| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GroupLayout.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/GrayFilter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/GroupLayout.Alignment.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/GroupLayout.html)    [**NO FRAMES**](http://docs.google.com/GroupLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#3rdcrjn) | [METHOD](#lnxbz9) |

## **javax.swing**

Class GroupLayout

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

**All Implemented Interfaces:** [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html), [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html)

public class **GroupLayout**extends [Object](http://docs.google.com/java/lang/Object.html)implements [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html)

GroupLayout is a LayoutManager that hierarchically groups components in order to position them in a Container. GroupLayout is intended for use by builders, but may be hand-coded as well. Grouping is done by instances of the [Group](http://docs.google.com/javax/swing/GroupLayout.Group.html) class. GroupLayout supports two types of groups. A sequential group positions its child elements sequentially, one after another. A parallel group aligns its child elements in one of four ways.

Each group may contain any number of elements, where an element is a Group, Component, or gap. A gap can be thought of as an invisible component with a minimum, preferred and maximum size. In addition GroupLayout supports a preferred gap, whose value comes from LayoutStyle.

Elements are similar to a spring. Each element has a range as specified by a minimum, preferred and maximum. Gaps have either a developer-specified range, or a range determined by LayoutStyle. The range for Components is determined from the Component's getMinimumSize, getPreferredSize and getMaximumSize methods. In addition, when adding Components you may specify a particular range to use instead of that from the component. The range for a Group is determined by the type of group. A ParallelGroup's range is the maximum of the ranges of its elements. A SequentialGroup's range is the sum of the ranges of its elements.

GroupLayout treats each axis independently. That is, there is a group representing the horizontal axis, and a group representing the vertical axis. The horizontal group is responsible for determining the minimum, preferred and maximum size along the horizontal axis as well as setting the x and width of the components contained in it. The vertical group is responsible for determining the minimum, preferred and maximum size along the vertical axis as well as setting the y and height of the components contained in it. Each Component must exist in both a horizontal and vertical group, otherwise an IllegalStateException is thrown during layout, or when the minimum, preferred or maximum size is requested.

The following diagram shows a sequential group along the horizontal axis. The sequential group contains three components. A parallel group was used along the vertical axis.



To reinforce that each axis is treated independently the diagram shows the range of each group and element along each axis. The range of each component has been projected onto the axes, and the groups are rendered in blue (horizontal) and red (vertical). For readability there is a gap between each of the elements in the sequential group.

The sequential group along the horizontal axis is rendered as a solid blue line. Notice the sequential group is the sum of the children elements it contains.

Along the vertical axis the parallel group is the maximum of the height of each of the components. As all three components have the same height, the parallel group has the same height.

The following diagram shows the same three components, but with the parallel group along the horizontal axis and the sequential group along the vertical axis.



As c1 is the largest of the three components, the parallel group is sized to c1. As c2 and c3 are smaller than c1 they are aligned based on the alignment specified for the component (if specified) or the default alignment of the parallel group. In the diagram c2 and c3 were created with an alignment of LEADING. If the component orientation were right-to-left then c2 and c3 would be positioned on the opposite side.

The following diagram shows a sequential group along both the horizontal and vertical axis.



GroupLayout provides the ability to insert gaps between Components. The size of the gap is determined by an instance of LayoutStyle. This may be turned on using the setAutoCreateGaps method. Similarly, you may use the setAutoCreateContainerGaps method to insert gaps between components that touch the edge of the parent container and the container.

The following builds a panel consisting of two labels in one column, followed by two textfields in the next column:

JComponent panel = ...;  
 GroupLayout layout = new GroupLayout(panel);  
 panel.setLayout(layout);  
   
 // Turn on automatically adding gaps between components  
 layout.setAutoCreateGaps(true);  
   
 // Turn on automatically creating gaps between components that touch  
 // the edge of the container and the container.  
 layout.setAutoCreateContainerGaps(true);  
   
 // Create a sequential group for the horizontal axis.  
   
