| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JList.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/JLayeredPane.AccessibleJLayeredPane.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/JList.AccessibleJList.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/JList.html)    [**NO FRAMES**](http://docs.google.com/JList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | [FIELD](#2s8eyo1) | [CONSTR](#lnxbz9) | [METHOD](#35nkun2) | DETAIL: [FIELD](#3j2qqm3) | [CONSTR](#1ci93xb) | [METHOD](#1pxezwc) |

## **javax.swing**

Class JList

[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.JList**

**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), [Accessible](http://docs.google.com/javax/accessibility/Accessible.html), [Scrollable](http://docs.google.com/javax/swing/Scrollable.html)

public class **JList**extends [JComponent](http://docs.google.com/javax/swing/JComponent.html)implements [Scrollable](http://docs.google.com/javax/swing/Scrollable.html), [Accessible](http://docs.google.com/javax/accessibility/Accessible.html)

A component that displays a list of objects and allows the user to select one or more items. A separate model, ListModel, maintains the contents of the list.

It's easy to display an array or Vector of objects, using the JList constructor that automatically builds a read-only ListModel instance for you:

// Create a JList that displays strings from an array  
  
 String[] data = {"one", "two", "three", "four"};  
 JList myList = new JList(data);  
  
 // Create a JList that displays the superclasses of JList.class, by  
 // creating it with a Vector populated with this data  
  
 Vector superClasses = new Vector();  
 Class rootClass = javax.swing.JList.class;  
 for(Class cls = rootClass; cls != null; cls = cls.getSuperclass()) {  
 superClasses.addElement(cls);  
 }  
 JList myList = new JList(superClasses);  
   
 // The automatically created model is stored in JList's "model"  
 // property, which you can retrieve  
  
 ListModel model = myList.getModel();  
 for(int i = 0; i < model.getSize(); i++) {  
 System.out.println(model.getElementAt(i));  
 }

A ListModel can be supplied directly to a JList by way of a constructor or the setModel method. The contents need not be static - the number of items, and the values of items can change over time. A correct ListModel implementation notifies the set of javax.swing.event.ListDataListeners that have been added to it, each time a change occurs. These changes are characterized by a javax.swing.event.ListDataEvent, which identifies the range of list indices that have been modified, added, or removed. JList's ListUI is responsible for keeping the visual representation up to date with changes, by listening to the model.

Simple, dynamic-content, JList applications can use the DefaultListModel class to maintain list elements. This class implements the ListModel interface and also provides a java.util.Vector-like API. Applications that need a more custom ListModel implementation may instead wish to subclass AbstractListModel, which provides basic support for managing and notifying listeners. For example, a read-only implementation of AbstractListModel:

// This list model has about 2^16 elements. Enjoy scrolling.  
  
 ListModel bigData = new AbstractListModel() {  
 public int getSize() { return Short.MAX\_VALUE; }  
 public Object getElementAt(int index) { return "Index " + index; }  
 };

The selection state of a JList is managed by another separate model, an instance of ListSelectionModel. JList is initialized with a selection model on construction, and also contains methods to query or set this selection model. Additionally, JList provides convenient methods for easily managing the selection. These methods, such as setSelectedIndex and getSelectedValue, are cover methods that take care of the details of interacting with the selection model. By default, JList's selection model is configured to allow any combination of items to be selected at a time; selection mode MULTIPLE\_INTERVAL\_SELECTION. The selection mode can be changed on the selection model directly, or via JList's cover method. Responsibility for updating the selection model in response to user gestures lies with the list's ListUI.

A correct ListSelectionModel implementation notifies the set of javax.swing.event.ListSelectionListeners that have been added to it each time a change to the selection occurs. These changes are characterized by a javax.swing.event.ListSelectionEvent, which identifies the range of the selection change.

The preferred way to listen for changes in list selection is to add ListSelectionListeners directly to the JList. JList then takes care of listening to the the selection model and notifying your listeners of change.

Responsibility for listening to selection changes in order to keep the list's visual representation up to date lies with the list's ListUI.

Painting of cells in a JList is handled by a delegate called a cell renderer, installed on the list as the cellRenderer property. The renderer provides a java.awt.Component that is used like a "rubber stamp" to paint the cells. Each time a cell needs to be painted, the list's ListUI asks the cell renderer for the component, moves it into place, and has it paint the contents of the cell by way of its paint method. A default cell renderer, which uses a JLabel component to render, is installed by the lists's ListUI. You can substitute your own renderer using code like this:

// Display an icon and a string for each object in the list.  
  
 class MyCellRenderer extends JLabel implements ListCellRenderer {  
 final static ImageIcon longIcon = new ImageIcon("long.gif");  
 final static ImageIcon shortIcon = new ImageIcon("short.gif");  
  
 // This is the only method defined by ListCellRenderer.  
 // We just reconfigure the JLabel each time we're called.  
  
 public Component getListCellRendererComponent(  
 JList list, // the list  
 Object value, // value to display  
 int index, // cell index  
 boolean isSelected, // is the cell selected  
 boolean cellHasFocus) // does the cell have focus  
 {  
 String s = value.toString();  
 setText(s);  
 setIcon((s.length() > 10) ? longIcon : shortIcon);  
 if (isSelected) {  
 setBackground(list.getSelectionBackground());  
 setForeground(list.getSelectionForeground());  
 } else {  
 setBackground(list.getBackground());  
 setForeground(list.getForeground());  
 }  
 setEnabled(list.isEnabled());  
 setFont(list.getFont());  
 setOpaque(true);  
 return this;  
 }  
 }  
  
 myList.setCellRenderer(new MyCellRenderer());

Another job for the cell renderer is in helping to determine sizing information for the list. By default, the list's ListUI determines the size of cells by asking the cell renderer for its preferred size for each list item. This can be expensive for large lists of items. To avoid these calculations, you can set a fixedCellWidth and fixedCellHeight on the list, or have these values calculated automatically based on a single prototype value:

JList bigDataList = new JList(bigData);  
  
 // We don't want the JList implementation to compute the width  
 // or height of all of the list cells, so we give it a string  
 // that's as big as we'll need for any cell. It uses this to  
 // compute values for the fixedCellWidth and fixedCellHeight  
 // properties.  
  
 bigDataList.setPrototypeCellValue("Index 1234567890");

JList doesn't implement scrolling directly. To create a list that scrolls, make it the viewport view of a JScrollPane. For example:

JScrollPane scrollPane = new JScrollPane(myList);  
  
 // Or in two steps:  
 JScrollPane scrollPane = new JScrollPane();  
 scrollPane.getViewport().setView(myList);

JList doesn't provide any special handling of double or triple (or N) mouse clicks, but it's easy to add a MouseListener if you wish to take action on these events. Use the locationToIndex method to determine what cell was clicked. For example:

MouseListener mouseListener = new MouseAdapter() {  
 public void mouseClicked(MouseEvent e) {  
 if (e.getClickCount() == 2) {  
 int index = list.locationToIndex(e.getPoint());  
 System.out.println("Double clicked on Item " + index);  
 }  
 }  
 };  
 list.addMouseListener(mouseListener);

**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 [How to Use Lists](http://java.sun.com/docs/books/tutorial/uiswing/components/list.html) in [*The Java Tutorial*](http://java.sun.com/Series/Tutorial/index.html) for further documentation. Also see the article [Advanced JList Programming](http://java.sun.com/products/jfc/tsc/tech_topics/jlist_1/jlist.html) in [*The Swing Connection*](http://java.sun.com/products/jfc/tsc).

