| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SizeSequence.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/SizeRequirements.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/SortingFocusTraversalPolicy.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/SizeSequence.html)    [**NO FRAMES**](http://docs.google.com/SizeSequence.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#3rdcrjn) |

## **javax.swing**

Class SizeSequence

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

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

A SizeSequence object efficiently maintains an ordered list of sizes and corresponding positions. One situation for which SizeSequence might be appropriate is in a component that displays multiple rows of unequal size. In this case, a single SizeSequence object could be used to track the heights and Y positions of all rows.

Another example would be a multi-column component, such as a JTable, in which the column sizes are not all equal. The JTable might use a single SizeSequence object to store the widths and X positions of all the columns. The JTable could then use the SizeSequence object to find the column corresponding to a certain position. The JTable could update the SizeSequence object whenever one or more column sizes changed.

The following figure shows the relationship between size and position data for a multi-column component.



In the figure, the first index (0) corresponds to the first column, the second index (1) to the second column, and so on. The first column's position starts at 0, and the column occupies *size0* pixels, where *size0* is the value returned by getSize(0). Thus, the first column ends at *size0* - 1. The second column then begins at the position *size0* and occupies *size1* (getSize(1)) pixels.

Note that a SizeSequence object simply represents intervals along an axis. In our examples, the intervals represent height or width in pixels. However, any other unit of measure (for example, time in days) could be just as valid.

#### Implementation Notes

Normally when storing the size and position of entries, one would choose between storing the sizes or storing their positions instead. The two common operations that are needed during rendering are: getIndex(position) and setSize(index, size). Whichever choice of internal format is made one of these operations is costly when the number of entries becomes large. If sizes are stored, finding the index of the entry that encloses a particular position is linear in the number of entries. If positions are stored instead, setting the size of an entry at a particular index requires updating the positions of the affected entries, which is also a linear calculation.

Like the above techniques this class holds an array of N integers internally but uses a hybrid encoding, which is halfway between the size-based and positional-based approaches. The result is a data structure that takes the same space to store the information but can perform most operations in Log(N) time instead of O(N), where N is the number of entries in the list.

Two operations that remain O(N) in the number of entries are the insertEntries and removeEntries methods, both of which are implemented by converting the internal array to a set of integer sizes, copying it into the new array, and then reforming the hybrid representation in place.

**Since:** 1.3

| **Constructor Summary** | |
| --- | --- |
| [**SizeSequence**](http://docs.google.com/javax/swing/SizeSequence.html#SizeSequence())()            Creates a new SizeSequence object that contains no entries. |
| [**SizeSequence**](http://docs.google.com/javax/swing/SizeSequence.html#SizeSequence(int))(int numEntries)            Creates a new SizeSequence object that contains the specified number of entries, all initialized to have size 0. |
| [**SizeSequence**](http://docs.google.com/javax/swing/SizeSequence.html#SizeSequence(int%5B%5D))(int[] sizes)            Creates a new SizeSequence object that contains the specified sizes. |
| [**SizeSequence**](http://docs.google.com/javax/swing/SizeSequence.html#SizeSequence(int,%20int))(int numEntries, int value)            Creates a new SizeSequence object that contains the specified number of entries, all initialized to have size value. |

| **Method Summary** | |
| --- | --- |
| int | [**getIndex**](http://docs.google.com/javax/swing/SizeSequence.html#getIndex(int))(int position)            Returns the index of the entry that corresponds to the specified position. |
| int | [**getPosition**](http://docs.google.com/javax/swing/SizeSequence.html#getPosition(int))(int index)            Returns the start position for the specified entry. |
| int | [**getSize**](http://docs.google.com/javax/swing/SizeSequence.html#getSize(int))(int index)            Returns the size of the specified entry. |
| int[] | [**getSizes**](http://docs.google.com/javax/swing/SizeSequence.html#getSizes())()            Returns the size of all entries. |
| void | [**insertEntries**](http://docs.google.com/javax/swing/SizeSequence.html#insertEntries(int,%20int,%20int))(int start, int length, int value)            Adds a contiguous group of entries to this SizeSequence. |
| void | [**removeEntries**](http://docs.google.com/javax/swing/SizeSequence.html#removeEntries(int,%20int))(int start, int length)            Removes a contiguous group of entries from this SizeSequence. |
| void | [**setSize**](http://docs.google.com/javax/swing/SizeSequence.html#setSize(int,%20int))(int index, int size)            Sets the size of the specified entry. |
| void | [**setSizes**](http://docs.google.com/javax/swing/SizeSequence.html#setSizes(int%5B%5D))(int[] sizes)            Resets this SizeSequence object, using the data in the sizes argument. |

| **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** |
| --- |

### SizeSequence

public **SizeSequence**()

Creates a new SizeSequence object that contains no entries. To add entries, you can use insertEntries or setSizes.

