| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CompositeView.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/text/ComponentView.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/DateFormatter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/CompositeView.html)    [**NO FRAMES**](http://docs.google.com/CompositeView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

## **javax.swing.text**

Class CompositeView

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.swing.text.View](http://docs.google.com/javax/swing/text/View.html)  
 **javax.swing.text.CompositeView**

**All Implemented Interfaces:** [SwingConstants](http://docs.google.com/javax/swing/SwingConstants.html) **Direct Known Subclasses:** [BoxView](http://docs.google.com/javax/swing/text/BoxView.html)

public abstract class **CompositeView**extends [View](http://docs.google.com/javax/swing/text/View.html)

CompositeView is an abstract View implementation which manages one or more child views. (Note that CompositeView is intended for managing relatively small numbers of child views.) CompositeView is intended to be used as a starting point for View implementations, such as BoxView, that will contain child Views. Subclasses that wish to manage the collection of child Views should use the [replace(int, int, javax.swing.text.View[])](http://docs.google.com/javax/swing/text/CompositeView.html#replace(int,%20int,%20javax.swing.text.View%5B%5D)) method. As View invokes replace during DocumentListener notification, you normally won't need to directly invoke replace.

While CompositeView does not impose a layout policy on its child Views, it does allow for inseting the child Views it will contain. The insets can be set by either [setInsets(short, short, short, short)](http://docs.google.com/javax/swing/text/CompositeView.html#setInsets(short,%20short,%20short,%20short)) or [setParagraphInsets(javax.swing.text.AttributeSet)](http://docs.google.com/javax/swing/text/CompositeView.html#setParagraphInsets(javax.swing.text.AttributeSet)).

In addition to the abstract methods of [View](http://docs.google.com/javax/swing/text/View.html), subclasses of CompositeView will need to override:

* [isBefore(int, int, java.awt.Rectangle)](http://docs.google.com/javax/swing/text/CompositeView.html#isBefore(int,%20int,%20java.awt.Rectangle)) - Used to test if a given View location is before the visual space of the CompositeView.
* [isAfter(int, int, java.awt.Rectangle)](http://docs.google.com/javax/swing/text/CompositeView.html#isAfter(int,%20int,%20java.awt.Rectangle)) - Used to test if a given View location is after the visual space of the CompositeView.
* [getViewAtPoint(int, int, java.awt.Rectangle)](http://docs.google.com/javax/swing/text/CompositeView.html#getViewAtPoint(int,%20int,%20java.awt.Rectangle)) - Returns the view at a given visual location.
* [childAllocation(int, java.awt.Rectangle)](http://docs.google.com/javax/swing/text/CompositeView.html#childAllocation(int,%20java.awt.Rectangle)) - Returns the bounds of a particular child View. getChildAllocation will invoke childAllocation after offseting the bounds by the Insets of the CompositeView.

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class javax.swing.text.**[**View**](http://docs.google.com/javax/swing/text/View.html) |
| --- |
| [BadBreakWeight](http://docs.google.com/javax/swing/text/View.html#BadBreakWeight), [ExcellentBreakWeight](http://docs.google.com/javax/swing/text/View.html#ExcellentBreakWeight), [ForcedBreakWeight](http://docs.google.com/javax/swing/text/View.html#ForcedBreakWeight), [GoodBreakWeight](http://docs.google.com/javax/swing/text/View.html#GoodBreakWeight), [X\_AXIS](http://docs.google.com/javax/swing/text/View.html#X_AXIS), [Y\_AXIS](http://docs.google.com/javax/swing/text/View.html#Y_AXIS) |