**See Also:**[ListModel](http://docs.google.com/javax/swing/ListModel.html), [AbstractListModel](http://docs.google.com/javax/swing/AbstractListModel.html), [DefaultListModel](http://docs.google.com/javax/swing/DefaultListModel.html), [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html), [DefaultListSelectionModel](http://docs.google.com/javax/swing/DefaultListSelectionModel.html), [ListCellRenderer](http://docs.google.com/javax/swing/ListCellRenderer.html), [DefaultListCellRenderer](http://docs.google.com/javax/swing/DefaultListCellRenderer.html)

| **Nested Class Summary** | |
| --- | --- |
| protected  class | [**JList.AccessibleJList**](http://docs.google.com/javax/swing/JList.AccessibleJList.html)            This class implements accessibility support for the JList class. |
| static class | [**JList.DropLocation**](http://docs.google.com/javax/swing/JList.DropLocation.html)            A subclass of TransferHandler.DropLocation representing a drop location for a JList. |

| **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 | [**HORIZONTAL\_WRAP**](http://docs.google.com/javax/swing/JList.html#HORIZONTAL_WRAP)            Indicates a "newspaper style" layout with cells flowing horizontally then vertically. |
| static int | [**VERTICAL**](http://docs.google.com/javax/swing/JList.html#VERTICAL)            Indicates a vertical layout of cells, in a single column; the default layout. |
| static int | [**VERTICAL\_WRAP**](http://docs.google.com/javax/swing/JList.html#VERTICAL_WRAP)            Indicates a "newspaper style" layout with cells flowing vertically then horizontally. |

| **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** | |
| --- | --- |
| [**JList**](http://docs.google.com/javax/swing/JList.html#JList())()            Constructs a JList with an empty, read-only, model. |
| [**JList**](http://docs.google.com/javax/swing/JList.html#JList(javax.swing.ListModel))([ListModel](http://docs.google.com/javax/swing/ListModel.html) dataModel)            Constructs a JList that displays elements from the specified, non-null, model. |
| [**JList**](http://docs.google.com/javax/swing/JList.html#JList(java.lang.Object%5B%5D))([Object](http://docs.google.com/java/lang/Object.html)[] listData)            Constructs a JList that displays the elements in the specified array. |
| [**JList**](http://docs.google.com/javax/swing/JList.html#JList(java.util.Vector))([Vector](http://docs.google.com/java/util/Vector.html)<?> listData)            Constructs a JList that displays the elements in the specified Vector. |

