| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractWriter.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/AbstractDocument.LeafElement.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/AsyncBoxView.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/AbstractWriter.html)    [**NO FRAMES**](http://docs.google.com/AbstractWriter.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#35nkun2) |

## **javax.swing.text**

Class AbstractWriter

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

**Direct Known Subclasses:** [HTMLWriter](http://docs.google.com/javax/swing/text/html/HTMLWriter.html), [MinimalHTMLWriter](http://docs.google.com/javax/swing/text/html/MinimalHTMLWriter.html)

public abstract class **AbstractWriter**extends [Object](http://docs.google.com/java/lang/Object.html)

AbstractWriter is an abstract class that actually does the work of writing out the element tree including the attributes. In terms of how much is written out per line, the writer defaults to 100. But this value can be set by subclasses.

| **Field Summary** | |
| --- | --- |
| protected static char | [**NEWLINE**](http://docs.google.com/javax/swing/text/AbstractWriter.html#NEWLINE)            How the text packages models newlines. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**AbstractWriter**](http://docs.google.com/javax/swing/text/AbstractWriter.html#AbstractWriter(java.io.Writer,%20javax.swing.text.Document))([Writer](http://docs.google.com/java/io/Writer.html) w, [Document](http://docs.google.com/javax/swing/text/Document.html) doc)            Creates a new AbstractWriter. |
| protected | [**AbstractWriter**](http://docs.google.com/javax/swing/text/AbstractWriter.html#AbstractWriter(java.io.Writer,%20javax.swing.text.Document,%20int,%20int))([Writer](http://docs.google.com/java/io/Writer.html) w, [Document](http://docs.google.com/javax/swing/text/Document.html) doc, int pos, int len)            Creates a new AbstractWriter. |
| protected | [**AbstractWriter**](http://docs.google.com/javax/swing/text/AbstractWriter.html#AbstractWriter(java.io.Writer,%20javax.swing.text.Element))([Writer](http://docs.google.com/java/io/Writer.html) w, [Element](http://docs.google.com/javax/swing/text/Element.html) root)            Creates a new AbstractWriter. |
| protected | [**AbstractWriter**](http://docs.google.com/javax/swing/text/AbstractWriter.html#AbstractWriter(java.io.Writer,%20javax.swing.text.Element,%20int,%20int))([Writer](http://docs.google.com/java/io/Writer.html) w, [Element](http://docs.google.com/javax/swing/text/Element.html) root, int pos, int len)            Creates a new AbstractWriter. |