| **Fields inherited from interface javax.swing.**[**SwingConstants**](http://docs.google.com/javax/swing/SwingConstants.html) |
| --- |
| [BOTTOM](http://docs.google.com/javax/swing/SwingConstants.html#BOTTOM), [CENTER](http://docs.google.com/javax/swing/SwingConstants.html#CENTER), [EAST](http://docs.google.com/javax/swing/SwingConstants.html#EAST), [HORIZONTAL](http://docs.google.com/javax/swing/SwingConstants.html#HORIZONTAL), [LEADING](http://docs.google.com/javax/swing/SwingConstants.html#LEADING), [LEFT](http://docs.google.com/javax/swing/SwingConstants.html#LEFT), [NEXT](http://docs.google.com/javax/swing/SwingConstants.html#NEXT), [NORTH](http://docs.google.com/javax/swing/SwingConstants.html#NORTH), [NORTH\_EAST](http://docs.google.com/javax/swing/SwingConstants.html#NORTH_EAST), [NORTH\_WEST](http://docs.google.com/javax/swing/SwingConstants.html#NORTH_WEST), [PREVIOUS](http://docs.google.com/javax/swing/SwingConstants.html#PREVIOUS), [RIGHT](http://docs.google.com/javax/swing/SwingConstants.html#RIGHT), [SOUTH](http://docs.google.com/javax/swing/SwingConstants.html#SOUTH), [SOUTH\_EAST](http://docs.google.com/javax/swing/SwingConstants.html#SOUTH_EAST), [SOUTH\_WEST](http://docs.google.com/javax/swing/SwingConstants.html#SOUTH_WEST), [TOP](http://docs.google.com/javax/swing/SwingConstants.html#TOP), [TRAILING](http://docs.google.com/javax/swing/SwingConstants.html#TRAILING), [VERTICAL](http://docs.google.com/javax/swing/SwingConstants.html#VERTICAL), [WEST](http://docs.google.com/javax/swing/SwingConstants.html#WEST) |

| **Constructor Summary** | |
| --- | --- |
| [**CompositeView**](http://docs.google.com/javax/swing/text/CompositeView.html#CompositeView(javax.swing.text.Element))([Element](http://docs.google.com/javax/swing/text/Element.html) elem)            Constructs a CompositeView for the given element. |