| **Method Summary** | |
| --- | --- |
| void | [**addListSelectionListener**](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))([ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html) listener)            Adds a listener to the list, to be notified each time a change to the selection occurs; the preferred way of listening for selection state changes. |
| void | [**addSelectionInterval**](http://docs.google.com/javax/swing/JList.html#addSelectionInterval(int,%20int))(int anchor, int lead)            Sets the selection to be the union of the specified interval with current selection. |
| void | [**clearSelection**](http://docs.google.com/javax/swing/JList.html#clearSelection())()            Clears the selection; after calling this method, isSelectionEmpty will return true. |
| protected  [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) | [**createSelectionModel**](http://docs.google.com/javax/swing/JList.html#createSelectionModel())()            Returns an instance of DefaultListSelectionModel; called during construction to initialize the list's selection model property. |
| void | [**ensureIndexIsVisible**](http://docs.google.com/javax/swing/JList.html#ensureIndexIsVisible(int))(int index)            Scrolls the list within an enclosing viewport to make the specified cell completely visible. |
| protected  void | [**fireSelectionValueChanged**](http://docs.google.com/javax/swing/JList.html#fireSelectionValueChanged(int,%20int,%20boolean))(int firstIndex, int lastIndex, boolean isAdjusting)            Notifies ListSelectionListeners added directly to the list of selection changes made to the selection model. |
| [AccessibleContext](http://docs.google.com/javax/accessibility/AccessibleContext.html) | [**getAccessibleContext**](http://docs.google.com/javax/swing/JList.html#getAccessibleContext())()            Gets the AccessibleContext associated with this JList. |
| int | [**getAnchorSelectionIndex**](http://docs.google.com/javax/swing/JList.html#getAnchorSelectionIndex())()            Returns the anchor selection index. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getCellBounds**](http://docs.google.com/javax/swing/JList.html#getCellBounds(int,%20int))(int index0, int index1)            Returns the bounding rectangle, in the list's coordinate system, for the range of cells specified by the two indices. |
| [ListCellRenderer](http://docs.google.com/javax/swing/ListCellRenderer.html) | [**getCellRenderer**](http://docs.google.com/javax/swing/JList.html#getCellRenderer())()            Returns the object responsible for painting list items. |
| boolean | [**getDragEnabled**](http://docs.google.com/javax/swing/JList.html#getDragEnabled())()            Returns whether or not automatic drag handling is enabled. |
| [JList.DropLocation](http://docs.google.com/javax/swing/JList.DropLocation.html) | [**getDropLocation**](http://docs.google.com/javax/swing/JList.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/JList.html#getDropMode())()            Returns the drop mode for this component. |
| int | [**getFirstVisibleIndex**](http://docs.google.com/javax/swing/JList.html#getFirstVisibleIndex())()            Returns the smallest list index that is currently visible. |
| int | [**getFixedCellHeight**](http://docs.google.com/javax/swing/JList.html#getFixedCellHeight())()            Returns the value of the fixedCellHeight property. |
| int | [**getFixedCellWidth**](http://docs.google.com/javax/swing/JList.html#getFixedCellWidth())()            Returns the value of the fixedCellWidth property. |
| int | [**getLastVisibleIndex**](http://docs.google.com/javax/swing/JList.html#getLastVisibleIndex())()            Returns the largest list index that is currently visible. |
| int | [**getLayoutOrientation**](http://docs.google.com/javax/swing/JList.html#getLayoutOrientation())()            Returns the layout orientation property for the list: VERTICAL if the layout is a single column of cells, VERTICAL\_WRAP if the layout is "newspaper style" with the content flowing vertically then horizontally, or HORIZONTAL\_WRAP if the layout is "newspaper style" with the content flowing horizontally then vertically. |
| int | [**getLeadSelectionIndex**](http://docs.google.com/javax/swing/JList.html#getLeadSelectionIndex())()            Returns the lead selection index. |
| [ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html)[] | [**getListSelectionListeners**](http://docs.google.com/javax/swing/JList.html#getListSelectionListeners())()            Returns an array of all the ListSelectionListeners added to this JList by way of addListSelectionListener. |
| int | [**getMaxSelectionIndex**](http://docs.google.com/javax/swing/JList.html#getMaxSelectionIndex())()            Returns the largest selected cell index, or -1 if the selection is empty. |
| int | [**getMinSelectionIndex**](http://docs.google.com/javax/swing/JList.html#getMinSelectionIndex())()            Returns the smallest selected cell index, or -1 if the selection is empty. |
| [ListModel](http://docs.google.com/javax/swing/ListModel.html) | [**getModel**](http://docs.google.com/javax/swing/JList.html#getModel())()            Returns the data model that holds the list of items displayed by the JList component. |
| int | [**getNextMatch**](http://docs.google.com/javax/swing/JList.html#getNextMatch(java.lang.String,%20int,%20javax.swing.text.Position.Bias))([String](http://docs.google.com/java/lang/String.html) prefix, int startIndex, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) bias)            Returns the next list element whose toString value starts with the given prefix. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredScrollableViewportSize**](http://docs.google.com/javax/swing/JList.html#getPreferredScrollableViewportSize())()            Computes the size of viewport needed to display visibleRowCount rows. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getPrototypeCellValue**](http://docs.google.com/javax/swing/JList.html#getPrototypeCellValue())()            Returns the "prototypical" cell value -- a value used to calculate a fixed width and height for cells. |
| int | [**getScrollableBlockIncrement**](http://docs.google.com/javax/swing/JList.html#getScrollableBlockIncrement(java.awt.Rectangle,%20int,%20int))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect, int orientation, int direction)            Returns the distance to scroll to expose the next or previous block. |
| boolean | [**getScrollableTracksViewportHeight**](http://docs.google.com/javax/swing/JList.html#getScrollableTracksViewportHeight())()            Returns true if this JList is displayed in a JViewport and the viewport is taller than the list's preferred height, or if the layout orientation is VERTICAL\_WRAP and visibleRowCount <= 0; otherwise returns false. |
| boolean | [**getScrollableTracksViewportWidth**](http://docs.google.com/javax/swing/JList.html#getScrollableTracksViewportWidth())()            Returns true if this JList is displayed in a JViewport and the viewport is wider than the list's preferred width, or if the layout orientation is HORIZONTAL\_WRAP and visibleRowCount <= 0; otherwise returns false. |
| int | [**getScrollableUnitIncrement**](http://docs.google.com/javax/swing/JList.html#getScrollableUnitIncrement(java.awt.Rectangle,%20int,%20int))([Rectangle](http://docs.google.com/java/awt/Rectangle.html) visibleRect, int orientation, int direction)            Returns the distance to scroll to expose the next or previous row (for vertical scrolling) or column (for horizontal scrolling). |
| int | [**getSelectedIndex**](http://docs.google.com/javax/swing/JList.html#getSelectedIndex())()            Returns the smallest selected cell index; *the selection* when only a single item is selected in the list. |
| int[] | [**getSelectedIndices**](http://docs.google.com/javax/swing/JList.html#getSelectedIndices())()            Returns an array of all of the selected indices, in increasing order. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getSelectedValue**](http://docs.google.com/javax/swing/JList.html#getSelectedValue())()            Returns the value for the smallest selected cell index; *the selected value* when only a single item is selected in the list. |
| [Object](http://docs.google.com/java/lang/Object.html)[] | [**getSelectedValues**](http://docs.google.com/javax/swing/JList.html#getSelectedValues())()            Returns an array of all the selected values, in increasing order based on their indices in the list. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectionBackground**](http://docs.google.com/javax/swing/JList.html#getSelectionBackground())()            Returns the color used to draw the background of selected items. |
| [Color](http://docs.google.com/java/awt/Color.html) | [**getSelectionForeground**](http://docs.google.com/javax/swing/JList.html#getSelectionForeground())()            Returns the color used to draw the foreground of selected items. |
| int | [**getSelectionMode**](http://docs.google.com/javax/swing/JList.html#getSelectionMode())()            Returns the current selection mode for the list. |
| [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) | [**getSelectionModel**](http://docs.google.com/javax/swing/JList.html#getSelectionModel())()            Returns the current selection model. |
| [String](http://docs.google.com/java/lang/String.html) | [**getToolTipText**](http://docs.google.com/javax/swing/JList.html#getToolTipText(java.awt.event.MouseEvent))([MouseEvent](http://docs.google.com/java/awt/event/MouseEvent.html) event)            Returns the tooltip text to be used for the given event. |
| [ListUI](http://docs.google.com/javax/swing/plaf/ListUI.html) | [**getUI**](http://docs.google.com/javax/swing/JList.html#getUI())()            Returns the ListUI, the look and feel object that renders this component. |
| [String](http://docs.google.com/java/lang/String.html) | [**getUIClassID**](http://docs.google.com/javax/swing/JList.html#getUIClassID())()            Returns "ListUI", the UIDefaults key used to look up the name of the javax.swing.plaf.ListUI class that defines the look and feel for this component. |
| boolean | [**getValueIsAdjusting**](http://docs.google.com/javax/swing/JList.html#getValueIsAdjusting())()            Returns the value of the selection model's isAdjusting property. |
| int | [**getVisibleRowCount**](http://docs.google.com/javax/swing/JList.html#getVisibleRowCount())()            Returns the value of the visibleRowCount property. |
| [Point](http://docs.google.com/java/awt/Point.html) | [**indexToLocation**](http://docs.google.com/javax/swing/JList.html#indexToLocation(int))(int index)            Returns the origin of the specified item in the list's coordinate system. |
| boolean | [**isSelectedIndex**](http://docs.google.com/javax/swing/JList.html#isSelectedIndex(int))(int index)            Returns true if the specified index is selected, else false. |
| boolean | [**isSelectionEmpty**](http://docs.google.com/javax/swing/JList.html#isSelectionEmpty())()            Returns true if nothing is selected, else false. |
| int | [**locationToIndex**](http://docs.google.com/javax/swing/JList.html#locationToIndex(java.awt.Point))([Point](http://docs.google.com/java/awt/Point.html) location)            Returns the cell index closest to the given location in the list's coordinate system. |
| protected  [String](http://docs.google.com/java/lang/String.html) | [**paramString**](http://docs.google.com/javax/swing/JList.html#paramString())()            Returns a String representation of this JList. |
| void | [**removeListSelectionListener**](http://docs.google.com/javax/swing/JList.html#removeListSelectionListener(javax.swing.event.ListSelectionListener))([ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html) listener)            Removes a selection listener from the list. |
| void | [**removeSelectionInterval**](http://docs.google.com/javax/swing/JList.html#removeSelectionInterval(int,%20int))(int index0, int index1)            Sets the selection to be the set difference of the specified interval and the current selection. |
| void | [**setCellRenderer**](http://docs.google.com/javax/swing/JList.html#setCellRenderer(javax.swing.ListCellRenderer))([ListCellRenderer](http://docs.google.com/javax/swing/ListCellRenderer.html) cellRenderer)            Sets the delegate that is used to paint each cell in the list. |
| void | [**setDragEnabled**](http://docs.google.com/javax/swing/JList.html#setDragEnabled(boolean))(boolean b)            Turns on or off automatic drag handling. |
| void | [**setDropMode**](http://docs.google.com/javax/swing/JList.html#setDropMode(javax.swing.DropMode))([DropMode](http://docs.google.com/javax/swing/DropMode.html) dropMode)            Sets the drop mode for this component. |
| void | [**setFixedCellHeight**](http://docs.google.com/javax/swing/JList.html#setFixedCellHeight(int))(int height)            Sets a fixed value to be used for the height of every cell in the list. |
| void | [**setFixedCellWidth**](http://docs.google.com/javax/swing/JList.html#setFixedCellWidth(int))(int width)            Sets a fixed value to be used for the width of every cell in the list. |
| void | [**setLayoutOrientation**](http://docs.google.com/javax/swing/JList.html#setLayoutOrientation(int))(int layoutOrientation)            Defines the way list cells are layed out. |
| void | [**setListData**](http://docs.google.com/javax/swing/JList.html#setListData(java.lang.Object%5B%5D))([Object](http://docs.google.com/java/lang/Object.html)[] listData)            Constructs a read-only ListModel from an array of objects, and calls setModel with this model. |
| void | [**setListData**](http://docs.google.com/javax/swing/JList.html#setListData(java.util.Vector))([Vector](http://docs.google.com/java/util/Vector.html)<?> listData)            Constructs a read-only ListModel from a Vector and calls setModel with this model. |
| void | [**setModel**](http://docs.google.com/javax/swing/JList.html#setModel(javax.swing.ListModel))([ListModel](http://docs.google.com/javax/swing/ListModel.html) model)            Sets the model that represents the contents or "value" of the list, notifies property change listeners, and then clears the list's selection. |
| void | [**setPrototypeCellValue**](http://docs.google.com/javax/swing/JList.html#setPrototypeCellValue(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) prototypeCellValue)            Sets the prototypeCellValue property, and then (if the new value is non-null), computes the fixedCellWidth and fixedCellHeight properties by requesting the cell renderer component for the given value (and index 0) from the cell renderer, and using that component's preferred size. |
| void | [**setSelectedIndex**](http://docs.google.com/javax/swing/JList.html#setSelectedIndex(int))(int index)            Selects a single cell. |
| void | [**setSelectedIndices**](http://docs.google.com/javax/swing/JList.html#setSelectedIndices(int%5B%5D))(int[] indices)            Changes the selection to be the set of indices specified by the given array. |
| void | [**setSelectedValue**](http://docs.google.com/javax/swing/JList.html#setSelectedValue(java.lang.Object,%20boolean))([Object](http://docs.google.com/java/lang/Object.html) anObject, boolean shouldScroll)            Selects the specified object from the list. |
| void | [**setSelectionBackground**](http://docs.google.com/javax/swing/JList.html#setSelectionBackground(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) selectionBackground)            Sets the color used to draw the background of selected items, which cell renderers can use fill selected cells. |
| void | [**setSelectionForeground**](http://docs.google.com/javax/swing/JList.html#setSelectionForeground(java.awt.Color))([Color](http://docs.google.com/java/awt/Color.html) selectionForeground)            Sets the color used to draw the foreground of selected items, which cell renderers can use to render text and graphics. |
| void | [**setSelectionInterval**](http://docs.google.com/javax/swing/JList.html#setSelectionInterval(int,%20int))(int anchor, int lead)            Selects the specified interval. |
| void | [**setSelectionMode**](http://docs.google.com/javax/swing/JList.html#setSelectionMode(int))(int selectionMode)            Sets the selection mode for the list. |
| void | [**setSelectionModel**](http://docs.google.com/javax/swing/JList.html#setSelectionModel(javax.swing.ListSelectionModel))([ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html) selectionModel)            Sets the selectionModel for the list to a non-null ListSelectionModel implementation. |
| void | [**setUI**](http://docs.google.com/javax/swing/JList.html#setUI(javax.swing.plaf.ListUI))([ListUI](http://docs.google.com/javax/swing/plaf/ListUI.html) ui)            Sets the ListUI, the look and feel object that renders this component. |
| void | [**setValueIsAdjusting**](http://docs.google.com/javax/swing/JList.html#setValueIsAdjusting(boolean))(boolean b)            Sets the selection model's valueIsAdjusting property. |
| void | [**setVisibleRowCount**](http://docs.google.com/javax/swing/JList.html#setVisibleRowCount(int))(int visibleRowCount)            Sets the visibleRowCount property, which has different meanings depending on the layout orientation: For a VERTICAL layout orientation, this sets the preferred number of rows to display without requiring scrolling; for other orientations, it affects the wrapping of cells. |
| void | [**updateUI**](http://docs.google.com/javax/swing/JList.html#updateUI())()            Resets the ListUI property by setting it to the value provided by the current look and feel. |

| **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)), [addNotify](http://docs.google.com/javax/swing/JComponent.html#addNotify()), [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)), [processKeyBinding](http://docs.google.com/javax/swing/JComponent.html#processKeyBinding(javax.swing.KeyStroke,%20java.awt.event.KeyEvent,%20int,%20boolean)), [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)), [removeNotify](http://docs.google.com/javax/swing/JComponent.html#removeNotify()), [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)), [doLayout](http://docs.google.com/java/awt/Container.html#doLayout()), [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** |
| --- |

### VERTICAL

public static final int **VERTICAL**

Indicates a vertical layout of cells, in a single column; the default layout.