| **Method Summary** | |
| --- | --- |
| protected  void | [**decrIndent**](http://docs.google.com/javax/swing/text/AbstractWriter.html#decrIndent())()            Decrements the indent level. |
| protected  boolean | [**getCanWrapLines**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getCanWrapLines())()            Returns whether or not the lines can be wrapped. |
| protected  int | [**getCurrentLineLength**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getCurrentLineLength())()            Returns the current line length. |
| protected  [Document](http://docs.google.com/javax/swing/text/Document.html) | [**getDocument**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getDocument())()            Fetches the document. |
| protected  [ElementIterator](http://docs.google.com/javax/swing/text/ElementIterator.html) | [**getElementIterator**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getElementIterator())()            Fetches the ElementIterator. |
| int | [**getEndOffset**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getEndOffset())()            Returns the last offset to be output. |
| protected  int | [**getIndentLevel**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getIndentLevel())()            Returns the current indentation level. |
| protected  int | [**getIndentSpace**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getIndentSpace())()            Returns the amount of space to indent. |
| protected  int | [**getLineLength**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getLineLength())()            Returns the maximum line length. |
| [String](http://docs.google.com/java/lang/String.html) | [**getLineSeparator**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getLineSeparator())()            Returns the string used to represent newlines. |
| int | [**getStartOffset**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getStartOffset())()            Returns the first offset to be output. |
| protected  [String](http://docs.google.com/java/lang/String.html) | [**getText**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getText(javax.swing.text.Element))([Element](http://docs.google.com/javax/swing/text/Element.html) elem)            Returns the text associated with the element. |
| protected  [Writer](http://docs.google.com/java/io/Writer.html) | [**getWriter**](http://docs.google.com/javax/swing/text/AbstractWriter.html#getWriter())()            Returns the Writer that is used to output the content. |
| protected  void | [**incrIndent**](http://docs.google.com/javax/swing/text/AbstractWriter.html#incrIndent())()            Increments the indent level. |
| protected  void | [**indent**](http://docs.google.com/javax/swing/text/AbstractWriter.html#indent())()            Does indentation. |
| protected  boolean | [**inRange**](http://docs.google.com/javax/swing/text/AbstractWriter.html#inRange(javax.swing.text.Element))([Element](http://docs.google.com/javax/swing/text/Element.html) next)            This method determines whether the current element is in the range specified. |
| protected  boolean | [**isLineEmpty**](http://docs.google.com/javax/swing/text/AbstractWriter.html#isLineEmpty())()            Returns true if the current line should be considered empty. |
| protected  void | [**output**](http://docs.google.com/javax/swing/text/AbstractWriter.html#output(char%5B%5D,%20int,%20int))(char[] content, int start, int length)            The last stop in writing out content. |
| protected  void | [**setCanWrapLines**](http://docs.google.com/javax/swing/text/AbstractWriter.html#setCanWrapLines(boolean))(boolean newValue)            Sets whether or not lines can be wrapped. |
| protected  void | [**setCurrentLineLength**](http://docs.google.com/javax/swing/text/AbstractWriter.html#setCurrentLineLength(int))(int length)            Sets the current line length. |
| protected  void | [**setIndentSpace**](http://docs.google.com/javax/swing/text/AbstractWriter.html#setIndentSpace(int))(int space)            Enables subclasses to specify how many spaces an indent maps to. |
| protected  void | [**setLineLength**](http://docs.google.com/javax/swing/text/AbstractWriter.html#setLineLength(int))(int l)            Enables subclasses to set the number of characters they want written per line. |
| void | [**setLineSeparator**](http://docs.google.com/javax/swing/text/AbstractWriter.html#setLineSeparator(java.lang.String))([String](http://docs.google.com/java/lang/String.html) value)            Sets the String used to reprsent newlines. |
| protected  void | [**text**](http://docs.google.com/javax/swing/text/AbstractWriter.html#text(javax.swing.text.Element))([Element](http://docs.google.com/javax/swing/text/Element.html) elem)            Writes out text. |
| protected abstract  void | [**write**](http://docs.google.com/javax/swing/text/AbstractWriter.html#write())()            This abstract method needs to be implemented by subclasses. |
| protected  void | [**write**](http://docs.google.com/javax/swing/text/AbstractWriter.html#write(char))(char ch)            Writes out a character. |
| protected  void | [**write**](http://docs.google.com/javax/swing/text/AbstractWriter.html#write(char%5B%5D,%20int,%20int))(char[] chars, int startIndex, int length)            All write methods call into this one. |
| protected  void | [**write**](http://docs.google.com/javax/swing/text/AbstractWriter.html#write(java.lang.String))([String](http://docs.google.com/java/lang/String.html) content)            Writes out a string. |
| protected  void | [**writeAttributes**](http://docs.google.com/javax/swing/text/AbstractWriter.html#writeAttributes(javax.swing.text.AttributeSet))([AttributeSet](http://docs.google.com/javax/swing/text/AttributeSet.html) attr)            Writes out the set of attributes as " =" pairs. |
| protected  void | [**writeLineSeparator**](http://docs.google.com/javax/swing/text/AbstractWriter.html#writeLineSeparator())()            Writes the line separator. |

| **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)) |

| **Field Detail** |
| --- |

### NEWLINE

protected static final char **NEWLINE**

How the text packages models newlines.