| **Method Summary** | |
| --- | --- |
| protected abstract  void | [**childAllocation**](http://docs.google.com/javax/swing/text/CompositeView.html#childAllocation(int,%20java.awt.Rectangle))(int index, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) a)            Returns the allocation for a given child. |
| protected  boolean | [**flipEastAndWestAtEnds**](http://docs.google.com/javax/swing/text/CompositeView.html#flipEastAndWestAtEnds(int,%20javax.swing.text.Position.Bias))(int position, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) bias)            Determines in which direction the next view lays. |
| protected  short | [**getBottomInset**](http://docs.google.com/javax/swing/text/CompositeView.html#getBottomInset())()            Gets the bottom inset. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getChildAllocation**](http://docs.google.com/javax/swing/text/CompositeView.html#getChildAllocation(int,%20java.awt.Shape))(int index, [Shape](http://docs.google.com/java/awt/Shape.html) a)            Fetches the allocation for the given child view to render into. |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getInsideAllocation**](http://docs.google.com/javax/swing/text/CompositeView.html#getInsideAllocation(java.awt.Shape))([Shape](http://docs.google.com/java/awt/Shape.html) a)            Translates the immutable allocation given to the view to a mutable allocation that represents the interior allocation (i.e. |
| protected  short | [**getLeftInset**](http://docs.google.com/javax/swing/text/CompositeView.html#getLeftInset())()            Gets the left inset. |
| protected  int | [**getNextEastWestVisualPositionFrom**](http://docs.google.com/javax/swing/text/CompositeView.html#getNextEastWestVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D))(int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b, [Shape](http://docs.google.com/java/awt/Shape.html) a, int direction, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)            Returns the next visual position for the cursor, in either the east or west direction. |
| protected  int | [**getNextNorthSouthVisualPositionFrom**](http://docs.google.com/javax/swing/text/CompositeView.html#getNextNorthSouthVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D))(int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b, [Shape](http://docs.google.com/java/awt/Shape.html) a, int direction, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)            Returns the next visual position for the cursor, in either the north or south direction. |
| int | [**getNextVisualPositionFrom**](http://docs.google.com/javax/swing/text/CompositeView.html#getNextVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D))(int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b, [Shape](http://docs.google.com/java/awt/Shape.html) a, int direction, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)            Provides a way to determine the next visually represented model location that one might place a caret. |
| protected  short | [**getRightInset**](http://docs.google.com/javax/swing/text/CompositeView.html#getRightInset())()            Gets the right inset. |
| protected  short | [**getTopInset**](http://docs.google.com/javax/swing/text/CompositeView.html#getTopInset())()            Gets the top inset. |
| [View](http://docs.google.com/javax/swing/text/View.html) | [**getView**](http://docs.google.com/javax/swing/text/CompositeView.html#getView(int))(int n)            Returns the n-th view in this container. |
| protected abstract  [View](http://docs.google.com/javax/swing/text/View.html) | [**getViewAtPoint**](http://docs.google.com/javax/swing/text/CompositeView.html#getViewAtPoint(int,%20int,%20java.awt.Rectangle))(int x, int y, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc)            Fetches the child view at the given coordinates. |
| protected  [View](http://docs.google.com/javax/swing/text/View.html) | [**getViewAtPosition**](http://docs.google.com/javax/swing/text/CompositeView.html#getViewAtPosition(int,%20java.awt.Rectangle))(int pos, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) a)            Fetches the child view that represents the given position in the model. |
| int | [**getViewCount**](http://docs.google.com/javax/swing/text/CompositeView.html#getViewCount())()            Returns the number of child views of this view. |
| int | [**getViewIndex**](http://docs.google.com/javax/swing/text/CompositeView.html#getViewIndex(int,%20javax.swing.text.Position.Bias))(int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)            Returns the child view index representing the given position in the model. |
| protected  int | [**getViewIndexAtPosition**](http://docs.google.com/javax/swing/text/CompositeView.html#getViewIndexAtPosition(int))(int pos)            Fetches the child view index representing the given position in the model. |
| protected abstract  boolean | [**isAfter**](http://docs.google.com/javax/swing/text/CompositeView.html#isAfter(int,%20int,%20java.awt.Rectangle))(int x, int y, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc)            Tests whether a point lies after the rectangle range. |
| protected abstract  boolean | [**isBefore**](http://docs.google.com/javax/swing/text/CompositeView.html#isBefore(int,%20int,%20java.awt.Rectangle))(int x, int y, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc)            Tests whether a point lies before the rectangle range. |
| protected  void | [**loadChildren**](http://docs.google.com/javax/swing/text/CompositeView.html#loadChildren(javax.swing.text.ViewFactory))([ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)            Loads all of the children to initialize the view. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**modelToView**](http://docs.google.com/javax/swing/text/CompositeView.html#modelToView(int,%20javax.swing.text.Position.Bias,%20int,%20javax.swing.text.Position.Bias,%20java.awt.Shape))(int p0, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b0, int p1, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b1, [Shape](http://docs.google.com/java/awt/Shape.html) a)            Provides a mapping from the document model coordinate space to the coordinate space of the view mapped to it. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**modelToView**](http://docs.google.com/javax/swing/text/CompositeView.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias))(int pos, [Shape](http://docs.google.com/java/awt/Shape.html) a, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)            Provides a mapping from the document model coordinate space to the coordinate space of the view mapped to it. |
| void | [**replace**](http://docs.google.com/javax/swing/text/CompositeView.html#replace(int,%20int,%20javax.swing.text.View%5B%5D))(int offset, int length, [View](http://docs.google.com/javax/swing/text/View.html)[] views)            Replaces child views. |
| protected  void | [**setInsets**](http://docs.google.com/javax/swing/text/CompositeView.html#setInsets(short,%20short,%20short,%20short))(short top, short left, short bottom, short right)            Sets the insets for the view. |
| protected  void | [**setParagraphInsets**](http://docs.google.com/javax/swing/text/CompositeView.html#setParagraphInsets(javax.swing.text.AttributeSet))([AttributeSet](http://docs.google.com/javax/swing/text/AttributeSet.html) attr)            Sets the insets from the paragraph attributes specified in the given attributes. |
| void | [**setParent**](http://docs.google.com/javax/swing/text/CompositeView.html#setParent(javax.swing.text.View))([View](http://docs.google.com/javax/swing/text/View.html) parent)            Sets the parent of the view. |
| int | [**viewToModel**](http://docs.google.com/javax/swing/text/CompositeView.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D))(float x, float y, [Shape](http://docs.google.com/java/awt/Shape.html) a, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] bias)            Provides a mapping from the view coordinate space to the logical coordinate space of the model. |