**Since:** 1.4 **See Also:**[setLayoutOrientation(int)](http://docs.google.com/javax/swing/JList.html#setLayoutOrientation(int)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JList.VERTICAL)

### VERTICAL\_WRAP

public static final int **VERTICAL\_WRAP**

Indicates a "newspaper style" layout with cells flowing vertically then horizontally.

**Since:** 1.4 **See Also:**[setLayoutOrientation(int)](http://docs.google.com/javax/swing/JList.html#setLayoutOrientation(int)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JList.VERTICAL_WRAP)

### HORIZONTAL\_WRAP

public static final int **HORIZONTAL\_WRAP**

Indicates a "newspaper style" layout with cells flowing horizontally then vertically.

**Since:** 1.4 **See Also:**[setLayoutOrientation(int)](http://docs.google.com/javax/swing/JList.html#setLayoutOrientation(int)), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.JList.HORIZONTAL_WRAP)

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

### JList

public **JList**([ListModel](http://docs.google.com/javax/swing/ListModel.html) dataModel)

Constructs a JList that displays elements from the specified, non-null, model. All JList constructors delegate to this one.

This constructor registers the list with the ToolTipManager, allowing for tooltips to be provided by the cell renderers.

**Parameters:**dataModel - the model for the list **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the model is null

### JList

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

Constructs a JList that displays the elements in the specified array. This constructor creates a read-only model for the given array, and then delegates to the constructor that takes a ListModel.

Attempts to pass a null value to this method results in undefined behavior and, most likely, exceptions. The created model references the given array directly. Attempts to modify the array after constructing the list results in undefined behavior.

**Parameters:**listData - the array of Objects to be loaded into the data model, non-null

### JList

public **JList**([Vector](http://docs.google.com/java/util/Vector.html)<?> listData)

Constructs a JList that displays the elements in the specified Vector. This constructor creates a read-only model for the given Vector, and then delegates to the constructor that takes a ListModel.

Attempts to pass a null value to this method results in undefined behavior and, most likely, exceptions. The created model references the given Vector directly. Attempts to modify the Vector after constructing the list results in undefined behavior.

**Parameters:**listData - the Vector to be loaded into the data model, non-null

### JList

public **JList**()

Constructs a JList with an empty, read-only, model.

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

### getUI

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

Returns the ListUI, the look and feel object that renders this component.

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

### setUI

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

Sets the ListUI, the look and feel object that renders this component.

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

### updateUI

public void **updateUI**()

Resets the ListUI property by setting it to the value provided by the current look and feel. If the current cell renderer was installed by the developer (rather than the look and feel itself), this also causes the cell renderer and its children to be updated, by calling SwingUtilities.updateComponentTreeUI on it.

**Overrides:**[updateUI](http://docs.google.com/javax/swing/JComponent.html#updateUI()) in class [JComponent](http://docs.google.com/javax/swing/JComponent.html) **See Also:**[UIManager.getUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/UIManager.html#getUI(javax.swing.JComponent)), [SwingUtilities.updateComponentTreeUI(java.awt.Component)](http://docs.google.com/javax/swing/SwingUtilities.html#updateComponentTreeUI(java.awt.Component))

### getUIClassID

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

Returns "ListUI", the UIDefaults key used to look up the name of the javax.swing.plaf.ListUI class that defines the look and feel for 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 "ListUI"**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))

### getPrototypeCellValue

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

Returns the "prototypical" cell value -- a value used to calculate a fixed width and height for cells. This can be null if there is no such value.

**Returns:**the value of the prototypeCellValue property**See Also:**[setPrototypeCellValue(java.lang.Object)](http://docs.google.com/javax/swing/JList.html#setPrototypeCellValue(java.lang.Object))

### setPrototypeCellValue

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

Sets the prototypeCellValue property, and then (if the new value is non-null), computes the fixedCellWidth and fixedCellHeight properties by requesting the cell renderer component for the given value (and index 0) from the cell renderer, and using that component's preferred size.

This method is useful when the list is too long to allow the ListUI to compute the width/height of each cell, and there is a single cell value that is known to occupy as much space as any of the others, a so-called prototype.

While all three of the prototypeCellValue, fixedCellHeight, and fixedCellWidth properties may be modified by this method, PropertyChangeEvent notifications are only sent when the prototypeCellValue property changes.

To see an example which sets this property, see the [class description](#2et92p0) above.

The default value of this property is null.

This is a JavaBeans bound property.

**Parameters:**prototypeCellValue - the value on which to base fixedCellWidth and fixedCellHeight**See Also:**[getPrototypeCellValue()](http://docs.google.com/javax/swing/JList.html#getPrototypeCellValue()), [setFixedCellWidth(int)](http://docs.google.com/javax/swing/JList.html#setFixedCellWidth(int)), [setFixedCellHeight(int)](http://docs.google.com/javax/swing/JList.html#setFixedCellHeight(int)), [Container.addPropertyChangeListener(java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.beans.PropertyChangeListener))

### getFixedCellWidth

public int **getFixedCellWidth**()

Returns the value of the fixedCellWidth property.

**Returns:**the fixed cell width**See Also:**[setFixedCellWidth(int)](http://docs.google.com/javax/swing/JList.html#setFixedCellWidth(int))

### setFixedCellWidth

public void **setFixedCellWidth**(int width)

Sets a fixed value to be used for the width of every cell in the list. If width is -1, cell widths are computed in the ListUI by applying getPreferredSize to the cell renderer component for each list element.

The default value of this property is -1.

This is a JavaBeans bound property.

**Parameters:**width - the width to be used for all cells in the list**See Also:**[setPrototypeCellValue(java.lang.Object)](http://docs.google.com/javax/swing/JList.html#setPrototypeCellValue(java.lang.Object)), [setFixedCellWidth(int)](http://docs.google.com/javax/swing/JList.html#setFixedCellWidth(int)), [Container.addPropertyChangeListener(java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.beans.PropertyChangeListener))

### getFixedCellHeight

public int **getFixedCellHeight**()

Returns the value of the fixedCellHeight property.

**Returns:**the fixed cell height**See Also:**[setFixedCellHeight(int)](http://docs.google.com/javax/swing/JList.html#setFixedCellHeight(int))

### setFixedCellHeight

public void **setFixedCellHeight**(int height)

Sets a fixed value to be used for the height of every cell in the list. If height is -1, cell heights are computed in the ListUI by applying getPreferredSize to the cell renderer component for each list element.

The default value of this property is -1.

This is a JavaBeans bound property.

**Parameters:**height - the height to be used for for all cells in the list**See Also:**[setPrototypeCellValue(java.lang.Object)](http://docs.google.com/javax/swing/JList.html#setPrototypeCellValue(java.lang.Object)), [setFixedCellWidth(int)](http://docs.google.com/javax/swing/JList.html#setFixedCellWidth(int)), [Container.addPropertyChangeListener(java.beans.PropertyChangeListener)](http://docs.google.com/java/awt/Container.html#addPropertyChangeListener(java.beans.PropertyChangeListener))

### getCellRenderer

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

Returns the object responsible for painting list items.

**Returns:**the value of the cellRenderer property**See Also:**[setCellRenderer(javax.swing.ListCellRenderer)](http://docs.google.com/javax/swing/JList.html#setCellRenderer(javax.swing.ListCellRenderer))

### setCellRenderer

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

Sets the delegate that is used to paint each cell in the list. The job of a cell renderer is discussed in detail in the [class level documentation](#3znysh7).

If the prototypeCellValue property is non-null, setting the cell renderer also causes the fixedCellWidth and fixedCellHeight properties to be re-calculated. Only one PropertyChangeEvent is generated however - for the cellRenderer property.

The default value of this property is provided by the ListUI delegate, i.e. by the look and feel implementation.

This is a JavaBeans bound property.

**Parameters:**cellRenderer - the ListCellRenderer that paints list cells**See Also:**[getCellRenderer()](http://docs.google.com/javax/swing/JList.html#getCellRenderer())

### getSelectionForeground

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

Returns the color used to draw the foreground of selected items. DefaultListCellRenderer uses this color to draw the foreground of items in the selected state, as do the renderers installed by most ListUI implementations.

