| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.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 PACKAGE**](http://docs.google.com/javax/sql/rowset/spi/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/swing/border/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package javax.swing

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms.

**See:**

[**Description**](#2et92p0)

| **Interface Summary** | |
| --- | --- |
| [**Action**](http://docs.google.com/javax/swing/Action.html) | The Action interface provides a useful extension to the ActionListener interface in cases where the same functionality may be accessed by several controls. |
| [**BoundedRangeModel**](http://docs.google.com/javax/swing/BoundedRangeModel.html) | Defines the data model used by components like Sliders and ProgressBars. |
| [**ButtonModel**](http://docs.google.com/javax/swing/ButtonModel.html) | State model for buttons. |
| [**CellEditor**](http://docs.google.com/javax/swing/CellEditor.html) | This interface defines the methods any general editor should be able to implement. |
| [**ComboBoxEditor**](http://docs.google.com/javax/swing/ComboBoxEditor.html) | The editor component used for JComboBox components. |
| [**ComboBoxModel**](http://docs.google.com/javax/swing/ComboBoxModel.html) | A data model for a combo box. |
| [**DesktopManager**](http://docs.google.com/javax/swing/DesktopManager.html) | DesktopManager objects are owned by a JDesktopPane object. |
| [**Icon**](http://docs.google.com/javax/swing/Icon.html) | A small fixed size picture, typically used to decorate components. |
| [**JComboBox.KeySelectionManager**](http://docs.google.com/javax/swing/JComboBox.KeySelectionManager.html) | The interface that defines a KeySelectionManager. |
| [**ListCellRenderer**](http://docs.google.com/javax/swing/ListCellRenderer.html) | Identifies components that can be used as "rubber stamps" to paint the cells in a JList. |
| [**ListModel**](http://docs.google.com/javax/swing/ListModel.html) | This interface defines the methods components like JList use to get the value of each cell in a list and the length of the list. |
| [**ListSelectionModel**](http://docs.google.com/javax/swing/ListSelectionModel.html) | This interface represents the current state of the selection for any of the components that display a list of values with stable indices. |
| [**MenuElement**](http://docs.google.com/javax/swing/MenuElement.html) | Any component that can be placed into a menu should implement this interface. |
| [**MutableComboBoxModel**](http://docs.google.com/javax/swing/MutableComboBoxModel.html) | A mutable version of ComboBoxModel. |
| [**Renderer**](http://docs.google.com/javax/swing/Renderer.html) | Defines the requirements for an object responsible for "rendering" (displaying) a value. |
| [**RootPaneContainer**](http://docs.google.com/javax/swing/RootPaneContainer.html) | This interface is implemented by components that have a single JRootPane child: JDialog, JFrame, JWindow, JApplet, JInternalFrame. |
| [**Scrollable**](http://docs.google.com/javax/swing/Scrollable.html) | An interface that provides information to a scrolling container like JScrollPane. |
| [**ScrollPaneConstants**](http://docs.google.com/javax/swing/ScrollPaneConstants.html) | Constants used with the JScrollPane component. |
| [**SingleSelectionModel**](http://docs.google.com/javax/swing/SingleSelectionModel.html) | A model that supports at most one indexed selection. |
| [**SpinnerModel**](http://docs.google.com/javax/swing/SpinnerModel.html) | A model for a potentially unbounded sequence of object values. |
| [**SwingConstants**](http://docs.google.com/javax/swing/SwingConstants.html) | A collection of constants generally used for positioning and orienting components on the screen. |
| [**UIDefaults.ActiveValue**](http://docs.google.com/javax/swing/UIDefaults.ActiveValue.html) | This class enables one to store an entry in the defaults table that's constructed each time it's looked up with one of the getXXX(key) methods. |
| [**UIDefaults.LazyValue**](http://docs.google.com/javax/swing/UIDefaults.LazyValue.html) | This class enables one to store an entry in the defaults table that isn't constructed until the first time it's looked up with one of the getXXX(key) methods. |
| [**WindowConstants**](http://docs.google.com/javax/swing/WindowConstants.html) | Constants used to control the window-closing operation. |