 GroupLayout.SequentialGroup hGroup = layout.createSequentialGroup();  
   
 // The sequential group in turn contains two parallel groups.  
 // One parallel group contains the labels, the other the text fields.  
 // Putting the labels in a parallel group along the horizontal axis  
 // positions them at the same x location.  
 //  
 // Variable indentation is used to reinforce the level of grouping.  
 hGroup.addGroup(layout.createParallelGroup().  
 addComponent(label1).addComponent(label2));  
 hGroup.addGroup(layout.createParallelGroup().  
 addComponent(tf1).addComponent(tf2));  
 layout.setHorizontalGroup(hGroup);  
   
 // Create a sequential group for the vertical axis.  
 GroupLayout.SequentialGroup vGroup = layout.createSequentialGroup();  
   
 // The sequential group contains two parallel groups that align  
 // the contents along the baseline. The first parallel group contains  
 // the first label and text field, and the second parallel group contains  
 // the second label and text field. By using a sequential group  
 // the labels and text fields are positioned vertically after one another.  
 vGroup.addGroup(layout.createParallelGroup(Alignment.BASELINE).  
 addComponent(label1).addComponent(tf1));  
 vGroup.addGroup(layout.createParallelGroup(Alignment.BASELINE).  
 addComponent(label2).addComponent(tf2));  
 layout.setVerticalGroup(vGroup);

When run the following is produced.



This layout consists of the following.

* The horizontal axis consists of a sequential group containing two parallel groups. The first parallel group contains the labels, and the second parallel group contains the text fields.
* The vertical axis consists of a sequential group containing two parallel groups. The parallel groups are configured to align their components along the baseline. The first parallel group contains the first label and first text field, and the second group consists of the second label and second text field.

There are a couple of things to notice in this code:

* You need not explicitly add the components to the container; this is indirectly done by using one of the add methods of Group.
* The various add methods return the caller. This allows for easy chaining of invocations. For example, group.addComponent(label1).addComponent(label2); is equivalent to group.addComponent(label1); group.addComponent(label2);.
* There are no public constructors for Groups; instead use the create methods of GroupLayout.

**Since:** 1.6

| **Nested Class Summary** | |
| --- | --- |
| static class | [**GroupLayout.Alignment**](http://docs.google.com/javax/swing/GroupLayout.Alignment.html)            Enumeration of the possible ways ParallelGroup can align its children. |
| class | [**GroupLayout.Group**](http://docs.google.com/javax/swing/GroupLayout.Group.html)            Group provides the basis for the two types of operations supported by GroupLayout: laying out components one after another ([SequentialGroup](http://docs.google.com/javax/swing/GroupLayout.SequentialGroup.html)) or aligned ([ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html)). |
| class | [**GroupLayout.ParallelGroup**](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html)            A Group that aligns and sizes it's children. |
| class | [**GroupLayout.SequentialGroup**](http://docs.google.com/javax/swing/GroupLayout.SequentialGroup.html)            A Group that positions and sizes its elements sequentially, one after another. |

| **Field Summary** | |
| --- | --- |
| static int | [**DEFAULT\_SIZE**](http://docs.google.com/javax/swing/GroupLayout.html#DEFAULT_SIZE)            Indicates the size from the component or gap should be used for a particular range value. |
| static int | [**PREFERRED\_SIZE**](http://docs.google.com/javax/swing/GroupLayout.html#PREFERRED_SIZE)            Indicates the preferred size from the component or gap should be used for a particular range value. |

| **Constructor Summary** | |
| --- | --- |
| [**GroupLayout**](http://docs.google.com/javax/swing/GroupLayout.html#GroupLayout(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) host)            Creates a GroupLayout for the specified Container. |