**Returns:**the color to draw the foreground of selected items**See Also:**[setSelectionForeground(java.awt.Color)](http://docs.google.com/javax/swing/JList.html#setSelectionForeground(java.awt.Color)), [DefaultListCellRenderer](http://docs.google.com/javax/swing/DefaultListCellRenderer.html)

### setSelectionForeground

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

Sets the color used to draw the foreground of selected items, which cell renderers can use to render text and graphics. DefaultListCellRenderer uses this color to draw the foreground of items in the selected state, as do the renderers installed by most ListUI implementations.

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

This is a JavaBeans bound property.

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

### getSelectionBackground

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

Returns the color used to draw the background of selected items. DefaultListCellRenderer uses this color to draw the background of items in the selected state, as do the renderers installed by most ListUI implementations.

**Returns:**the color to draw the background of selected items**See Also:**[setSelectionBackground(java.awt.Color)](http://docs.google.com/javax/swing/JList.html#setSelectionBackground(java.awt.Color)), [DefaultListCellRenderer](http://docs.google.com/javax/swing/DefaultListCellRenderer.html)

### setSelectionBackground

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

Sets the color used to draw the background of selected items, which cell renderers can use fill selected cells. DefaultListCellRenderer uses this color to fill the background of items in the selected state, as do the renderers installed by most ListUI implementations.

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

This is a JavaBeans bound property.

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

### getVisibleRowCount

public int **getVisibleRowCount**()

Returns the value of the visibleRowCount property. See the documentation for [setVisibleRowCount(int)](http://docs.google.com/javax/swing/JList.html#setVisibleRowCount(int)) for details on how to interpret this value.

**Returns:**the value of the visibleRowCount property.**See Also:**[setVisibleRowCount(int)](http://docs.google.com/javax/swing/JList.html#setVisibleRowCount(int))

### setVisibleRowCount

public void **setVisibleRowCount**(int visibleRowCount)

Sets the visibleRowCount property, which has different meanings depending on the layout orientation: For a VERTICAL layout orientation, this sets the preferred number of rows to display without requiring scrolling; for other orientations, it affects the wrapping of cells.

In VERTICAL orientation:

Setting this property affects the return value of the [getPreferredScrollableViewportSize()](http://docs.google.com/javax/swing/JList.html#getPreferredScrollableViewportSize()) method, which is used to calculate the preferred size of an enclosing viewport. See that method's documentation for more details.

In HORIZONTAL\_WRAP and VERTICAL\_WRAP orientations:

This affects how cells are wrapped. See the documentation of [setLayoutOrientation(int)](http://docs.google.com/javax/swing/JList.html#setLayoutOrientation(int)) for more details.

The default value of this property is 8.

Calling this method with a negative value results in the property being set to 0.

This is a JavaBeans bound property.

**Parameters:**visibleRowCount - an integer specifying the preferred number of rows to display without requiring scrolling**See Also:**[getVisibleRowCount()](http://docs.google.com/javax/swing/JList.html#getVisibleRowCount()), [getPreferredScrollableViewportSize()](http://docs.google.com/javax/swing/JList.html#getPreferredScrollableViewportSize()), [setLayoutOrientation(int)](http://docs.google.com/javax/swing/JList.html#setLayoutOrientation(int)), [JComponent.getVisibleRect()](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect()), [JViewport](http://docs.google.com/javax/swing/JViewport.html)

### getLayoutOrientation

public int **getLayoutOrientation**()

Returns the layout orientation property for the list: VERTICAL if the layout is a single column of cells, VERTICAL\_WRAP if the layout is "newspaper style" with the content flowing vertically then horizontally, or HORIZONTAL\_WRAP if the layout is "newspaper style" with the content flowing horizontally then vertically.

**Returns:**the value of the layoutOrientation property**Since:** 1.4 **See Also:**[setLayoutOrientation(int)](http://docs.google.com/javax/swing/JList.html#setLayoutOrientation(int))

### setLayoutOrientation

public void **setLayoutOrientation**(int layoutOrientation)

Defines the way list cells are layed out. Consider a JList with five cells. Cells can be layed out in one of the following ways:

VERTICAL: 0  
 1  
 2  
 3  
 4  
  
 HORIZONTAL\_WRAP: 0 1 2  
 3 4  
  
 VERTICAL\_WRAP: 0 3  
 1 4  
 2

A description of these layouts follows:

| Value | Description |
| --- | --- |
| VERTICAL | Cells are layed out vertically in a single column. |
| HORIZONTAL\_WRAP | Cells are layed out horizontally, wrapping to a new row as necessary. If the visibleRowCount property is less than or equal to zero, wrapping is determined by the width of the list; otherwise wrapping is done in such a way as to ensure visibleRowCount rows in the list. |
| VERTICAL\_WRAP | Cells are layed out vertically, wrapping to a new column as necessary. If the visibleRowCount property is less than or equal to zero, wrapping is determined by the height of the list; otherwise wrapping is done at visibleRowCount rows. |

The default value of this property is VERTICAL.

**Parameters:**layoutOrientation - the new layout orientation, one of: VERTICAL, HORIZONTAL\_WRAP or VERTICAL\_WRAP **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if layoutOrientation isn't one of the allowable values**Since:** 1.4 **See Also:**[getLayoutOrientation()](http://docs.google.com/javax/swing/JList.html#getLayoutOrientation()), [setVisibleRowCount(int)](http://docs.google.com/javax/swing/JList.html#setVisibleRowCount(int)), [getScrollableTracksViewportHeight()](http://docs.google.com/javax/swing/JList.html#getScrollableTracksViewportHeight()), [getScrollableTracksViewportWidth()](http://docs.google.com/javax/swing/JList.html#getScrollableTracksViewportWidth())

### getFirstVisibleIndex

public int **getFirstVisibleIndex**()

Returns the smallest list index that is currently visible. In a left-to-right componentOrientation, the first visible cell is found closest to the list's upper-left corner. In right-to-left orientation, it is found closest to the upper-right corner. If nothing is visible or the list is empty, -1 is returned. Note that the returned cell may only be partially visible.

**Returns:**the index of the first visible cell**See Also:**[getLastVisibleIndex()](http://docs.google.com/javax/swing/JList.html#getLastVisibleIndex()), [JComponent.getVisibleRect()](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect())

### getLastVisibleIndex

public int **getLastVisibleIndex**()

Returns the largest list index that is currently visible. If nothing is visible or the list is empty, -1 is returned. Note that the returned cell may only be partially visible.

**Returns:**the index of the last visible cell**See Also:**[getFirstVisibleIndex()](http://docs.google.com/javax/swing/JList.html#getFirstVisibleIndex()), [JComponent.getVisibleRect()](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect())

### ensureIndexIsVisible

public void **ensureIndexIsVisible**(int index)

Scrolls the list within an enclosing viewport to make the specified cell completely visible. This calls scrollRectToVisible with the bounds of the specified cell. For this method to work, the JList must be within a JViewport.

If the given index is outside the list's range of cells, this method results in nothing.

**Parameters:**index - the index of the cell to make visible**See Also:**[JComponent.scrollRectToVisible(java.awt.Rectangle)](http://docs.google.com/javax/swing/JComponent.html#scrollRectToVisible(java.awt.Rectangle)), [JComponent.getVisibleRect()](http://docs.google.com/javax/swing/JComponent.html#getVisibleRect())

### 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 list'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 list's ListUI. 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 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 list'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/JList.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/JList.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 list.

JList supports the following drop modes:

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

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/JList.html#getDropMode()), [getDropLocation()](http://docs.google.com/javax/swing/JList.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/JList.html#setDropMode(javax.swing.DropMode))

### getDropLocation

public final [JList.DropLocation](http://docs.google.com/javax/swing/JList.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.

By default, responsibility for listening for changes to this property and indicating the drop location visually lies with the list's ListUI, which may paint it directly and/or install a cell renderer to do so. Developers wishing to implement custom drop location painting and/or replace the default cell renderer, may need to honor this property.

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

### getNextMatch

public int **getNextMatch**([String](http://docs.google.com/java/lang/String.html) prefix,  
 int startIndex,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) bias)

Returns the next list element whose toString value starts with the given prefix.

**Parameters:**prefix - the string to test for a matchstartIndex - the index for starting the searchbias - the search direction, either Position.Bias.Forward or Position.Bias.Backward. **Returns:**the index of the next list element that starts with the prefix; otherwise -1 **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if prefix is null or startIndex is out of bounds**Since:** 1.4

### getToolTipText

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

Returns the tooltip text to be used for the given event. This overrides JComponent's getToolTipText to first check the cell renderer component for the cell over which the event occurred, returning its tooltip text, if any. This implementation allows you to specify tooltip text on the cell level, by using setToolTipText on your cell renderer component.

