| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/WrappedPlainView.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/ViewFactory.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/ZoneView.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/WrappedPlainView.html)    [**NO FRAMES**](http://docs.google.com/WrappedPlainView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#26in1rg) | [METHOD](#1ksv4uv) |

## **javax.swing.text**

Class WrappedPlainView

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

**All Implemented Interfaces:** [SwingConstants](http://docs.google.com/javax/swing/SwingConstants.html), [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html)

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

View of plain text (text with only one font and color) that does line-wrapping. This view expects that its associated element has child elements that represent the lines it should be wrapping. It is implemented as a vertical box that contains logical line views. The logical line views are nested classes that render the logical line as multiple physical line if the logical line is too wide to fit within the allocation. The line views draw upon the outer class for its state to reduce their memory requirements.

The line views do all of their rendering through the drawLine method which in turn does all of its rendering through the drawSelectedText and drawUnselectedText methods. This enables subclasses to easily specialize the rendering without concern for the layout aspects.

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

| **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** | |
| --- | --- |
| [**WrappedPlainView**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#WrappedPlainView(javax.swing.text.Element))([Element](http://docs.google.com/javax/swing/text/Element.html) elem)            Creates a new WrappedPlainView. |
| [**WrappedPlainView**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#WrappedPlainView(javax.swing.text.Element,%20boolean))([Element](http://docs.google.com/javax/swing/text/Element.html) elem, boolean wordWrap)            Creates a new WrappedPlainView. |

| **Method Summary** | |
| --- | --- |
| protected  int | [**calculateBreakPosition**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#calculateBreakPosition(int,%20int))(int p0, int p1)            This is called by the nested wrapped line views to determine the break location. |
| void | [**changedUpdate**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#changedUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))([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)            Gives notification from the document that attributes were changed in a location that this view is responsible for. |
| protected  void | [**drawLine**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#drawLine(int,%20int,%20java.awt.Graphics,%20int,%20int))(int p0, int p1, [Graphics](http://docs.google.com/java/awt/Graphics.html) g, int x, int y)            Renders a line of text, suppressing whitespace at the end and expanding any tabs. |
| protected  int | [**drawSelectedText**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#drawSelectedText(java.awt.Graphics,%20int,%20int,%20int,%20int))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, int x, int y, int p0, int p1)            Renders the given range in the model as selected text. |
| protected  int | [**drawUnselectedText**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#drawUnselectedText(java.awt.Graphics,%20int,%20int,%20int,%20int))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, int x, int y, int p0, int p1)            Renders the given range in the model as normal unselected text. |
| protected  [Segment](http://docs.google.com/javax/swing/text/Segment.html) | [**getLineBuffer**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#getLineBuffer())()            Gives access to a buffer that can be used to fetch text from the associated document. |
| float | [**getMaximumSpan**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#getMaximumSpan(int))(int axis)            Determines the maximum span for this view along an axis. |
| float | [**getMinimumSpan**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#getMinimumSpan(int))(int axis)            Determines the minimum span for this view along an axis. |
| float | [**getPreferredSpan**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#getPreferredSpan(int))(int axis)            Determines the preferred span for this view along an axis. |
| protected  int | [**getTabSize**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#getTabSize())()            Returns the tab size set for the document, defaulting to 8. |
| void | [**insertUpdate**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))([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)            Gives notification that something was inserted into the document in a location that this view is responsible for. |
| protected  void | [**loadChildren**](http://docs.google.com/javax/swing/text/WrappedPlainView.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. |
| float | [**nextTabStop**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#nextTabStop(float,%20int))(float x, int tabOffset)            Returns the next tab stop position after a given reference position. |
| void | [**paint**](http://docs.google.com/javax/swing/text/WrappedPlainView.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) a)            Renders using the given rendering surface and area on that surface. |
| void | [**removeUpdate**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))([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)            Gives notification that something was removed from the document in a location that this view is responsible for. |
| void | [**setSize**](http://docs.google.com/javax/swing/text/WrappedPlainView.html#setSize(float,%20float))(float width, float height)            Sets the size of the view. |