| **Methods inherited from class javax.swing.text.**[**View**](http://docs.google.com/javax/swing/text/View.html) |
| --- |
| [append](http://docs.google.com/javax/swing/text/View.html#append(javax.swing.text.View)), [breakView](http://docs.google.com/javax/swing/text/View.html#breakView(int,%20int,%20float,%20float)), [changedUpdate](http://docs.google.com/javax/swing/text/View.html#changedUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [createFragment](http://docs.google.com/javax/swing/text/View.html#createFragment(int,%20int)), [forwardUpdate](http://docs.google.com/javax/swing/text/View.html#forwardUpdate(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [forwardUpdateToView](http://docs.google.com/javax/swing/text/View.html#forwardUpdateToView(javax.swing.text.View,%20javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [getAlignment](http://docs.google.com/javax/swing/text/View.html#getAlignment(int)), [getAttributes](http://docs.google.com/javax/swing/text/View.html#getAttributes()), [getBreakWeight](http://docs.google.com/javax/swing/text/View.html#getBreakWeight(int,%20float,%20float)), [getContainer](http://docs.google.com/javax/swing/text/View.html#getContainer()), [getDocument](http://docs.google.com/javax/swing/text/View.html#getDocument()), [getElement](http://docs.google.com/javax/swing/text/View.html#getElement()), [getEndOffset](http://docs.google.com/javax/swing/text/View.html#getEndOffset()), [getGraphics](http://docs.google.com/javax/swing/text/View.html#getGraphics()), [getMaximumSpan](http://docs.google.com/javax/swing/text/View.html#getMaximumSpan(int)), [getMinimumSpan](http://docs.google.com/javax/swing/text/View.html#getMinimumSpan(int)), [getParent](http://docs.google.com/javax/swing/text/View.html#getParent()), [getPreferredSpan](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int)), [getResizeWeight](http://docs.google.com/javax/swing/text/View.html#getResizeWeight(int)), [getStartOffset](http://docs.google.com/javax/swing/text/View.html#getStartOffset()), [getToolTipText](http://docs.google.com/javax/swing/text/View.html#getToolTipText(float,%20float,%20java.awt.Shape)), [getViewFactory](http://docs.google.com/javax/swing/text/View.html#getViewFactory()), [getViewIndex](http://docs.google.com/javax/swing/text/View.html#getViewIndex(float,%20float,%20java.awt.Shape)), [insert](http://docs.google.com/javax/swing/text/View.html#insert(int,%20javax.swing.text.View)), [insertUpdate](http://docs.google.com/javax/swing/text/View.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [isVisible](http://docs.google.com/javax/swing/text/View.html#isVisible()), [modelToView](http://docs.google.com/javax/swing/text/View.html#modelToView(int,%20java.awt.Shape)), [paint](http://docs.google.com/javax/swing/text/View.html#paint(java.awt.Graphics,%20java.awt.Shape)), [preferenceChanged](http://docs.google.com/javax/swing/text/View.html#preferenceChanged(javax.swing.text.View,%20boolean,%20boolean)), [remove](http://docs.google.com/javax/swing/text/View.html#remove(int)), [removeAll](http://docs.google.com/javax/swing/text/View.html#removeAll()), [removeUpdate](http://docs.google.com/javax/swing/text/View.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [setSize](http://docs.google.com/javax/swing/text/View.html#setSize(float,%20float)), [updateChildren](http://docs.google.com/javax/swing/text/View.html#updateChildren(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20javax.swing.text.ViewFactory)), [updateLayout](http://docs.google.com/javax/swing/text/View.html#updateLayout(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape)), [viewToModel](http://docs.google.com/javax/swing/text/View.html#viewToModel(float,%20float,%20java.awt.Shape)) |

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

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

### CompositeView

public **CompositeView**([Element](http://docs.google.com/javax/swing/text/Element.html) elem)

Constructs a CompositeView for the given element.

**Parameters:**elem - the element this view is responsible for

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

### loadChildren

protected void **loadChildren**([ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)

Loads all of the children to initialize the view. This is called by the [setParent(javax.swing.text.View)](http://docs.google.com/javax/swing/text/CompositeView.html#setParent(javax.swing.text.View)) method. Subclasses can reimplement this to initialize their child views in a different manner. The default implementation creates a child view for each child element.

