

# MODULE IV

## Part III

# JAVA LAYOUTMANAGERS

- The LayoutManagers are used to arrange components in a particular manner.
- LayoutManager is an interface that is implemented by all the classes of layout managers.
- There are following classes that represents the awt layout managers:
  - java.awt.BorderLayout
  - java.awt.FlowLayout
  - java.awt.GridLayout
  - java.awt.CardLayout
  - java.awt.GridBagLayout



# JAVA FLOWLAYOUT

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



## 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



# FIELDS OF FLOWLAYOUT CLASS

Modifier and Type	Field and Description
static int	<b>CENTER:</b> This value indicates that each row of components should be centered.
static int	<b>LEADING:</b> This value indicates that each row of components should be justified to the leading edge of the container's orientation, for example, to the left in left-to-right orientations.
static int	<b>LEFT:</b> This value indicates that each row of components should be left-justified.
static int	<b>RIGHT:</b> This value indicates that each row of components should be right-justified.
static int	<b>TRAILING:</b> This value indicates that each row of components should be justified to the trailing edge of the container's orientation, for example, to the right in left-to-right orientations.

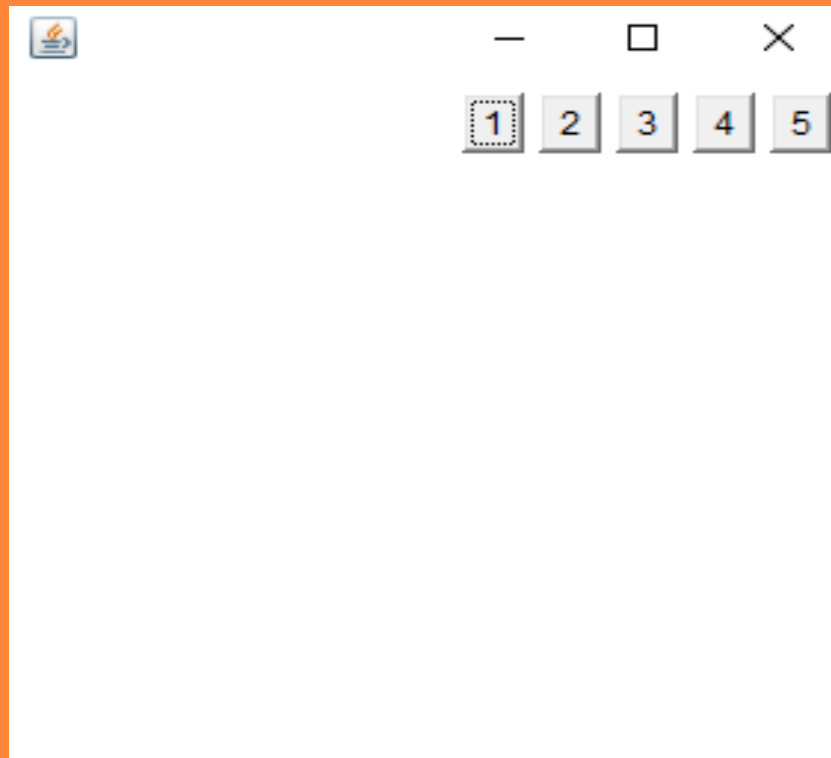


## EXAMPLE

```
import java.awt.*;
public class Example{
    Frame f;
    Example(){
        f=new Frame();
        Button b1=new Button("1");
        Button b2=new Button("2");
        Button b3=new Button("3");
        Button b4=new Button("4");
        Button b5=new Button("5");
        f.add(b1);f.add(b2);f.add(b3);f.add(b4);f.add(b5);
        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 Example();
    }
}
```



## OUTPUT



# 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 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.

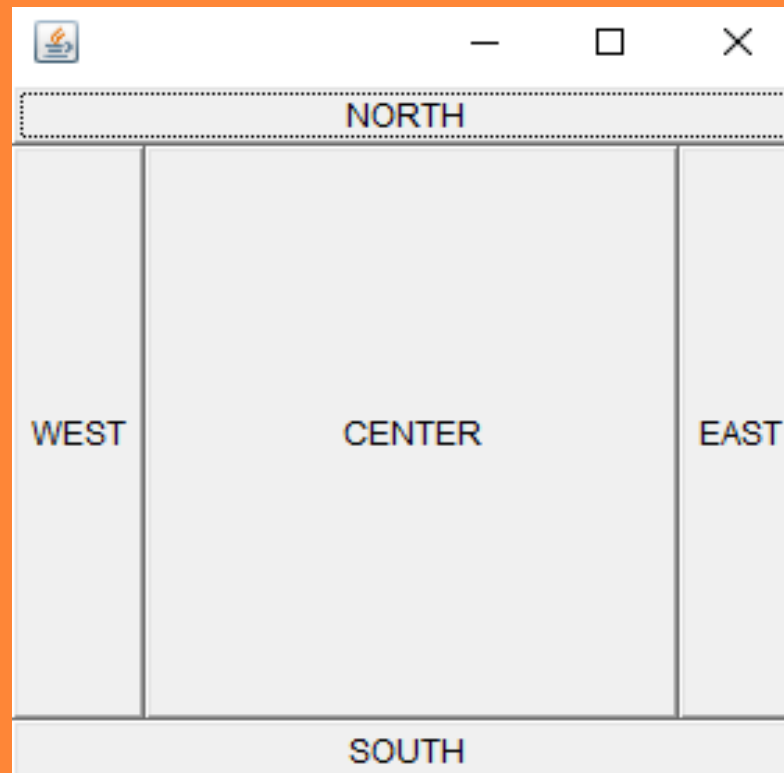


## EXAMPLE

```
import java.awt.*;
public class Example {
    Frame f;
    Example(){
        f=new Frame();
        Button b1=new Button("NORTH");;
        Button b2=new Button("SOUTH");;
        Button b3=new Button("EAST");;
        Button b4=new Button("WEST");;
        Button b5=new Button("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 Example();
    }
}
```



## OUTPUT



# JAVA GRIDLAYOUT

- The GridLayout is used to arrange the components in 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 alongwith given horizontal and vertical gaps.