| **Class Summary** | |
| --- | --- |
| [**AbstractAction**](http://docs.google.com/javax/swing/AbstractAction.html) | This class provides default implementations for the JFC Action interface. |
| [**AbstractButton**](http://docs.google.com/javax/swing/AbstractButton.html) | Defines common behaviors for buttons and menu items. |
| [**AbstractCellEditor**](http://docs.google.com/javax/swing/AbstractCellEditor.html) |  |
| [**AbstractListModel**](http://docs.google.com/javax/swing/AbstractListModel.html) | The abstract definition for the data model that provides a List with its contents. |
| [**AbstractSpinnerModel**](http://docs.google.com/javax/swing/AbstractSpinnerModel.html) | This class provides the ChangeListener part of the SpinnerModel interface that should be suitable for most concrete SpinnerModel implementations. |
| [**ActionMap**](http://docs.google.com/javax/swing/ActionMap.html) | ActionMap provides mappings from Objects (called *keys* or *Action names*) to Actions. |
| [**BorderFactory**](http://docs.google.com/javax/swing/BorderFactory.html) | Factory class for vending standard Border objects. |
| [**Box**](http://docs.google.com/javax/swing/Box.html) | A lightweight container that uses a BoxLayout object as its layout manager. |
| [**Box.Filler**](http://docs.google.com/javax/swing/Box.Filler.html) | An implementation of a lightweight component that participates in layout but has no view. |
| [**BoxLayout**](http://docs.google.com/javax/swing/BoxLayout.html) | A layout manager that allows multiple components to be laid out either vertically or horizontally. |
| [**ButtonGroup**](http://docs.google.com/javax/swing/ButtonGroup.html) | This class is used to create a multiple-exclusion scope for a set of buttons. |
| [**CellRendererPane**](http://docs.google.com/javax/swing/CellRendererPane.html) | This class is inserted in between cell renderers and the components that use them. |
| [**ComponentInputMap**](http://docs.google.com/javax/swing/ComponentInputMap.html) | A ComponentInputMap is an InputMap associated with a particular JComponent. |
| [**DebugGraphics**](http://docs.google.com/javax/swing/DebugGraphics.html) | Graphics subclass supporting graphics debugging. |
| [**DefaultBoundedRangeModel**](http://docs.google.com/javax/swing/DefaultBoundedRangeModel.html) | A generic implementation of BoundedRangeModel. |
| [**DefaultButtonModel**](http://docs.google.com/javax/swing/DefaultButtonModel.html) | The default implementation of a Button component's data model. |
| [**DefaultCellEditor**](http://docs.google.com/javax/swing/DefaultCellEditor.html) | The default editor for table and tree cells. |
| [**DefaultComboBoxModel**](http://docs.google.com/javax/swing/DefaultComboBoxModel.html) | The default model for combo boxes. |
| [**DefaultDesktopManager**](http://docs.google.com/javax/swing/DefaultDesktopManager.html) | This is an implementation of the DesktopManager. |
| [**DefaultFocusManager**](http://docs.google.com/javax/swing/DefaultFocusManager.html) | This class has been obsoleted by the 1.4 focus APIs. |
| [**DefaultListCellRenderer**](http://docs.google.com/javax/swing/DefaultListCellRenderer.html) | Renders an item in a list. |
| [**DefaultListCellRenderer.UIResource**](http://docs.google.com/javax/swing/DefaultListCellRenderer.UIResource.html) | A subclass of DefaultListCellRenderer that implements UIResource. |
| [**DefaultListModel**](http://docs.google.com/javax/swing/DefaultListModel.html) | This class loosely implements the java.util.Vector API, in that it implements the 1.1.x version of java.util.Vector, has no collection class support, and notifies the ListDataListeners when changes occur. |
| [**DefaultListSelectionModel**](http://docs.google.com/javax/swing/DefaultListSelectionModel.html) | Default data model for list selections. |
| [**DefaultRowSorter<M,I>**](http://docs.google.com/javax/swing/DefaultRowSorter.html) | An implementation of RowSorter that provides sorting and filtering around a grid-based data model. |
| [**DefaultRowSorter.ModelWrapper<M,I>**](http://docs.google.com/javax/swing/DefaultRowSorter.ModelWrapper.html) | DefaultRowSorter.ModelWrapper is responsible for providing the data that gets sorted by DefaultRowSorter. |
| [**DefaultSingleSelectionModel**](http://docs.google.com/javax/swing/DefaultSingleSelectionModel.html) | A generic implementation of SingleSelectionModel. |
| [**FocusManager**](http://docs.google.com/javax/swing/FocusManager.html) | This class has been obsoleted by the 1.4 focus APIs. |
