BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI
First Semester, 2016-17
O Object-Oriented Programming (CS F213)
LAB-9 (GUI and Event handling)

AGENDA

1. **Introduction to GUI design in java**
2. **Event handling**

1.Graphical User Interface in Java

There are 2 ways to create a Graphical User Interface in java, using AWT and Swing

1. AWT

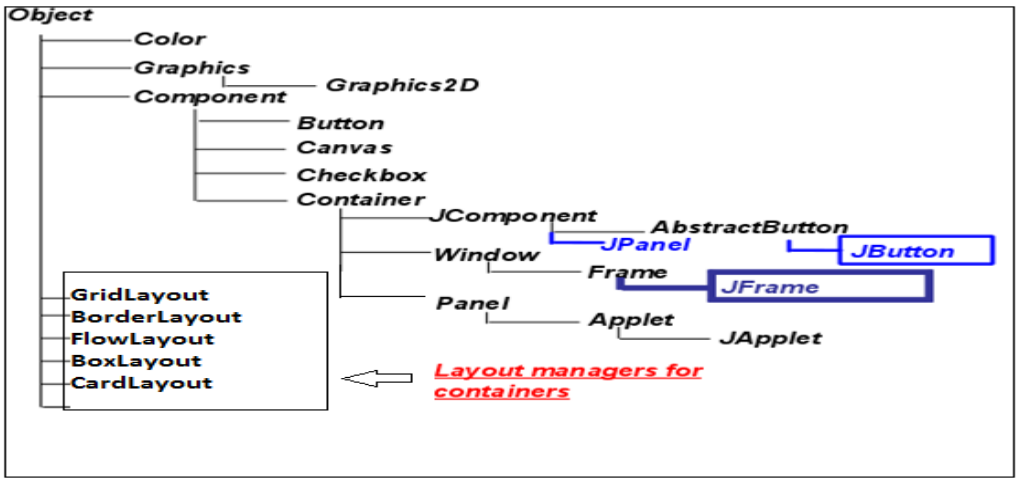
The Abstract Window Toolkit (AWT) *is a* Java package and can be used in any Java program by importing java.awt.* via the import keyword.
AWT contains classes for implementing Color, Graphics, Button, Canvas, Frame, Panel, Window, BorderLayout, FlowLayout etc.

2. Swing

Swing library is an official advanced Java GUI toolkit having a rich set of widgets. From basic widgets like buttons, labels, scrollbars to advanced widgets like trees and tables. It has more powerful and flexible components than AWT.

Swing classes begin with a 'J'. eg: JFrame, JPanel, JButton, JTextArea, JTextField, etc.

3. GUI Hierarchy



4.JFrame:

It is an extended version of java.awt.Frame. A graphical user interface has to start with a top-level container. JFrame will be used to create a simple top-level window for a Java application.

5.JPanel

The JPanel is a container that is used to hold the 'widgets' of Swing. They are used to let the developer have more control over positioning and style of widgets on the screen.

6.Layout Managers in Java

Java technology provides the following Layout Managers:

1.FlowLayout : FlowLayout places component in rows from left to right. Components towards the end of row are written on next row, if there is not enough space in the current row.

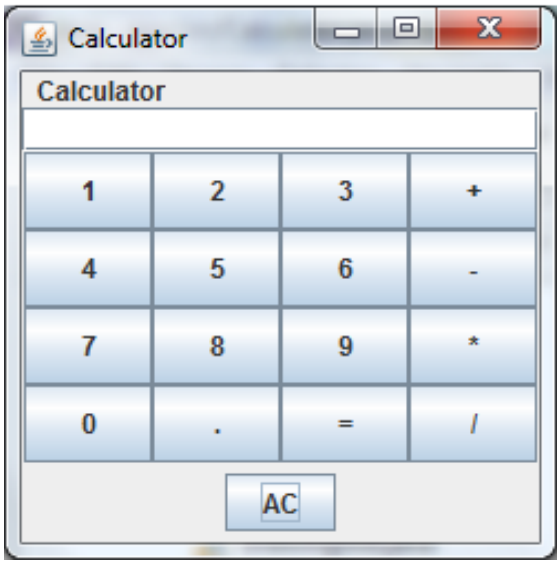
2.BorderLayout : A BorderLayout Manager divides the window into five regions - North, East, West, South and Center. A component can be explicitly added to one of the regions using the add() method of the Container class.

3.GridLayout : A GridLayout Manager places the components in a rectangular grid. Each component's position is identified by a column and row.

