SWING - GROUPLAYOUT CLASS

Introduction

The class **GroupLayout** hierarchically groups components in order to position them in a Container .

Class declaration

Following is the declaration for **javax.swing.GroupLayout** class:

```
public class GroupLayout
   extends Object
      implements LayoutManager2
```

Field

Following are the fields for **javax.swing.GroupLayout** class:

- **static int DEFAULT_SIZE** -- Indicates the size from the component or gap should be used for a particular range value.
- **static int PREFERRED_SIZE** -- Indicates the preferred size from the component or gap should be used for a particular range value.

Class constructors

S.N.	Constructor & Description
1	**GroupLayout***Containerhost* Creates a GroupLayout for the specified Container.

Class methods

S.N.	Method & Description
1	**void addLayoutComponent***Componentcomponent, Objectconstraints* Notification that a Component has been added to the parent container.
2	**void addLayoutComponent***Stringname, Componentcomponent* Notification that a Component has been added to the parent container.
3	**GroupLayout.ParallelGroup createBaselineGroup** *booleanresizable, booleananchorBaselineToTop* Creates and returns a ParallelGroup that aligns it's elements along the baseline.
4	**GroupLayout.ParallelGroup createParallelGroup** Creates and returns a ParallelGroup with an alignment of Alignment.LEADING.

5 **GroupLayout.ParallelGroup createParallelGroup***GroupLayout. Alignmentalignment*

Creates and returns a ParallelGroup with the specified alignment.

6 **GroupLayout.ParallelGroup createParallelGroup**
GroupLayout. Alignmentalignment, booleanresizable

Creates and returns a ParallelGroup with the specified alignment and resize behavior.

7 **GroupLayout.SequentialGroup createSequentialGroup**

Creates and returns a SequentialGroup.

8 **boolean getAutoCreateContainerGaps**

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

9 **boolean getAutoCreateGaps**

Returns true if gaps between components are automatically created.

10 **boolean getHonorsVisibility**

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

11 **float getLayoutAlignmentX***Containerparent*

Returns the alignment along the x axis.

12 **float getLayoutAlignmentY***Containerparent*

Returns the alignment along the y axis.

13 **LayoutStyle getLayoutStyle**

Returns the LayoutStyle used for calculating the preferred gap between components.

14 **void invalidateLayout***Containerparent*

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

15 **void layoutContainer***Containerparent*

Lays out the specified container.

16 **void linkSize***Component. . . components*

Forces the specified components to have the same size regardless of their preferred, minimum or maximum sizes.

17 **void linkSize***intaxis, Component. . . components*

Forces the specified components to have the same size along the specified axis regardless of their preferred, minimum or maximum sizes.

18 **Dimension maximumLayoutSize***Containerparent*

Returns the maximum size for the specified container.

19 **Dimension minimumLayoutSize***Containerparent*

Returns the minimum size for the specified container.

20 **Dimension preferredLayoutSize***Containerparent*

Returns the preferred size for the specified container.

21 **void removeLayoutComponent***Componentcomponent*

Notification that a Component has been removed from the parent container.

22 **void replace***ComponentexistingComponent, ComponentnewComponent*

Replaces an existing component with a new one.

23 **void setAutoCreateContainerGaps***booleanautoCreateContainerPadding*

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

24 **void setAutoCreateGaps***booleanautoCreatePadding*

Sets whether a gap between components should automatically be created.

25 **void setHonorsVisibility***booleanhonorsVisibility*

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

26 **void setHonorsVisibility***Componentcomponent, BooleanhonorsVisibility*

Sets whether the component's visiblity is considered for sizing and positioning.

27 **void setHorizontalGroup***GroupLayout. Groupgroup*

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

28 **void setLayoutStyle***LayoutStylelayoutStyle*

Sets the LayoutStyle used to calculate the preferred gaps between components.

29 **void setVerticalGroup***GroupLayout. Groupgroup*

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

30 **String toString**

Returns a string representation of this GroupLayout.

Methods inherited

This class inherits methods from the following classes:

- java.lang.Object

GroupLayout Example

Create the following java program using any editor of your choice in say **D:/ > SWING > com > tutorialspoint > gui >**

SwingLayoutDemo.java

```
package com.tutorialspoint.gui;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class SwingLayoutDemo {
   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;
   private JLabel msglabel;

   public SwingLayoutDemo(){
      prepareGUI();
   }

   public static void main(String[] args){
      SwingLayoutDemo swingLayoutDemo = new SwingLayoutDemo();
      swingLayoutDemo.showGroupLayoutDemo();
   }

   private void prepareGUI(){
      mainFrame = new JFrame("Java SWING Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));

      headerLabel = new JLabel("",JLabel.CENTER );
      statusLabel = new JLabel("",JLabel.CENTER);

      statusLabel.setSize(350,100);
      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());

      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }

   private void showGroupLayoutDemo(){
      headerLabel.setText("Layout in action: GroupLayout");

      JPanel panel = new JPanel();
      // panel.setBackground(Color.darkGray);
      panel.setSize(200,200);
      GroupLayout layout = new GroupLayout(panel);
      layout.setAutoCreateGaps(true);
      layout.setAutoCreateContainerGaps(true);
      JButton btn1 = new JButton("Button 1");
      JButton btn2 = new JButton("Button 2");
      JButton btn3 = new JButton("Button 3");

      layout.setHorizontalGroup(layout.createSequentialGroup()
```

```
            .addComponent(btn1)
            .addGroup(layout.createSequentialGroup()
               .addGroup(layout.createParallelGroup(
                  GroupLayout.Alignment.LEADING)
                  .addComponent(btn2)
                  .addComponent(btn3)
               )
            )
         );

         layout.setVerticalGroup(layout.createSequentialGroup()
            .addComponent(btn1)
            .addComponent(btn2)
            .addComponent(btn3)
         );
         panel.setLayout(layout);
         controlPanel.add(panel);

         mainFrame.setVisible(true);
      }
   }
```

Compile the program using command prompt. Go to **D:/ > SWING** and type the following command.

```
D:\SWING>javac com\tutorialspoint\gui\SwingLayoutDemo.java
```

If no error comes that means compilation is successful. Run the program using following command.

```
D:\SWING>java com.tutorialspoint.gui.SwingLayoutDemo
```

Verify the following output