Layouts

Layouts

- The LayoutManagers are used to arrange components in a particular manner. The **Java LayoutManagers** facilitates us to control the positioning and size of the components in GUI forms. LayoutManager is an interface that is implemented by all the classes of layout managers.

1. java.awt.BorderLayout
2. java.awt.FlowLayout
3. java.awt.GridLayout
4. java.awt.CardLayout
5. java.awt.GridBagLayout
6. javax.swing.BoxLayout
7. javax.swing.GroupLayout
8. javax.swing.ScrollPaneLayout
9. javax.swing.SpringLayout

Java BorderLayout

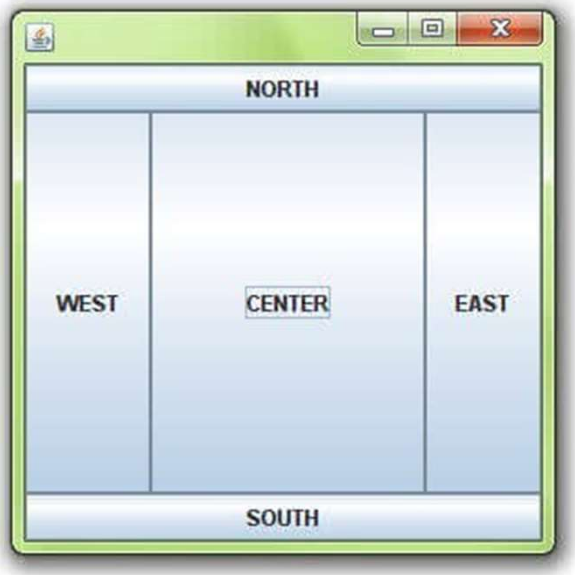
- The BorderLayout is used to arrange the components in five regions: north, south, east, west, and center. Each region (area) may contain one component only. It is the default layout of a frame or window. The BorderLayout provides five constants for each region:
  - **public static final int NORTH**
  - **public static final int SOUTH**
  - **public static final int EAST**
  - **public static final int WEST**
  - **public static final int CENTER**

Constructors of BorderLayout class

- **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 of BorderLayout

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

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

    // creating buttons
   JButton b1 = new JButton("NORTH");; // the button will be labeled as NORTH
   JButton b2 = new JButton("SOUTH");; // the button will be labeled as SOUTH
   JButton b3 = new JButton("EAST");; // the button will be labeled as EAST
   JButton b4 = new JButton("WEST");; // the button will be labeled as WEST
   JButton b5 = new JButton("CENTER");; // the button will be labeled as CENTER

   f.add(b1, BorderLayout.NORTH); // b1 will be placed in the North Direction
   f.add(b2, BorderLayout.SOUTH);  // b2 will be placed in the South Direction
   f.add(b3, BorderLayout.EAST);  // b2 will be placed in the East Direction
   f.add(b4, BorderLayout.WEST);  // b2 will be placed in the West Direction
   f.add(b5, BorderLayout.CENTER);  // b2 will be placed in the Center

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



Java GridLayout

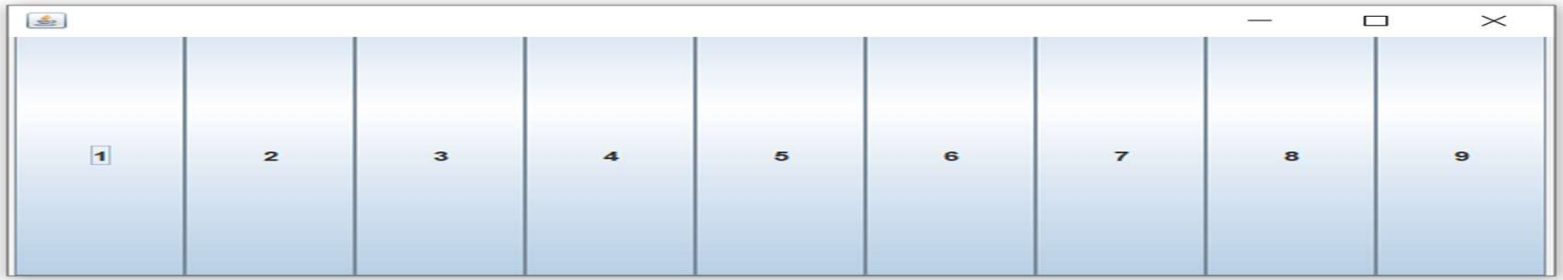
- The Java GridLayout class is used to arrange the components in a rectangular grid. One component is displayed in each rectangle.