Note: For JList to properly display the tooltips of its renderers in this manner, JList must be a registered component with the ToolTipManager. This registration is done automatically in the constructor. However, if at a later point JList is unregistered, by way of a call to setToolTipText(null), tips from the renderers will no longer display.

**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) **Parameters:**event - the MouseEvent to fetch the tooltip text for**See Also:**[JComponent.setToolTipText(java.lang.String)](http://docs.google.com/javax/swing/JComponent.html#setToolTipText(java.lang.String)), [JComponent.getToolTipText()](http://docs.google.com/javax/swing/JComponent.html#getToolTipText())

### locationToIndex

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

Returns the cell index closest to the given location in the list's coordinate system. To determine if the cell actually contains the specified location, compare the point against the cell's bounds, as provided by getCellBounds. This method returns -1 if the model is empty

This is a cover method that delegates to the method of the same name in the list's ListUI. It returns -1 if the list has no ListUI.

**Parameters:**location - the coordinates of the point **Returns:**the cell index closest to the given location, or -1

### indexToLocation

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

Returns the origin of the specified item in the list's coordinate system. This method returns null if the index isn't valid.

This is a cover method that delegates to the method of the same name in the list's ListUI. It returns null if the list has no ListUI.

**Parameters:**index - the cell index **Returns:**the origin of the cell, or null

### getCellBounds

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getCellBounds**(int index0,  
 int index1)

Returns the bounding rectangle, in the list's coordinate system, for the range of cells specified by the two indices. These indices can be supplied in any order.

If the smaller index is outside the list's range of cells, this method returns null. If the smaller index is valid, but the larger index is outside the list's range, the bounds of just the first index is returned. Otherwise, the bounds of the valid range is returned.

This is a cover method that delegates to the method of the same name in the list's ListUI. It returns null if the list has no ListUI.

**Parameters:**index0 - the first index in the rangeindex1 - the second index in the range **Returns:**the bounding rectangle for the range of cells, or null

### getModel

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

Returns the data model that holds the list of items displayed by the JList component.

**Returns:**the ListModel that provides the displayed list of items**See Also:**[setModel(javax.swing.ListModel)](http://docs.google.com/javax/swing/JList.html#setModel(javax.swing.ListModel))

### setModel

public void **setModel**([ListModel](http://docs.google.com/javax/swing/ListModel.html) model)

Sets the model that represents the contents or "value" of the list, notifies property change listeners, and then clears the list's selection.

This is a JavaBeans bound property.

**Parameters:**model - the ListModel that provides the list of items for display **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if model is null**See Also:**[getModel()](http://docs.google.com/javax/swing/JList.html#getModel()), [clearSelection()](http://docs.google.com/javax/swing/JList.html#clearSelection())

### setListData

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

Constructs a read-only ListModel from an array of objects, and calls setModel with this model.

Attempts to pass a null value to this method results in undefined behavior and, most likely, exceptions. The created model references the given array directly. Attempts to modify the array after invoking this method results in undefined behavior.

**Parameters:**listData - an array of Objects containing the items to display in the list**See Also:**[setModel(javax.swing.ListModel)](http://docs.google.com/javax/swing/JList.html#setModel(javax.swing.ListModel))

### setListData

public void **setListData**([Vector](http://docs.google.com/java/util/Vector.html)<?> listData)

Constructs a read-only ListModel from a Vector and calls setModel with this model.

Attempts to pass a null value to this method results in undefined behavior and, most likely, exceptions. The created model references the given Vector directly. Attempts to modify the Vector after invoking this method results in undefined behavior.

**Parameters:**listData - a Vector containing the items to display in the list**See Also:**[setModel(javax.swing.ListModel)](http://docs.google.com/javax/swing/JList.html#setModel(javax.swing.ListModel))

### createSelectionModel

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

Returns an instance of DefaultListSelectionModel; called during construction to initialize the list's selection model property.

**Returns:**a DefaultListSelecitonModel, used to initialize the list's selection model property during construction**See Also:**[setSelectionModel(javax.swing.ListSelectionModel)](http://docs.google.com/javax/swing/JList.html#setSelectionModel(javax.swing.ListSelectionModel)), [DefaultListSelectionModel](http://docs.google.com/javax/swing/DefaultListSelectionModel.html)

### getSelectionModel

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

Returns the current selection model. The selection model maintains the selection state of the list. See the class level documentation for more details.

**Returns:**the ListSelectionModel that maintains the list's selections**See Also:**[setSelectionModel(javax.swing.ListSelectionModel)](http://docs.google.com/javax/swing/JList.html#setSelectionModel(javax.swing.ListSelectionModel)), [ListSelectionModel](http://docs.google.com/javax/swing/ListSelectionModel.html)

### fireSelectionValueChanged

protected void **fireSelectionValueChanged**(int firstIndex,  
 int lastIndex,  
 boolean isAdjusting)

Notifies ListSelectionListeners added directly to the list of selection changes made to the selection model. JList listens for changes made to the selection in the selection model, and forwards notification to listeners added to the list directly, by calling this method.

This method constructs a ListSelectionEvent with this list as the source, and the specified arguments, and sends it to the registered ListSelectionListeners.

**Parameters:**firstIndex - the first index in the range, <= lastIndexlastIndex - the last index in the range, >= firstIndexisAdjusting - whether or not this is one in a series of multiple events, where changes are still being made**See Also:**[addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener)), [removeListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#removeListSelectionListener(javax.swing.event.ListSelectionListener)), [ListSelectionEvent](http://docs.google.com/javax/swing/event/ListSelectionEvent.html), [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html)

### addListSelectionListener

public void **addListSelectionListener**([ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html) listener)

Adds a listener to the list, to be notified each time a change to the selection occurs; the preferred way of listening for selection state changes. JList takes care of listening for selection state changes in the selection model, and notifies the given listener of each change. ListSelectionEvents sent to the listener have a source property set to this list.

**Parameters:**listener - the ListSelectionListener to add**See Also:**[getSelectionModel()](http://docs.google.com/javax/swing/JList.html#getSelectionModel()), [getListSelectionListeners()](http://docs.google.com/javax/swing/JList.html#getListSelectionListeners())

### removeListSelectionListener

public void **removeListSelectionListener**([ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html) listener)

Removes a selection listener from the list.

**Parameters:**listener - the ListSelectionListener to remove**See Also:**[addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener)), [getSelectionModel()](http://docs.google.com/javax/swing/JList.html#getSelectionModel())

### getListSelectionListeners

public [ListSelectionListener](http://docs.google.com/javax/swing/event/ListSelectionListener.html)[] **getListSelectionListeners**()

Returns an array of all the ListSelectionListeners added to this JList by way of addListSelectionListener.

**Returns:**all of the ListSelectionListeners on this list, or an empty array if no listeners have been added**Since:** 1.4 **See Also:**[addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))

### setSelectionModel

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

Sets the selectionModel for the list to a non-null ListSelectionModel implementation. The selection model handles the task of making single selections, selections of contiguous ranges, and non-contiguous selections.

This is a JavaBeans bound property.

**Parameters:**selectionModel - the ListSelectionModel that implements the selections **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if selectionModel is null**See Also:**[getSelectionModel()](http://docs.google.com/javax/swing/JList.html#getSelectionModel())

### setSelectionMode

public void **setSelectionMode**(int selectionMode)

Sets the selection mode for the list. This is a cover method that sets the selection mode directly on the selection model.

The following list describes the accepted selection modes:

* ListSelectionModel.SINGLE\_SELECTION - Only one list index can be selected at a time. In this mode, setSelectionInterval and addSelectionInterval are equivalent, both replacing the current selection with the index represented by the second argument (the "lead").
* ListSelectionModel.SINGLE\_INTERVAL\_SELECTION - Only one contiguous interval can be selected at a time. In this mode, addSelectionInterval behaves like setSelectionInterval (replacing the current selection}, unless the given interval is immediately adjacent to or overlaps the existing selection, and can be used to grow the selection.
* ListSelectionModel.MULTIPLE\_INTERVAL\_SELECTION - In this mode, there's no restriction on what can be selected. This mode is the default.

**Parameters:**selectionMode - the selection mode **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the selection mode isn't one of those allowed**See Also:**[getSelectionMode()](http://docs.google.com/javax/swing/JList.html#getSelectionMode())

### getSelectionMode

public int **getSelectionMode**()

Returns the current selection mode for the list. This is a cover method that delegates to the method of the same name on the list's selection model.

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

### getAnchorSelectionIndex

public int **getAnchorSelectionIndex**()

Returns the anchor selection index. This is a cover method that delegates to the method of the same name on the list's selection model.