| **Method Summary** | |
| --- | --- |
| void | [**addLayoutComponent**](http://docs.google.com/javax/swing/GroupLayout.html#addLayoutComponent(java.awt.Component,%20java.lang.Object))([Component](http://docs.google.com/java/awt/Component.html) component, [Object](http://docs.google.com/java/lang/Object.html) constraints)            Notification that a Component has been added to the parent container. |
| void | [**addLayoutComponent**](http://docs.google.com/javax/swing/GroupLayout.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) component)            Notification that a Component has been added to the parent container. |
| [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) | [**createBaselineGroup**](http://docs.google.com/javax/swing/GroupLayout.html#createBaselineGroup(boolean,%20boolean))(boolean resizable, boolean anchorBaselineToTop)            Creates and returns a ParallelGroup that aligns it's elements along the baseline. |
| [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) | [**createParallelGroup**](http://docs.google.com/javax/swing/GroupLayout.html#createParallelGroup())()            Creates and returns a ParallelGroup with an alignment of Alignment.LEADING. |
| [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) | [**createParallelGroup**](http://docs.google.com/javax/swing/GroupLayout.html#createParallelGroup(javax.swing.GroupLayout.Alignment))([GroupLayout.Alignment](http://docs.google.com/javax/swing/GroupLayout.Alignment.html) alignment)            Creates and returns a ParallelGroup with the specified alignment. |
| [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) | [**createParallelGroup**](http://docs.google.com/javax/swing/GroupLayout.html#createParallelGroup(javax.swing.GroupLayout.Alignment,%20boolean))([GroupLayout.Alignment](http://docs.google.com/javax/swing/GroupLayout.Alignment.html) alignment, boolean resizable)            Creates and returns a ParallelGroup with the specified alignment and resize behavior. |
| [GroupLayout.SequentialGroup](http://docs.google.com/javax/swing/GroupLayout.SequentialGroup.html) | [**createSequentialGroup**](http://docs.google.com/javax/swing/GroupLayout.html#createSequentialGroup())()            Creates and returns a SequentialGroup. |
| boolean | [**getAutoCreateContainerGaps**](http://docs.google.com/javax/swing/GroupLayout.html#getAutoCreateContainerGaps())()            Returns true if gaps between the container and components that border the container are automatically created. |
| boolean | [**getAutoCreateGaps**](http://docs.google.com/javax/swing/GroupLayout.html#getAutoCreateGaps())()            Returns true if gaps between components are automatically created. |
| boolean | [**getHonorsVisibility**](http://docs.google.com/javax/swing/GroupLayout.html#getHonorsVisibility())()            Returns whether component visiblity is considered when sizing and positioning components. |
| float | [**getLayoutAlignmentX**](http://docs.google.com/javax/swing/GroupLayout.html#getLayoutAlignmentX(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Returns the alignment along the x axis. |
| float | [**getLayoutAlignmentY**](http://docs.google.com/javax/swing/GroupLayout.html#getLayoutAlignmentY(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Returns the alignment along the y axis. |
| [LayoutStyle](http://docs.google.com/javax/swing/LayoutStyle.html) | [**getLayoutStyle**](http://docs.google.com/javax/swing/GroupLayout.html#getLayoutStyle())()            Returns the LayoutStyle used for calculating the preferred gap between components. |
| void | [**invalidateLayout**](http://docs.google.com/javax/swing/GroupLayout.html#invalidateLayout(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Invalidates the layout, indicating that if the layout manager has cached information it should be discarded. |
| void | [**layoutContainer**](http://docs.google.com/javax/swing/GroupLayout.html#layoutContainer(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Lays out the specified container. |
| void | [**linkSize**](http://docs.google.com/javax/swing/GroupLayout.html#linkSize(java.awt.Component...))([Component](http://docs.google.com/java/awt/Component.html)... components)            Forces the specified components to have the same size regardless of their preferred, minimum or maximum sizes. |
| void | [**linkSize**](http://docs.google.com/javax/swing/GroupLayout.html#linkSize(int,%20java.awt.Component...))(int axis, [Component](http://docs.google.com/java/awt/Component.html)... components)            Forces the specified components to have the same size along the specified axis regardless of their preferred, minimum or maximum sizes. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**maximumLayoutSize**](http://docs.google.com/javax/swing/GroupLayout.html#maximumLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Returns the maximum size for the specified container. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**minimumLayoutSize**](http://docs.google.com/javax/swing/GroupLayout.html#minimumLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Returns the minimum size for the specified container. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**preferredLayoutSize**](http://docs.google.com/javax/swing/GroupLayout.html#preferredLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) parent)            Returns the preferred size for the specified container. |
| void | [**removeLayoutComponent**](http://docs.google.com/javax/swing/GroupLayout.html#removeLayoutComponent(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) component)            Notification that a Component has been removed from the parent container. |
| void | [**replace**](http://docs.google.com/javax/swing/GroupLayout.html#replace(java.awt.Component,%20java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) existingComponent, [Component](http://docs.google.com/java/awt/Component.html) newComponent)            Replaces an existing component with a new one. |
| void | [**setAutoCreateContainerGaps**](http://docs.google.com/javax/swing/GroupLayout.html#setAutoCreateContainerGaps(boolean))(boolean autoCreateContainerPadding)            Sets whether a gap between the container and components that touch the border of the container should automatically be created. |
| void | [**setAutoCreateGaps**](http://docs.google.com/javax/swing/GroupLayout.html#setAutoCreateGaps(boolean))(boolean autoCreatePadding)            Sets whether a gap between components should automatically be created. |
| void | [**setHonorsVisibility**](http://docs.google.com/javax/swing/GroupLayout.html#setHonorsVisibility(boolean))(boolean honorsVisibility)            Sets whether component visiblity is considered when sizing and positioning components. |
| void | [**setHonorsVisibility**](http://docs.google.com/javax/swing/GroupLayout.html#setHonorsVisibility(java.awt.Component,%20java.lang.Boolean))([Component](http://docs.google.com/java/awt/Component.html) component, [Boolean](http://docs.google.com/java/lang/Boolean.html) honorsVisibility)            Sets whether the component's visiblity is considered for sizing and positioning. |
| void | [**setHorizontalGroup**](http://docs.google.com/javax/swing/GroupLayout.html#setHorizontalGroup(javax.swing.GroupLayout.Group))([GroupLayout.Group](http://docs.google.com/javax/swing/GroupLayout.Group.html) group)            Sets the Group that positions and sizes components along the horizontal axis. |
| void | [**setLayoutStyle**](http://docs.google.com/javax/swing/GroupLayout.html#setLayoutStyle(javax.swing.LayoutStyle))([LayoutStyle](http://docs.google.com/javax/swing/LayoutStyle.html) layoutStyle)            Sets the LayoutStyle used to calculate the preferred gaps between components. |
| void | [**setVerticalGroup**](http://docs.google.com/javax/swing/GroupLayout.html#setVerticalGroup(javax.swing.GroupLayout.Group))([GroupLayout.Group](http://docs.google.com/javax/swing/GroupLayout.Group.html) group)            Sets the Group that positions and sizes components along the vertical axis. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/javax/swing/GroupLayout.html#toString())()            Returns a string representation of this GroupLayout. |