Constructors of GridLayout class

- **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.

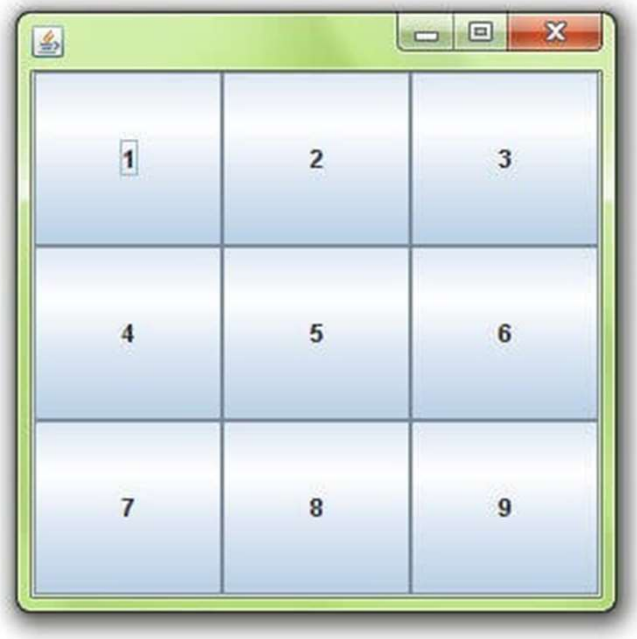
Example of GridLayout

```
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");

// adding buttons to the frame
// since, we are using the parameterless constructor, therfore;
// the number of columns is equal to the number of buttons we
// are adding to the frame. The row count remains one.
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 parameterless constructor
frameObj.setLayout(new GridLayout());


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

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



Example of GridLayout class: Using GridLayout(int rows, int columns) Constructor

