SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Information Technology

19CS204 OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Topic – Layout Manager

Layout Manager

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 layout managers:

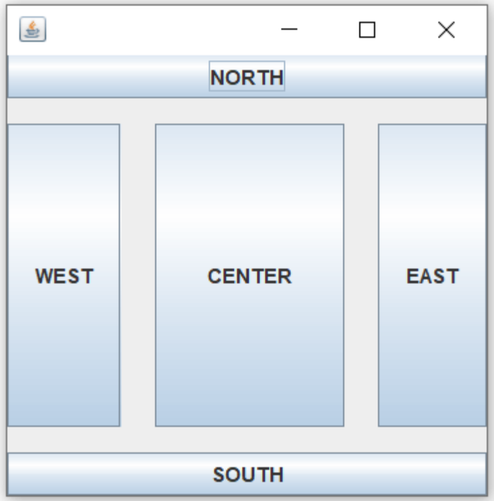
1. java.awt.BorderLayout
2. java.awt.FlowLayout
3. java.awt.GridLayout
4. java.awt.CardLayout
5. java.awt.GridBagLayout
6. javax.swing.BoxLayout
7. javax.swing.GroupLayout
8. javax.swing.ScrollPaneLayout
9. javax.swing.SpringLayout etc.

Layout Manager

Border Layout

- The **default layout** manager for **every JFrame is BorderLayout.**
- It places components in upto five places which is top, bottom, left, right and center

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

public class BorderLO {
JFrame f;
BorderLO(){
   f=new JFrame();
   JButton b1=new JButton("TOP");;
   JButton b2=new JButton("LEFT");;
   JButton b3=new JButton("CENTER");;
   JButton b4=new JButton("RIGHT");;
   JButton b5=new JButton("BOTTOM");;

   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 BorderLO();
}
}
```



Layout Manager

Flow Layout

FlowLayout simply **lays the components in a row one after the other,** it is the default layout manager for every JPanel.

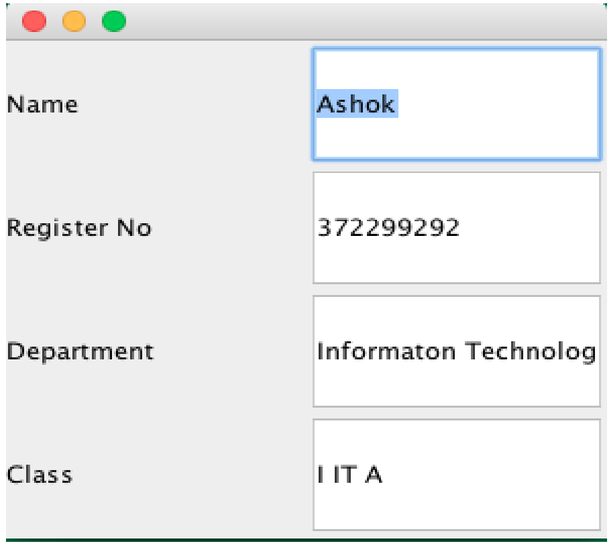

```
import java.awt.*;
import javax.swing.*;

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

   JTextField b1=new JTextField("1");
   JButton b2=new JButton("2");
   JButton b3=new JButton("3");
   JButton b4=new JButton("4");
   JLabel b5=new JLabel("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 MyFlowLayout();
}
}
```

Layout Manager

GridBag Layout

GridBagLayout **places the components in a grid which allows the components to span more than one cell.**

It arrange the **components in rectangular grid**. One component is displayed in each rectangle.




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

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

   JLabel b1=new JLabel("Name");
   JTextField b2=new JTextField();
  JLabel b3=new JLabel("Register No");
  JTextField b4=new JTextField();
   JLabel b5=new JLabel("Department");
   JTextField b6=new JTextField();
   JLabel b7=new JLabel("Class");
    JTextField b8=new JTextField();
      //JButton b9=new JButton("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.setLayout(new GridLayout(4,2));
   //setting grid layout of 4 rows and 2 columns

   f.setSize(300,150);
   f.setVisible(true);
}
public static void main(String[] args) {
   new GridLO();
}
}
```

Event Handler

- Any program that uses **GUI (graphical user interface) such as Java application written for windows, is** event driven.

- **Event describes the change in state of any object.**

- For Example : Pressing a button, Entering a character in Textbox, Clicking or Dragging a mouse, etc.

Components of Event Handling

Event handling has three main components,

- **Events** : An event is a **change in state of an object**. For example, mouseClicked, mousePressed

- **Events Source :** Event source is an **object** that generates an **event.** Example: a button, frame, textfield.

- **Listeners** : A listener is an **object that listens to the event. A listener gets notified when an event occurs.**

Event Handling

Some of the event classes and Listener interfaces are listed below.

Event Classes	Generated when	Listener Interfaces
ActionEvent	button is pressed, menu-item is selected, list-item is double clicked	Action Listener
MouseEvent	mouse is dragged, moved, clicked, pressed or released and also when it enters or exit a component	Mouse Listener and Mouse Motion Listener
MouseWheelEvent	mouse wheel is moved	Mouse Wheel Listener
KeyEvent	input is received from keyboard	Key Listener
ItemEvent	check-box or list item is clicked	Item Listener
TextEvent	value of textarea or textfield is changed	Text Listener
AdjustmentEvent	scroll bar is manipulated	Adjustment Listener
WindowEvent	window is activated, deactivated, deiconified, iconified, opened or closed	Window Listener
ComponentEvent	component is hidden, moved, resized or set visible	Component Listener
ContainerEvent	component is added or removed from container	Container Listener
FocusEvent	component gains or losses keyboard focus	Focus Listener

Event Handling

How Events are handled?

