Containers and Layout Managers

Layout Managers allow components to be added using different layouts. All these classes are known as container classes. We have the following layout managers:

- BorderLayout
- BoxLayout
- FlowLayout
- GridLayout
- GroupLayout
- Coordinate Layout
- Table Layout

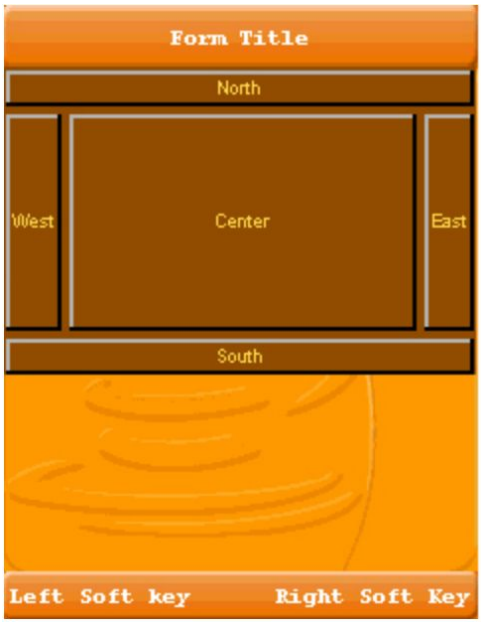
But we will currently be covering only three of these:

BorderLayout : A BorderLayout object has five areas. These areas are specified by the BorderLayout constants:

- Center
- East
- North
- South
- West

When adding a component to a container, specify the component's location (for example, BorderLayout.CENTER) as one of the arguments to the addComponent method. If this component is missing from a container, controlled by a BorderLayout object, make sure that the component's location was specified and that no other component was placed in the same location.
addComponent(BorderLayout.CENTER, component) // preferred
or
addComponent("Center", component) // valid but error prone

The center area gets as much of the available space as possible. The other areas expand only as much as necessary to fit the components that have been added to it. Often a container uses only one or two of the areas of the BorderLayout object — just the center, or the center and the bottom.




```
//Sample Code
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.BorderLayout;

public class BorderLayout_Example extends JFrame
{
   public static final int WIDTH=500;
   public static final int HEIGHT=600;


public BorderLayout_Example()
   {
         super("Border layout example");
         setSize(WIDTH,HEIGHT);
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         setLayout(new BorderLayout());

         /* Components are added to North, South, Center, East
         and West directions
         The space is divided between regions as follows: Regions
         are allocated space in the order first north and south,
         second east and west, and last center. So, in particular,
         if there is nothing in the north region, then the east and
         west regions will extend to the top of the space.*/
```

```
        JLabel Label1=new JLabel("Label North");
        add(Label1,BorderLayout.NORTH);

        JLabel Label2=new JLabel("Label South");
        add(Label2,BorderLayout.SOUTH);

        JLabel Label3=new JLabel("Label Center");
        add(Label3,BorderLayout.CENTER);

        JLabel Label4=new JLabel("Label East");
        add(Label4,BorderLayout.EAST);

        JLabel Label5=new JLabel("Label West");
        add(Label5,BorderLayout.WEST);

        /* Later we will learn that using Panels we can
        group items so that more than one item can  be
        placed in each region.

        */
    }
}

//Demo  Code
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.BorderLayout;

public class BorderLayout_Demo
{
    public static void main(String[] args)
    {
        BorderLayout_Example B1=new BorderLayout_Example();
        B1.setVisible(true);
    }
}
//Sample Output
```

Constructors

- BorderLayout()
  Constructs a new border layout with no gaps between components.

- BorderLayout(int hgap, int vgap)
  Constructs a border layout with the specified gaps between components.

FlowLayout

The FlowLayout class provides a very simple layout manager that is the **default layout manager** for Container objects.
The FlowLayout class puts components in a row, sized at their preferred size. If the horizontal space in the container is too small to put all the components in one row, the FlowLayout class uses multiple rows. To align the row to the left, right, or center, use a FlowLayout constructor that takes an alignment argument.

When constructing a FlowLayout manager you can select either the **Left, Right, or Center** option to set up the component's orientation. The default alignment is Left.

Constructors

- FlowLayout()
  Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.

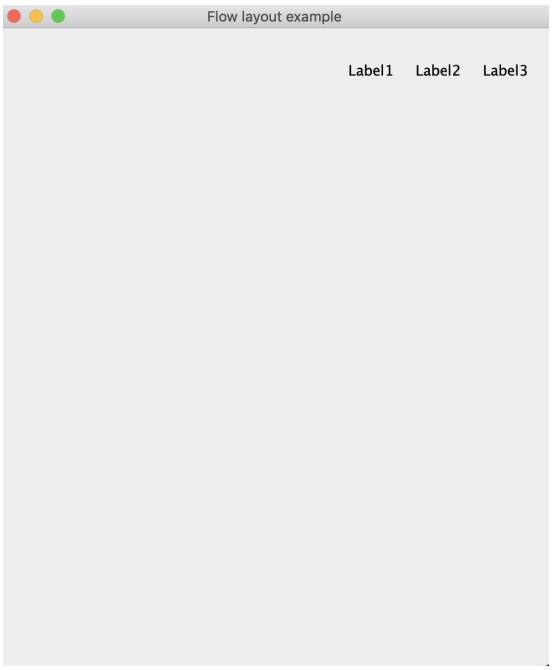
- FlowLayout(int align)
  Constructs a new FlowLayout with the specified alignment and a default 5-unit horizontal and vertical gap.