```
import java.awt.*;
import javax.swing.*;
public 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");
     // 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 MyGridLayout();
}
}
```



Java FlowLayout

- The Java FlowLayout class is used to arrange the components in a line, one after another (in a flow). It is the default layout of the applet or panel.

Fields of FlowLayout class

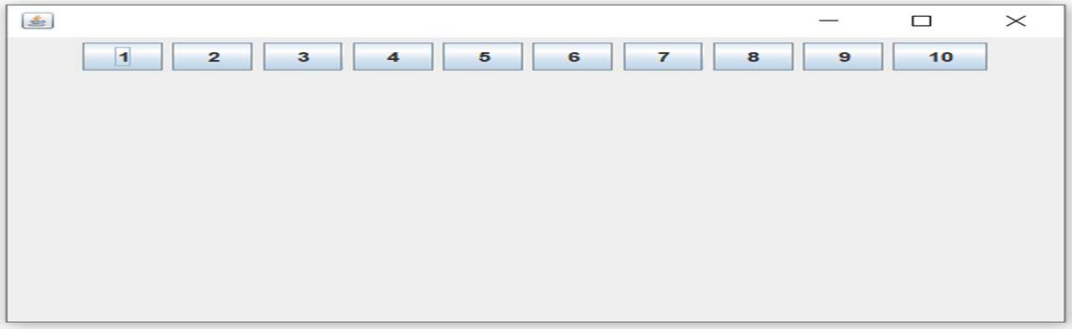
- **public static final int LEFT**
- **public static final int RIGHT**
- **public static final int CENTER**
- **public static final int LEADING**
- **public static final int TRAILING**

Constructors of FlowLayout class

- **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.

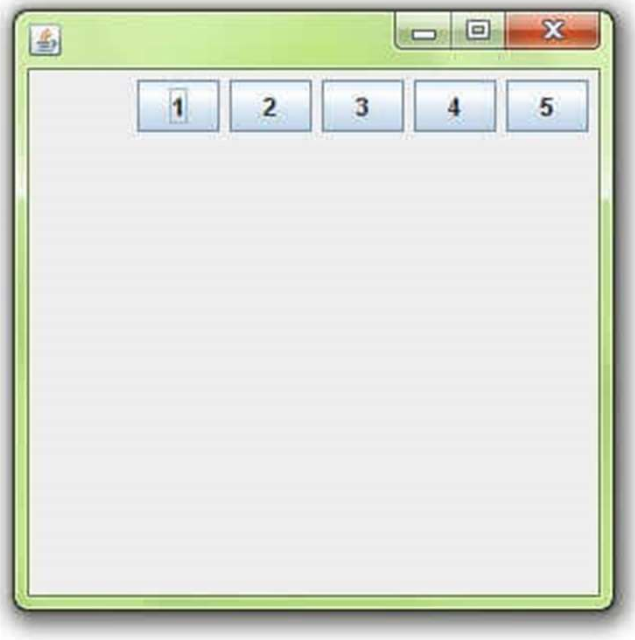
Example of FlowLayout

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

public class FlowLayoutExample
{

JFrame frameObj;

FlowLayoutExample()
{

   frameObj = 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");
   JButton b10 = new JButton("10");

   frameObj.add(b1); frameObj.add(b2); frameObj.add(b3); frameObj.add
      (b4);
   frameObj.add(b5); frameObj.add(b6);  frameObj.add(b7);  frameObj.a
