| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BoxView.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/BadLocationException.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/Caret.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/BoxView.html)    [**NO FRAMES**](http://docs.google.com/BoxView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#lnxbz9) |

## **javax.swing.text**

Class BoxView

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

**All Implemented Interfaces:** [SwingConstants](http://docs.google.com/javax/swing/SwingConstants.html) **Direct Known Subclasses:** [BlockView](http://docs.google.com/javax/swing/text/html/BlockView.html), [FlowView](http://docs.google.com/javax/swing/text/FlowView.html), [TableView](http://docs.google.com/javax/swing/text/TableView.html), [TableView.TableCell](http://docs.google.com/javax/swing/text/TableView.TableCell.html), [TableView.TableRow](http://docs.google.com/javax/swing/text/TableView.TableRow.html), [WrappedPlainView](http://docs.google.com/javax/swing/text/WrappedPlainView.html), [ZoneView](http://docs.google.com/javax/swing/text/ZoneView.html)

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

A view that arranges its children into a box shape by tiling its children along an axis. The box is somewhat like that found in TeX where there is alignment of the children, flexibility of the children is considered, etc. This is a building block that might be useful to represent things like a collection of lines, paragraphs, lists, columns, pages, etc. The axis along which the children are tiled is considered the major axis. The orthoginal axis is the minor axis.

Layout for each axis is handled separately by the methods layoutMajorAxis and layoutMinorAxis. Subclasses can change the layout algorithm by reimplementing these methods. These methods will be called as necessary depending upon whether or not there is cached layout information and the cache is considered valid. These methods are typically called if the given size along the axis changes, or if layoutChanged is called to force an updated layout. The layoutChanged method invalidates cached layout information, if there is any. The requirements published to the parent view are calculated by the methods calculateMajorAxisRequirements and calculateMinorAxisRequirements. If the layout algorithm is changed, these methods will likely need to be reimplemented.

| **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** | |
| --- | --- |
| [**BoxView**](http://docs.google.com/javax/swing/text/BoxView.html#BoxView(javax.swing.text.Element,%20int))([Element](http://docs.google.com/javax/swing/text/Element.html) elem, int axis)            Constructs a BoxView. |