**Parameters:**f - the view factory**See Also:**[setParent(javax.swing.text.View)](http://docs.google.com/javax/swing/text/CompositeView.html#setParent(javax.swing.text.View))

### setParent

public void **setParent**([View](http://docs.google.com/javax/swing/text/View.html) parent)

Sets the parent of the view. This is reimplemented to provide the superclass behavior as well as calling the loadChildren method if this view does not already have children. The children should not be loaded in the constructor because the act of setting the parent may cause them to try to search up the hierarchy (to get the hosting Container for example). If this view has children (the view is being moved from one place in the view hierarchy to another), the loadChildren method will not be called.

**Overrides:**[setParent](http://docs.google.com/javax/swing/text/View.html#setParent(javax.swing.text.View)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**parent - the parent of the view, null if none

### getViewCount

public int **getViewCount**()

Returns the number of child views of this view.

**Overrides:**[getViewCount](http://docs.google.com/javax/swing/text/View.html#getViewCount()) in class [View](http://docs.google.com/javax/swing/text/View.html) **Returns:**the number of views >= 0**See Also:**[getView(int)](http://docs.google.com/javax/swing/text/CompositeView.html#getView(int))

### getView

public [View](http://docs.google.com/javax/swing/text/View.html) **getView**(int n)

Returns the n-th view in this container.

**Overrides:**[getView](http://docs.google.com/javax/swing/text/View.html#getView(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**n - the number of the desired view, >= 0 && < getViewCount() **Returns:**the view at index n

### replace

public void **replace**(int offset,  
 int length,  
 [View](http://docs.google.com/javax/swing/text/View.html)[] views)

Replaces child views. If there are no views to remove this acts as an insert. If there are no views to add this acts as a remove. Views being removed will have the parent set to null, and the internal reference to them removed so that they may be garbage collected.

**Overrides:**[replace](http://docs.google.com/javax/swing/text/View.html#replace(int,%20int,%20javax.swing.text.View%5B%5D)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**offset - the starting index into the child views to insert the new views; >= 0 and <= getViewCountlength - the number of existing child views to remove; this should be a value >= 0 and <= (getViewCount() - offset)views - the child views to add; this value can be null to indicate no children are being added (useful to remove)

### getChildAllocation

public [Shape](http://docs.google.com/java/awt/Shape.html) **getChildAllocation**(int index,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a)

Fetches the allocation for the given child view to render into. This enables finding out where various views are located.

**Overrides:**[getChildAllocation](http://docs.google.com/javax/swing/text/View.html#getChildAllocation(int,%20java.awt.Shape)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**index - the index of the child, >= 0 && < getViewCount()a - the allocation to this view **Returns:**the allocation to the child

### modelToView

public [Shape](http://docs.google.com/java/awt/Shape.html) **modelToView**(int pos,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Provides a mapping from the document model coordinate space to the coordinate space of the view mapped to it.

**Specified by:**[modelToView](http://docs.google.com/javax/swing/text/View.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**pos - the position to convert >= 0a - the allocated region to render intob - a bias value of either Position.Bias.Forward or Position.Bias.Backward **Returns:**the bounding box of the given position **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the given position does not represent a valid location in the associated document**See Also:**[View.modelToView(int, java.awt.Shape, javax.swing.text.Position.Bias)](http://docs.google.com/javax/swing/text/View.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias))

### modelToView

public [Shape](http://docs.google.com/java/awt/Shape.html) **modelToView**(int p0,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b0,  
 int p1,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b1,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Provides a mapping from the document model coordinate space to the coordinate space of the view mapped to it.

**Overrides:**[modelToView](http://docs.google.com/javax/swing/text/View.html#modelToView(int,%20javax.swing.text.Position.Bias,%20int,%20javax.swing.text.Position.Bias,%20java.awt.Shape)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**p0 - the position to convert >= 0b0 - the bias toward the previous character or the next character represented by p0, in case the position is a boundary of two views; either Position.Bias.Forward or Position.Bias.Backwardp1 - the position to convert >= 0b1 - the bias toward the previous character or the next character represented by p1, in case the position is a boundary of two viewsa - the allocated region to render into **Returns:**the bounding box of the given position is returned **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the given position does not represent a valid location in the associated document [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid bias argument**See Also:**[View.viewToModel(float, float, java.awt.Shape, javax.swing.text.Position.Bias[])](http://docs.google.com/javax/swing/text/View.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D))

### viewToModel

public int **viewToModel**(float x,  
 float y,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] bias)

Provides a mapping from the view coordinate space to the logical coordinate space of the model.