- FlowLayout(int align, int hgap, int vgap)
  Creates a new flow layout manager with the indicated alignment and the indicated horizontal and vertical gaps.


```
public FlowLayout_Example()
  {
        super("Flow layout example");
        setSize(WIDTH,HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout(FlowLayout.RIGHT,20,30));
        /*sets alignment for layout to right (or center/left),
        horizontal gap, vertical gap */

        JLabel Label1=new JLabel("Label1");
        add(Label1);


        JLabel Label2=new JLabel("Label2");
        add(Label2);
```

```
        JLabel Label3=new JLabel("Label3");
        add(Label3);
    }
```

//Sample Output



GridLayout

A GridLayout object places components in a grid of cells. Each component takes all the available space within its cell, and each cell is exactly the same size.

Constructors

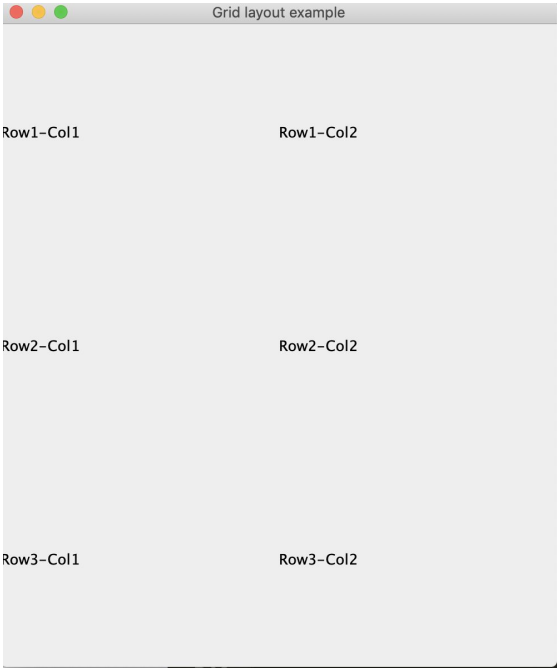
- GridLayout()
  Creates a grid layout with a default of one column per component, in a single row.

- GridLayout(int rows, int cols)
  Creates a grid layout with the specified number of rows and columns.

- GridLayout(int rows, int cols, int hgap, int vgap)
  Creates a grid layout with the specified number of rows and columns.


```
//Sample Code
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.*;

public class GridLayout_Example extends JFrame
{
   public static final int WIDTH=500;
   public static final int HEIGHT=600;

   public GridLayout_Example()
   {
         super("Grid layout example");
         setSize(WIDTH,HEIGHT);
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         setLayout(new GridLayout(3,2));
         /*sets grid to 3 rows and 2 columns */

         /* When using a GridLayout manager, each component
          is stretched so that it completely fills its grid
          position.*/
         JLabel Label1=new JLabel("Row1-Col1");
         add(Label1);
```

```
        JLabel Label2=new JLabel("Row1-Col2");
        add(Label2);

        JLabel Label3=new JLabel("Row2-Col1");
        add(Label3);

        JLabel Label4=new JLabel("Row2-Col2");
        add(Label4);

        JLabel Label5=new JLabel("Row3-Col1");
        add(Label5);

        JLabel Label6=new JLabel("Row3-Col2");
        add(Label6);

        /*If you add seven or eight items, a third column
         is automatically added, and so forth. If you add
         fewer than six components, there will be two rows
         and a reduced number of columns.*/

        //JLabel Label7=new JLabel("Row-Col1");

    }

}

//Sample Output
```



Panel: Panel is the simplest container class. A panel provides space in which an application can attach any other component, including other panels. The default layout manager for a panel is the FlowLayout layout manager.

A panel is an object of the class **JPanel**, which is a very simple container class that does little more than group objects. A JPanel object is analogous to the braces used to combine a number of simpler Java statements into a single larger Java statement. It groups smaller objects, such as buttons and labels, into a larger component (the JPanel). You can then put the JPanel object in a JFrame. **Thus, one of the main functions of JPanel objects is to subdivide a JFrame (or other container) into different areas.**

Constructors

- Panel()
  Creates a new panel using the default layout manager.

- Panel(LayoutManager layout)
  Creates a new panel with the specified layout manager.

//Sample Code

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.*;
```

```
import javax.swing.JPanel;

public class Panel_Example1 extends JFrame
{
   public static final int WIDTH=500;
   public static final int HEIGHT=600;

   public Panel_Example1()
   {
         super("Panel example");
         setSize(WIDTH,HEIGHT);
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         setLayout(new BorderLayout());

         /* Components are Panel based on layout*/
         JPanel PanelNorth=new JPanel(new GridLayout());
         JLabel Label1North=new JLabel("Label1 North");
         PanelNorth.add(Label1North);
         JLabel Label2North=new JLabel("Label2 North");
         PanelNorth.add(Label2North);
         PanelNorth.setBackground(Color.PINK);
         add(PanelNorth,BorderLayout.NORTH);

         JPanel PanelSouth=new JPanel(new FlowLayout(FlowLayout.CENTER,20,30));

         JLabel Label1South=new JLabel("Label1 South");
         PanelSouth.add(Label1South);
         PanelSouth.setBackground(Color.PINK);

         JLabel Label2South=new JLabel("Label2 South");
         PanelSouth.add(Label2South);

         add(PanelSouth,BorderLayout.SOUTH);

         JLabel Label4=new JLabel("Label East");
         add(Label4,BorderLayout.EAST);

         JLabel Label5=new JLabel("Label West");
         add(Label5,BorderLayout.WEST);

         /* Later we will learn that using Panels we can
         group items so that more than one item can  be
         placed in each region.
```

```
        */
    }
}
```

//Sample Output