- A source generates an **Event and send it to one or more listeners registered with the source.** Once event is received by the listener, they process the event and then return.
- Events are **supported by a number of Java packages**, like **java.util**, **java.awt** and **java.awt.event**.

Steps to handle events:

- Implement appropriate interface in the class.
- Register the component with the listener.

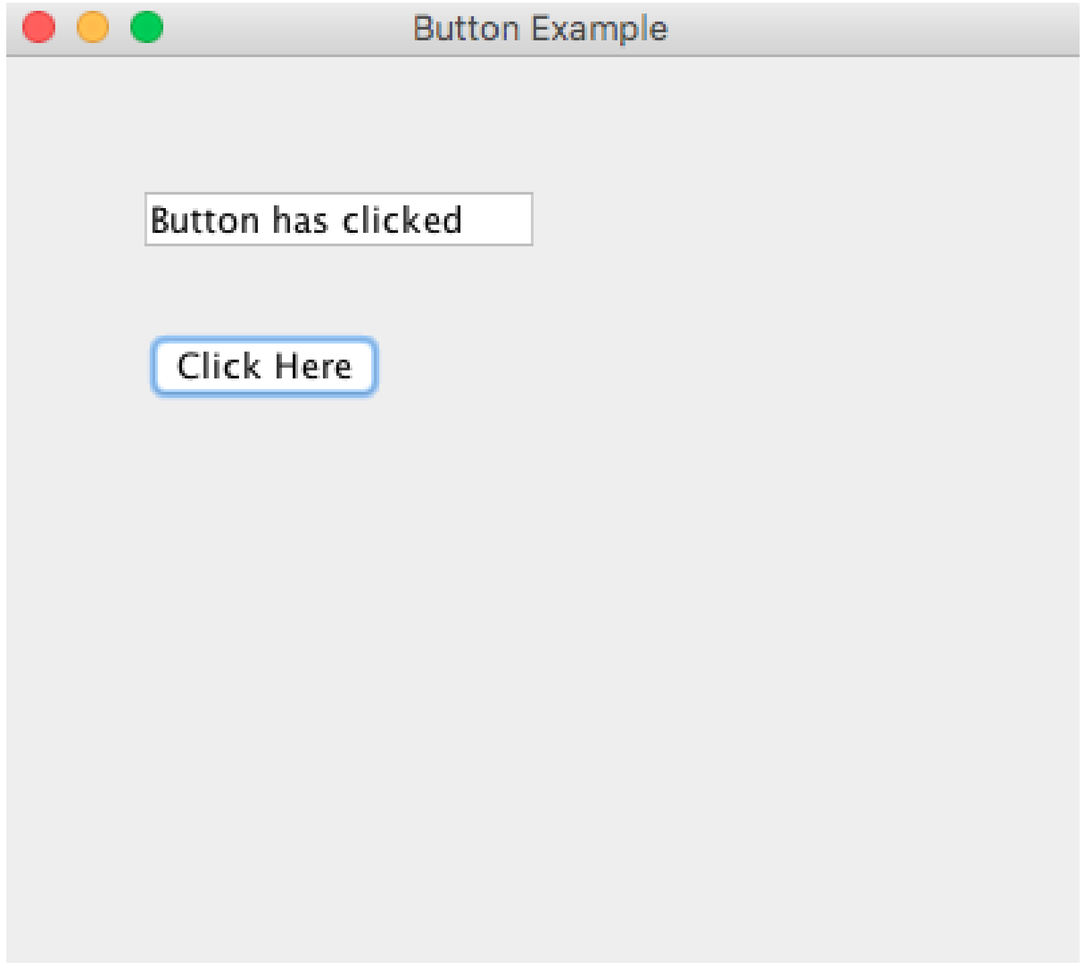
Event Handling

Action Event – Action Listener for Button

- The Java **ActionListener** is notified whenever you click on the button or menu item.
- It is notified against **ActionEvent.** The ActionListener interface is found in java.awt.event package.
- It has only one method: actionPerformed(). actionPerformed() method
- The actionPerformed() method is invoked automatically whenever you click on the registered component.

```
import java.awt.event.*;
import javax.swing.*;
public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    final JTextField tf=new JTextField();
    tf.setBounds(50,50, 150,20);
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    b.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
            tf.setText("Button has clicked");
        }
    });
    f.add(b);f.add(tf);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Event Handling



Event Handling

Java MouseListener Interface

The Java MouseListener is notified whenever you change the state of mouse.

It is notified against MouseEvent.

The MouseListener interface is found in java.awt.event package. It has five methods.

Methods of MouseListener interface

The signature of 5 methods found in MouseListener interface are given below:

```
public abstract void mouseClicked(MouseEvent e);
public abstract void mouseEntered(MouseEvent e);
public abstract void mouseExited(MouseEvent e);
public abstract void mousePressed(MouseEvent e);
public abstract void mouseReleased(MouseEvent e);
```

Event Handling

```
import java.awt.*;
import java.awt.event.*;
public class MouseListenerExample extends Frame impleme
nts MouseListener{
    Label l;
    MouseListenerExample(){
        addMouseListener(this);

        l=new Label();
        l.setBounds(20,50,100,20);
        add(l);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
```

```
    public void mouseClicked(MouseEvent e) {
        l.setText("Mouse Clicked");
    }
    public void mouseEntered(MouseEvent e) {
        l.setText("Mouse Entered");
    }
    public void mouseExited(MouseEvent e) {
        l.setText("Mouse Exited");
    }
    public void mousePressed(MouseEvent e) {
        l.setText("Mouse Pressed");
    }
    public void mouseReleased(MouseEvent e) {
        l.setText("Mouse Released");
    }
public static void main(String[] args) {
    new MouseListenerExample();
}
}
```

THANK YOU