**Specified by:**[viewToModel](http://docs.google.com/javax/swing/text/View.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**x - x coordinate of the view location to convert >= 0y - y coordinate of the view location to convert >= 0a - the allocated region to render intobias - either Position.Bias.Forward or Position.Bias.Backward **Returns:**the location within the model that best represents the given point in the view >= 0**See Also:**[View.viewToModel(float, float, java.awt.Shape, javax.swing.text.Position.Bias[])](http://docs.google.com/javax/swing/text/View.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D))

### getNextVisualPositionFrom

public int **getNextVisualPositionFrom**(int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 int direction,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Provides a way to determine the next visually represented model location that one might place a caret. Some views may not be visible, they might not be in the same order found in the model, or they just might not allow access to some of the locations in the model. This is a convenience method for [getNextNorthSouthVisualPositionFrom(int, javax.swing.text.Position.Bias, java.awt.Shape, int, javax.swing.text.Position.Bias[])](http://docs.google.com/javax/swing/text/CompositeView.html#getNextNorthSouthVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D)) and [getNextEastWestVisualPositionFrom(int, javax.swing.text.Position.Bias, java.awt.Shape, int, javax.swing.text.Position.Bias[])](http://docs.google.com/javax/swing/text/CompositeView.html#getNextEastWestVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D)).

**Overrides:**[getNextVisualPositionFrom](http://docs.google.com/javax/swing/text/View.html#getNextVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**pos - the position to convert >= 0b - a bias value of either Position.Bias.Forward or Position.Bias.Backwarda - the allocated region to render intodirection - the direction from the current position that can be thought of as the arrow keys typically found on a keyboard; this may be one of the following:

* SwingConstants.WEST
* SwingConstants.EAST
* SwingConstants.NORTH
* SwingConstants.SOUTH

biasRet - an array containing the bias that was checked **Returns:**the location within the model that best represents the next location visual position **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if direction is invalid

### getViewIndex

public int **getViewIndex**(int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)

Returns the child view index representing the given position in the model. This is implemented to call the getViewIndexByPosition method for backward compatibility.

**Overrides:**[getViewIndex](http://docs.google.com/javax/swing/text/View.html#getViewIndex(int,%20javax.swing.text.Position.Bias)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**pos - the position >= 0 **Returns:**index of the view representing the given position, or -1 if no view represents that position**Since:** 1.3

### isBefore

protected abstract boolean **isBefore**(int x,  
 int y,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc)

Tests whether a point lies before the rectangle range.

**Parameters:**x - the X coordinate >= 0y - the Y coordinate >= 0alloc - the rectangle **Returns:**true if the point is before the specified range

### isAfter

protected abstract boolean **isAfter**(int x,  
 int y,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc)

Tests whether a point lies after the rectangle range.

**Parameters:**x - the X coordinate >= 0y - the Y coordinate >= 0alloc - the rectangle **Returns:**true if the point is after the specified range

### getViewAtPoint

protected abstract [View](http://docs.google.com/javax/swing/text/View.html) **getViewAtPoint**(int x,  
 int y,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc)

Fetches the child view at the given coordinates.

**Parameters:**x - the X coordinate >= 0y - the Y coordinate >= 0alloc - the parent's allocation on entry, which should be changed to the child's allocation on exit **Returns:**the child view

### childAllocation

protected abstract void **childAllocation**(int index,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) a)

Returns the allocation for a given child.

**Parameters:**index - the index of the child, >= 0 && < getViewCount()a - the allocation to the interior of the box on entry, and the allocation of the child view at the index on exit.

### getViewAtPosition

protected [View](http://docs.google.com/javax/swing/text/View.html) **getViewAtPosition**(int pos,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) a)

Fetches the child view that represents the given position in the model. This is implemented to fetch the view in the case where there is a child view for each child element.

**Parameters:**pos - the position >= 0a - the allocation to the interior of the box on entry, and the allocation of the view containing the position on exit **Returns:**the view representing the given position, or null if there isn't one

### getViewIndexAtPosition

protected int **getViewIndexAtPosition**(int pos)

Fetches the child view index representing the given position in the model. This is implemented to fetch the view in the case where there is a child view for each child element.