| **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)) |

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

### DEFAULT\_SIZE

public static final int **DEFAULT\_SIZE**

Indicates the size from the component or gap should be used for a particular range value.

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

### PREFERRED\_SIZE

public static final int **PREFERRED\_SIZE**

Indicates the preferred size from the component or gap should be used for a particular range value.

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

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

### GroupLayout

public **GroupLayout**([Container](http://docs.google.com/java/awt/Container.html) host)

Creates a GroupLayout for the specified Container.

**Parameters:**host - the Container the GroupLayout is the LayoutManager for **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if host is null

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

### setHonorsVisibility

public void **setHonorsVisibility**(boolean honorsVisibility)

Sets whether component visiblity is considered when sizing and positioning components. A value of true indicates that non-visible components should not be treated as part of the layout. A value of false indicates that components should be positioned and sized regardless of visibility.

A value of false is useful when the visibility of components is dynamically adjusted and you don't want surrounding components and the sizing to change.

The specified value is used for components that do not have an explicit visibility specified.

The default is true.

**Parameters:**honorsVisibility - whether component visiblity is considered when sizing and positioning components**See Also:**[setHonorsVisibility(Component,Boolean)](http://docs.google.com/javax/swing/GroupLayout.html#setHonorsVisibility(java.awt.Component,%20java.lang.Boolean))

### getHonorsVisibility

public boolean **getHonorsVisibility**()

Returns whether component visiblity is considered when sizing and positioning components.