**See Also:**[insertEntries(int, int, int)](http://docs.google.com/javax/swing/SizeSequence.html#insertEntries(int,%20int,%20int)), [setSizes(int, int)](http://docs.google.com/javax/swing/SizeSequence.html#setSizes(int,%20int))

### SizeSequence

public **SizeSequence**(int numEntries)

Creates a new SizeSequence object that contains the specified number of entries, all initialized to have size 0.

**Parameters:**numEntries - the number of sizes to track **Throws:** [NegativeArraySizeException](http://docs.google.com/java/lang/NegativeArraySizeException.html) - if numEntries < 0

### SizeSequence

public **SizeSequence**(int numEntries,  
 int value)

Creates a new SizeSequence object that contains the specified number of entries, all initialized to have size value.

**Parameters:**numEntries - the number of sizes to trackvalue - the initial value of each size

### SizeSequence

public **SizeSequence**(int[] sizes)

Creates a new SizeSequence object that contains the specified sizes.

**Parameters:**sizes - the array of sizes to be contained in the SizeSequence

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

### setSizes

public void **setSizes**(int[] sizes)

Resets this SizeSequence object, using the data in the sizes argument. This method reinitializes this object so that it contains as many entries as the sizes array. Each entry's size is initialized to the value of the corresponding item in sizes.

**Parameters:**sizes - the array of sizes to be contained in this SizeSequence

### getSizes

public int[] **getSizes**()

Returns the size of all entries.

**Returns:**a new array containing the sizes in this object

### getPosition

public int **getPosition**(int index)

Returns the start position for the specified entry. For example, getPosition(0) returns 0, getPosition(1) is equal to getSize(0), getPosition(2) is equal to getSize(0) + getSize(1), and so on.

Note that if index is greater than length the value returned may be meaningless.

**Parameters:**index - the index of the entry whose position is desired **Returns:**the starting position of the specified entry

### getIndex

public int **getIndex**(int position)

Returns the index of the entry that corresponds to the specified position. For example, getIndex(0) is 0, since the first entry always starts at position 0.

**Parameters:**position - the position of the entry **Returns:**the index of the entry that occupies the specified position

### getSize

public int **getSize**(int index)

Returns the size of the specified entry. If index is out of the range (0 <= index < getSizes().length) the behavior is unspecified.

**Parameters:**index - the index corresponding to the entry **Returns:**the size of the entry

### setSize

public void **setSize**(int index,  
 int size)

Sets the size of the specified entry. Note that if the value of index does not fall in the range: (0 <= index < getSizes().length) the behavior is unspecified.

**Parameters:**index - the index corresponding to the entrysize - the size of the entry

### insertEntries

public void **insertEntries**(int start,  
 int length,  
 int value)

Adds a contiguous group of entries to this SizeSequence. Note that the values of start and length must satisfy the following conditions: (0 <= start < getSizes().length) AND (length >= 0). If these conditions are not met, the behavior is unspecified and an exception may be thrown.

**Parameters:**start - the index to be assigned to the first entry in the grouplength - the number of entries in the groupvalue - the size to be assigned to each new entry **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if the parameters are outside of the range: (0 <= start < (getSizes().length)) AND (length >= 0)

### removeEntries

public void **removeEntries**(int start,  
 int length)

Removes a contiguous group of entries from this SizeSequence. Note that the values of start and length must satisfy the following conditions: (0 <= start < getSizes().length) AND (length >= 0). If these conditions are not met, the behavior is unspecified and an exception may be thrown.

**Parameters:**start - the index of the first entry to be removedlength - the number of entries to be removed

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/SizeSequence.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/SizeRequirements.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/SortingFocusTraversalPolicy.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/SizeSequence.html)    [**NO FRAMES**](http://docs.google.com/SizeSequence.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#3rdcrjn) |

[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).