4.BoxLayout : It allows multiple components to be laid out either vertically or horizontally.

Example 1.1:

Following example makes use of different layouts to display a calculator as shown in figure.



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

public class Calculator extends JFrame {
     JTextField resultField;

     Calculator() {
          super("Calculator");
          //creating panels
          JPanel p = new JPanel();
          JPanel p1 = new JPanel();
          JPanel p2 = new JPanel();
          JPanel p3 = new JPanel();
          p.setSize(100, 100);
        //creating labels,textfield and buttons
          JLabel titleLabel = new JLabel("Calculator",
                                                JLabel.CENTER);
          resultField = new JTextField(10);
          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 b0 = new JButton("0");
          JButton bpo = new JButton(".");
          JButton be = new JButton("=");
          JButton bp = new JButton("+");
          JButton bs = new JButton("-");
          JButton bm = new JButton("*");
          JButton bd = new JButton("/");
          JButton bclr = new JButton("AC");
        //setting layout for panels
          p.setLayout(new BorderLayout());//divides panel into 5
regions
```

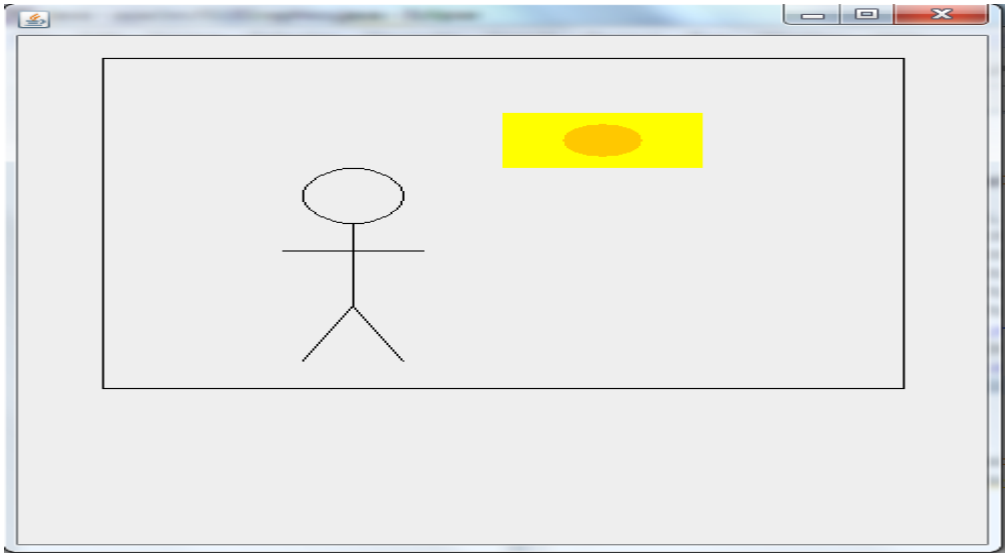
```
            p1.setLayout(new BoxLayout(p1, BoxLayout.Y_AXIS));
//adds components vertically
            p2.setLayout(new GridLayout(4, 4)); //making a grid of
4 by 4
            add(p);
            p1.add(titleLabel);
            p1.add(resultField);
            p.add(BorderLayout.NORTH, p1);

            p2.add(b1);
            p2.add(b2);
            p2.add(b3);
            p2.add(bp);
            p2.add(b4);
            p2.add(b5);
            p2.add(b6);
            p2.add(bs);
            p2.add(b7);
            p2.add(b8);
            p2.add(b9);
            p2.add(bm);
            p2.add(b0);
            p2.add(bpo);
            p2.add(be);
            p2.add(bd);
            p3.add(bclr);
            p.add(BorderLayout.CENTER, p2);
            p.add(BorderLayout.SOUTH, p3);
}