| [**GrayFilter**](http://docs.google.com/javax/swing/GrayFilter.html) | An image filter that "disables" an image by turning it into a grayscale image, and brightening the pixels in the image. |
| [**GroupLayout**](http://docs.google.com/javax/swing/GroupLayout.html) | GroupLayout is a LayoutManager that hierarchically groups components in order to position them in a Container. |
| [**ImageIcon**](http://docs.google.com/javax/swing/ImageIcon.html) | An implementation of the Icon interface that paints Icons from Images. |
| [**InputMap**](http://docs.google.com/javax/swing/InputMap.html) | InputMap provides a binding between an input event (currently only KeyStrokes are used) and an Object. |
| [**InputVerifier**](http://docs.google.com/javax/swing/InputVerifier.html) | The purpose of this class is to help clients support smooth focus navigation through GUIs with text fields. |
| [**InternalFrameFocusTraversalPolicy**](http://docs.google.com/javax/swing/InternalFrameFocusTraversalPolicy.html) | A FocusTraversalPolicy which can optionally provide an algorithm for determining a JInternalFrame's initial Component. |
| [**JApplet**](http://docs.google.com/javax/swing/JApplet.html) | An extended version of java.applet.Applet that adds support for the JFC/Swing component architecture. |
| [**JButton**](http://docs.google.com/javax/swing/JButton.html) | An implementation of a "push" button. |
| [**JCheckBox**](http://docs.google.com/javax/swing/JCheckBox.html) | An implementation of a check box -- an item that can be selected or deselected, and which displays its state to the user. |
| [**JCheckBoxMenuItem**](http://docs.google.com/javax/swing/JCheckBoxMenuItem.html) | A menu item that can be selected or deselected. |
| [**JColorChooser**](http://docs.google.com/javax/swing/JColorChooser.html) | JColorChooser provides a pane of controls designed to allow a user to manipulate and select a color. |
| [**JComboBox**](http://docs.google.com/javax/swing/JComboBox.html) | A component that combines a button or editable field and a drop-down list. |
| [**JComponent**](http://docs.google.com/javax/swing/JComponent.html) | The base class for all Swing components except top-level containers. |
| [**JDesktopPane**](http://docs.google.com/javax/swing/JDesktopPane.html) | A container used to create a multiple-document interface or a virtual desktop. |
| [**JDialog**](http://docs.google.com/javax/swing/JDialog.html) | The main class for creating a dialog window. |
| [**JEditorPane**](http://docs.google.com/javax/swing/JEditorPane.html) | A text component to edit various kinds of content. |
| [**JFileChooser**](http://docs.google.com/javax/swing/JFileChooser.html) | JFileChooser provides a simple mechanism for the user to choose a file. |
| [**JFormattedTextField**](http://docs.google.com/javax/swing/JFormattedTextField.html) | JFormattedTextField extends JTextField adding support for formatting arbitrary values, as well as retrieving a particular object once the user has edited the text. |
| [**JFormattedTextField.AbstractFormatter**](http://docs.google.com/javax/swing/JFormattedTextField.AbstractFormatter.html) | Instances of AbstractFormatter are used by JFormattedTextField to handle the conversion both from an Object to a String, and back from a String to an Object. |
| [**JFormattedTextField.AbstractFormatterFactory**](http://docs.google.com/javax/swing/JFormattedTextField.AbstractFormatterFactory.html) | Instances of AbstractFormatterFactory are used by JFormattedTextField to obtain instances of AbstractFormatter which in turn are used to format values. |
| [**JFrame**](http://docs.google.com/javax/swing/JFrame.html) | An extended version of java.awt.Frame that adds support for the JFC/Swing component architecture. |
| [**JInternalFrame**](http://docs.google.com/javax/swing/JInternalFrame.html) | A lightweight object that provides many of the features of a native frame, including dragging, closing, becoming an icon, resizing, title display, and support for a menu bar. |
| [**JInternalFrame.JDesktopIcon**](http://docs.google.com/javax/swing/JInternalFrame.JDesktopIcon.html) | This component represents an iconified version of a JInternalFrame. |
| [**JLabel**](http://docs.google.com/javax/swing/JLabel.html) | A display area for a short text string or an image, or both. |
| [**JLayeredPane**](http://docs.google.com/javax/swing/JLayeredPane.html) | JLayeredPane adds depth to a JFC/Swing container, allowing components to overlap each other when needed. |
