor)

### setEditingColumn

public void **setEditingColumn**(int aColumn)

Sets the editingColumn variable.

**Parameters:**aColumn - the column of the cell to be edited**See Also:**[editingColumn](http://docs.google.com/javax/swing/JTable.html#editingColumn)

### setEditingRow

public void **setEditingRow**(int aRow)

Sets the editingRow variable.

**Parameters:**aRow - the row of the cell to be edited**See Also:**[editingRow](http://docs.google.com/javax/swing/JTable.html#editingRow)

### getCellRenderer

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

Returns an appropriate renderer for the cell specified by this row and column. If the TableColumn for this column has a non-null renderer, returns that. If not, finds the class of the data in this column (using getColumnClass) and returns the default renderer for this type of data.

**Note:** Throughout the table package, the internal implementations always use this method to provide renderers so that this default behavior can be safely overridden by a subclass.

**Parameters:**row - the row of the cell to render, where 0 is the first rowcolumn - the column of the cell to render, where 0 is the first column **Returns:**the assigned renderer; if null returns the default renderer for this type of object**See Also:**[DefaultTableCellRenderer](http://docs.google.com/javax/swing/table/DefaultTableCellRenderer.html), [TableColumn.setCellRenderer(javax.swing.table.TableCellRenderer)](http://docs.google.com/javax/swing/table/TableColumn.html#setCellRenderer(javax.swing.table.TableCellRenderer)), [setDefaultRenderer(java.lang.Class, javax.swing.table.TableCellRenderer)](http://docs.google.com/javax/swing/JTable.html#setDefaultRenderer(java.lang.Class,%20javax.swing.table.TableCellRenderer))

### prepareRenderer

public [Component](http://docs.google.com/java/awt/Component.html) **prepareRenderer**([TableCellRenderer](http://docs.google.com/javax/swing/table/TableCellRenderer.html) renderer,  
 int row,  
 int column)

Prepares the renderer by querying the data model for the value and selection state of the cell at row, column. Returns the component (may be a Component or a JComponent) under the event location.

During a printing operation, this method will configure the renderer without indicating selection or focus, to prevent them from appearing in the printed output. To do other customizations based on whether or not the table is being printed, you can check the value of [JComponent.isPaintingForPrint()](http://docs.google.com/javax/swing/JComponent.html#isPaintingForPrint()), either here or within custom renderers.

**Note:** Throughout the table package, the internal implementations always use this method to prepare renderers so that this default behavior can be safely overridden by a subclass.

**Parameters:**renderer - the TableCellRenderer to preparerow - the row of the cell to render, where 0 is the first rowcolumn - the column of the cell to render, where 0 is the first column **Returns:**the Component under the event location

### getCellEditor

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

Returns an appropriate editor for the cell specified by row and column. If the TableColumn for this column has a non-null editor, returns that. If not, finds the class of the data in this column (using getColumnClass) and returns the default editor for this type of data.

**Note:** Throughout the table package, the internal implementations always use this method to provide editors so that this default behavior can be safely overridden by a subclass.

**Parameters:**row - the row of the cell to edit, where 0 is the first rowcolumn - the column of the cell to edit, where 0 is the first column **Returns:**the editor for this cell; if null return the default editor for this type of cell**See Also:**[DefaultCellEditor](http://docs.google.com/javax/swing/DefaultCellEditor.html)

### prepareEditor

public [Component](http://docs.google.com/java/awt/Component.html) **prepareEditor**([TableCellEditor](http://docs.google.com/javax/swing/table/TableCellEditor.html) editor,  
 int row,  
 int column)

Prepares the editor by querying the data model for the value and selection state of the cell at row, column.

**Note:** Throughout the table package, the internal implementations always use this method to prepare editors so that this default behavior can be safely overridden by a subclass.

**Parameters:**editor - the TableCellEditor to set uprow - the row of the cell to edit, where 0 is the first rowcolumn - the column of the cell to edit, where 0 is the first column **Returns:**the Component being edited

### removeEditor

public void **removeEditor**()

Discards the editor object and frees the real estate it used for cell rendering.

### paramString

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

Returns a string representation of this table. This method is intended to be used only for debugging purposes, and the content and format of the returned string may vary between implementations. The returned string may be empty but may not be null.

