Layout Management

- The LayoutManagement refers to the way in which the GUI components are arranged in a particular manner.
- Layout Managers deals with the location and the size of the windows.i.e. it controls the position and the size of components in GUI forms.
- It is an interface that is implemented by all the classes of layout managers.
- The **setLayout()** method determines the layout manager.If setLayout() is not called, the default layout manager is used.The general form of setLayout() is:

  void setLayout(Layout Manager obj)

1.Flow Layout:

It allows to arrange components in a single row.Components are arranged from upper left to lower right, and top to bottom.When there are no more components that will fit on a line, the following component will appear on the next line.

Constructor:

FlowLayout(): Creates a flow arrangement with centered alignment and a default horizontal and vertical gap of 5 units.

FlowLayout(int align): generates a flow layout with the specified alignment and a 5-unit horizontal and vertical gap as a default.

FlowLayout(int align, int hgap, int vgap): Creates a flow layout with the specified alignment and horizontal and vertical gaps .

Example:

```
import java.awt.*;
import javax.swing.*;
public class FlowLayout1
{
JFrame f;
JFrame f1;
public FlowLayout1() {
f=new JFrame("FlowLayout");
JButton b1=new JButton("A");
JButton b2=new JButton("B");
JButton b3=new JButton("C");
JButton b4=new JButton("D");
f.add(b1);
f.add(b2);
```

```
f.add(b3);
f.add(b4);
f.setLayout(new FlowLayout(FlowLayout.CENTER));
f.setSize(250,250);
f.setVisible(true);
}
public static void main(String[] args){
new FlowLayout1();
}
}
```

Output:

2. Border Layout:

The BorderLayout 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.

It consists of five constants.

1. public static final int NORTH
2. public static final int SOUTH
3. public static final int EAST
4. public static final int WEST
5. public static final int CENTER

Constructors of BorderLayout class:

- BorderLayout(): creates a border layout 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.

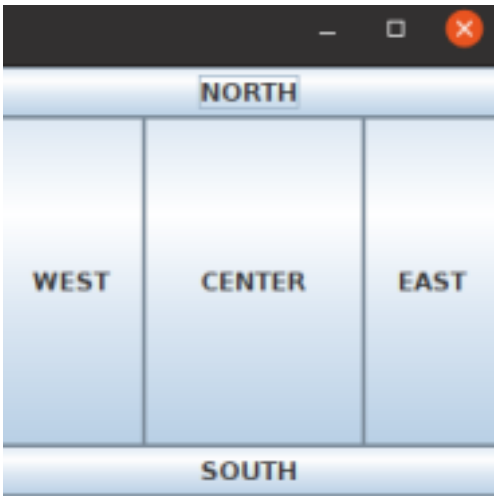
Example:

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

```
import javax.swing.*;


public class BorderExample

{

JFrame f;

BorderExample()

{

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(250, 250);

f.setVisible(true);

}
```

```
public static void main(String[] args) {

new BorderExample();

}

}
```

Output:



3.Grid Layout:

- It allows to arrange the components in a two-dimensional grid i.e. arranges them in the desired number of rows and columns.
- A cell can only contain one component.
- All components are of equal size and cover the entire volume of the container. ● Components are added from the first row to the last row and from the first column to the last column.

Constructors:

GridLayout(): It creates a grid layout with one column per component in a row.

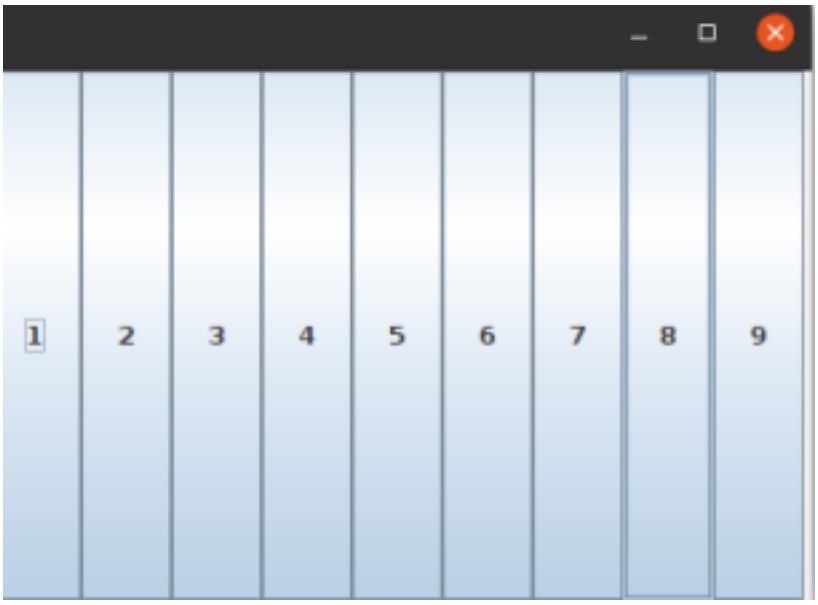
GridLayout(int rows,int cols): Creates a grid layout with the given rows and columns but no gaps between the components.