## EXAMPLE

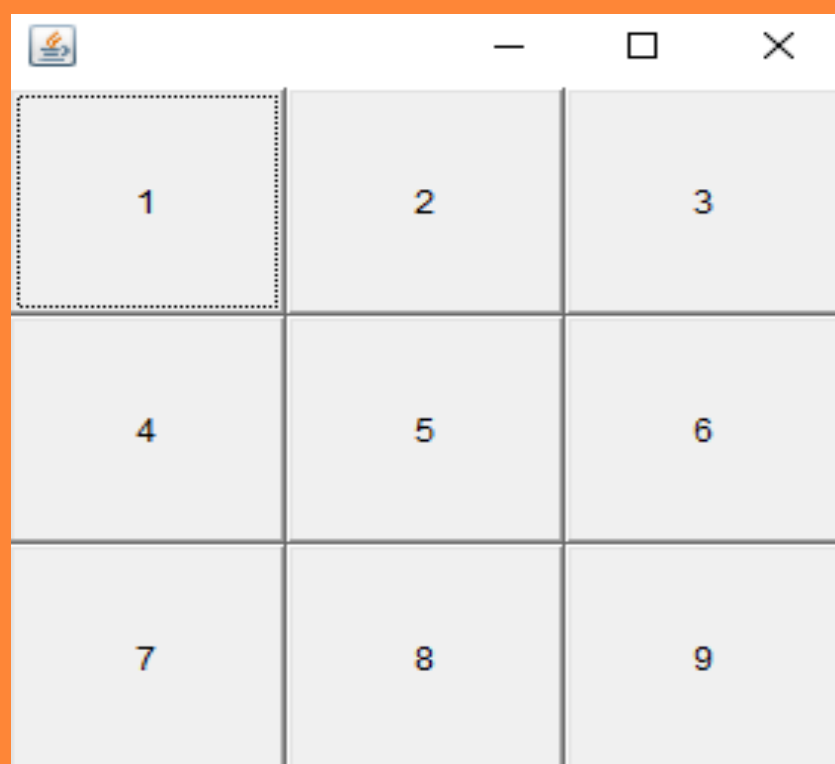
```
import java.awt.*;

public class Example{
    Frame f;
    Example(){
        f=new Frame();
        Button b1=new Button("1");
        Button b2=new Button("2");
        Button b3=new Button("3");
        Button b4=new Button("4");
        Button b5=new Button("5");
        Button b6=new Button("6");
        Button b7=new Button("7");
        Button b8=new Button("8");
        Button b9=new Button("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));
        //setting grid layout of 3 rows and 3 columns
        f.setSize(300,300);
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new Example();
    }
}
```



## OUTPUT



A screenshot of a graphical user interface window. The window has a white title bar with a small icon on the left and standard minimize, maximize, and close buttons on the right. The main content area is a 3x3 grid of light gray squares. The top-left square, containing the number '1', is highlighted with a dotted border. The other squares contain numbers 2 through 9 in a sequential order from top-left to bottom-right.

1	2	3
4	5	6
7	8	9

# 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

- **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.



## EXAMPLE-CARDLAYOUT

[NOTE: SINCE WE NEED TO MOVE FROM ONE BUTTON TO ANOTHER WE ARE USING EVENT HANDLING AND SWING PACKAGE HERE. TRY TO UNDERSTAND IT BY USING THE COMMENTS. THESE TOPICS WILL BE DISCUSSED IN DETAIL IN COMING LECTURES]

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
// class extends JFrame and implements ActionListener
public class Cardlayout extends JFrame implements
ActionListener {
    // Declaration of objects of CardLayout class.
    CardLayout card;
    // Declaration of objects of Button class.
    Button b1, b2, b3;
    // Declaration of objects
    // of Container class.
    Container c;
```



Cardlayout()

```
{
    // to get the content
    c = getContentPane();
    // Initialization of object "card"
    // of CardLayout class with 40
    // horizontal space and 30 vertical space .
    card = new CardLayout(40, 30);
    // set the layout
    c.setLayout(card);
    // Initialization of object "b1" of Button class.
    b1 = new Button("ONE");
    // Initialization of object "b2" of Button class.
    b2 = new Button("TWO");
    // Initialization of object "b3" of Button class.
    b3 = new Button("THREE");
    // this Keyword refers to current object.
    // Adding button "b1" on JFrame using ActionListener.
    b1.addActionListener(this);
    // Adding button "b2" on JFrame using ActionListener.
    b2.addActionListener(this);
    // Adding button "b3" on JFrame using ActionListener.
    b3.addActionListener(this);
    // Adding the Button "b1"
    c.add("a", b1);
    // Adding the Button "b2"
    c.add("b", b2);
    // Adding the Button "b1"
    c.add("c", b3);
}
```



```
public void actionPerformed(ActionEvent e)
{

    // call the next card
    card.next(c);
}

// Main Method
public static void main(String[] args)
{

    // Creating Object of CardLayout class.
    Cardlayout cl = new Cardlayout();

    // Function to set size of JFrame.
    cl.setSize(400, 400);

    // Function to set visibility of JFrame.
    cl.setVisible(true);

    // Function to set default operation of JFrame.
    cl.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}
```



# OUTPUT



[On Clicking you can see the next buttons also]