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

## **javax.swing**

Class Spring

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

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

An instance of the Spring class holds three properties that characterize its behavior: the *minimum*, *preferred*, and *maximum* values. Each of these properties may be involved in defining its fourth, *value*, property based on a series of rules.

An instance of the Spring class can be visualized as a mechanical spring that provides a corrective force as the spring is compressed or stretched away from its preferred value. This force is modelled as linear function of the distance from the preferred value, but with two different constants -- one for the compressional force and one for the tensional one. Those constants are specified by the minimum and maximum values of the spring such that a spring at its minimum value produces an equal and opposite force to that which is created when it is at its maximum value. The difference between the *preferred* and *minimum* values, therefore, represents the ease with which the spring can be compressed and the difference between its *maximum* and *preferred* values, indicates the ease with which the Spring can be extended. See the [sum(javax.swing.Spring, javax.swing.Spring)](http://docs.google.com/javax/swing/Spring.html#sum(javax.swing.Spring,%20javax.swing.Spring)) method for details.

By defining simple arithmetic operations on Springs, the behavior of a collection of Springs can be reduced to that of an ordinary (non-compound) Spring. We define the "+", "-", *max*, and *min* operators on Springs so that, in each case, the result is a Spring whose characteristics bear a useful mathematical relationship to its constituent springs.

A Spring can be treated as a pair of intervals with a single common point: the preferred value. The following rules define some of the arithmetic operators that can be applied to intervals ([a, b] refers to the interval from a to b, where a <= b).

[a1, b1] + [a2, b2] = [a1 + a2, b1 + b2]  
  
 -[a, b] = [-b, -a]  
  
 max([a1, b1], [a2, b2]) = [max(a1, a2), max(b1, b2)]

If we denote Springs as [a, b, c], where a <= b <= c, we can define the same arithmetic operators on Springs:

[a1, b1, c1] + [a2, b2, c2] = [a1 + a2, b1 + b2, c1 + c2]  
  
 -[a, b, c] = [-c, -b, -a]  
  
 max([a1, b1, c1], [a2, b2, c2]) = [max(a1, a2), max(b1, b2), max(c1, c2)]

With both intervals and Springs we can define "-" and *min* in terms of negation:

X - Y = X + (-Y)  
  
 min(X, Y) = -max(-X, -Y)

For the static methods in this class that embody the arithmetic operators, we do not actually perform the operation in question as that would snapshot the values of the properties of the method's arguments at the time the static method is called. Instead, the static methods create a new Spring instance containing references to the method's arguments so that the characteristics of the new spring track the potentially changing characteristics of the springs from which it was made. This is a little like the idea of a *lazy value* in a functional language.

If you are implementing a SpringLayout you can find further information and examples in [How to Use SpringLayout](http://java.sun.com/docs/books/tutorial/uiswing/layout/spring.html), a section in *The Java Tutorial.*

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

**Since:** 1.4 **See Also:**[SpringLayout](http://docs.google.com/javax/swing/SpringLayout.html), [SpringLayout.Constraints](http://docs.google.com/javax/swing/SpringLayout.Constraints.html)

| **Field Summary** | |
| --- | --- |
| static int | [**UNSET**](http://docs.google.com/javax/swing/Spring.html#UNSET)            An integer value signifying that a property value has not yet been calculated. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**Spring**](http://docs.google.com/javax/swing/Spring.html#Spring())()            Used by factory methods to create a Spring. |