| [**JList**](http://docs.google.com/javax/swing/JList.html) | A component that displays a list of objects and allows the user to select one or more items. |
| [**JList.DropLocation**](http://docs.google.com/javax/swing/JList.DropLocation.html) | A subclass of TransferHandler.DropLocation representing a drop location for a JList. |
| [**JMenu**](http://docs.google.com/javax/swing/JMenu.html) | An implementation of a menu -- a popup window containing JMenuItems that is displayed when the user selects an item on the JMenuBar. |
| [**JMenuBar**](http://docs.google.com/javax/swing/JMenuBar.html) | An implementation of a menu bar. |
| [**JMenuItem**](http://docs.google.com/javax/swing/JMenuItem.html) | An implementation of an item in a menu. |
| [**JOptionPane**](http://docs.google.com/javax/swing/JOptionPane.html) | JOptionPane makes it easy to pop up a standard dialog box that prompts users for a value or informs them of something. |
| [**JPanel**](http://docs.google.com/javax/swing/JPanel.html) | JPanel is a generic lightweight container. |
| [**JPasswordField**](http://docs.google.com/javax/swing/JPasswordField.html) | JPasswordField is a lightweight component that allows the editing of a single line of text where the view indicates something was typed, but does not show the original characters. |
| [**JPopupMenu**](http://docs.google.com/javax/swing/JPopupMenu.html) | An implementation of a popup menu -- a small window that pops up and displays a series of choices. |
| [**JPopupMenu.Separator**](http://docs.google.com/javax/swing/JPopupMenu.Separator.html) | A popup menu-specific separator. |
| [**JProgressBar**](http://docs.google.com/javax/swing/JProgressBar.html) | A component that visually displays the progress of some task. |
| [**JRadioButton**](http://docs.google.com/javax/swing/JRadioButton.html) | An implementation of a radio button -- an item that can be selected or deselected, and which displays its state to the user. |
| [**JRadioButtonMenuItem**](http://docs.google.com/javax/swing/JRadioButtonMenuItem.html) | An implementation of a radio button menu item. |
| [**JRootPane**](http://docs.google.com/javax/swing/JRootPane.html) | A lightweight container used behind the scenes by JFrame, JDialog, JWindow, JApplet, and JInternalFrame. |
| [**JScrollBar**](http://docs.google.com/javax/swing/JScrollBar.html) | An implementation of a scrollbar. |
| [**JScrollPane**](http://docs.google.com/javax/swing/JScrollPane.html) | Provides a scrollable view of a lightweight component. |
| [**JSeparator**](http://docs.google.com/javax/swing/JSeparator.html) | JSeparator provides a general purpose component for implementing divider lines - most commonly used as a divider between menu items that breaks them up into logical groupings. |
| [**JSlider**](http://docs.google.com/javax/swing/JSlider.html) | A component that lets the user graphically select a value by sliding a knob within a bounded interval. |
| [**JSpinner**](http://docs.google.com/javax/swing/JSpinner.html) | A single line input field that lets the user select a number or an object value from an ordered sequence. |
| [**JSpinner.DateEditor**](http://docs.google.com/javax/swing/JSpinner.DateEditor.html) | An editor for a JSpinner whose model is a SpinnerDateModel. |
| [**JSpinner.DefaultEditor**](http://docs.google.com/javax/swing/JSpinner.DefaultEditor.html) | A simple base class for more specialized editors that displays a read-only view of the model's current value with a JFormattedTextField. |
| [**JSpinner.ListEditor**](http://docs.google.com/javax/swing/JSpinner.ListEditor.html) | An editor for a JSpinner whose model is a SpinnerListModel. |
| [**JSpinner.NumberEditor**](http://docs.google.com/javax/swing/JSpinner.NumberEditor.html) | An editor for a JSpinner whose model is a SpinnerNumberModel. |
| [**JSplitPane**](http://docs.google.com/javax/swing/JSplitPane.html) | JSplitPane is used to divide two (and only two) Components. |
| [**JTabbedPane**](http://docs.google.com/javax/swing/JTabbedPane.html) | A component that lets the user switch between a group of components by clicking on a tab with a given title and/or icon. |
| [**JTable**](http://docs.google.com/javax/swing/JTable.html) | The JTable is used to display and edit regular two-dimensional tables of cells. |
| [**JTable.DropLocation**](http://docs.google.com/javax/swing/JTable.DropLocation.html) | A subclass of TransferHandler.DropLocation representing a drop location for a JTable. |
