SNS COLLEGE OF TECHNOLOGY

(an autonomous institution)

Coimbatore - 35

19CST102 - OBJECT ORIENTED PROGRAMMING

I YEAR / II SEMESTER

UNIT V – DESIGNING JAVA APPLETS

TOPIC : LAYOUT HANDLERS

Guided by :

Mr.Selvakumar. N

AP/CSE

Presented by:

Rakshana Lakshmi G

713522CS121

LAYOUT HANDLERS

- Layout refers to the arrangement of components within the container or we can say that layout is placing the components at a particular **position within the container**.

- The task of laying out the controls is done automatically by the Layout Manager

TYPES OF LAYOUT HANDLER :

- Flow layout
- Grid layout
- Border layout
- Card layout
- Box layout
- Grid bag layout

BORDER LAYOUT

- The Border Layout is used to arrange the components in five regions: north, south, east, west, and center.
- Each region (area) may contain one component only.
- It is the default layout of a frame or window.

Constructors :

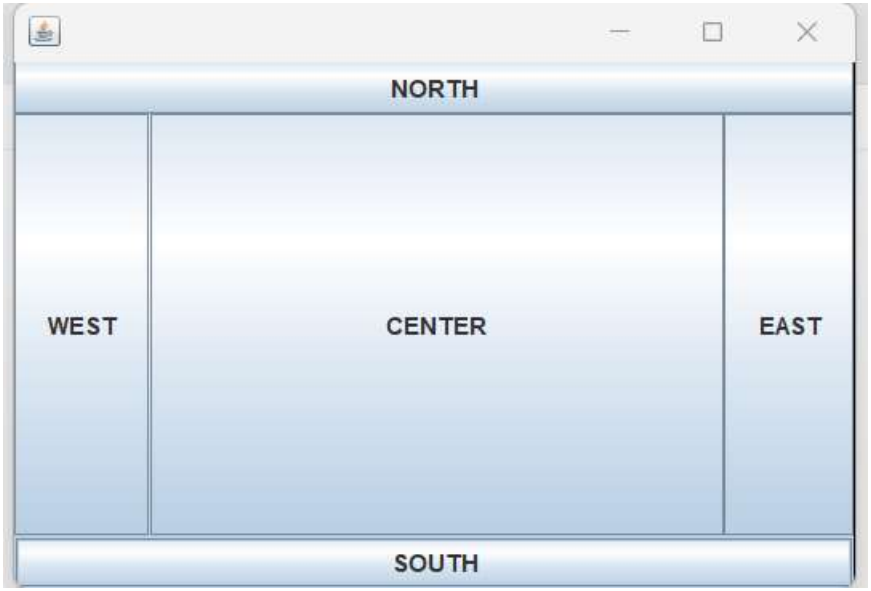
- **BorderLayout():** creates a border layout but with no gaps between the components.
- **BorderLayout(int hgap, int vgap):** creates a border layout with the given horizontal and vertical gaps between the components.

Example program :

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

class Border
{
JFrame f;
Border()
{
    f = new JFrame();

     // creating buttons
    JButton b1 = new JButton("NORTH");
    JButton b2 = new JButton("SOUTH");
    JButton b3 = new JButton("EAST");
    JButton b4 = new JButton("WEST");
    JButton b5 = new JButton("CENTER");
```

```
 f.add(b1, BorderLayout.NORTH);
    f.add(b2, BorderLayout.SOUTH);
    f.add(b3, BorderLayout.EAST);
    f.add(b4, BorderLayout.WEST);
    f.add(b5, BorderLayout.CENTER);
    f.setSize(300, 300);
    f.setVisible(true);
}
public static void main(String[] args) {
    new Border();
}
}
```

OUTPUT:



FLOW LAYOUT

- Arrange in row or a column
- If there is no enough space it wraps the next row or a column

Constructors

- **FlowLayout():** creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.
- **FlowLayout(int align):** creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.
- **FlowLayout(int align, int hgap, int vgap):** creates a flow layout with the given alignment and the given horizontal and vertical gap.

Example program :

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

 class MyFlowLayout{
JFrame f;
MyFlowLayout(){
    f=new JFrame();

    JButton b1=new JButton("1");
    JButton b2=new JButton("2");
    JButton b3=new JButton("3");
    JButton b4=new JButton("4");
    JButton b5=new JButton("5");
```

Output

```
f.add(b1); f.add(b2); f.add(b3); f.add(b4); f.add(b5);


    f.setLayout(new FlowLayout(FlowLayout.RIGHT));

    f.setSize(300,300);
    f.setVisible(true);
}
public static void main(String[] args) {
    new MyFlowLayout();
}
}
```