**Returns:**whether component visiblity is considered when sizing and positioning components

### setHonorsVisibility

public void **setHonorsVisibility**([Component](http://docs.google.com/java/awt/Component.html) component,  
 [Boolean](http://docs.google.com/java/lang/Boolean.html) honorsVisibility)

Sets whether the component's visiblity is considered for sizing and positioning. A value of Boolean.TRUE indicates that if component is not visible it should not be treated as part of the layout. A value of false indicates that component is positioned and sized regardless of it's visibility. A value of null indicates the value specified by the single argument method setHonorsVisibility should be used.

If component is not a child of the Container this GroupLayout is managine, it will be added to the Container.

**Parameters:**component - the componenthonorsVisibility - whether component's visiblity should be considered for sizing and positioning **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if component is null**See Also:**[setHonorsVisibility(Component,Boolean)](http://docs.google.com/javax/swing/GroupLayout.html#setHonorsVisibility(java.awt.Component,%20java.lang.Boolean))

### setAutoCreateGaps

public void **setAutoCreateGaps**(boolean autoCreatePadding)

Sets whether a gap between components should automatically be created. For example, if this is true and you add two components to a SequentialGroup a gap between the two components is automatically be created. The default is false.

**Parameters:**autoCreatePadding - whether a gap between components is automatically created

### getAutoCreateGaps

public boolean **getAutoCreateGaps**()

Returns true if gaps between components are automatically created.

**Returns:**true if gaps between components are automatically created

### setAutoCreateContainerGaps

public void **setAutoCreateContainerGaps**(boolean autoCreateContainerPadding)

Sets whether a gap between the container and components that touch the border of the container should automatically be created. The default is false.

**Parameters:**autoCreateContainerPadding - whether a gap between the container and components that touch the border of the container should automatically be created

### getAutoCreateContainerGaps

public boolean **getAutoCreateContainerGaps**()

Returns true if gaps between the container and components that border the container are automatically created.

**Returns:**true if gaps between the container and components that border the container are automatically created

### setHorizontalGroup

public void **setHorizontalGroup**([GroupLayout.Group](http://docs.google.com/javax/swing/GroupLayout.Group.html) group)

Sets the Group that positions and sizes components along the horizontal axis.

**Parameters:**group - the Group that positions and sizes components along the horizontal axis **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if group is null

### setVerticalGroup

public void **setVerticalGroup**([GroupLayout.Group](http://docs.google.com/javax/swing/GroupLayout.Group.html) group)

Sets the Group that positions and sizes components along the vertical axis.

**Parameters:**group - the Group that positions and sizes components along the vertical axis **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if group is null

### createSequentialGroup

public [GroupLayout.SequentialGroup](http://docs.google.com/javax/swing/GroupLayout.SequentialGroup.html) **createSequentialGroup**()

Creates and returns a SequentialGroup.

**Returns:**a new SequentialGroup

### createParallelGroup

public [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) **createParallelGroup**()

Creates and returns a ParallelGroup with an alignment of Alignment.LEADING. This is a cover method for the more general createParallelGroup(Alignment) method.

**Returns:**a new ParallelGroup**See Also:**[createParallelGroup(Alignment)](http://docs.google.com/javax/swing/GroupLayout.html#createParallelGroup(javax.swing.GroupLayout.Alignment))

### createParallelGroup

public [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) **createParallelGroup**([GroupLayout.Alignment](http://docs.google.com/javax/swing/GroupLayout.Alignment.html) alignment)

Creates and returns a ParallelGroup with the specified alignment. This is a cover method for the more general createParallelGroup(Alignment,boolean) method with true supplied for the second argument.

**Parameters:**alignment - the alignment for the elements of the group **Returns:**a new ParallelGroup **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if alignment is null**See Also:**[createBaselineGroup(boolean, boolean)](http://docs.google.com/javax/swing/GroupLayout.html#createBaselineGroup(boolean,%20boolean)), [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html)

### createParallelGroup

public [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) **createParallelGroup**([GroupLayout.Alignment](http://docs.google.com/javax/swing/GroupLayout.Alignment.html) alignment,  
 boolean resizable)

Creates and returns a ParallelGroup with the specified alignment and resize behavior. The alignment argument specifies how children elements are positioned that do not fill the group. For example, if a ParallelGroup with an alignment of TRAILING is given 100 and a child only needs 50, the child is positioned at the position 50 (with a component orientation of left-to-right).