**See Also:**[getLineSeparator()](http://docs.google.com/javax/swing/text/AbstractWriter.html#getLineSeparator()), [Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.text.AbstractWriter.NEWLINE)

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

### AbstractWriter

protected **AbstractWriter**([Writer](http://docs.google.com/java/io/Writer.html) w,  
 [Document](http://docs.google.com/javax/swing/text/Document.html) doc)

Creates a new AbstractWriter. Initializes the ElementIterator with the default root of the document.

**Parameters:**w - a Writer.doc - a Document

### AbstractWriter

protected **AbstractWriter**([Writer](http://docs.google.com/java/io/Writer.html) w,  
 [Document](http://docs.google.com/javax/swing/text/Document.html) doc,  
 int pos,  
 int len)

Creates a new AbstractWriter. Initializes the ElementIterator with the element passed in.

**Parameters:**w - a Writerdoc - an Elementpos - The location in the document to fetch the content.len - The amount to write out.

### AbstractWriter

protected **AbstractWriter**([Writer](http://docs.google.com/java/io/Writer.html) w,  
 [Element](http://docs.google.com/javax/swing/text/Element.html) root)

Creates a new AbstractWriter. Initializes the ElementIterator with the element passed in.

**Parameters:**w - a Writerroot - an Element

### AbstractWriter

protected **AbstractWriter**([Writer](http://docs.google.com/java/io/Writer.html) w,  
 [Element](http://docs.google.com/javax/swing/text/Element.html) root,  
 int pos,  
 int len)

Creates a new AbstractWriter. Initializes the ElementIterator with the element passed in.

**Parameters:**w - a Writerroot - an Elementpos - The location in the document to fetch the content.len - The amount to write out.

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

### getStartOffset

public int **getStartOffset**()

Returns the first offset to be output.

**Since:** 1.3

### getEndOffset

public int **getEndOffset**()

Returns the last offset to be output.

**Since:** 1.3

### getElementIterator

protected [ElementIterator](http://docs.google.com/javax/swing/text/ElementIterator.html) **getElementIterator**()

Fetches the ElementIterator.

**Returns:**the ElementIterator.

### getWriter

protected [Writer](http://docs.google.com/java/io/Writer.html) **getWriter**()

Returns the Writer that is used to output the content.

**Since:** 1.3

### getDocument

protected [Document](http://docs.google.com/javax/swing/text/Document.html) **getDocument**()

Fetches the document.

**Returns:**the Document.

### inRange

protected boolean **inRange**([Element](http://docs.google.com/javax/swing/text/Element.html) next)

This method determines whether the current element is in the range specified. When no range is specified, the range is initialized to be the entire document. inRange() returns true if the range specified intersects with the element's range.

**Parameters:**next - an Element. **Returns:**boolean that indicates whether the element is in the range.

### write

protected abstract void **write**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

This abstract method needs to be implemented by subclasses. Its responsibility is to iterate over the elements and use the write() methods to generate output in the desired format.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

### getText

protected [String](http://docs.google.com/java/lang/String.html) **getText**([Element](http://docs.google.com/javax/swing/text/Element.html) elem)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Returns the text associated with the element. The assumption here is that the element is a leaf element. Throws a BadLocationException when encountered.

**Parameters:**elem - an Element **Returns:**the text as a String **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if pos represents an invalid location within the document

### text

protected void **text**([Element](http://docs.google.com/javax/swing/text/Element.html) elem)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html),  
 [IOException](http://docs.google.com/java/io/IOException.html)

Writes out text. If a range is specified when the constructor is invoked, then only the appropriate range of text is written out.

**Parameters:**elem - an Element. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - on any I/O error [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if pos represents an invalid location within the document.

### setLineLength

protected void **setLineLength**(int l)

Enables subclasses to set the number of characters they want written per line. The default is 100.

**Parameters:**l - the maximum line length.

### getLineLength

protected int **getLineLength**()

Returns the maximum line length.

**Since:** 1.3

### setCurrentLineLength

protected void **setCurrentLineLength**(int length)

Sets the current line length.