**Parameters:**pos - the position >= 0 **Returns:**index of the view representing the given position, or -1 if no view represents that position

### getInsideAllocation

protected [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getInsideAllocation**([Shape](http://docs.google.com/java/awt/Shape.html) a)

Translates the immutable allocation given to the view to a mutable allocation that represents the interior allocation (i.e. the bounds of the given allocation with the top, left, bottom, and right insets removed. It is expected that the returned value would be further mutated to represent an allocation to a child view. This is implemented to reuse an instance variable so it avoids creating excessive Rectangles. Typically the result of calling this method would be fed to the childAllocation method.

**Parameters:**a - the allocation given to the view **Returns:**the allocation that represents the inside of the view after the margins have all been removed; if the given allocation was null, the return value is null

### setParagraphInsets

protected void **setParagraphInsets**([AttributeSet](http://docs.google.com/javax/swing/text/AttributeSet.html) attr)

Sets the insets from the paragraph attributes specified in the given attributes.

**Parameters:**attr - the attributes

### setInsets

protected void **setInsets**(short top,  
 short left,  
 short bottom,  
 short right)

Sets the insets for the view.

**Parameters:**top - the top inset >= 0left - the left inset >= 0bottom - the bottom inset >= 0right - the right inset >= 0

### getLeftInset

protected short **getLeftInset**()

Gets the left inset.

**Returns:**the inset >= 0

### getRightInset

protected short **getRightInset**()

Gets the right inset.

**Returns:**the inset >= 0

### getTopInset

protected short **getTopInset**()

Gets the top inset.

**Returns:**the inset >= 0

### getBottomInset

protected short **getBottomInset**()

Gets the bottom inset.

**Returns:**the inset >= 0

### getNextNorthSouthVisualPositionFrom

protected int **getNextNorthSouthVisualPositionFrom**(int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 int direction,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Returns the next visual position for the cursor, in either the north or south direction.

**Parameters:**pos - the position to convert >= 0b - a bias value of either Position.Bias.Forward or Position.Bias.Backwarda - the allocated region to render intodirection - the direction from the current position that can be thought of as the arrow keys typically found on a keyboard; this may be one of the following:

* SwingConstants.NORTH
* SwingConstants.SOUTH

biasRet - an array containing the bias that was checked **Returns:**the location within the model that best represents the next north or south location **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if direction is invalid**See Also:**[getNextVisualPositionFrom(int, javax.swing.text.Position.Bias, java.awt.Shape, int, javax.swing.text.Position.Bias[])](http://docs.google.com/javax/swing/text/CompositeView.html#getNextVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D))

### getNextEastWestVisualPositionFrom

protected int **getNextEastWestVisualPositionFrom**(int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 int direction,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Returns the next visual position for the cursor, in either the east or west direction.

**Parameters:**pos - the position to convert >= 0b - a bias value of either Position.Bias.Forward or Position.Bias.Backwarda - the allocated region to render intodirection - the direction from the current position that can be thought of as the arrow keys typically found on a keyboard; this may be one of the following:

* SwingConstants.WEST
* SwingConstants.EAST

biasRet - an array containing the bias that was checked **Returns:**the location within the model that best represents the next west or east location **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if direction is invalid**See Also:**[getNextVisualPositionFrom(int, javax.swing.text.Position.Bias, java.awt.Shape, int, javax.swing.text.Position.Bias[])](http://docs.google.com/javax/swing/text/CompositeView.html#getNextVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D))

### flipEastAndWestAtEnds

protected boolean **flipEastAndWestAtEnds**(int position,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) bias)

Determines in which direction the next view lays. Consider the View at index n. Typically the Views are layed out from left to right, so that the View to the EAST will be at index n + 1, and the View to the WEST will be at index n - 1. In certain situations, such as with bidirectional text, it is possible that the View to EAST is not at index n + 1, but rather at index n - 1, or that the View to the WEST is not at index n - 1, but index n + 1. In this case this method would return true, indicating the Views are layed out in descending order.

This unconditionally returns false, subclasses should override this method if there is the possibility for laying Views in descending order.

**Parameters:**position - position into the modelbias - either Position.Bias.Forward or Position.Bias.Backward **Returns:**false

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CompositeView.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/text/ComponentView.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/DateFormatter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/CompositeView.html)    [**NO FRAMES**](http://docs.google.com/CompositeView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

[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).