| [**JTextArea**](http://docs.google.com/javax/swing/JTextArea.html) | A JTextArea is a multi-line area that displays plain text. |
| [**JTextField**](http://docs.google.com/javax/swing/JTextField.html) | JTextField is a lightweight component that allows the editing of a single line of text. |
| [**JTextPane**](http://docs.google.com/javax/swing/JTextPane.html) | A text component that can be marked up with attributes that are represented graphically. |
| [**JToggleButton**](http://docs.google.com/javax/swing/JToggleButton.html) | An implementation of a two-state button. |
| [**JToggleButton.ToggleButtonModel**](http://docs.google.com/javax/swing/JToggleButton.ToggleButtonModel.html) | The ToggleButton model |
| [**JToolBar**](http://docs.google.com/javax/swing/JToolBar.html) | JToolBar provides a component that is useful for displaying commonly used Actions or controls. |
| [**JToolBar.Separator**](http://docs.google.com/javax/swing/JToolBar.Separator.html) | A toolbar-specific separator. |
| [**JToolTip**](http://docs.google.com/javax/swing/JToolTip.html) | Used to display a "Tip" for a Component. |
| [**JTree**](http://docs.google.com/javax/swing/JTree.html) | A control that displays a set of hierarchical data as an outline. |
| [**JTree.DropLocation**](http://docs.google.com/javax/swing/JTree.DropLocation.html) | A subclass of TransferHandler.DropLocation representing a drop location for a JTree. |
| [**JTree.DynamicUtilTreeNode**](http://docs.google.com/javax/swing/JTree.DynamicUtilTreeNode.html) | DynamicUtilTreeNode can wrap vectors/hashtables/arrays/strings and create the appropriate children tree nodes as necessary. |
| [**JTree.EmptySelectionModel**](http://docs.google.com/javax/swing/JTree.EmptySelectionModel.html) | EmptySelectionModel is a TreeSelectionModel that does not allow anything to be selected. |
| [**JViewport**](http://docs.google.com/javax/swing/JViewport.html) | The "viewport" or "porthole" through which you see the underlying information. |
| [**JWindow**](http://docs.google.com/javax/swing/JWindow.html) | A JWindow is a container that can be displayed anywhere on the user's desktop. |
| [**KeyStroke**](http://docs.google.com/javax/swing/KeyStroke.html) | A KeyStroke represents a key action on the keyboard, or equivalent input device. |
| [**LayoutFocusTraversalPolicy**](http://docs.google.com/javax/swing/LayoutFocusTraversalPolicy.html) | A SortingFocusTraversalPolicy which sorts Components based on their size, position, and orientation. |
| [**LayoutStyle**](http://docs.google.com/javax/swing/LayoutStyle.html) | LayoutStyle provides information about how to position components. |
| [**LookAndFeel**](http://docs.google.com/javax/swing/LookAndFeel.html) | LookAndFeel, as the name implies, encapsulates a look and feel. |
| [**MenuSelectionManager**](http://docs.google.com/javax/swing/MenuSelectionManager.html) | A MenuSelectionManager owns the selection in menu hierarchy. |
| [**OverlayLayout**](http://docs.google.com/javax/swing/OverlayLayout.html) | A layout manager to arrange components over the top of each other. |
| [**Popup**](http://docs.google.com/javax/swing/Popup.html) | Popups are used to display a Component to the user, typically on top of all the other Components in a particular containment hierarchy. |
| [**PopupFactory**](http://docs.google.com/javax/swing/PopupFactory.html) | PopupFactory, as the name implies, is used to obtain instances of Popups. |
| [**ProgressMonitor**](http://docs.google.com/javax/swing/ProgressMonitor.html) | A class to monitor the progress of some operation. |
| [**ProgressMonitorInputStream**](http://docs.google.com/javax/swing/ProgressMonitorInputStream.html) | Monitors the progress of reading from some InputStream. |
| [**RepaintManager**](http://docs.google.com/javax/swing/RepaintManager.html) | This class manages repaint requests, allowing the number of repaints to be minimized, for example by collapsing multiple requests into a single repaint for members of a component tree. |
| [**RowFilter<M,I>**](http://docs.google.com/javax/swing/RowFilter.html) | RowFilter is used to filter out entries from the model so that they are not shown in the view. |
| [**RowFilter.Entry<M,I>**](http://docs.google.com/javax/swing/RowFilter.Entry.html) | An Entry object is passed to instances of RowFilter, allowing the filter to get the value of the entry's data, and thus to determine whether the entry should be shown. |
| [**RowSorter<M>**](http://docs.google.com/javax/swing/RowSorter.html) | RowSorter provides the basis for sorting and filtering. |