Baseline alignment is only useful when used along the vertical axis. A ParallelGroup created with a baseline alignment along the horizontal axis is treated as LEADING.

Refer to [ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) for details on the behavior of baseline groups.

**Parameters:**alignment - the alignment for the elements of the groupresizable - true if the group is resizable; if the group is not resizable the preferred size is used for the minimum and maximum size of the group **Returns:**a new ParallelGroup **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if alignment is null**See Also:**[createBaselineGroup(boolean, boolean)](http://docs.google.com/javax/swing/GroupLayout.html#createBaselineGroup(boolean,%20boolean)), [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html)

### createBaselineGroup

public [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html) **createBaselineGroup**(boolean resizable,  
 boolean anchorBaselineToTop)

Creates and returns a ParallelGroup that aligns it's elements along the baseline.

**Parameters:**resizable - whether the group is resizableanchorBaselineToTop - whether the baseline is anchored to the top or bottom of the group**See Also:**[createBaselineGroup(boolean, boolean)](http://docs.google.com/javax/swing/GroupLayout.html#createBaselineGroup(boolean,%20boolean)), [GroupLayout.ParallelGroup](http://docs.google.com/javax/swing/GroupLayout.ParallelGroup.html)

### linkSize

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

Forces the specified components to have the same size regardless of their preferred, minimum or maximum sizes. Components that are linked are given the maximum of the preferred size of each of the linked components. For example, if you link two components with a preferred width of 10 and 20, both components are given a width of 20.

This can be used multiple times to force any number of components to share the same size.

Linked Components are not be resizable.

**Parameters:**components - the Components that are to have the same size **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if components is null, or contains null**See Also:**[linkSize(int,Component[])](http://docs.google.com/javax/swing/GroupLayout.html#linkSize(int,%20java.awt.Component...))

### linkSize

public void **linkSize**(int axis,  
 [Component](http://docs.google.com/java/awt/Component.html)... components)

Forces the specified components to have the same size along the specified axis regardless of their preferred, minimum or maximum sizes. Components that are linked are given the maximum of the preferred size of each of the linked components. For example, if you link two components along the horizontal axis and the preferred width is 10 and 20, both components are given a width of 20.

This can be used multiple times to force any number of components to share the same size.

Linked Components are not be resizable.

**Parameters:**components - the Components that are to have the same sizeaxis - the axis to link the size along; one of SwingConstants.HORIZONTAL or SwingConstans.VERTICAL **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if components is null, or contains null; or axis is not SwingConstants.HORIZONTAL or SwingConstants.VERTICAL

### replace

public void **replace**([Component](http://docs.google.com/java/awt/Component.html) existingComponent,  
 [Component](http://docs.google.com/java/awt/Component.html) newComponent)

Replaces an existing component with a new one.

**Parameters:**existingComponent - the component that should be removed and replaced with newComponentnewComponent - the component to put in existingComponent's place **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if either of the components are null or existingComponent is not being managed by this layout manager

### setLayoutStyle

public void **setLayoutStyle**([LayoutStyle](http://docs.google.com/javax/swing/LayoutStyle.html) layoutStyle)

Sets the LayoutStyle used to calculate the preferred gaps between components. A value of null indicates the shared instance of LayoutStyle should be used.

**Parameters:**layoutStyle - the LayoutStyle to use**See Also:**[LayoutStyle](http://docs.google.com/javax/swing/LayoutStyle.html)

### getLayoutStyle

public [LayoutStyle](http://docs.google.com/javax/swing/LayoutStyle.html) **getLayoutStyle**()

Returns the LayoutStyle used for calculating the preferred gap between components. This returns the value specified to setLayoutStyle, which may be null.

**Returns:**the LayoutStyle used for calculating the preferred gap between components

### addLayoutComponent

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

Notification that a Component has been added to the parent container. You should not invoke this method directly, instead you should use one of the Group methods to add a Component.