      public static void main(String[] args) {
            JFrame frame = new Calculator();
            frame.setVisible(true);
            // make program quit when you close the window
            frame.setDefaultCloseOperation(frame.EXIT_ON_CLOSE);
            frame.setSize(250, 250);
      }
}
```

Graphics and the Paint Method

A graphics context is an object belonging to the class, Graphics. Instance methods are provided in this class for drawing shapes, text, and images. The Graphics class is an abstract class.

The **paint()** method is called by the JVM implicitly in two circumstances – one when the first time frame is created and displayed and the other when the frame is resized by the user. If the programmer would like to call the paint() method, to draw some graphics on the frame or applet window in the middle of the coding, he is permitted to call **repaint()** method and not paint() method directly.

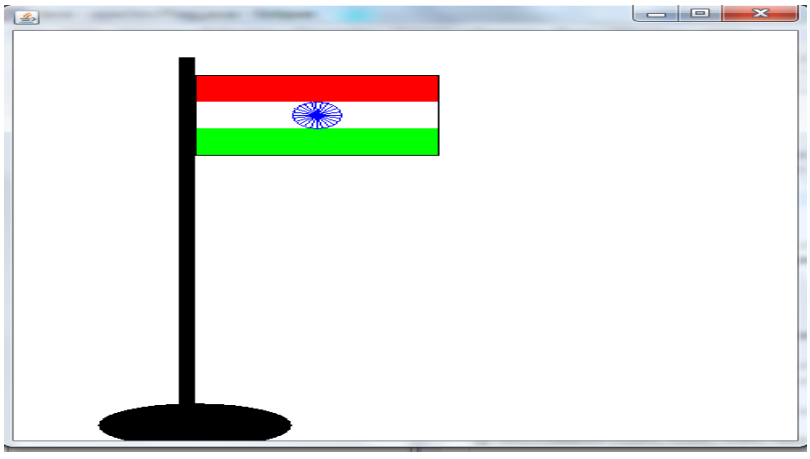
Example1.2: Construct Graphics as given in the figure below:



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

public class GUIGraphics extends JFrame{
     GUIGraphics()
     {
          super();
          setSize(500,500) ;
          setDefaultCloseOperation(EXIT_ON_CLOSE) ;
             show();
     }
     public void paint(Graphics g)
```

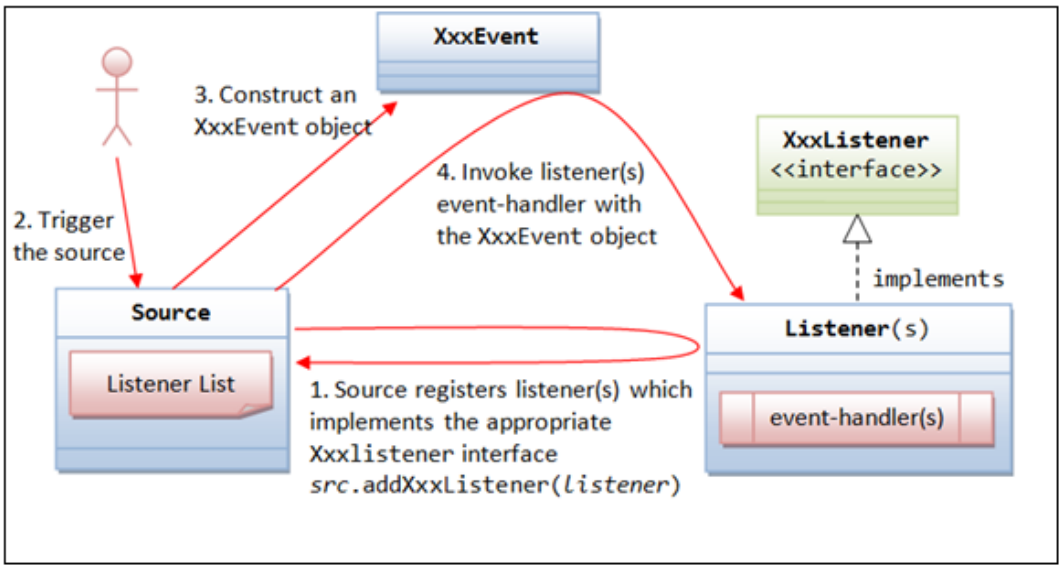
```
    {    g.drawRect(50, 50, 400, 300);
         g.drawOval(150, 150, 50, 50);//face of stick man
         g.drawLine(175, 200, 175, 275);//height of stick man
         g.drawLine(140,225,210,225);//hands of stick man
         g.drawLine(175,275,150,325);//left leg of stick man
         g.drawLine(175,275,200,325);//right leg of stick man
         g.setColor(Color.yellow);
         g.fillRect(250, 100, 100, 50);//yellow rectangle
         g.setColor(Color.orange);
         g.fillOval(280, 110, 40, 30);//orange oval
    }

    public static void main(String args[])
    {    GUIGraphics graphics=new GUIGraphics();
    }
}
```

Exercise 1.1 :
Draw Indian flag using graphics as shown in figure.



Hint to draw 24 equally spaced spokes in circle: Consider a line of length r with end-points (x,y) and (x1,y1) making an angle d (in radian) with X-axis. Given a point (x,y), we can find point(x1,y1) as follows:
x1=x+(**double**)r*Math.*cos*(d);
y1=y+(**double**)r*Math.*sin*(d);
Considering point(x,y) as centre of the circle repeat this procedure 24 times to find point(x1,y1) for different angles d.

