| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GridLayout.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/java/awt/GridBagLayoutInfo.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/HeadlessException.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/GridLayout.html)    [**NO FRAMES**](http://docs.google.com/GridLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#17dp8vu) |

## **java.awt**

Class GridLayout

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.awt.GridLayout**

**All Implemented Interfaces:** [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

public class **GridLayout**extends [Object](http://docs.google.com/java/lang/Object.html)implements [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

The GridLayout class is a layout manager that lays out a container's components in a rectangular grid. The container is divided into equal-sized rectangles, and one component is placed in each rectangle. For example, the following is an applet that lays out six buttons into three rows and two columns:

import java.awt.\*;  
 import java.applet.Applet;  
 public class ButtonGrid extends Applet {  
 public void init() {  
 setLayout(new GridLayout(3,2));  
 add(new Button("1"));  
 add(new Button("2"));  
 add(new Button("3"));  
 add(new Button("4"));  
 add(new Button("5"));  
 add(new Button("6"));  
 }  
 }

If the container's ComponentOrientation property is horizontal and left-to-right, the above example produces the output shown in Figure 1. If the container's ComponentOrientation property is horizontal and right-to-left, the example produces the output shown in Figure 2.

|  |  |
| --- | --- |
| Figure 1: Horizontal, Left-to-Right | Figure 2: Horizontal, Right-to-Left |

When both the number of rows and the number of columns have been set to non-zero values, either by a constructor or by the setRows and setColumns methods, the number of columns specified is ignored. Instead, the number of columns is determined from the specified number of rows and the total number of components in the layout. So, for example, if three rows and two columns have been specified and nine components are added to the layout, they will be displayed as three rows of three columns. Specifying the number of columns affects the layout only when the number of rows is set to zero.

**Since:** JDK1.0 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.awt.GridLayout)

| **Constructor Summary** | |
| --- | --- |
| [**GridLayout**](http://docs.google.com/java/awt/GridLayout.html#GridLayout())()            Creates a grid layout with a default of one column per component, in a single row. |
| [**GridLayout**](http://docs.google.com/java/awt/GridLayout.html#GridLayout(int,%20int))(int rows, int cols)            Creates a grid layout with the specified number of rows and columns. |
| [**GridLayout**](http://docs.google.com/java/awt/GridLayout.html#GridLayout(int,%20int,%20int,%20int))(int rows, int cols, int hgap, int vgap)            Creates a grid layout with the specified number of rows and columns. |

| **Method Summary** | |
| --- | --- |
| void | [**addLayoutComponent**](http://docs.google.com/java/awt/GridLayout.html#addLayoutComponent(java.lang.String,%20java.awt.Component))([String](http://docs.google.com/java/lang/String.html) name, [Component](http://docs.google.com/java/awt/Component.html) comp)            Adds the specified component with the specified name to the layout. |
| int | [**getColumns**](http://docs.google.com/java/awt/GridLayout.html#getColumns())()            Gets the number of columns in this layout. |
| int | [**getHgap**](http://docs.google.com/java/awt/GridLayout.html#getHgap())()            Gets the horizontal gap between components. |
| int | [**getRows**](http://docs.google.com/java/awt/GridLayout.html#getRows())()            Gets the number of rows in this layout. |
| int | [**getVgap**](http://docs.google.com/java/awt/GridLayout.html#getVgap())()            Gets the vertical gap between components. |
| void | [**layoutContainer**](http://docs.google.com/java/awt/GridLayout.html#layoutContainer(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Lays out the specified container using this layout. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**minimumLayoutSize**](http://docs.google.com/java/awt/GridLayout.html#minimumLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Determines the minimum size of the container argument using this grid layout. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**preferredLayoutSize**](http://docs.google.com/java/awt/GridLayout.html#preferredLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Determines the preferred size of the container argument using this grid layout. |
| void | [**removeLayoutComponent**](http://docs.google.com/java/awt/GridLayout.html#removeLayoutComponent(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) comp)            Removes the specified component from the layout. |
| void | [**setColumns**](http://docs.google.com/java/awt/GridLayout.html#setColumns(int))(int cols)            Sets the number of columns in this layout to the specified value. |
| void | [**setHgap**](http://docs.google.com/java/awt/GridLayout.html#setHgap(int))(int hgap)            Sets the horizontal gap between components to the specified value. |
| void | [**setRows**](http://docs.google.com/java/awt/GridLayout.html#setRows(int))(int rows)            Sets the number of rows in this layout to the specified value. |
| void | [**setVgap**](http://docs.google.com/java/awt/GridLayout.html#setVgap(int))(int vgap)            Sets the vertical gap between components to the specified value. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/awt/GridLayout.html#toString())()            Returns the string representation of this grid layout's values. |

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

### GridLayout

public **GridLayout**()

Creates a grid layout with a default of one column per component, in a single row.

**Since:** JDK1.1

### GridLayout

public **GridLayout**(int rows,  
 int cols)

Creates a grid layout with the specified number of rows and columns. All components in the layout are given equal size.

One, but not both, of rows and cols can be zero, which means that any number of objects can be placed in a row or in a column.

**Parameters:**rows - the rows, with the value zero meaning any number of rows.cols - the columns, with the value zero meaning any number of columns.

### GridLayout

public **GridLayout**(int rows,  
 int cols,  
 int hgap,  
 int vgap)

Creates a grid layout with the specified number of rows and columns. All components in the layout are given equal size.

In addition, the horizontal and vertical gaps are set to the specified values. Horizontal gaps are placed between each of the columns. Vertical gaps are placed between each of the rows.