| **Method Summary** | |
| --- | --- |
| protected  void | [**baselineLayout**](http://docs.google.com/javax/swing/text/BoxView.html#baselineLayout(int,%20int,%20int%5B%5D,%20int%5B%5D))(int targetSpan, int axis, int[] offsets, int[] spans)            Computes the location and extent of each child view in this BoxView given the targetSpan, which is the width (or height) of the region we have to work with. |
| protected  [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) | [**baselineRequirements**](http://docs.google.com/javax/swing/text/BoxView.html#baselineRequirements(int,%20javax.swing.SizeRequirements))(int axis, [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) r)            Calculates the size requirements for this BoxView by examining the size of each child view. |
| protected  [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) | [**calculateMajorAxisRequirements**](http://docs.google.com/javax/swing/text/BoxView.html#calculateMajorAxisRequirements(int,%20javax.swing.SizeRequirements))(int axis, [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) r)            Calculates the size requirements for the major axis axis. |
| protected  [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) | [**calculateMinorAxisRequirements**](http://docs.google.com/javax/swing/text/BoxView.html#calculateMinorAxisRequirements(int,%20javax.swing.SizeRequirements))(int axis, [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) r)            Calculates the size requirements for the minor axis axis. |
| protected  void | [**childAllocation**](http://docs.google.com/javax/swing/text/BoxView.html#childAllocation(int,%20java.awt.Rectangle))(int index, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc)            Allocates a region for a child view. |
| protected  boolean | [**flipEastAndWestAtEnds**](http://docs.google.com/javax/swing/text/BoxView.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  void | [**forwardUpdate**](http://docs.google.com/javax/swing/text/BoxView.html#forwardUpdate(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))([DocumentEvent.ElementChange](http://docs.google.com/javax/swing/event/DocumentEvent.ElementChange.html) ec, [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) e, [Shape](http://docs.google.com/java/awt/Shape.html) a, [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)            Forwards the given DocumentEvent to the child views that need to be notified of the change to the model. |
| float | [**getAlignment**](http://docs.google.com/javax/swing/text/BoxView.html#getAlignment(int))(int axis)            Determines the desired alignment for this view along an axis. |
| int | [**getAxis**](http://docs.google.com/javax/swing/text/BoxView.html#getAxis())()            Fetches the tile axis property. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getChildAllocation**](http://docs.google.com/javax/swing/text/BoxView.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. |
| int | [**getHeight**](http://docs.google.com/javax/swing/text/BoxView.html#getHeight())()            Returns the current height of the box. |
| float | [**getMaximumSpan**](http://docs.google.com/javax/swing/text/BoxView.html#getMaximumSpan(int))(int axis)            Determines the maximum span for this view along an axis. |
| float | [**getMinimumSpan**](http://docs.google.com/javax/swing/text/BoxView.html#getMinimumSpan(int))(int axis)            Determines the minimum span for this view along an axis. |
| protected  int | [**getOffset**](http://docs.google.com/javax/swing/text/BoxView.html#getOffset(int,%20int))(int axis, int childIndex)            Fetches the offset of a particular child's current layout. |
| float | [**getPreferredSpan**](http://docs.google.com/javax/swing/text/BoxView.html#getPreferredSpan(int))(int axis)            Determines the preferred span for this view along an axis. |
| int | [**getResizeWeight**](http://docs.google.com/javax/swing/text/BoxView.html#getResizeWeight(int))(int axis)            Gets the resize weight. |
| protected  int | [**getSpan**](http://docs.google.com/javax/swing/text/BoxView.html#getSpan(int,%20int))(int axis, int childIndex)            Fetches the span of a particular childs current layout. |
| protected  [View](http://docs.google.com/javax/swing/text/View.html) | [**getViewAtPoint**](http://docs.google.com/javax/swing/text/BoxView.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. |
| int | [**getWidth**](http://docs.google.com/javax/swing/text/BoxView.html#getWidth())()            Returns the current width of the box. |
| protected  boolean | [**isAfter**](http://docs.google.com/javax/swing/text/BoxView.html#isAfter(int,%20int,%20java.awt.Rectangle))(int x, int y, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) innerAlloc)            Determines if a point falls after an allocated region. |
| protected  boolean | [**isAllocationValid**](http://docs.google.com/javax/swing/text/BoxView.html#isAllocationValid())()            Are the allocations for the children still valid? |