**Returns:**the anchor selection index**See Also:**[ListSelectionModel.getAnchorSelectionIndex()](http://docs.google.com/javax/swing/ListSelectionModel.html#getAnchorSelectionIndex())

### getLeadSelectionIndex

public int **getLeadSelectionIndex**()

Returns the lead selection index. This is a cover method that delegates to the method of the same name on the list's selection model.

**Returns:**the lead selection index**See Also:**[ListSelectionModel.getLeadSelectionIndex()](http://docs.google.com/javax/swing/ListSelectionModel.html#getLeadSelectionIndex())

### getMinSelectionIndex

public int **getMinSelectionIndex**()

Returns the smallest selected cell index, or -1 if the selection is empty. This is a cover method that delegates to the method of the same name on the list's selection model.

**Returns:**the smallest selected cell index, or -1**See Also:**[ListSelectionModel.getMinSelectionIndex()](http://docs.google.com/javax/swing/ListSelectionModel.html#getMinSelectionIndex())

### getMaxSelectionIndex

public int **getMaxSelectionIndex**()

Returns the largest selected cell index, or -1 if the selection is empty. This is a cover method that delegates to the method of the same name on the list's selection model.

**Returns:**the largest selected cell index**See Also:**[ListSelectionModel.getMaxSelectionIndex()](http://docs.google.com/javax/swing/ListSelectionModel.html#getMaxSelectionIndex())

### isSelectedIndex

public boolean **isSelectedIndex**(int index)

Returns true if the specified index is selected, else false. This is a cover method that delegates to the method of the same name on the list's selection model.

**Parameters:**index - index to be queried for selection state **Returns:**true if the specified index is selected, else false**See Also:**[ListSelectionModel.isSelectedIndex(int)](http://docs.google.com/javax/swing/ListSelectionModel.html#isSelectedIndex(int)), [setSelectedIndex(int)](http://docs.google.com/javax/swing/JList.html#setSelectedIndex(int))

### isSelectionEmpty

public boolean **isSelectionEmpty**()

Returns true if nothing is selected, else false. This is a cover method that delegates to the method of the same name on the list's selection model.

**Returns:**true if nothing is selected, else false**See Also:**[ListSelectionModel.isSelectionEmpty()](http://docs.google.com/javax/swing/ListSelectionModel.html#isSelectionEmpty()), [clearSelection()](http://docs.google.com/javax/swing/JList.html#clearSelection())

### clearSelection

public void **clearSelection**()

Clears the selection; after calling this method, isSelectionEmpty will return true. This is a cover method that delegates to the method of the same name on the list's selection model.

**See Also:**[ListSelectionModel.clearSelection()](http://docs.google.com/javax/swing/ListSelectionModel.html#clearSelection()), [isSelectionEmpty()](http://docs.google.com/javax/swing/JList.html#isSelectionEmpty())

### setSelectionInterval

public void **setSelectionInterval**(int anchor,  
 int lead)

Selects the specified interval. Both anchor and lead indices are included. anchor doesn't have to be less than or equal to lead. This is a cover method that delegates to the method of the same name on the list's selection model.

Refer to the documentation of the selection model class being used for details on how values less than 0 are handled.

**Parameters:**anchor - the first index to selectlead - the last index to select**See Also:**[ListSelectionModel.setSelectionInterval(int, int)](http://docs.google.com/javax/swing/ListSelectionModel.html#setSelectionInterval(int,%20int)), [DefaultListSelectionModel.setSelectionInterval(int, int)](http://docs.google.com/javax/swing/DefaultListSelectionModel.html#setSelectionInterval(int,%20int)), [createSelectionModel()](http://docs.google.com/javax/swing/JList.html#createSelectionModel()), [addSelectionInterval(int, int)](http://docs.google.com/javax/swing/JList.html#addSelectionInterval(int,%20int)), [removeSelectionInterval(int, int)](http://docs.google.com/javax/swing/JList.html#removeSelectionInterval(int,%20int))

### addSelectionInterval

public void **addSelectionInterval**(int anchor,  
 int lead)

Sets the selection to be the union of the specified interval with current selection. Both the anchor and lead indices are included. anchor doesn't have to be less than or equal to lead. This is a cover method that delegates to the method of the same name on the list's selection model.

Refer to the documentation of the selection model class being used for details on how values less than 0 are handled.

**Parameters:**anchor - the first index to add to the selectionlead - the last index to add to the selection**See Also:**[ListSelectionModel.addSelectionInterval(int, int)](http://docs.google.com/javax/swing/ListSelectionModel.html#addSelectionInterval(int,%20int)), [DefaultListSelectionModel.addSelectionInterval(int, int)](http://docs.google.com/javax/swing/DefaultListSelectionModel.html#addSelectionInterval(int,%20int)), [createSelectionModel()](http://docs.google.com/javax/swing/JList.html#createSelectionModel()), [setSelectionInterval(int, int)](http://docs.google.com/javax/swing/JList.html#setSelectionInterval(int,%20int)), [removeSelectionInterval(int, int)](http://docs.google.com/javax/swing/JList.html#removeSelectionInterval(int,%20int))

### removeSelectionInterval

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

Sets the selection to be the set difference of the specified interval and the current selection. Both the index0 and index1 indices are removed. index0 doesn't have to be less than or equal to index1. This is a cover method that delegates to the method of the same name on the list's selection model.

Refer to the documentation of the selection model class being used for details on how values less than 0 are handled.

**Parameters:**index0 - the first index to remove from the selectionindex1 - the last index to remove from the selection**See Also:**[ListSelectionModel.removeSelectionInterval(int, int)](http://docs.google.com/javax/swing/ListSelectionModel.html#removeSelectionInterval(int,%20int)), [DefaultListSelectionModel.removeSelectionInterval(int, int)](http://docs.google.com/javax/swing/DefaultListSelectionModel.html#removeSelectionInterval(int,%20int)), [createSelectionModel()](http://docs.google.com/javax/swing/JList.html#createSelectionModel()), [setSelectionInterval(int, int)](http://docs.google.com/javax/swing/JList.html#setSelectionInterval(int,%20int)), [addSelectionInterval(int, int)](http://docs.google.com/javax/swing/JList.html#addSelectionInterval(int,%20int))

### setValueIsAdjusting

public void **setValueIsAdjusting**(boolean b)

Sets the selection model's valueIsAdjusting property. When true, upcoming changes to selection should be considered part of a single change. This property is used internally and developers typically need not call this method. For example, when the model is being updated in response to a user drag, the value of the property is set to true when the drag is initiated and set to false when the drag is finished. This allows listeners to update only when a change has been finalized, rather than handling all of the intermediate values.

You may want to use this directly if making a series of changes that should be considered part of a single change.

This is a cover method that delegates to the method of the same name on the list's selection model. See the documentation for [ListSelectionModel.setValueIsAdjusting(boolean)](http://docs.google.com/javax/swing/ListSelectionModel.html#setValueIsAdjusting(boolean)) for more details.

**Parameters:**b - the new value for the property**See Also:**[ListSelectionModel.setValueIsAdjusting(boolean)](http://docs.google.com/javax/swing/ListSelectionModel.html#setValueIsAdjusting(boolean)), [ListSelectionEvent.getValueIsAdjusting()](http://docs.google.com/javax/swing/event/ListSelectionEvent.html#getValueIsAdjusting()), [getValueIsAdjusting()](http://docs.google.com/javax/swing/JList.html#getValueIsAdjusting())

### getValueIsAdjusting

public boolean **getValueIsAdjusting**()

Returns the value of the selection model's isAdjusting property.

This is a cover method that delegates to the method of the same name on the list's selection model.

**Returns:**the value of the selection model's isAdjusting property.**See Also:**[setValueIsAdjusting(boolean)](http://docs.google.com/javax/swing/JList.html#setValueIsAdjusting(boolean)), [ListSelectionModel.getValueIsAdjusting()](http://docs.google.com/javax/swing/ListSelectionModel.html#getValueIsAdjusting())

### getSelectedIndices

public int[] **getSelectedIndices**()

Returns an array of all of the selected indices, in increasing order.

**Returns:**all of the selected indices, in increasing order, or an empty array if nothing is selected**See Also:**[removeSelectionInterval(int, int)](http://docs.google.com/javax/swing/JList.html#removeSelectionInterval(int,%20int)), [addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))

### setSelectedIndex

public void **setSelectedIndex**(int index)

Selects a single cell. Does nothing if the given index is greater than or equal to the model size. This is a convenience method that uses setSelectionInterval on the selection model. Refer to the documentation for the selection model class being used for details on how values less than 0 are handled.