| [**RowSorter.SortKey**](http://docs.google.com/javax/swing/RowSorter.SortKey.html) | SortKey describes the sort order for a particular column. |
| [**ScrollPaneLayout**](http://docs.google.com/javax/swing/ScrollPaneLayout.html) | The layout manager used by JScrollPane. |
| [**ScrollPaneLayout.UIResource**](http://docs.google.com/javax/swing/ScrollPaneLayout.UIResource.html) | The UI resource version of ScrollPaneLayout. |
| [**SizeRequirements**](http://docs.google.com/javax/swing/SizeRequirements.html) | For the convenience of layout managers, calculates information about the size and position of components. |
| [**SizeSequence**](http://docs.google.com/javax/swing/SizeSequence.html) | A SizeSequence object efficiently maintains an ordered list of sizes and corresponding positions. |
| [**SortingFocusTraversalPolicy**](http://docs.google.com/javax/swing/SortingFocusTraversalPolicy.html) | A FocusTraversalPolicy that determines traversal order by sorting the Components of a focus traversal cycle based on a given Comparator. |
| [**SpinnerDateModel**](http://docs.google.com/javax/swing/SpinnerDateModel.html) | A SpinnerModel for sequences of Dates. |
| [**SpinnerListModel**](http://docs.google.com/javax/swing/SpinnerListModel.html) | A simple implementation of SpinnerModel whose values are defined by an array or a List. |
| [**SpinnerNumberModel**](http://docs.google.com/javax/swing/SpinnerNumberModel.html) | A SpinnerModel for sequences of numbers. |
| [**Spring**](http://docs.google.com/javax/swing/Spring.html) | An instance of the Spring class holds three properties that characterize its behavior: the *minimum*, *preferred*, and *maximum* values. |
| [**SpringLayout**](http://docs.google.com/javax/swing/SpringLayout.html) | A SpringLayout lays out the children of its associated container according to a set of constraints. |
| [**SpringLayout.Constraints**](http://docs.google.com/javax/swing/SpringLayout.Constraints.html) | A Constraints object holds the constraints that govern the way a component's size and position change in a container controlled by a SpringLayout. |
| [**SwingUtilities**](http://docs.google.com/javax/swing/SwingUtilities.html) | A collection of utility methods for Swing. |
| [**SwingWorker<T,V>**](http://docs.google.com/javax/swing/SwingWorker.html) | An abstract class to perform lengthy GUI-interacting tasks in a dedicated thread. |
| [**Timer**](http://docs.google.com/javax/swing/Timer.html) | Fires one or more ActionEvents at specified intervals. |
| [**ToolTipManager**](http://docs.google.com/javax/swing/ToolTipManager.html) | Manages all the ToolTips in the system. |
| [**TransferHandler**](http://docs.google.com/javax/swing/TransferHandler.html) | This class is used to handle the transfer of a Transferable to and from Swing components. |
| [**TransferHandler.DropLocation**](http://docs.google.com/javax/swing/TransferHandler.DropLocation.html) | Represents a location where dropped data should be inserted. |
| [**TransferHandler.TransferSupport**](http://docs.google.com/javax/swing/TransferHandler.TransferSupport.html) | This class encapsulates all relevant details of a clipboard or drag and drop transfer, and also allows for customizing aspects of the drag and drop experience. |
| [**UIDefaults**](http://docs.google.com/javax/swing/UIDefaults.html) | A table of defaults for Swing components. |
| [**UIDefaults.LazyInputMap**](http://docs.google.com/javax/swing/UIDefaults.LazyInputMap.html) | LazyInputMap will create a InputMap in its createValue method. |
| [**UIDefaults.ProxyLazyValue**](http://docs.google.com/javax/swing/UIDefaults.ProxyLazyValue.html) | This class provides an implementation of LazyValue which can be used to delay loading of the Class for the instance to be created. |
| [**UIManager**](http://docs.google.com/javax/swing/UIManager.html) | UIManager manages the current look and feel, the set of available look and feels, PropertyChangeListeners that are notified when the look and feel changes, look and feel defaults, and convenience methods for obtaining various default values. |
| [**UIManager.LookAndFeelInfo**](http://docs.google.com/javax/swing/UIManager.LookAndFeelInfo.html) | Provides a little information about an installed LookAndFeel for the sake of configuring a menu or for initial application set up. |
| [**ViewportLayout**](http://docs.google.com/javax/swing/ViewportLayout.html) | The default layout manager for JViewport. |