| protected  boolean | [**isBefore**](http://docs.google.com/javax/swing/text/BoxView.html#isBefore(int,%20int,%20java.awt.Rectangle))(int x, int y, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) innerAlloc)            Determines if a point falls before an allocated region. |
| protected  boolean | [**isLayoutValid**](http://docs.google.com/javax/swing/text/BoxView.html#isLayoutValid(int))(int axis)            Determines if the layout is valid along the given axis. |
| protected  void | [**layout**](http://docs.google.com/javax/swing/text/BoxView.html#layout(int,%20int))(int width, int height)            Perform layout on the box |
| void | [**layoutChanged**](http://docs.google.com/javax/swing/text/BoxView.html#layoutChanged(int))(int axis)            Invalidates the layout along an axis. |
| protected  void | [**layoutMajorAxis**](http://docs.google.com/javax/swing/text/BoxView.html#layoutMajorAxis(int,%20int,%20int%5B%5D,%20int%5B%5D))(int targetSpan, int axis, int[] offsets, int[] spans)            Performs layout for the major axis of the box (i.e. |
| protected  void | [**layoutMinorAxis**](http://docs.google.com/javax/swing/text/BoxView.html#layoutMinorAxis(int,%20int,%20int%5B%5D,%20int%5B%5D))(int targetSpan, int axis, int[] offsets, int[] spans)            Performs layout for the minor axis of the box (i.e. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**modelToView**](http://docs.google.com/javax/swing/text/BoxView.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 | [**paint**](http://docs.google.com/javax/swing/text/BoxView.html#paint(java.awt.Graphics,%20java.awt.Shape))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [Shape](http://docs.google.com/java/awt/Shape.html) allocation)            Renders the BoxView using the given rendering surface and area on that surface. |
| protected  void | [**paintChild**](http://docs.google.com/javax/swing/text/BoxView.html#paintChild(java.awt.Graphics,%20java.awt.Rectangle,%20int))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc, int index)            Paints a child. |
| void | [**preferenceChanged**](http://docs.google.com/javax/swing/text/BoxView.html#preferenceChanged(javax.swing.text.View,%20boolean,%20boolean))([View](http://docs.google.com/javax/swing/text/View.html) child, boolean width, boolean height)            This is called by a child to indicate its preferred span has changed. |
| void | [**replace**](http://docs.google.com/javax/swing/text/BoxView.html#replace(int,%20int,%20javax.swing.text.View%5B%5D))(int index, int length, [View](http://docs.google.com/javax/swing/text/View.html)[] elems)            Invalidates the layout and resizes the cache of requests/allocations. |
| void | [**setAxis**](http://docs.google.com/javax/swing/text/BoxView.html#setAxis(int))(int axis)            Sets the tile axis property. |
| void | [**setSize**](http://docs.google.com/javax/swing/text/BoxView.html#setSize(float,%20float))(float width, float height)            Sets the size of the view. |
| int | [**viewToModel**](http://docs.google.com/javax/swing/text/BoxView.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.**[**CompositeView**](http://docs.google.com/javax/swing/text/CompositeView.html) |
| --- |
| [getBottomInset](http://docs.google.com/javax/swing/text/CompositeView.html#getBottomInset()), [getInsideAllocation](http://docs.google.com/javax/swing/text/CompositeView.html#getInsideAllocation(java.awt.Shape)), [getLeftInset](http://docs.google.com/javax/swing/text/CompositeView.html#getLeftInset()), [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)), [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)), [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)), [getRightInset](http://docs.google.com/javax/swing/text/CompositeView.html#getRightInset()), [getTopInset](http://docs.google.com/javax/swing/text/CompositeView.html#getTopInset()), [getView](http://docs.google.com/javax/swing/text/CompositeView.html#getView(int)), [getViewAtPosition](http://docs.google.com/javax/swing/text/CompositeView.html#getViewAtPosition(int,%20java.awt.Rectangle)), [getViewCount](http://docs.google.com/javax/swing/text/CompositeView.html#getViewCount()), [getViewIndex](http://docs.google.com/javax/swing/text/CompositeView.html#getViewIndex(int,%20javax.swing.text.Position.Bias)), [getViewIndexAtPosition](http://docs.google.com/javax/swing/text/CompositeView.html#getViewIndexAtPosition(int)), [loadChildren](http://docs.google.com/javax/swing/text/CompositeView.html#loadChildren(javax.swing.text.ViewFactory)), [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)), [setInsets](http://docs.google.com/javax/swing/text/CompositeView.html#setInsets(short,%20short,%20short,%20short)), [setParagraphInsets](http://docs.google.com/javax/swing/text/CompositeView.html#setParagraphInsets(javax.swing.text.AttributeSet)), [setParent](http://docs.google.com/javax/swing/text/CompositeView.html#setParent(javax.swing.text.View)) |