| **Method Summary** | |
| --- | --- |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**constant**](http://docs.google.com/javax/swing/Spring.html#constant(int))(int pref)            Returns a strut -- a spring whose *minimum*, *preferred*, and *maximum* values each have the value pref. |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**constant**](http://docs.google.com/javax/swing/Spring.html#constant(int,%20int,%20int))(int min, int pref, int max)            Returns a spring whose *minimum*, *preferred*, and *maximum* values have the values: min, pref, and max respectively. |
| abstract  int | [**getMaximumValue**](http://docs.google.com/javax/swing/Spring.html#getMaximumValue())()            Returns the *maximum* value of this Spring. |
| abstract  int | [**getMinimumValue**](http://docs.google.com/javax/swing/Spring.html#getMinimumValue())()            Returns the *minimum* value of this Spring. |
| abstract  int | [**getPreferredValue**](http://docs.google.com/javax/swing/Spring.html#getPreferredValue())()            Returns the *preferred* value of this Spring. |
| abstract  int | [**getValue**](http://docs.google.com/javax/swing/Spring.html#getValue())()            Returns the current *value* of this Spring. |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**height**](http://docs.google.com/javax/swing/Spring.html#height(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) c)            Returns a spring whose *minimum*, *preferred*, *maximum* and *value* properties are defined by the heights of the *minimumSize*, *preferredSize*, *maximumSize* and *size* properties of the supplied component. |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**max**](http://docs.google.com/javax/swing/Spring.html#max(javax.swing.Spring,%20javax.swing.Spring))([Spring](http://docs.google.com/javax/swing/Spring.html) s1, [Spring](http://docs.google.com/javax/swing/Spring.html) s2)            Returns max(s1, s2): a spring whose value is always greater than (or equal to) the values of both s1 and s2. |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**minus**](http://docs.google.com/javax/swing/Spring.html#minus(javax.swing.Spring))([Spring](http://docs.google.com/javax/swing/Spring.html) s)            Returns -s: a spring running in the opposite direction to s. |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**scale**](http://docs.google.com/javax/swing/Spring.html#scale(javax.swing.Spring,%20float))([Spring](http://docs.google.com/javax/swing/Spring.html) s, float factor)            Returns a spring whose *minimum*, *preferred*, *maximum* and *value* properties are each multiples of the properties of the argument spring, s. |
| abstract  void | [**setValue**](http://docs.google.com/javax/swing/Spring.html#setValue(int))(int value)            Sets the current *value* of this Spring to value. |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**sum**](http://docs.google.com/javax/swing/Spring.html#sum(javax.swing.Spring,%20javax.swing.Spring))([Spring](http://docs.google.com/javax/swing/Spring.html) s1, [Spring](http://docs.google.com/javax/swing/Spring.html) s2)            Returns s1+s2: a spring representing s1 and s2 in series. |
| static [Spring](http://docs.google.com/javax/swing/Spring.html) | [**width**](http://docs.google.com/javax/swing/Spring.html#width(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) c)            Returns a spring whose *minimum*, *preferred*, *maximum* and *value* properties are defined by the widths of the *minimumSize*, *preferredSize*, *maximumSize* and *size* properties of the supplied component. |

| **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** |
| --- |

### UNSET

public static final int **UNSET**

An integer value signifying that a property value has not yet been calculated.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.Spring.UNSET)

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

### Spring

protected **Spring**()

Used by factory methods to create a Spring.

**See Also:**[constant(int)](http://docs.google.com/javax/swing/Spring.html#constant(int)), [constant(int, int, int)](http://docs.google.com/javax/swing/Spring.html#constant(int,%20int,%20int)), [max(javax.swing.Spring, javax.swing.Spring)](http://docs.google.com/javax/swing/Spring.html#max(javax.swing.Spring,%20javax.swing.Spring)), [minus(javax.swing.Spring)](http://docs.google.com/javax/swing/Spring.html#minus(javax.swing.Spring)), [sum(javax.swing.Spring, javax.swing.Spring)](http://docs.google.com/javax/swing/Spring.html#sum(javax.swing.Spring,%20javax.swing.Spring)), [SpringLayout.Constraints](http://docs.google.com/javax/swing/SpringLayout.Constraints.html)

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

### getMinimumValue

public abstract int **getMinimumValue**()

Returns the *minimum* value of this Spring.

**Returns:**the minimumValue property of this Spring

### getPreferredValue

public abstract int **getPreferredValue**()

Returns the *preferred* value of this Spring.

**Returns:**the preferredValue of this Spring

### getMaximumValue

public abstract int **getMaximumValue**()

Returns the *maximum* value of this Spring.

**Returns:**the maximumValue property of this Spring

### getValue

public abstract int **getValue**()

Returns the current *value* of this Spring.

**Returns:**the value property of this Spring**See Also:**[setValue(int)](http://docs.google.com/javax/swing/Spring.html#setValue(int))

### setValue

public abstract void **setValue**(int value)

Sets the current *value* of this Spring to value.

**Parameters:**value - the new setting of the value property**See Also:**[getValue()](http://docs.google.com/javax/swing/Spring.html#getValue())

### constant

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **constant**(int pref)

Returns a strut -- a spring whose *minimum*, *preferred*, and *maximum* values each have the value pref.

