**LayoutManager in Java:**

* The layout will specify the format or the order in which the components have got to be placed on the container.
* Layout Manager may be a class or component that’s responsible to rearrange the components on the container
* A layout manager is an instance of any class that implements the LayoutManager interface.
* The layout manager is set by the setLayout( ) method.
* If no call to setLayout( ) is formed, then the default layout manager is employed. The setLayout( ) method has the subsequent

**void setLayout(LayoutManager layoutObj**)

If we don’t want to set any layout manager, then

**setLayout (null);** is called.

To set the position of a component manually

**void setBounds(int x,int y,int width,int height) is used, which is a method of every component class.**

Button1.setBounds(100,100,80,20);

**Predefined LayoutManager class**

1. FlowLayout
2. BorderLayout
3. CardLayout
4. GridLayout

##### ****Flow Layout****

* This layout will display the components in sequence from left to right, from top to bottom.
* The components will always be displayed in first-line and if the first line is filled, these components displayed to the next line automatically.

**Note:** If the row contains only one component then the component is aligned in the center position of that row.

**FlowLayout f1 = new FlowLayout();**  
**FlowLayout f1 = new FlowLayout(int align);**  
**FlowLayout f1 = new FlowLayout(int align, int hgap, int vgap);**

**Examp.**

import java.awt.\*;

import javax.swing.\*;

public class FlowLayoutDemo

{

JFrame f;

FlowLayoutDemo ()

{

f = new JFrame ();

JLabel l1 = new JLabel ("Enter Name");

JTextField tf1 = new JTextField (10);

JButton b1 = new JButton ("click here");

f.add (l1);

f.add (tf1);

f.add (b1);

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

//setting flow layout of right alignment

f.setSize (300, 300);

f.setVisible (true);

}

public static void main (String[]args)

{

new FlowLayoutDemo ();

}

}

##### ****Border Layout****

* This layout will display the components along the border of the container.
* This layout contains five locations where the component can be displayed. Locations are North, South, East, west, and Center.
* The default region is the center.

**BorderLayout bl = new BorderLayout();**  
**BorderLayout bl = new BorderLayout(int vgap, int hgap);**

**Frame class contains two types of add methods:**

1. **add(component):**This method default aligns component in the center region.
2. **add(component, region name):**Internally add method handovers a given component (i.e object) to the container and container user peer class of that component to communicate with OS library and then created component is aligned on the window.

import java.awt.\*;

public class BorderLayoutDemo

{

public static void main (String[]args)

{

Frame f1 = new Frame ();

f1.setSize (250, 250);

Button b1 = new Button ("Button1");

Button b2 = new Button ("Button2");

Button b3 = new Button ("Button3");

Button b4 = new Button ("Button4");

Button b5 = new Button ("Button5");

f1.add (b1, BorderLayout.NORTH);

f1.add (b2, BorderLayout.EAST);

f1.add (b3, BorderLayout.WEST);

f1.add (b4, BorderLayout.SOUTH);

f1.add (b5);

f1.setVisible (true);

}

}

##### ****Grid Layout****

* The layout will display the components in the format of rows and columns statically.
* The container will be divided into a table of rows and columns.
* The intersection of a row and column cell and every cell contains only one component and all the cells are of equal size.
* According to Grid Layout Manager, the grid cannot be empty.

**GridLayout gl = new GridLayout(int rows, int cols);**  
**GridLayout gl = new GridLayout(int rows, int cols, int vgap, int hgap);**

import java.awt.\*;

import javax.swing.\*;

public class GridLayoutDemo

{

public static void main (String[]args)

{

Frame f1 = new Frame ();

f1.setSize (300, 300);

GridLayout ob = new GridLayout (2, 2);

f1.setLayout (ob);

Panel p1 = new Panel ();

Label l1 = new Label ("Enter name");

TextField tf = new TextField (10);

Button b1 = new Button ("Submit");

p1.add (l1);

p1.add (tf);

p1.add (b1);

f1.add (p1);

Panel p2 = new Panel ();

f1.add (p2);

Panel p3 = new Panel ();

f1.add (p3);

Label l2 = new Label ("Welcome to Java");

f1.add (l2);

f1.setVisible (true);

}

}

##### ****Card Layout****

* A card layout represents a stack of cards displayed on a container.
* At a time only one card can be visible and each can contain the only component.

**CardLayout cl = new CardLayout();**  
**CardLayout cl = new CardLayout(int hgap, int vgap);**

**Methods of CardLayout**

1. **first(Container):**It is used to flip to the first card of the given container.
2. **last(Container):**It is used to flip to the last card of the given container.
3. **next(Container):**It is used to flip to the next card of the given container.
4. **previous(Container):** It is used to flip to the previous card of the given container.
5. **show(Container, cardname):** It is used to flip to the specified card with the given name.

import java.awt.\*;

import javax.swing.\*;

import javax.swing.JButton;

import java.awt.event.\*;

public class CardLayoutDemo extends JFrame implements ActionListener

{

JButton b1, b2, b3, b4, b5;

CardLayout cl;

Container c;

CardLayoutDemo ()

{

b1 = new JButton ("Button1");

b2 = new JButton ("Button2");

b3 = new JButton ("Button3");

b4 = new JButton ("Button4");

b5 = new JButton ("Button5");

c = this.getContentPane ();

cl = new CardLayout (10, 20);

c.setLayout (cl);

c.add ("Card1", b1);

c.add ("Card2", b2);

c.add ("Card3", b3);

b1.addActionListener (this);

b2.addActionListener (this);

b3.addActionListener (this);

setVisible (true);

setSize (400, 400);

setTitle ("Card Layout");

setDefaultCloseOperation (JFrame.EXIT\_ON\_CLOSE);

}

public void actionPerformed (ActionEvent ae)

{

cl.next (c);

}

public static void main (String[]args)

{

new CardLayoutDemo ();

}

}