OUTPUT:

GRID LAYOUT

Grid layout is to arrange components in rectangular grid

Constructors

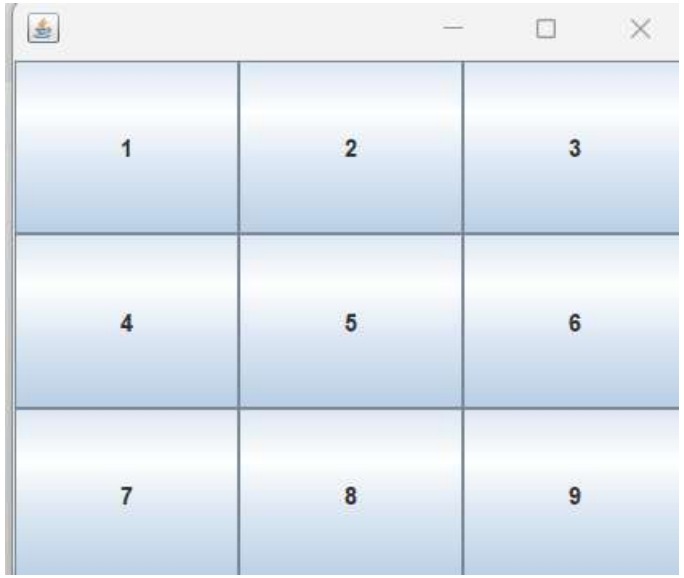
- **GridLayout():** creates a grid layout with one column per component in a row.
- **GridLayout(int rows, int columns):** creates a grid layout with the given rows and columns but no gaps between the components.
- **GridLayout(int rows, int columns, int hgap, int vgap):** creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.

Example program :

```
import java.awt.*;
import javax.swing.*;
 class MyGridLayout{
JFrame f;
MyGridLayout(){
    f=new JFrame();
    JButton b1=new JButton("1");
    JButton b2=new JButton("2");
    JButton b3=new JButton("3");
    JButton b4=new JButton("4");
    JButton b5=new JButton("5");
    JButton b6=new JButton("6");
    JButton b7=new JButton("7");
    JButton b8=new JButton("8");
    JButton b9=new JButton("9");
```

```
 f.add(b1); f.add(b2); f.add(b3);
    f.add(b4); f.add(b5); f.add(b6);
    f.add(b7); f.add(b8); f.add(b9);


    f.setLayout(new GridLayout(3,3));
    f.setSize(300,300);
    f.setVisible(true);
}
public static void main(String[] args) {
    new MyGridLayout();
}
}
```

OUTPUT:



CARD LAYOUT

- It treats components as a card
- This cardlayout manages the components in such a manner that only one component is visible at a time

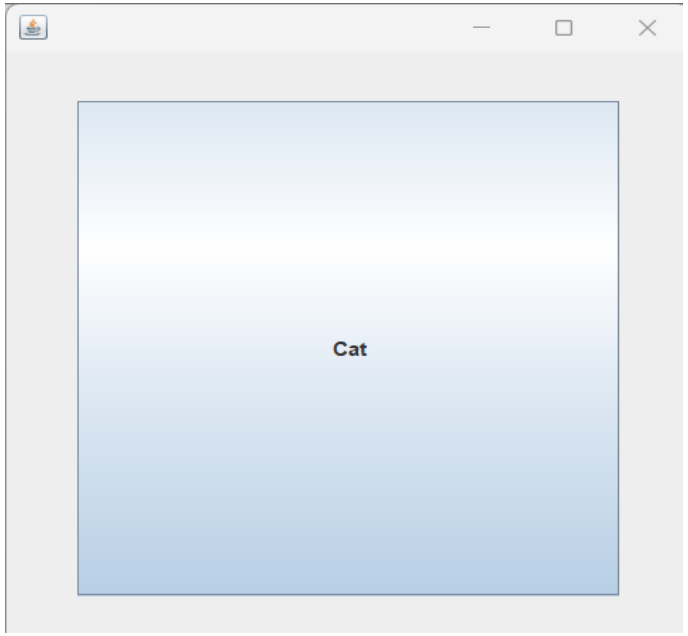
Constructors

- **CardLayout():** creates a card layout with zero horizontal and vertical gap.
- **CardLayout(int hgap, int vgap):** creates a card layout with the given horizontal and vertical gap