**Parameters:**pref - the *minimum*, *preferred*, and *maximum* values of the new spring **Returns:**a spring whose *minimum*, *preferred*, and *maximum* values each have the value pref**See Also:**[Spring](http://docs.google.com/javax/swing/Spring.html)

### constant

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **constant**(int min,  
 int pref,  
 int max)

Returns a spring whose *minimum*, *preferred*, and *maximum* values have the values: min, pref, and max respectively.

**Parameters:**min - the *minimum* value of the new springpref - the *preferred* value of the new springmax - the *maximum* value of the new spring **Returns:**a spring whose *minimum*, *preferred*, and *maximum* values have the values: min, pref, and max respectively**See Also:**[Spring](http://docs.google.com/javax/swing/Spring.html)

### minus

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **minus**([Spring](http://docs.google.com/javax/swing/Spring.html) s)

Returns -s: a spring running in the opposite direction to s.

**Returns:**-s: a spring running in the opposite direction to s**See Also:**[Spring](http://docs.google.com/javax/swing/Spring.html)

### sum

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **sum**([Spring](http://docs.google.com/javax/swing/Spring.html) s1,  
 [Spring](http://docs.google.com/javax/swing/Spring.html) s2)

Returns s1+s2: a spring representing s1 and s2 in series. In a sum, s3, of two springs, s1 and s2, the *strains* of s1, s2, and s3 are maintained at the same level (to within the precision implied by their integer *value*s). The strain of a spring in compression is:

value - pref  
 ------------  
 pref - min

and the strain of a spring in tension is:

value - pref  
 ------------  
 max - pref

When setValue is called on the sum spring, s3, the strain in s3 is calculated using one of the formulas above. Once the strain of the sum is known, the *value*s of s1 and s2 are then set so that they are have a strain equal to that of the sum. The formulas are evaluated so as to take rounding errors into account and ensure that the sum of the *value*s of s1 and s2 is exactly equal to the *value* of s3.

**Returns:**s1+s2: a spring representing s1 and s2 in series**See Also:**[Spring](http://docs.google.com/javax/swing/Spring.html)

### max

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **max**([Spring](http://docs.google.com/javax/swing/Spring.html) s1,  
 [Spring](http://docs.google.com/javax/swing/Spring.html) s2)

Returns max(s1, s2): a spring whose value is always greater than (or equal to) the values of both s1 and s2.

**Returns:**max(s1, s2): a spring whose value is always greater than (or equal to) the values of both s1 and s2**See Also:**[Spring](http://docs.google.com/javax/swing/Spring.html)

### scale

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **scale**([Spring](http://docs.google.com/javax/swing/Spring.html) s,  
 float factor)

Returns a spring whose *minimum*, *preferred*, *maximum* and *value* properties are each multiples of the properties of the argument spring, s. Minimum and maximum properties are swapped when factor is negative (in accordance with the rules of interval arithmetic).

When factor is, for example, 0.5f the result represents 'the mid-point' of its input - an operation that is useful for centering components in a container.

**Parameters:**s - the spring to scalefactor - amount to scale by. **Returns:**a spring whose properties are those of the input spring s multiplied by factor **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if s is null**Since:** 1.5

### width

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **width**([Component](http://docs.google.com/java/awt/Component.html) c)

Returns a spring whose *minimum*, *preferred*, *maximum* and *value* properties are defined by the widths of the *minimumSize*, *preferredSize*, *maximumSize* and *size* properties of the supplied component. The returned spring is a 'wrapper' implementation whose methods call the appropriate size methods of the supplied component. The minimum, preferred, maximum and value properties of the returned spring therefore report the current state of the appropriate properties in the component and track them as they change.

**Parameters:**c - Component used for calculating size **Returns:**a spring whose properties are defined by the horizontal component of the component's size methods. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if c is null**Since:** 1.5

### height

public static [Spring](http://docs.google.com/javax/swing/Spring.html) **height**([Component](http://docs.google.com/java/awt/Component.html) c)

Returns a spring whose *minimum*, *preferred*, *maximum* and *value* properties are defined by the heights of the *minimumSize*, *preferredSize*, *maximumSize* and *size* properties of the supplied component. The returned spring is a 'wrapper' implementation whose methods call the appropriate size methods of the supplied component. The minimum, preferred, maximum and value properties of the returned spring therefore report the current state of the appropriate properties in the component and track them as they change.

**Parameters:**c - Component used for calculating size **Returns:**a spring whose properties are defined by the vertical component of the component's size methods. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if c is null**Since:** 1.5

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