**Overrides:**[paramString](http://docs.google.com/javax/swing/JComponent.html#paramString()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **Returns:**a string representation of this table

### print

public boolean **print**()  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html)

A convenience method that displays a printing dialog, and then prints this JTable in mode PrintMode.FIT\_WIDTH, with no header or footer text. A modal progress dialog, with an abort option, will be shown for the duration of printing.

Note: In headless mode, no dialogs are shown and printing occurs on the default printer.

**Returns:**true, unless printing is cancelled by the user **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if this thread is not allowed to initiate a print job request [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted**Since:** 1.5 **See Also:**[print(JTable.PrintMode, MessageFormat, MessageFormat, boolean, PrintRequestAttributeSet, boolean, PrintService)](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean,%20javax.print.PrintService)), [getPrintable(javax.swing.JTable.PrintMode, java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat))

### print

public boolean **print**([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode)  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html)

A convenience method that displays a printing dialog, and then prints this JTable in the given printing mode, with no header or footer text. A modal progress dialog, with an abort option, will be shown for the duration of printing.

Note: In headless mode, no dialogs are shown and printing occurs on the default printer.

**Parameters:**printMode - the printing mode that the printable should use **Returns:**true, unless printing is cancelled by the user **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if this thread is not allowed to initiate a print job request [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted**Since:** 1.5 **See Also:**[print(JTable.PrintMode, MessageFormat, MessageFormat, boolean, PrintRequestAttributeSet, boolean, PrintService)](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean,%20javax.print.PrintService)), [getPrintable(javax.swing.JTable.PrintMode, java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat))

### print

public boolean **print**([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html)

A convenience method that displays a printing dialog, and then prints this JTable in the given printing mode, with the specified header and footer text. A modal progress dialog, with an abort option, will be shown for the duration of printing.

Note: In headless mode, no dialogs are shown and printing occurs on the default printer.

**Parameters:**printMode - the printing mode that the printable should useheaderFormat - a MessageFormat specifying the text to be used in printing a header, or null for nonefooterFormat - a MessageFormat specifying the text to be used in printing a footer, or null for none **Returns:**true, unless printing is cancelled by the user **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if this thread is not allowed to initiate a print job request [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted**Since:** 1.5 **See Also:**[print(JTable.PrintMode, MessageFormat, MessageFormat, boolean, PrintRequestAttributeSet, boolean, PrintService)](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean,%20javax.print.PrintService)), [getPrintable(javax.swing.JTable.PrintMode, java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat))

### print

public boolean **print**([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat,  
 boolean showPrintDialog,  
 [PrintRequestAttributeSet](http://docs.google.com/javax/print/attribute/PrintRequestAttributeSet.html) attr,  
 boolean interactive)  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html),  
 [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html)

Prints this table, as specified by the fully featured [print](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean,%20javax.print.PrintService)) method, with the default printer specified as the print service.

**Parameters:**printMode - the printing mode that the printable should useheaderFormat - a MessageFormat specifying the text to be used in printing a header, or null for nonefooterFormat - a MessageFormat specifying the text to be used in printing a footer, or null for noneshowPrintDialog - whether or not to display a print dialogattr - a PrintRequestAttributeSet specifying any printing attributes, or null for noneinteractive - whether or not to print in an interactive mode **Returns:**true, unless printing is cancelled by the user **Throws:** [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html) - if the method is asked to show a printing dialog or run interactively, and GraphicsEnvironment.isHeadless returns true [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if this thread is not allowed to initiate a print job request [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted**Since:** 1.5 **See Also:**[print(JTable.PrintMode, MessageFormat, MessageFormat, boolean, PrintRequestAttributeSet, boolean, PrintService)](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean,%20javax.print.PrintService)), [getPrintable(javax.swing.JTable.PrintMode, java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat))

### print

public boolean **print**([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat,  
 boolean showPrintDialog,  
 [PrintRequestAttributeSet](http://docs.google.com/javax/print/attribute/PrintRequestAttributeSet.html) attr,  
 boolean interactive,  
 [PrintService](http://docs.google.com/javax/print/PrintService.html) service)  
 throws [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html),  
 [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html)

Prints this JTable. Takes steps that the majority of developers would take in order to print a JTable. In short, it prepares the table, calls getPrintable to fetch an appropriate Printable, and then sends it to the printer.