**Since:** 1.3

### getCurrentLineLength

protected int **getCurrentLineLength**()

Returns the current line length.

**Since:** 1.3

### isLineEmpty

protected boolean **isLineEmpty**()

Returns true if the current line should be considered empty. This is true when getCurrentLineLength == 0 || indent has been invoked on an empty line.

**Since:** 1.3

### setCanWrapLines

protected void **setCanWrapLines**(boolean newValue)

Sets whether or not lines can be wrapped. This can be toggled during the writing of lines. For example, outputting HTML might set this to false when outputting a quoted string.

**Since:** 1.3

### getCanWrapLines

protected boolean **getCanWrapLines**()

Returns whether or not the lines can be wrapped. If this is false no lineSeparator's will be output.

**Since:** 1.3

### setIndentSpace

protected void **setIndentSpace**(int space)

Enables subclasses to specify how many spaces an indent maps to. When indentation takes place, the indent level is multiplied by this mapping. The default is 2.

**Parameters:**space - an int representing the space to indent mapping.

### getIndentSpace

protected int **getIndentSpace**()

Returns the amount of space to indent.

**Since:** 1.3

### setLineSeparator

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

Sets the String used to reprsent newlines. This is initialized in the constructor from either the Document, or the System property line.separator.

**Since:** 1.3

### getLineSeparator

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

Returns the string used to represent newlines.

**Since:** 1.3

### incrIndent

protected void **incrIndent**()

Increments the indent level. If indenting would cause getIndentSpace() \*getIndentLevel() to be > than getLineLength() this will not cause an indent.

### decrIndent

protected void **decrIndent**()

Decrements the indent level.

### getIndentLevel

protected int **getIndentLevel**()

Returns the current indentation level. That is, the number of times incrIndent has been invoked minus the number of times decrIndent has been invoked.

**Since:** 1.3

### indent

protected void **indent**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Does indentation. The number of spaces written out is indent level times the space to map mapping. If the current line is empty, this will not make it so that the current line is still considered empty.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - on any I/O error

### write

protected void **write**(char ch)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes out a character. This is implemented to invoke the write method that takes a char[].

**Parameters:**ch - a char. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - on any I/O error

### write

protected void **write**([String](http://docs.google.com/java/lang/String.html) content)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes out a string. This is implemented to invoke the write method that takes a char[].

**Parameters:**content - a String. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - on any I/O error

### writeLineSeparator

protected void **writeLineSeparator**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes the line separator. This invokes output directly as well as setting the lineLength to 0.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html)**Since:** 1.3

### write

protected void **write**(char[] chars,  
 int startIndex,  
 int length)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

All write methods call into this one. If getCanWrapLines() returns false, this will call output with each sequence of chars that doesn't contain a NEWLINE, followed by a call to writeLineSeparator. On the other hand, if getCanWrapLines() returns true, this will split the string, as necessary, so getLineLength is honored. The only exception is if the current string contains no whitespace, and won't fit in which case the line length will exceed getLineLength.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html)**Since:** 1.3

### writeAttributes

protected void **writeAttributes**([AttributeSet](http://docs.google.com/javax/swing/text/AttributeSet.html) attr)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Writes out the set of attributes as " =" pairs. It throws an IOException when encountered.

**Parameters:**attr - an AttributeSet. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - on any I/O error

### output

protected void **output**(char[] content,  
 int start,  
 int length)  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

The last stop in writing out content. All the write methods eventually make it to this method, which invokes write on the Writer.

This method also updates the line length based on length. If this is invoked to output a newline, the current line length will need to be reset as will no longer be valid. If it is up to the caller to do this. Use writeLineSeparator to write out a newline, which will property update the current line length.

**Throws:** [IOException](http://docs.google.com/java/io/IOException.html)**Since:** 1.3

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractWriter.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/AbstractDocument.LeafElement.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/AsyncBoxView.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/AbstractWriter.html)    [**NO FRAMES**](http://docs.google.com/AbstractWriter.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#35nkun2) |

[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).