```
import java.awt.*;
public class myframe extends Frame
{
public static void main(String[] args)
{
   myframe f=new myframe();
   Button b1=new Button("Click Here");
   f.add(b1);
   f.setSize(400,400);
   f.setVisible(true);
}
}
```

LAYOUT MANAGER:

A layout manager an instance of any class that implements the LayoutManager interface.the layout manager is set by the **setLayout()**

Method.

Layouts are following types

1.**FlowLayout**: the default layout manager is FlowLayout.it implements a simple layout style, which is similar to how words flow in a text editor. Components are laid out from the upper-left corner, left to right and top to bottom. When no more components fit on a line, the next one appears on the next line. a small space left between each component, above and below, as well as left and right.

The **constructors:**

1.FlowLayout()

2. FlowLayout**(**int how**)**

3. FlowLayout**(**int how,int horz,int vert)

Alignment as follows:

FlowLayout.LEFT

FlowLayout.CENTER

FlowLayout.RIGHT

EX:PROGRAM TO IMPLEMENT FLOE LAYOUT:

```
import  java.awt.*;

import  java.applet.*;

public class ButtonExample extends Applet
```

```
{
   public void init()
{
  setLayout(new FlowLayout(FlowLayout.CENTER));
    Button b1=new Button();
    Button b2=new Button("eee");
   Button b3=new Button("it");

  add(b1);
  add(b2);
add(b3);
}
}
/*
<html>
<applet code="ButtonExample.class" width=200 height=200>
</applet>
</html>      */
```

2.**BorderLayout**:this class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north,south.east,west.the middle area is called the center.

Constructors:

1. BorderLayout()
2. BorderLayout(int horz,int vert)

Regions

1. BorderLayout.CENTER
2. BorderLayout.SOUTH
3. BorderLayout.NORTH
4. BorderLayout.EAST
5. BorderLayout.WEST

PROGRAM:

```
import  java.awt.*;
import  java.applet.*;
public class ButtonExample1 extends Applet
{
  public void init()
{
  setLayout(new BorderLayout());
  Button b1=new Button();
  Button b2=new Button("eee");
```

```
  Button b3=new Button("it");
Button b4=new Button("cse");
Button b5=new Button("sreyas");
add( b1,BorderLayout.EAST);
  add( b2,BorderLayout.SOUTH);
add( b3,BorderLayout.NORTH);
add( b4,BorderLayout.WEST);
add( b5,BorderLayout.CENTER);
}
}
/*
<html>
<applet code="ButtonExample1.class" width=200 height=200>
</applet>  </html>    */
```

3.GridLayout: it lays out components in a two dimensional grid.when you instantiate a grid layout,you define the number of rows and columns.

Constructors:

1.GridLayout ()

2. GridLayout (int numrows,int numcolumns)

3. GridLayout (int numrows,int numcolumns,int horz,int vert)

The first form creats a single column grid layout.the second form creats a grid layout with the specified number of rows and columns.the third form allows you to specify the horizontal and vertical space left between components in horz and vert,respectively.

Program:

```
import  java.awt.*;
import  java.applet.*;
public class ButtonExample2 extends Applet
{
public void init()
{
   setLayout(new GridLayout(2,2));
   Button b1=new Button("eee");
  Button b2=new Button("it");
Button b3=new Button("cse");
Button b4=new Button("sreyas");
add(b1);
  add(b2);
add(b3);
add( b4);
```

```
}

}

/*

<applet code="ButtonExample2.class" width=200 height=200>

</applet>

*/
```

Java CardLayout

The CardLayout class manages 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.

Commonly used methods of CardLayout class

- **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.

```
import  java.awt.*;
import  java.applet.*;
public class ButtonExample2 extends Applet
{
public void init()
{
   setLayout(new CardLayout(40,30));
   Button b1=new Button("eee");
  Button b2=new Button("it");
Button b3=new Button("cse");
Button b4=new Button("sreyas");
 add(b1);
  add(b2);
add(b3);
add(b4);
}
}
/*<html>
```

<applet code="ButtonExample2.class" width=200 height=200>

</applet> </html>

***/**

Java GridBagLayou

This is the most flexible layout manager class.The object of GridBagLayout aligns the component vertically,horizontally or along their baseline without requiring the components of same size.

The Java GridBagLayout class is used to align components vertically, horizontally or along their baseline.

The components may not be of 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 constraints object we arrange component's display area on the grid. The GridBagLayout manages each component's minimum and preferred sizes in order to determine component's size

```
import  java.awt.*;

import  java.applet.*;

public class gridExample1 extends Applet

{

public void init()

{

GridBagLayout g=new GridBagLayout();
```

```
GridBagConstraints c=new GridBagConstraints();
setLayout(g);

        c.insets = new Insets(2, 2, 2, 2);
   // column 0
        c.gridx = 0;
  // row 0
        c.gridy = 0;
 // increases components width by 10 pixels
        c.ipadx = 15;
 // increases components height by 50 pixels
        c.ipady = 50;
   // constraints passed in
        add(new Button("Java Swing"), c);

        // column 1
        c.gridx = 1;
 // increases components width by 70 pixels
        c.ipadx = 90;
   // increases components height by 40 pixels
        c.ipady = 40;
```

```
// constraints passed in
        add(new Button("Layout"), c);
   // column 0
        c.gridx = 0;
 // row 1
        c.gridy = 1;
 // increases components width by 20 pixels
        c.ipadx = 20;
  // increases components height by 20 pixels
        c.ipady = 20;
  // constraints passed in
        add(new Button("Manager"), c);
   // increases components width by 10 pixels
        c.ipadx = 10;
   // column 1
        c.gridx = 1;
    // constraints passed in
        add(new Button("Demo"), c);
}
}/*<html>
<applet code="gridExample1.class" width=200 height=200>
```

</applet> </html>

***/**

BorderLayout The borderlayout arranges the components to fit in the five regions: east, west, north, south and center.	
2	CardLayout The CardLayout object treats each component in the container as a card. Only one card is visible at a time.
3	FlowLayout The FlowLayout is the default layout.It layouts the components in a directional flow.
4	GridLayout The GridLayout manages the components in form of a rectangular grid.
5	GridBagLayout This is the most flexible layout manager class.The object of GridBagLayout aligns the component vertically,horizontally or along their baseline without requiring the components of same size.