A boolean parameter allows you to specify whether or not a printing dialog is displayed to the user. When it is, the user may use the dialog to change the destination printer or printing attributes, or even to cancel the print. Another two parameters allow for a PrintService and printing attributes to be specified. These parameters can be used either to provide initial values for the print dialog, or to specify values when the dialog is not shown.

A second boolean parameter allows you to specify whether or not to perform printing in an interactive mode. If true, a modal progress dialog, with an abort option, is displayed for the duration of printing . This dialog also prevents any user action which may affect the table. However, it can not prevent the table from being modified by code (for example, another thread that posts updates using SwingUtilities.invokeLater). It is therefore the responsibility of the developer to ensure that no other code modifies the table in any way during printing (invalid modifications include changes in: size, renderers, or underlying data). Printing behavior is undefined when the table is changed during printing.

If false is specified for this parameter, no dialog will be displayed and printing will begin immediately on the event-dispatch thread. This blocks any other events, including repaints, from being processed until printing is complete. Although this effectively prevents the table from being changed, it doesn't provide a good user experience. For this reason, specifying false is only recommended when printing from an application with no visible GUI.

Note: Attempting to show the printing dialog or run interactively, while in headless mode, will result in a HeadlessException.

Before fetching the printable, this method will gracefully terminate editing, if necessary, to prevent an editor from showing in the printed result. Additionally, JTable will prepare its renderers during printing such that selection and focus are not indicated. As far as customizing further how the table looks in the printout, developers can provide custom renderers or paint code that conditionalize on the value of [JComponent.isPaintingForPrint()](http://docs.google.com/javax/swing/JComponent.html#isPaintingForPrint()).

See [getPrintable(javax.swing.JTable.PrintMode, java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat)) for more description on how the table is printed.

**Parameters:**printMode - the printing mode that the printable should useheaderFormat - a MessageFormat specifying the text to be used in printing a header, or null for nonefooterFormat - a MessageFormat specifying the text to be used in printing a footer, or null for noneshowPrintDialog - whether or not to display a print dialogattr - a PrintRequestAttributeSet specifying any printing attributes, or null for noneinteractive - whether or not to print in an interactive modeservice - the destination PrintService, or null to use the default printer **Returns:**true, unless printing is cancelled by the user **Throws:** [HeadlessException](http://docs.google.com/java/awt/HeadlessException.html) - if the method is asked to show a printing dialog or run interactively, and GraphicsEnvironment.isHeadless returns true [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and its [SecurityManager.checkPrintJobAccess()](http://docs.google.com/java/lang/SecurityManager.html#checkPrintJobAccess()) method disallows this thread from creating a print job request [PrinterException](http://docs.google.com/java/awt/print/PrinterException.html) - if an error in the print system causes the job to be aborted**Since:** 1.6 **See Also:**[getPrintable(javax.swing.JTable.PrintMode, java.text.MessageFormat, java.text.MessageFormat)](http://docs.google.com/javax/swing/JTable.html#getPrintable(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat)), [GraphicsEnvironment.isHeadless()](http://docs.google.com/java/awt/GraphicsEnvironment.html#isHeadless())

### getPrintable

public [Printable](http://docs.google.com/java/awt/print/Printable.html) **getPrintable**([JTable.PrintMode](http://docs.google.com/javax/swing/JTable.PrintMode.html) printMode,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) headerFormat,  
 [MessageFormat](http://docs.google.com/java/text/MessageFormat.html) footerFormat)

Return a Printable for use in printing this JTable.

This method is meant for those wishing to customize the default Printable implementation used by JTable's print methods. Developers wanting simply to print the table should use one of those methods directly.

The Printable can be requested in one of two printing modes. In both modes, it spreads table rows naturally in sequence across multiple pages, fitting as many rows as possible per page. PrintMode.NORMAL specifies that the table be printed at its current size. In this mode, there may be a need to spread columns across pages in a similar manner to that of the rows. When the need arises, columns are distributed in an order consistent with the table's ComponentOrientation. PrintMode.FIT\_WIDTH specifies that the output be scaled smaller, if necessary, to fit the table's entire width (and thereby all columns) on each page. Width and height are scaled equally, maintaining the aspect ratio of the output.