**Parameters:**index - the index of the cell to select**See Also:**[ListSelectionModel.setSelectionInterval(int, int)](http://docs.google.com/javax/swing/ListSelectionModel.html#setSelectionInterval(int,%20int)), [isSelectedIndex(int)](http://docs.google.com/javax/swing/JList.html#isSelectedIndex(int)), [addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))

### setSelectedIndices

public void **setSelectedIndices**(int[] indices)

Changes the selection to be the set of indices specified by the given array. Indices greater than or equal to the model size are ignored. This is a convenience method that clears the selection and then uses addSelectionInterval on the selection model to add the indices. Refer to the documentation of the selection model class being used for details on how values less than 0 are handled.

**Parameters:**indices - an array of the indices of the cells to select, non-null **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the given array is null**See Also:**[ListSelectionModel.addSelectionInterval(int, int)](http://docs.google.com/javax/swing/ListSelectionModel.html#addSelectionInterval(int,%20int)), [isSelectedIndex(int)](http://docs.google.com/javax/swing/JList.html#isSelectedIndex(int)), [addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))

### getSelectedValues

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

Returns an array of all the selected values, in increasing order based on their indices in the list.

**Returns:**the selected values, or an empty array if nothing is selected**See Also:**[isSelectedIndex(int)](http://docs.google.com/javax/swing/JList.html#isSelectedIndex(int)), [getModel()](http://docs.google.com/javax/swing/JList.html#getModel()), [addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))

### getSelectedIndex

public int **getSelectedIndex**()

Returns the smallest selected cell index; *the selection* when only a single item is selected in the list. When multiple items are selected, it is simply the smallest selected index. Returns -1 if there is no selection.

This method is a cover that delegates to getMinSelectionIndex.

**Returns:**the smallest selected cell index**See Also:**[getMinSelectionIndex()](http://docs.google.com/javax/swing/JList.html#getMinSelectionIndex()), [addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))

### getSelectedValue

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

Returns the value for the smallest selected cell index; *the selected value* when only a single item is selected in the list. When multiple items are selected, it is simply the value for the smallest selected index. Returns null if there is no selection.

This is a convenience method that simply returns the model value for getMinSelectionIndex.

**Returns:**the first selected value**See Also:**[getMinSelectionIndex()](http://docs.google.com/javax/swing/JList.html#getMinSelectionIndex()), [getModel()](http://docs.google.com/javax/swing/JList.html#getModel()), [addListSelectionListener(javax.swing.event.ListSelectionListener)](http://docs.google.com/javax/swing/JList.html#addListSelectionListener(javax.swing.event.ListSelectionListener))

### setSelectedValue

public void **setSelectedValue**([Object](http://docs.google.com/java/lang/Object.html) anObject,  
 boolean shouldScroll)

Selects the specified object from the list.

**Parameters:**anObject - the object to selectshouldScroll - true if the list should scroll to display the selected object, if one exists; otherwise false

### getPreferredScrollableViewportSize

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

Computes the size of viewport needed to display visibleRowCount rows. The value returned by this method depends on the layout orientation:

**VERTICAL:**

This is trivial if both fixedCellWidth and fixedCellHeight have been set (either explicitly or by specifying a prototype cell value). The width is simply the fixedCellWidth plus the list's horizontal insets. The height is the fixedCellHeight multiplied by the visibleRowCount, plus the list's vertical insets.

If either fixedCellWidth or fixedCellHeight haven't been specified, heuristics are used. If the model is empty, the width is the fixedCellWidth, if greater than 0, or a hard-coded value of 256. The height is the fixedCellHeight multiplied by visibleRowCount, if fixedCellHeight is greater than 0, otherwise it is a hard-coded value of 16 multiplied by visibleRowCount.

If the model isn't empty, the width is the preferred size's width, typically the width of the widest list element. The height is the fixedCellHeight multiplied by the visibleRowCount, plus the list's vertical insets.

**VERTICAL\_WRAP or HORIZONTAL\_WRAP:**

This method simply returns the value from getPreferredSize. The list's ListUI is expected to override getPreferredSize to return an appropriate value.

**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 containing the size of the viewport needed to display visibleRowCount rows**See Also:**[getPreferredScrollableViewportSize()](http://docs.google.com/javax/swing/JList.html#getPreferredScrollableViewportSize()), [setPrototypeCellValue(java.lang.Object)](http://docs.google.com/javax/swing/JList.html#setPrototypeCellValue(java.lang.Object))

### getScrollableUnitIncrement

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

Returns the distance to scroll to expose the next or previous row (for vertical scrolling) or column (for horizontal scrolling).

For horizontal scrolling, if the layout orientation is VERTICAL, then the list's font size is returned (or 1 if the font is null).

**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 - SwingConstants.HORIZONTAL or SwingConstants.VERTICALdirection - less or equal to zero to scroll up/back, greater than zero for down/forward **Returns:**the "unit" increment for scrolling in the specified direction; always positive **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if visibleRect is null, or orientation isn't one of SwingConstants.VERTICAL or SwingConstants.HORIZONTAL**See Also:**[getScrollableBlockIncrement(java.awt.Rectangle, int, int)](http://docs.google.com/javax/swing/JList.html#getScrollableBlockIncrement(java.awt.Rectangle,%20int,%20int)), [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 the distance to scroll to expose the next or previous block.

For vertical scrolling, the following rules are used:

* if scrolling down, returns the distance to scroll so that the last visible element becomes the first completely visible element
* if scrolling up, returns the distance to scroll so that the first visible element becomes the last completely visible element
* returns visibleRect.height if the list is empty

For horizontal scrolling, when the layout orientation is either VERTICAL\_WRAP or HORIZONTAL\_WRAP:

* if scrolling right, returns the distance to scroll so that the last visible element becomes the first completely visible element
* if scrolling left, returns the distance to scroll so that the first visible element becomes the last completely visible element
* returns visibleRect.width if the list is empty

For horizontal scrolling and VERTICAL orientation, returns visibleRect.width.

Note that the value of visibleRect must be the equal to this.getVisibleRect().

**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 - SwingConstants.HORIZONTAL or SwingConstants.VERTICALdirection - less or equal to zero to scroll up/back, greater than zero for down/forward **Returns:**the "block" increment for scrolling in the specified direction; always positive **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if visibleRect is null, or orientation isn't one of SwingConstants.VERTICAL or SwingConstants.HORIZONTAL**See Also:**[getScrollableUnitIncrement(java.awt.Rectangle, int, int)](http://docs.google.com/javax/swing/JList.html#getScrollableUnitIncrement(java.awt.Rectangle,%20int,%20int)), [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 true if this JList is displayed in a JViewport and the viewport is wider than the list's preferred width, or if the layout orientation is HORIZONTAL\_WRAP and visibleRowCount <= 0; otherwise returns false.

If false, then don't track the viewport's width. This allows horizontal scrolling if the JViewport is itself embedded in a JScrollPane.

**Specified by:**[getScrollableTracksViewportWidth](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportWidth()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**whether or not an enclosing viewport should force the list's width to match its own**See Also:**[Scrollable.getScrollableTracksViewportWidth()](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportWidth())

### getScrollableTracksViewportHeight

public boolean **getScrollableTracksViewportHeight**()

Returns true if this JList is displayed in a JViewport and the viewport is taller than the list's preferred height, or if the layout orientation is VERTICAL\_WRAP and visibleRowCount <= 0; otherwise returns false.

If false, then don't track the viewport's height. This allows vertical scrolling if the JViewport is itself embedded in a JScrollPane.

**Specified by:**[getScrollableTracksViewportHeight](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportHeight()) in interface [Scrollable](http://docs.google.com/javax/swing/Scrollable.html) **Returns:**whether or not an enclosing viewport should force the list's height to match its own**See Also:**[Scrollable.getScrollableTracksViewportHeight()](http://docs.google.com/javax/swing/Scrollable.html#getScrollableTracksViewportHeight())

### paramString

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

Returns a String representation of this JList. 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 JList.

### getAccessibleContext

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

Gets the AccessibleContext associated with this JList. For JList, the AccessibleContext takes the form of an AccessibleJList.

A new AccessibleJList 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 AccessibleJList that serves as the AccessibleContext of this JList

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/JList.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/JLayeredPane.AccessibleJLayeredPane.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/JList.AccessibleJList.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/JList.html)    [**NO FRAMES**](http://docs.google.com/JList.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#tyjcwt) | [FIELD](#2s8eyo1) | [CONSTR](#lnxbz9) | [METHOD](#35nkun2) | DETAIL: [FIELD](#3j2qqm3) | [CONSTR](#1ci93xb) | [METHOD](#1pxezwc) |

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

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

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