One, but not both, of rows and cols can be zero, which means that any number of objects can be placed in a row or in a column.

All GridLayout constructors defer to this one.

**Parameters:**rows - the rows, with the value zero meaning any number of rowscols - the columns, with the value zero meaning any number of columnshgap - the horizontal gapvgap - the vertical gap **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the value of both rows and cols is set to zero

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

### getRows

public int **getRows**()

Gets the number of rows in this layout.

**Returns:**the number of rows in this layout**Since:** JDK1.1

### setRows

public void **setRows**(int rows)

Sets the number of rows in this layout to the specified value.

**Parameters:**rows - the number of rows in this layout **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the value of both rows and cols is set to zero**Since:** JDK1.1

### getColumns

public int **getColumns**()

Gets the number of columns in this layout.

**Returns:**the number of columns in this layout**Since:** JDK1.1

### setColumns

public void **setColumns**(int cols)

Sets the number of columns in this layout to the specified value. Setting the number of columns has no affect on the layout if the number of rows specified by a constructor or by the setRows method is non-zero. In that case, the number of columns displayed in the layout is determined by the total number of components and the number of rows specified.

**Parameters:**cols - the number of columns in this layout **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the value of both rows and cols is set to zero**Since:** JDK1.1

### getHgap

public int **getHgap**()

Gets the horizontal gap between components.

**Returns:**the horizontal gap between components**Since:** JDK1.1

### setHgap

public void **setHgap**(int hgap)

Sets the horizontal gap between components to the specified value.

**Parameters:**hgap - the horizontal gap between components**Since:** JDK1.1

### getVgap

public int **getVgap**()

Gets the vertical gap between components.

**Returns:**the vertical gap between components**Since:** JDK1.1

### setVgap

public void **setVgap**(int vgap)

Sets the vertical gap between components to the specified value.

**Parameters:**vgap - the vertical gap between components**Since:** JDK1.1

### addLayoutComponent

public void **addLayoutComponent**([String](http://docs.google.com/java/lang/String.html) name,  
 [Component](http://docs.google.com/java/awt/Component.html) comp)

Adds the specified component with the specified name to the layout.

**Specified by:**[addLayoutComponent](http://docs.google.com/java/awt/LayoutManager.html#addLayoutComponent(java.lang.String,%20java.awt.Component)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**name - the name of the componentcomp - the component to be added

### removeLayoutComponent

public void **removeLayoutComponent**([Component](http://docs.google.com/java/awt/Component.html) comp)

Removes the specified component from the layout.

**Specified by:**[removeLayoutComponent](http://docs.google.com/java/awt/LayoutManager.html#removeLayoutComponent(java.awt.Component)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**comp - the component to be removed

### preferredLayoutSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **preferredLayoutSize**([Container](http://docs.google.com/java/awt/Container.html) parent)

Determines the preferred size of the container argument using this grid layout.

The preferred width of a grid layout is the largest preferred width of all of the components in the container times the number of columns, plus the horizontal padding times the number of columns minus one, plus the left and right insets of the target container.

The preferred height of a grid layout is the largest preferred height of all of the components in the container times the number of rows, plus the vertical padding times the number of rows minus one, plus the top and bottom insets of the target container.

**Specified by:**[preferredLayoutSize](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**parent - the container in which to do the layout **Returns:**the preferred dimensions to lay out the subcomponents of the specified container**See Also:**[minimumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/GridLayout.html#minimumLayoutSize(java.awt.Container)), [Container.getPreferredSize()](http://docs.google.com/java/awt/Container.html#getPreferredSize())

### minimumLayoutSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **minimumLayoutSize**([Container](http://docs.google.com/java/awt/Container.html) parent)

Determines the minimum size of the container argument using this grid layout.

The minimum width of a grid layout is the largest minimum width of all of the components in the container times the number of columns, plus the horizontal padding times the number of columns minus one, plus the left and right insets of the target container.

The minimum height of a grid layout is the largest minimum height of all of the components in the container times the number of rows, plus the vertical padding times the number of rows minus one, plus the top and bottom insets of the target container.

**Specified by:**[minimumLayoutSize](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**parent - the container in which to do the layout **Returns:**the minimum dimensions needed to lay out the subcomponents of the specified container**See Also:**[preferredLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/GridLayout.html#preferredLayoutSize(java.awt.Container)), [Container.doLayout()](http://docs.google.com/java/awt/Container.html#doLayout())

### layoutContainer

public void **layoutContainer**([Container](http://docs.google.com/java/awt/Container.html) parent)

Lays out the specified container using this layout.

This method reshapes the components in the specified target container in order to satisfy the constraints of the GridLayout object.

The grid layout manager determines the size of individual components by dividing the free space in the container into equal-sized portions according to the number of rows and columns in the layout. The container's free space equals the container's size minus any insets and any specified horizontal or vertical gap. All components in a grid layout are given the same size.

**Specified by:**[layoutContainer](http://docs.google.com/java/awt/LayoutManager.html#layoutContainer(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**parent - the container in which to do the layout**See Also:**[Container](http://docs.google.com/java/awt/Container.html), [Container.doLayout()](http://docs.google.com/java/awt/Container.html#doLayout())

### toString

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

Returns the string representation of this grid layout's values.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of this grid layout

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GridLayout.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/java/awt/GridBagLayoutInfo.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/HeadlessException.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/GridLayout.html)    [**NO FRAMES**](http://docs.google.com/GridLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#17dp8vu) |

[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).