| **Methods inherited from class javax.swing.text.**[**BoxView**](http://docs.google.com/javax/swing/text/BoxView.html) |
| --- |
| [baselineLayout](http://docs.google.com/javax/swing/text/BoxView.html#baselineLayout(int,%20int,%20int%5B%5D,%20int%5B%5D)), [baselineRequirements](http://docs.google.com/javax/swing/text/BoxView.html#baselineRequirements(int,%20javax.swing.SizeRequirements)), [calculateMajorAxisRequirements](http://docs.google.com/javax/swing/text/BoxView.html#calculateMajorAxisRequirements(int,%20javax.swing.SizeRequirements)), [calculateMinorAxisRequirements](http://docs.google.com/javax/swing/text/BoxView.html#calculateMinorAxisRequirements(int,%20javax.swing.SizeRequirements)), [childAllocation](http://docs.google.com/javax/swing/text/BoxView.html#childAllocation(int,%20java.awt.Rectangle)), [flipEastAndWestAtEnds](http://docs.google.com/javax/swing/text/BoxView.html#flipEastAndWestAtEnds(int,%20javax.swing.text.Position.Bias)), [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)), [getAlignment](http://docs.google.com/javax/swing/text/BoxView.html#getAlignment(int)), [getAxis](http://docs.google.com/javax/swing/text/BoxView.html#getAxis()), [getChildAllocation](http://docs.google.com/javax/swing/text/BoxView.html#getChildAllocation(int,%20java.awt.Shape)), [getHeight](http://docs.google.com/javax/swing/text/BoxView.html#getHeight()), [getOffset](http://docs.google.com/javax/swing/text/BoxView.html#getOffset(int,%20int)), [getResizeWeight](http://docs.google.com/javax/swing/text/BoxView.html#getResizeWeight(int)), [getSpan](http://docs.google.com/javax/swing/text/BoxView.html#getSpan(int,%20int)), [getViewAtPoint](http://docs.google.com/javax/swing/text/BoxView.html#getViewAtPoint(int,%20int,%20java.awt.Rectangle)), [getWidth](http://docs.google.com/javax/swing/text/BoxView.html#getWidth()), [isAfter](http://docs.google.com/javax/swing/text/BoxView.html#isAfter(int,%20int,%20java.awt.Rectangle)), [isAllocationValid](http://docs.google.com/javax/swing/text/BoxView.html#isAllocationValid()), [isBefore](http://docs.google.com/javax/swing/text/BoxView.html#isBefore(int,%20int,%20java.awt.Rectangle)), [isLayoutValid](http://docs.google.com/javax/swing/text/BoxView.html#isLayoutValid(int)), [layout](http://docs.google.com/javax/swing/text/BoxView.html#layout(int,%20int)), [layoutChanged](http://docs.google.com/javax/swing/text/BoxView.html#layoutChanged(int)), [layoutMajorAxis](http://docs.google.com/javax/swing/text/BoxView.html#layoutMajorAxis(int,%20int,%20int%5B%5D,%20int%5B%5D)), [layoutMinorAxis](http://docs.google.com/javax/swing/text/BoxView.html#layoutMinorAxis(int,%20int,%20int%5B%5D,%20int%5B%5D)), [modelToView](http://docs.google.com/javax/swing/text/BoxView.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias)), [paintChild](http://docs.google.com/javax/swing/text/BoxView.html#paintChild(java.awt.Graphics,%20java.awt.Rectangle,%20int)), [preferenceChanged](http://docs.google.com/javax/swing/text/BoxView.html#preferenceChanged(javax.swing.text.View,%20boolean,%20boolean)), [replace](http://docs.google.com/javax/swing/text/BoxView.html#replace(int,%20int,%20javax.swing.text.View%5B%5D)), [setAxis](http://docs.google.com/javax/swing/text/BoxView.html#setAxis(int)), [viewToModel](http://docs.google.com/javax/swing/text/BoxView.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D)) |

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

### WrappedPlainView

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

Creates a new WrappedPlainView. Lines will be wrapped on character boundaries.

**Parameters:**elem - the element underlying the view

### WrappedPlainView

public **WrappedPlainView**([Element](http://docs.google.com/javax/swing/text/Element.html) elem,  
 boolean wordWrap)

Creates a new WrappedPlainView. Lines can be wrapped on either character or word boundaries depending upon the setting of the wordWrap parameter.

**Parameters:**elem - the element underlying the viewwordWrap - should lines be wrapped on word boundaries?

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

### getTabSize

protected int **getTabSize**()

Returns the tab size set for the document, defaulting to 8.

**Returns:**the tab size

### drawLine

protected void **drawLine**(int p0,  
 int p1,  
 [Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 int x,  
 int y)

Renders a line of text, suppressing whitespace at the end and expanding any tabs. This is implemented to make calls to the methods drawUnselectedText and drawSelectedText so that the way selected and unselected text are rendered can be customized.

**Parameters:**p0 - the starting document location to use >= 0p1 - the ending document location to use >= p1g - the graphics contextx - the starting X position >= 0y - the starting Y position >= 0**See Also:**[drawUnselectedText(java.awt.Graphics, int, int, int, int)](http://docs.google.com/javax/swing/text/WrappedPlainView.html#drawUnselectedText(java.awt.Graphics,%20int,%20int,%20int,%20int)), [drawSelectedText(java.awt.Graphics, int, int, int, int)](http://docs.google.com/javax/swing/text/WrappedPlainView.html#drawSelectedText(java.awt.Graphics,%20int,%20int,%20int,%20int))

### drawUnselectedText

protected int **drawUnselectedText**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 int x,  
 int y,  
 int p0,  
 int p1)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Renders the given range in the model as normal unselected text.