| **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)), [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)), [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()), [getParent](http://docs.google.com/javax/swing/text/View.html#getParent()), [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)), [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)), [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** |
| --- |

### BoxView

public **BoxView**([Element](http://docs.google.com/javax/swing/text/Element.html) elem,  
 int axis)

Constructs a BoxView.

**Parameters:**elem - the element this view is responsible foraxis - either View.X\_AXIS or View.Y\_AXIS

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

### getAxis

public int **getAxis**()

Fetches the tile axis property. This is the axis along which the child views are tiled.

**Returns:**the major axis of the box, either View.X\_AXIS or View.Y\_AXIS**Since:** 1.3

### setAxis

public void **setAxis**(int axis)

Sets the tile axis property. This is the axis along which the child views are tiled.

**Parameters:**axis - either View.X\_AXIS or View.Y\_AXIS**Since:** 1.3

### layoutChanged

public void **layoutChanged**(int axis)

Invalidates the layout along an axis. This happens automatically if the preferences have changed for any of the child views. In some cases the layout may need to be recalculated when the preferences have not changed. The layout can be marked as invalid by calling this method. The layout will be updated the next time the setSize method is called on this view (typically in paint).

**Parameters:**axis - either View.X\_AXIS or View.Y\_AXIS**Since:** 1.3

### isLayoutValid

protected boolean **isLayoutValid**(int axis)

Determines if the layout is valid along the given axis.

**Parameters:**axis - either View.X\_AXIS or View.Y\_AXIS**Since:** 1.4

### paintChild

protected void **paintChild**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) alloc,  
 int index)

Paints a child. By default that is all it does, but a subclass can use this to paint things relative to the child.

**Parameters:**g - the graphics contextalloc - the allocated region to paint intoindex - the child index, >= 0 && < getViewCount()

### replace

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

Invalidates the layout and resizes the cache of requests/allocations. The child allocations can still be accessed for the old layout, but the new children will have an offset and span of 0.

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

### forwardUpdate

protected void **forwardUpdate**([DocumentEvent.ElementChange](http://docs.google.com/javax/swing/event/DocumentEvent.ElementChange.html) ec,  
 [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) e,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)

Forwards the given DocumentEvent to the child views that need to be notified of the change to the model. If a child changed its requirements and the allocation was valid prior to forwarding the portion of the box from the starting child to the end of the box will be repainted.

**Overrides:**[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)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**ec - changes to the element this view is responsible for (may be null if there were no changes)e - the change information from the associated documenta - the current allocation of the viewf - the factory to use to rebuild if the view has children**Since:** 1.3 **See Also:**[View.insertUpdate(javax.swing.event.DocumentEvent, java.awt.Shape, javax.swing.text.ViewFactory)](http://docs.google.com/javax/swing/text/View.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [View.removeUpdate(javax.swing.event.DocumentEvent, java.awt.Shape, javax.swing.text.ViewFactory)](http://docs.google.com/javax/swing/text/View.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [View.changedUpdate(javax.swing.event.DocumentEvent, java.awt.Shape, javax.swing.text.ViewFactory)](http://docs.google.com/javax/swing/text/View.html#changedUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))

### preferenceChanged

public void **preferenceChanged**([View](http://docs.google.com/javax/swing/text/View.html) child,  
 boolean width,  
 boolean height)

This is called by a child to indicate its preferred span has changed. This is implemented to throw away cached layout information so that new calculations will be done the next time the children need an allocation.

**Overrides:**[preferenceChanged](http://docs.google.com/javax/swing/text/View.html#preferenceChanged(javax.swing.text.View,%20boolean,%20boolean)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**child - the child viewwidth - true if the width preference should changeheight - true if the height preference should change**See Also:**[JComponent.revalidate()](http://docs.google.com/javax/swing/JComponent.html#revalidate())

### getResizeWeight

public int **getResizeWeight**(int axis)

Gets the resize weight. A value of 0 or less is not resizable.

**Overrides:**[getResizeWeight](http://docs.google.com/javax/swing/text/View.html#getResizeWeight(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the weight **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis

### setSize

public void **setSize**(float width,  
 float height)

Sets the size of the view. This should cause layout of the view if the view caches any layout information. This is implemented to call the layout method with the sizes inside of the insets.

**Overrides:**[setSize](http://docs.google.com/javax/swing/text/View.html#setSize(float,%20float)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**width - the width >= 0height - the height >= 0

### paint

public void **paint**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [Shape](http://docs.google.com/java/awt/Shape.html) allocation)

Renders the BoxView using the given rendering surface and area on that surface. Only the children that intersect the clip bounds of the given Graphics will be rendered.

**Specified by:**[paint](http://docs.google.com/javax/swing/text/View.html#paint(java.awt.Graphics,%20java.awt.Shape)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**g - the rendering surface to useallocation - the allocated region to render into**See Also:**[View.paint(java.awt.Graphics, java.awt.Shape)](http://docs.google.com/javax/swing/text/View.html#paint(java.awt.Graphics,%20java.awt.Shape))

### 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. This enables finding out where various views are located. This is implemented to return null if the layout is invalid, otherwise the superclass behavior is executed.

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

### 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. This makes sure the allocation is valid before calling the superclass.