3.Event Handling

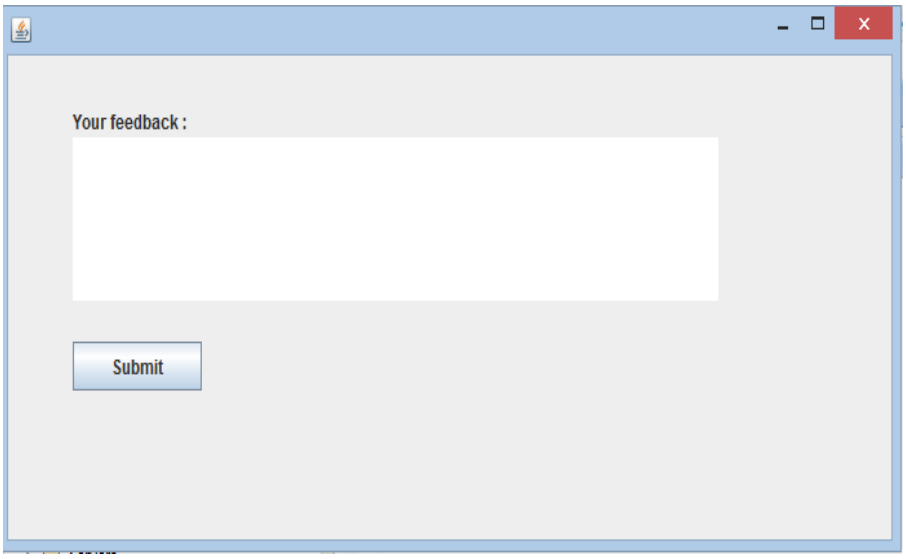


Three objects are involved in the event-handling: a *source*, a *listener*(s) and an *event* object.
1 .The *source* object (such as Button and Textfield) interacts with the user.
2. Upon triggered, it creates an *event* object.
3. This *event* object will be messaged to all the *registered listener* object(s), and an appropriate event-handler method of the listener(s) is called-back to provide the response.

The following table lists some of the events generated by components aand their Listener interfaces.

Component	Event it generates	Listener Interfaces
Button, TextField, Menu	ActionEvent	ActionListener
Frame	WindowEvent	WindowListener
Checkbox, Choice, List	ItemEvent	ItemListener
Scrollbar	AdjustmentEvent	AdjustmentListener
Mouse	MouseEvent	MouseListener, MouseMotionListener
Keyboard	KeyEvent	KeyListener

Exercise2.1:



Modify the code given below which constructs a GUI as shown in figure to perform the following tasks.

(1) Write implementation of method actionPerformed(ActionEvent e). It should do the following:

a) 'Submit' button disappears.

b) Text entered by the user becomes uneditable in text area.

(2) Add one more text area (which is not editable)in the frame.Add it to the commentPanel. Modify method actionPerformed(ActionEvent e)so that it performs the following:

a) When user clicks 'submit' button, text entered in old text area appears in new text area but in red color.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class GUITest implements ActionListener {

     JFrame frame;
     JPanel commentPanel;
     JButton button1;
     JTextArea text1;
     JLabel label;

     GUITest() {

          frame=new JFrame(); //create a new frame
```

```
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            // exists when you close the window
            //set the default size of the frame
            frame.setSize(700,700);
            //Display the frame on the screen
            frame.setVisible(true);

            //creating panel that contains label, text area and
            //button
            commentPanel= new JPanel();
            commentPanel.setLayout(null);

            label=new JLabel("Your feedback :");
            label.setLocation(50, 25);
            label.setSize(100, 30);
            commentPanel.add(label);

            text1=new JTextArea();
            text1.setLocation(50, 50);
            text1.setSize(500, 100);
            commentPanel.add(text1);

            button1 =new JButton("Submit");
            button1.setLocation(50,175);
            button1.setSize(100, 30);
            button1.addActionListener(this);
            commentPanel.add(button1);

            frame.add(commentPanel);
            frame.validate();
      }

      public static void main(String args[]){
            GUITest gui=new GUITest();
      }

      public void actionPerformed(ActionEvent e){
            //see (1)in exercise 2.1
      }
}
```

Exercise 2.2:

Modify the Example 1.1 Calculator.java to implement addition, subtraction, multiplication and division operations using event handling.