**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 string to be associated with the componentcomponent - the Component to be added

### removeLayoutComponent

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

Notification that a Component has been removed from the parent container. You should not invoke this method directly, instead invoke remove on the parent Container.

**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:**component - the component to be removed**See Also:**[Component.remove(java.awt.MenuComponent)](http://docs.google.com/java/awt/Component.html#remove(java.awt.MenuComponent))

### preferredLayoutSize

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

Returns the preferred size for the specified 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 to return the preferred size for **Returns:**the preferred size for parent **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if parent is not the same Container this was created with [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if any of the components added to this layout are not in both a horizontal and vertical group**See Also:**[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)

Returns the minimum size for the specified 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 to return the size for **Returns:**the minimum size for parent **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if parent is not the same Container that this was created with [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if any of the components added to this layout are not in both a horizontal and vertical group**See Also:**[Container.getMinimumSize()](http://docs.google.com/java/awt/Container.html#getMinimumSize())

### layoutContainer

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

Lays out the specified container.

**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 to be laid out **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if any of the components added to this layout are not in both a horizontal and vertical group

### addLayoutComponent

public void **addLayoutComponent**([Component](http://docs.google.com/java/awt/Component.html) component,  
 [Object](http://docs.google.com/java/lang/Object.html) constraints)

Notification that a Component has been added to the parent container. You should not invoke this method directly, instead you should use one of the Group methods to add a Component.

**Specified by:**[addLayoutComponent](http://docs.google.com/java/awt/LayoutManager2.html#addLayoutComponent(java.awt.Component,%20java.lang.Object)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html) **Parameters:**component - the component addedconstraints - description of where to place the component

### maximumLayoutSize

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

Returns the maximum size for the specified container.

**Specified by:**[maximumLayoutSize](http://docs.google.com/java/awt/LayoutManager2.html#maximumLayoutSize(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html) **Parameters:**parent - the container to return the size for **Returns:**the maximum size for parent **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if parent is not the same Container that this was created with [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if any of the components added to this layout are not in both a horizontal and vertical group**See Also:**[Container.getMaximumSize()](http://docs.google.com/java/awt/Container.html#getMaximumSize())

### getLayoutAlignmentX

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

Returns the alignment along the x axis. This specifies how the component would like to be aligned relative to other components. The value should be a number between 0 and 1 where 0 represents alignment along the origin, 1 is aligned the furthest away from the origin, 0.5 is centered, etc.

**Specified by:**[getLayoutAlignmentX](http://docs.google.com/java/awt/LayoutManager2.html#getLayoutAlignmentX(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html) **Parameters:**parent - the Container hosting this LayoutManager **Returns:**the alignment; this implementation returns .5 **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if parent is not the same Container that this was created with

### getLayoutAlignmentY

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

Returns the alignment along the y axis. This specifies how the component would like to be aligned relative to other components. The value should be a number between 0 and 1 where 0 represents alignment along the origin, 1 is aligned the furthest away from the origin, 0.5 is centered, etc.

**Specified by:**[getLayoutAlignmentY](http://docs.google.com/java/awt/LayoutManager2.html#getLayoutAlignmentY(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html) **Parameters:**parent - the Container hosting this LayoutManager **Returns:**alignment; this implementation returns .5 **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if parent is not the same Container that this was created with

### invalidateLayout

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

Invalidates the layout, indicating that if the layout manager has cached information it should be discarded.

**Specified by:**[invalidateLayout](http://docs.google.com/java/awt/LayoutManager2.html#invalidateLayout(java.awt.Container)) in interface [LayoutManager2](http://docs.google.com/java/awt/LayoutManager2.html) **Parameters:**parent - the Container hosting this LayoutManager **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if parent is not the same Container that this was created with

### toString

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

Returns a string representation of this GroupLayout. This method is intended to be used for debugging purposes, and the content and format of the returned string may vary between implementations.

**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 GroupLayout

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/GroupLayout.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/GrayFilter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/GroupLayout.Alignment.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/GroupLayout.html)    [**NO FRAMES**](http://docs.google.com/GroupLayout.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#4d34og8) | [CONSTR](#3rdcrjn) | [METHOD](#lnxbz9) |

[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).