| **Enum Summary** | |
| --- | --- |
| [**DropMode**](http://docs.google.com/javax/swing/DropMode.html) | Drop modes, used to determine the method by which a component tracks and indicates a drop location during drag and drop. |
| [**GroupLayout.Alignment**](http://docs.google.com/javax/swing/GroupLayout.Alignment.html) | Enumeration of the possible ways ParallelGroup can align its children. |
| [**JTable.PrintMode**](http://docs.google.com/javax/swing/JTable.PrintMode.html) | Printing modes, used in printing JTables. |
| [**LayoutStyle.ComponentPlacement**](http://docs.google.com/javax/swing/LayoutStyle.ComponentPlacement.html) | ComponentPlacement is an enumeration of the possible ways two components can be placed relative to each other. |
| [**RowFilter.ComparisonType**](http://docs.google.com/javax/swing/RowFilter.ComparisonType.html) | Enumeration of the possible comparison values supported by some of the default RowFilters. |
| [**SortOrder**](http://docs.google.com/javax/swing/SortOrder.html) | SortOrder is an enumeration of the possible sort orderings. |
| [**SwingWorker.StateValue**](http://docs.google.com/javax/swing/SwingWorker.StateValue.html) | Values for the state bound property. |

| **Exception Summary** | |
| --- | --- |
| [**UnsupportedLookAndFeelException**](http://docs.google.com/javax/swing/UnsupportedLookAndFeelException.html) | An exception that indicates the requested look & feel management classes are not present on the user's system. |