**Overrides:**[modelToView](http://docs.google.com/javax/swing/text/CompositeView.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.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))

### 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.

**Overrides:**[viewToModel](http://docs.google.com/javax/swing/text/CompositeView.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.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))

### getAlignment

public float **getAlignment**(int axis)

Determines the desired alignment for this view along an axis. This is implemented to give the total alignment needed to position the children with the alignment points lined up along the axis orthoginal to the axis that is being tiled. The axis being tiled will request to be centered (i.e. 0.5f).

**Overrides:**[getAlignment](http://docs.google.com/javax/swing/text/View.html#getAlignment(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the desired alignment >= 0.0f && <= 1.0f; this should be a value between 0.0 and 1.0 where 0 indicates alignment at the origin and 1.0 indicates alignment to the full span away from the origin; an alignment of 0.5 would be the center of the view **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis

### getPreferredSpan

public float **getPreferredSpan**(int axis)

Determines the preferred span for this view along an axis.

**Specified by:**[getPreferredSpan](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into >= 0; typically the view is told to render into the span that is returned, although there is no guarantee; the parent may choose to resize or break the view **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis type**See Also:**[View.getPreferredSpan(int)](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int))

### getMinimumSpan

public float **getMinimumSpan**(int axis)

Determines the minimum span for this view along an axis.

**Overrides:**[getMinimumSpan](http://docs.google.com/javax/swing/text/View.html#getMinimumSpan(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into >= 0; typically the view is told to render into the span that is returned, although there is no guarantee; the parent may choose to resize or break the view **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis type**See Also:**[View.getPreferredSpan(int)](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int))

### getMaximumSpan

public float **getMaximumSpan**(int axis)

Determines the maximum span for this view along an axis.

**Overrides:**[getMaximumSpan](http://docs.google.com/javax/swing/text/View.html#getMaximumSpan(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into >= 0; typically the view is told to render into the span that is returned, although there is no guarantee; the parent may choose to resize or break the view **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis type**See Also:**[View.getPreferredSpan(int)](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int))

### isAllocationValid

protected boolean **isAllocationValid**()

Are the allocations for the children still valid?

**Returns:**true if allocations still valid

### isBefore

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

Determines if a point falls before an allocated region.

**Specified by:**[isBefore](http://docs.google.com/javax/swing/text/CompositeView.html#isBefore(int,%20int,%20java.awt.Rectangle)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.html) **Parameters:**x - the X coordinate >= 0y - the Y coordinate >= 0innerAlloc - the allocated region; this is the area inside of the insets **Returns:**true if the point lies before the region else false

### isAfter

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

Determines if a point falls after an allocated region.

**Specified by:**[isAfter](http://docs.google.com/javax/swing/text/CompositeView.html#isAfter(int,%20int,%20java.awt.Rectangle)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.html) **Parameters:**x - the X coordinate >= 0y - the Y coordinate >= 0innerAlloc - the allocated region; this is the area inside of the insets **Returns:**true if the point lies after the region else false

### getViewAtPoint

protected [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.

**Specified by:**[getViewAtPoint](http://docs.google.com/javax/swing/text/CompositeView.html#getViewAtPoint(int,%20int,%20java.awt.Rectangle)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.html) **Parameters:**x - the X coordinate >= 0y - the Y coordinate >= 0alloc - the parents inner allocation on entry, which should be changed to the childs allocation on exit **Returns:**the view

### childAllocation

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

Allocates a region for a child view.

**Specified by:**[childAllocation](http://docs.google.com/javax/swing/text/CompositeView.html#childAllocation(int,%20java.awt.Rectangle)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.html) **Parameters:**index - the index of the child view to allocate, >= 0 && < getViewCount()alloc - the allocated region

### layout

protected void **layout**(int width,  
 int height)

Perform layout on the box

**Parameters:**width - the width (inside of the insets) >= 0height - the height (inside of the insets) >= 0

### getWidth

public int **getWidth**()

Returns the current width of the box. This is the width that it was last allocated.

**Returns:**the current width of the box

### getHeight

public int **getHeight**()

Returns the current height of the box. This is the height that it was last allocated.