**Parameters:**g - the graphics contextx - the starting X coordinate >= 0y - the starting Y coordinate >= 0p0 - the beginning position in the model >= 0p1 - the ending position in the model >= p0 **Returns:**the X location of the end of the range >= 0 **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the range is invalid

### drawSelectedText

protected int **drawSelectedText**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 int x,  
 int y,  
 int p0,  
 int p1)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Renders the given range in the model as selected text. This is implemented to render the text in the color specified in the hosting component. It assumes the highlighter will render the selected background.

**Parameters:**g - the graphics contextx - the starting X coordinate >= 0y - the starting Y coordinate >= 0p0 - the beginning position in the model >= 0p1 - the ending position in the model >= p0 **Returns:**the location of the end of the range. **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the range is invalid

### getLineBuffer

protected final [Segment](http://docs.google.com/javax/swing/text/Segment.html) **getLineBuffer**()

Gives access to a buffer that can be used to fetch text from the associated document.

**Returns:**the buffer

### calculateBreakPosition

protected int **calculateBreakPosition**(int p0,  
 int p1)

This is called by the nested wrapped line views to determine the break location. This can be reimplemented to alter the breaking behavior. It will either break at word or character boundaries depending upon the break argument given at construction.

### 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 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.

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

### nextTabStop

public float **nextTabStop**(float x,  
 int tabOffset)

Returns the next tab stop position after a given reference position. This implementation does not support things like centering so it ignores the tabOffset argument.

**Specified by:**[nextTabStop](http://docs.google.com/javax/swing/text/TabExpander.html#nextTabStop(float,%20int)) in interface [TabExpander](http://docs.google.com/javax/swing/text/TabExpander.html) **Parameters:**x - the current position >= 0tabOffset - the position within the text stream that the tab occurred at >= 0. **Returns:**the tab stop, measured in points >= 0

### paint

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

Renders using the given rendering surface and area on that surface. This is implemented to stash the selection positions, selection colors, and font metrics for the nested lines to use.

**Overrides:**[paint](http://docs.google.com/javax/swing/text/BoxView.html#paint(java.awt.Graphics,%20java.awt.Shape)) in class [BoxView](http://docs.google.com/javax/swing/text/BoxView.html) **Parameters:**g - the rendering surface to usea - 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))

### setSize

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

Sets the size of the view. This should cause layout of the view along the given axis, if it has any layout duties.

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

### getPreferredSpan

public float **getPreferredSpan**(int axis)

Determines the preferred span for this view along an axis. This is implemented to provide the superclass behavior after first making sure that the current font metrics are cached (for the nested lines which use the metrics to determine the height of the potentially wrapped lines).

**Overrides:**[getPreferredSpan](http://docs.google.com/javax/swing/text/BoxView.html#getPreferredSpan(int)) in class [BoxView](http://docs.google.com/javax/swing/text/BoxView.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into. 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.**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. This is implemented to provide the superclass behavior after first making sure that the current font metrics are cached (for the nested lines which use the metrics to determine the height of the potentially wrapped lines).

**Overrides:**[getMinimumSpan](http://docs.google.com/javax/swing/text/BoxView.html#getMinimumSpan(int)) in class [BoxView](http://docs.google.com/javax/swing/text/BoxView.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into. 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.**See Also:**[View.getMinimumSpan(int)](http://docs.google.com/javax/swing/text/View.html#getMinimumSpan(int))

### getMaximumSpan

public float **getMaximumSpan**(int axis)

Determines the maximum span for this view along an axis. This is implemented to provide the superclass behavior after first making sure that the current font metrics are cached (for the nested lines which use the metrics to determine the height of the potentially wrapped lines).

**Overrides:**[getMaximumSpan](http://docs.google.com/javax/swing/text/BoxView.html#getMaximumSpan(int)) in class [BoxView](http://docs.google.com/javax/swing/text/BoxView.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into. 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.**See Also:**[View.getMaximumSpan(int)](http://docs.google.com/javax/swing/text/View.html#getMaximumSpan(int))

### insertUpdate

public void **insertUpdate**([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)

Gives notification that something was inserted into the document in a location that this view is responsible for. This is implemented to simply update the children.

**Overrides:**[insertUpdate](http://docs.google.com/javax/swing/text/View.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**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**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))

### removeUpdate

public void **removeUpdate**([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)

Gives notification that something was removed from the document in a location that this view is responsible for. This is implemented to simply update the children.

**Overrides:**[removeUpdate](http://docs.google.com/javax/swing/text/View.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**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**See Also:**[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))

### changedUpdate

public void **changedUpdate**([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)

Gives notification from the document that attributes were changed in a location that this view is responsible for.

**Overrides:**[changedUpdate](http://docs.google.com/javax/swing/text/View.html#changedUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**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**See Also:**[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))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/WrappedPlainView.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/ViewFactory.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/ZoneView.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/WrappedPlainView.html)    [**NO FRAMES**](http://docs.google.com/WrappedPlainView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#26in1rg) | [METHOD](#1ksv4uv) |

[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).