GridLayout(int rows,int cols,int horz_gap,int vert_gap): It creates a grid layout with the given rows and columns along with given horizontal and vertical gaps

Example of GridLayout with parameterless constructor(GridLayout() Constructor)

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

public class GridLayoutExample
{
JFrame frameObj;

// constructor
GridLayoutExample()
{
frameObj = new JFrame();

// creating 9 buttons
JButton btn1 = new JButton("1");
JButton btn2 = new JButton("2");
JButton btn3 = new JButton("3");
JButton btn4 = new JButton("4");
JButton btn5 = new JButton("5");
JButton btn6 = new JButton("6");
JButton btn7 = new JButton("7");
JButton btn8 = new JButton("8");
JButton btn9 = new JButton("9");

frameObj.add(btn1); frameObj.add(btn2); frameObj.add(btn3);
frameObj.add(btn4); frameObj.add(btn5); frameObj.add(btn6);
frameObj.add(btn7); frameObj.add(btn8); frameObj.add(btn9);

// setting the grid layout using the default constructor
frameObj.setLayout(new GridLayout());


frameObj.setSize(300, 300);
frameObj.setVisible(true);
}

// main method
```

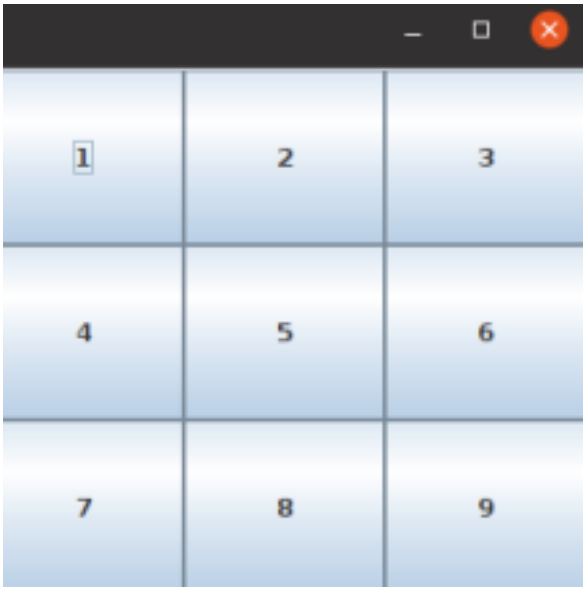
```
public static void main(String argvs[])
{
new GridLayoutExample();
}
}
```



Program 2:(Using GridLayout(int rows, int columns) Constructor)

```
import java.awt.*;
import javax.swing.*;
public class GridLayoutExample1{
JFrame f;
GridLayoutExample1(){
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");
// adding buttons to the frame
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);

// setting grid layout of 3 rows and 3 columns
f.setLayout(new GridLayout(3,3));
f.setSize(300,300);
```

```
f.setVisible(true);
}
public static void main(String[] args) {
new GridLayoutExample1();
}
}
```



4. **Box Layout**

It is used to arrange the components either vertically or horizontally.

Fields:

public static final int X_AXIS:

public static final int Y_AXIS:

public static final int LINE_AXIS:

public static final int PAGE_AXIS

Constructor of BoxLayout class

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

Example:

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

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

5. Card Layout:

It allows you to arrange the components in such a manner that only one component is visible at a time**.**

It treats each component as a card that is why it is known as CardLayout.

Constructors of CardLayout Class

1. CardLayout(): creates a card layout with zero horizontal and vertical gap. 2. CardLayout(int hgap, int vgap): creates a card layout with the given horizontal

and vertical gap.

Methods:

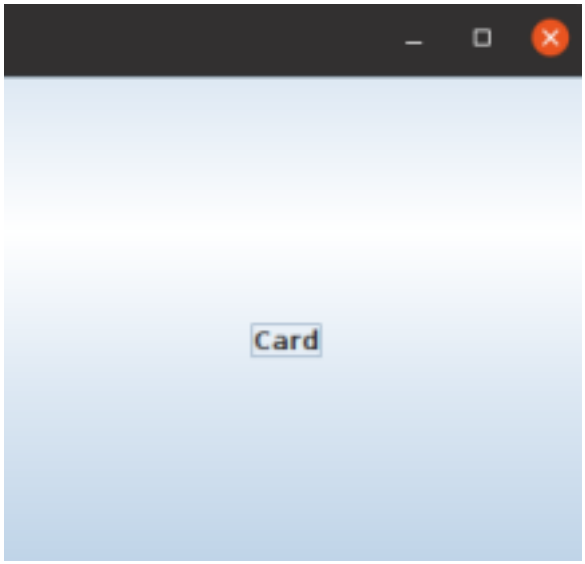
- **public void next(Container parent):** is used to flip to the next card of the given container.
- **public void previous(Container parent):** is used to flip to the previous card of the given container.
- **public void first(Container parent):** is used to flip to the first card of the given container.
- **public void last(Container parent):** is used to flip to the last card of the given container.
- **public void show(Container parent, String name):** is used to flip to the specified card with the given name.

Example:

```
import java.awt.*;

import javax.swing.*;

import java.awt.event.*;


public class CardLayoutExample1 extends JFrame implements

ActionListener {


CardLayout c;


// button variables to hold the references of buttons

JButton btn1, btn2, btn3;

Container cPane;


// constructor of the class

CardLayoutExample1()
```

```
{


cPane = getContentPane();


//default constructor used
// therefore, components will

// cover the whole area

c = new CardLayout();


cPane.setLayout(c);


// creating the buttons

btn1 = new JButton("Card");

btn2 = new JButton("Boy");

btn3 = new JButton("Cat");


// adding listeners to it

btn1.addActionListener(this);

btn2.addActionListener(this);

btn3.addActionListener(this);


cPane.add("a", btn1); // first card is the button btn1

cPane.add("b", btn2); // first card is the button btn2

cPane.add("c", btn3); // first card is the button btn3
```

```
}
public void actionPerformed(ActionEvent e)
{
// Upon clicking the button, the next card of the container is shown // after the
last card, again, the first card of the container is shown upon clicking
c.next(cPane);
}

// main method
public static void main(String argvs[])
{
// creating an object of the class CardLayoutExample1
CardLayoutExample1 cl = new CardLayoutExample1();

// size is 300 * 300
cl.setSize(300, 300);
cl.setVisible(true);
cl.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}
```


Card