      dd(b8);
   frameObj.add(b9);  frameObj.add(b10);


   frameObj.setLayout(new FlowLayout());

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

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



Example of FlowLayout class: Using FlowLayout(int align)

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

public 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");

    // adding buttons to the frame
   f.add(b1); f.add(b2); f.add(b3); f.add(b4); f.add(b5);

    // setting flow layout of right alignment
   f.setLayout(new FlowLayout(FlowLayout.RIGHT));

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

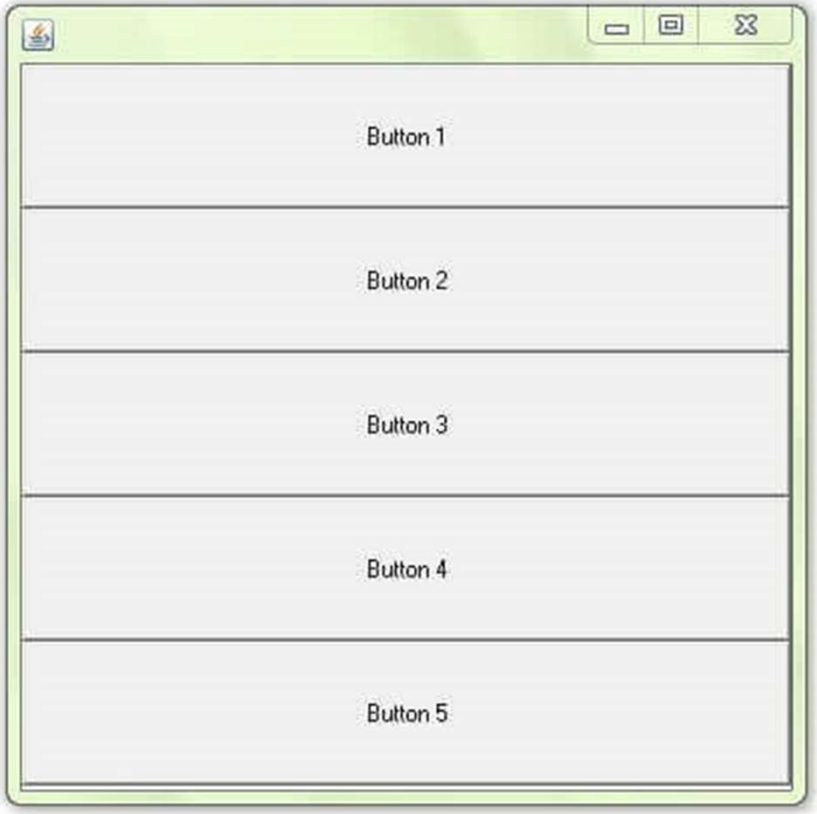


Java BoxLayout

- The **Java BoxLayout class** is used to arrange the components either vertically or horizontally. For this purpose, the BoxLayout class provides four constants. They are as follows:
  - **public static final int X_AXIS:**
  - **public static final int Y_AXIS:**
  - **public static final int LINE_AXIS:**
  - **public static final int PAGE_AXIS:**

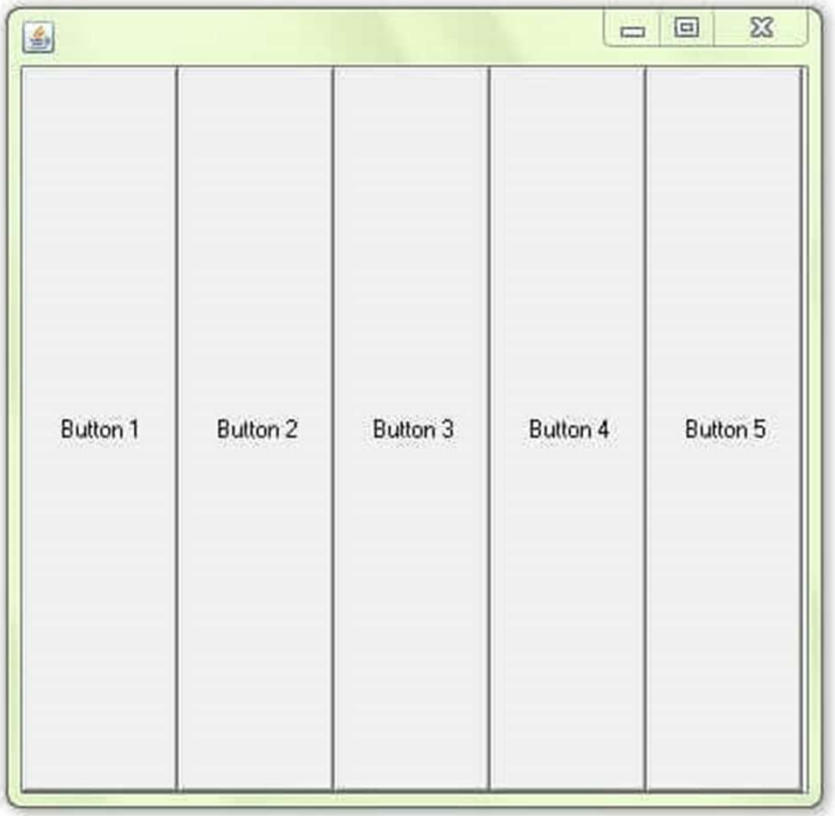
Example of BoxLayout class with Y-AXIS:

```
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();
}
}
```



Example of BoxLayout class with X-AXIS

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

public class BoxLayoutExample2 extends Frame {
 Button buttons[];

 public BoxLayoutExample2() {
  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 in the output will be aligned vertically
setLayout (new BoxLayout(this, BoxLayout.X_AXIS));
setSize(400,400);
setVisible(true);
}
 // main method
public static void main(String args[]){
BoxLayoutExample2 b=new BoxLayoutExample2();
}
}
```



Java CardLayout

- The **Java 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

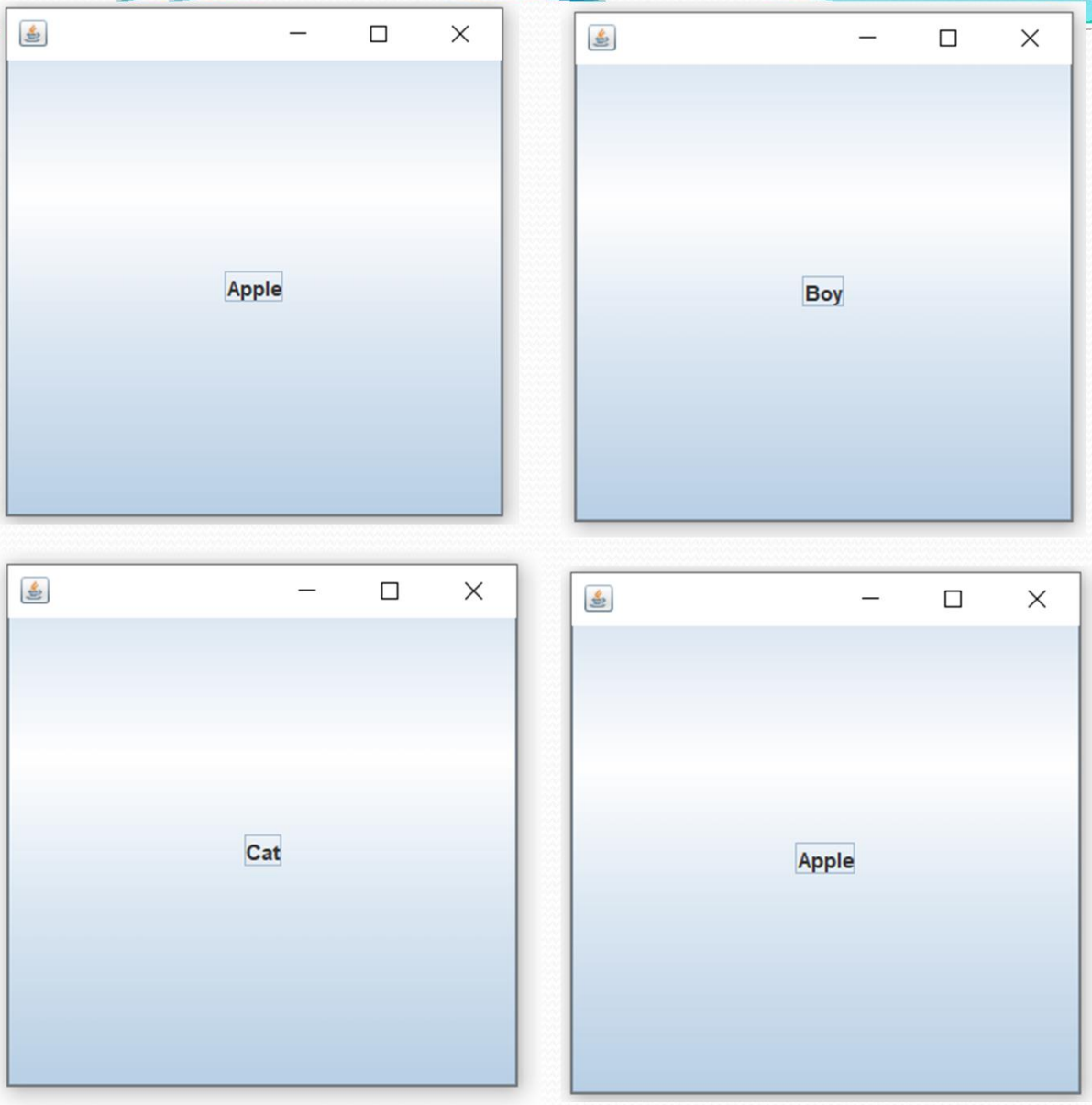
- **CardLayout():** creates a card layout with zero horizontal and vertical gap.
- **CardLayout(int hgap, int vgap):** creates a card layout with the given horizontal and vertical gap.

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.

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

public class CardLayoutExample1 extends JFrame implements ActionListener
{
CardLayout crd;
JButton btn1, btn2, btn3;
Container cPane;
CardLayoutExample1()
{
cPane = getContentPane();
  crd = new CardLayout();
cPane.setLayout(crd);
 btn1 = new JButton("Apple");
btn2 = new JButton("Boy");
btn3 = new JButton("Cat");
 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)
{
crd.next(cPane);
}

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

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


Apple
Boy
Cat
Apple

Thanks