The Printable heads the portion of table on each page with the appropriate section from the table's JTableHeader, if it has one.

Header and footer text can be added to the output by providing MessageFormat arguments. The printing code requests Strings from the formats, providing a single item which may be included in the formatted string: an Integer representing the current page number.

You are encouraged to read the documentation for MessageFormat as some characters, such as single-quote, are special and need to be escaped.

Here's an example of creating a MessageFormat that can be used to print "Duke's Table: Page - " and the current page number:

// notice the escaping of the single quote  
 // notice how the page number is included with "{0}"  
 MessageFormat format = new MessageFormat("Duke''s Table: Page - {0}");

The Printable constrains what it draws to the printable area of each page that it prints. Under certain circumstances, it may find it impossible to fit all of a page's content into that area. In these cases the output may be clipped, but the implementation makes an effort to do something reasonable. Here are a few situations where this is known to occur, and how they may be handled by this particular implementation:

* In any mode, when the header or footer text is too wide to fit completely in the printable area -- print as much of the text as possible starting from the beginning, as determined by the table's ComponentOrientation.
* In any mode, when a row is too tall to fit in the printable area -- print the upper-most portion of the row and paint no lower border on the table.
* In PrintMode.NORMAL when a column is too wide to fit in the printable area -- print the center portion of the column and leave the left and right borders off the table.

It is entirely valid for this Printable to be wrapped inside another in order to create complex reports and documents. You may even request that different pages be rendered into different sized printable areas. The implementation must be prepared to handle this (possibly by doing its layout calculations on the fly). However, providing different heights to each page will likely not work well with PrintMode.NORMAL when it has to spread columns across pages.

As far as customizing how the table looks in the printed result, JTable itself will take care of hiding the selection and focus during printing. For additional customizations, your renderers or painting code can customize the look based on the value of [JComponent.isPaintingForPrint()](http://docs.google.com/javax/swing/JComponent.html#isPaintingForPrint())

Also, *before* calling this method you may wish to *first* modify the state of the table, such as to cancel cell editing or have the user size the table appropriately. However, you must not modify the state of the table *after* this Printable has been fetched (invalid modifications include changes in size or underlying data). The behavior of the returned Printable is undefined once the table has been changed.

**Parameters:**printMode - the printing mode that the printable should useheaderFormat - a MessageFormat specifying the text to be used in printing a header, or null for nonefooterFormat - a MessageFormat specifying the text to be used in printing a footer, or null for none **Returns:**a Printable for printing this JTable**Since:** 1.5 **See Also:**[print(JTable.PrintMode, MessageFormat, MessageFormat, boolean, PrintRequestAttributeSet, boolean)](http://docs.google.com/javax/swing/JTable.html#print(javax.swing.JTable.PrintMode,%20java.text.MessageFormat,%20java.text.MessageFormat,%20boolean,%20javax.print.attribute.PrintRequestAttributeSet,%20boolean)), [Printable](http://docs.google.com/java/awt/print/Printable.html), [PrinterJob](http://docs.google.com/java/awt/print/PrinterJob.html)

### getAccessibleContext

public [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) **getAccessibleContext**()

Gets the AccessibleContext associated with this JTable. For tables, the AccessibleContext takes the form of an AccessibleJTable. A new AccessibleJTable instance is created if necessary.

**Specified by:**[getAccessibleContext](http://docs.google.com/javax/accessibility/Accessible.html#getAccessibleContext()) in interface [Accessible](http://docs.google.com/javax/accessibility/Accessible.html)**Overrides:**[getAccessibleContext](http://docs.google.com/javax/swing/JComponent.html#getAccessibleContext()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **Returns:**an AccessibleJTable that serves as the AccessibleContext of this JTable

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JTable.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/JTabbedPane.ModelListener.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/JTable.AccessibleJTable.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/JTable.html)    [**NO FRAMES**](http://docs.google.com/JTable.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#26in1rg) | DETAIL: [FIELD](#2jxsxqh) | [CONSTR](#3tbugp1) | [METHOD](#3l18frh) |

[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).