Example program :

```
import java.awt.*;
import java.awt.event.*;

import javax.swing.*;
    class CardLayoutExample2 extends JFrame
implements ActionListener{
CardLayout card;
JButton b1,b2,b3;
Container c;
   CardLayoutExample2(){
          c=getContentPane();
         card=new CardLayout(40,30);
    c.setLayout(card);
          b1=new JButton("Apple");
          b2=new JButton("Boy");
          b3=new JButton("Cat");
```

```
b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

        c.add("a",b1);c.add("b",b2);c.add("c",b3);

    }
    public void actionPerformed(ActionEvent e) {
    card.next(c);
    }

    public static void main(String[] args) {
        CardLayoutExample2 cl=new CardLayoutExample2();
        cl.setSize(400,400);
        cl.setVisible(true);
        cl.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```



BOX LAYOUT

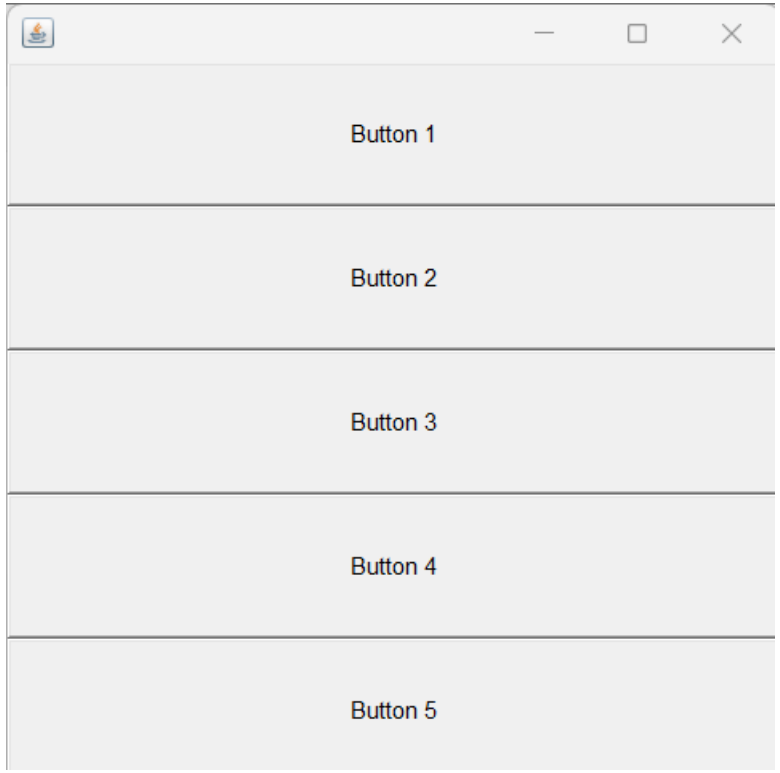
Constructors

- **BoxLayout(Container c, int axis):** creates a box layout that arranges the components with the given axis.

Example program :

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

 class BoxLayoutExample1 extends Frame {
 Button buttons[];

 public BoxLayoutExample1 () {
   buttons = new Button [5];

   for (int i = 0;i<5;i++) {
      buttons[i] = new Button ("Button " + (i + 1));
      // adding the buttons so that it can be
displayed
      add (buttons[i]);
    }
```

```
 // the buttons will be placed horizontally
setLayout (new BoxLayout (this, BoxLayout.Y_AXIS));
setSize(400,400);
setVisible(true);
}

public static void main(String args[]){
BoxLayoutExample1 b=new BoxLayoutExample1();
}
}       }
```



GRID BAG LAYOUT

- The Java GridBagLayout class is used to align components vertically, horizontally or along their baseline.
- The components may not be of the same size.
- Each GridBagLayout object maintains a dynamic, rectangular grid of cells. Each component occupies one or more cells known as its display area. Each component associates an instance of GridBagConstraints. With the help of the constraints object, we arrange the component's display area on the grid.
- The GridBagLayout manages each component's minimum and preferred sizes in order to determine the component's size. GridBagLayout components are also arranged in the rectangular grid but can have many different sizes and can occupy multiple rows or columns.

GRID BAG LAYOUT

TYPES OF LAYOUT HANDLER :

TYPES	EXPLANATION
Flow layout	This layout manager arranges components in a single row or column. If there is not enough space, the components wrap to the next row or column.
Grid layout	Components are arranged in a grid with a fixed number of rows and columns. All cells in the grid have the same size.
Grid bag layout	This layout manager allows you to create flexible and complex layouts. It uses constraints to specify the position and size of each component individually.
Border layout	Components are arranged in five regions: north, south, east, west, and center. Each region can contain only one component, and the center region expands to fill any remaining space
Card layout	This layout manager is useful when you want to stack multiple components on top of each other, like a deck of cards. You can switch between components by specifying a unique name for each component.
Box layout	Components are arranged in a single row or column. It supports both vertical and horizontal orientations.


Thank
you!