**Returns:**the current height of the box

### layoutMajorAxis

protected void **layoutMajorAxis**(int targetSpan,  
 int axis,  
 int[] offsets,  
 int[] spans)

Performs layout for the major axis of the box (i.e. the axis that it represents). The results of the layout (the offset and span for each children) are placed in the given arrays which represent the allocations to the children along the major axis.

**Parameters:**targetSpan - the total span given to the view, which would be used to layout the childrenaxis - the axis being layed outoffsets - the offsets from the origin of the view for each of the child views; this is a return value and is filled in by the implementation of this methodspans - the span of each child view; this is a return value and is filled in by the implementation of this method

### layoutMinorAxis

protected void **layoutMinorAxis**(int targetSpan,  
 int axis,  
 int[] offsets,  
 int[] spans)

Performs layout for the minor axis of the box (i.e. the axis orthoginal to the axis that it represents). The results of the layout (the offset and span for each children) are placed in the given arrays which represent the allocations to the children along the minor axis.

**Parameters:**targetSpan - the total span given to the view, which would be used to layout the childrenaxis - the axis being layed outoffsets - the offsets from the origin of the view for each of the child views; this is a return value and is filled in by the implementation of this methodspans - the span of each child view; this is a return value and is filled in by the implementation of this method

### calculateMajorAxisRequirements

protected [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) **calculateMajorAxisRequirements**(int axis,  
 [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) r)

Calculates the size requirements for the major axis axis.

**Parameters:**axis - the axis being studiedr - the SizeRequirements object; if null one will be created **Returns:**the newly initialized SizeRequirements object**See Also:**[SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html)

### calculateMinorAxisRequirements

protected [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) **calculateMinorAxisRequirements**(int axis,  
 [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) r)

Calculates the size requirements for the minor axis axis.

**Parameters:**axis - the axis being studiedr - the SizeRequirements object; if null one will be created **Returns:**the newly initialized SizeRequirements object**See Also:**[SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html)

### baselineLayout

protected void **baselineLayout**(int targetSpan,  
 int axis,  
 int[] offsets,  
 int[] spans)

Computes the location and extent of each child view in this BoxView given the targetSpan, which is the width (or height) of the region we have to work with.

**Parameters:**targetSpan - the total span given to the view, which would be used to layout the childrenaxis - the axis being studied, either View.X\_AXIS or View.Y\_AXISoffsets - an empty array filled by this method with values specifying the location of each child viewspans - an empty array filled by this method with values specifying the extent of each child view

### baselineRequirements

protected [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) **baselineRequirements**(int axis,  
 [SizeRequirements](http://docs.google.com/javax/swing/SizeRequirements.html) r)

Calculates the size requirements for this BoxView by examining the size of each child view.

**Parameters:**axis - the axis being studiedr - the SizeRequirements object; if null one will be created **Returns:**the newly initialized SizeRequirements object

### getOffset

protected int **getOffset**(int axis,  
 int childIndex)

Fetches the offset of a particular child's current layout.

**Parameters:**axis - the axis being studiedchildIndex - the index of the requested child **Returns:**the offset (location) for the specified child

### getSpan

protected int **getSpan**(int axis,  
 int childIndex)

Fetches the span of a particular childs current layout.

**Parameters:**axis - the axis being studiedchildIndex - the index of the requested child **Returns:**the span (width or height) of the specified child

### 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. Otherwise the method would return false indicating the Views are layed out in ascending order.

If the receiver is laying its Views along the Y\_AXIS, this will will return the value from invoking the same method on the View responsible for rendering position and bias. Otherwise this will return false.

**Overrides:**[flipEastAndWestAtEnds](http://docs.google.com/javax/swing/text/CompositeView.html#flipEastAndWestAtEnds(int,%20javax.swing.text.Position.Bias)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.html) **Parameters:**position - position into the modelbias - either Position.Bias.Forward or Position.Bias.Backward **Returns:**true if the Views surrounding the View responding for rendering position and bias are layed out in descending order; otherwise 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/BoxView.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/BadLocationException.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/Caret.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/BoxView.html)    [**NO FRAMES**](http://docs.google.com/BoxView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#lnxbz9) |

[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).