## Package javax.swing Description

Provides a set of "lightweight" (all-Java language) components that, to the maximum degree possible, work the same on all platforms. For a programmer's guide to using these components, see [Creating a GUI with JFC/Swing](http://java.sun.com/docs/books/tutorial/uiswing/index.html), a trail in *The Java Tutorial*. For other resources, see [Related Documentation](#_3dy6vkm).

## Swing's Threading Policy

In general Swing is not thread safe. All Swing components and related classes, unless otherwise documented, must be accessed on the event dispatching thread.

Typical Swing applications do processing in response to an event generated from a user gesture. For example, clicking on a JButton notifies all ActionListeners added to the JButton. As all events generated from a user gesture are dispatched on the event dispatching thread, most developers are not impacted by the restriction.

Where the impact lies, however, is in constructing and showing a Swing application. Calls to an application's main method, or methods in Applet, are not invoked on the event dispatching thread. As such, care must be taken to transfer control to the event dispatching thread when constructing and showing an application or applet. The preferred way to transfer control and begin working with Swing is to use invokeLater. The invokeLater method schedules a Runnable to be processed on the event dispatching thread. The following two examples work equally well for transferring control and starting up a Swing application:

public class MyApp implements Runnable {  
 public void run() {  
 // Invoked on the event dispatching thread.  
 // Construct and show GUI.  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.invokeLater(new MyApp(args));  
 }  
}

Or:

public class MyApp {  
 MyApp(String[] args) {  
 // Invoked on the event dispatching thread. Do any initialization  
 // here.  
 }  
  
 public void show() {  
 // Show the UI.  
 }  
  
 public static void main(final String[] args) {  
 // Schedule a job for the event-dispatching thread:  
 // creating and showing this application's GUI.  
 SwingUtilities.invokeLater(new Runnable() {  
 public void run() {  
 new MyApp(args).show();  
 }  
 });  
 }  
}

This restriction also applies to models attached to Swing components. For example, if a TableModel is attached to a JTable, the TableModel should only be modified on the event dispatching thread. If you modify the model on a separate thread you run the risk of exceptions and possible display corruption.

As all events are delivered on the event dispatching thread, care must be taken in event processing. In particular, a long running task, such as network io or computational intensive processing, executed on the event dispatching thread blocks the event dispatching thread from dispatching any other events. While the event dispatching thread is blocked the application is completely unresponsive to user input. Refer to SwingWorker for the preferred way to do such processing when working with Swing.

More information on this topic can be found in the [Swing tutorial](http://java.sun.com/docs/books/tutorial/uiswing/), in particular the section on [How to Use Threads](http://java.sun.com/docs/books/tutorial/uiswing/misc/threads.html).

## Related Documentation

For overviews, tutorials, examples, guides, and other documentation, please see:

* [The Swing Connection](http://java.sun.com/products/jfc/tsc/)
* [The Java Tutorial](http://java.sun.com/docs/books/tutorial/)
* [Online Training](http://developer.java.sun.com/developer/onlineTraining/) at the Java Developer ConnectionSM
* [Java Foundation Classes (JFC)](http://java.sun.com/products/jfc/) home page

| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.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 PACKAGE**](http://docs.google.com/javax/sql/rowset/spi/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/javax/swing/border/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

[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).