

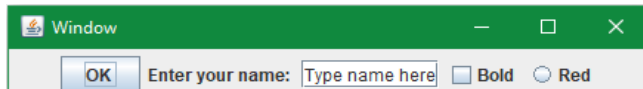
Theory (Frame)

To create a user interface(UI), it is needed to create a frame or an applet to hold the user interface components. To make a frame with Java it is needed to import javax.swing.* package. JFrame is a method in java to make up frame in java.

Code

```
package myframe;
import javax.swing.*;
public class MyFrame {
    public static void main(String[] args) {
        JFrame frame1 = new CustomFrame();
        frame1.setTitle("Window");
        frame1.setLocation(200,100);
        frame1.setVisible(true);
    }
}
class CustomFrame extends JFrame {
    public CustomFrame() {
        JButton Jbt = new JButton("OK ");
        JLabel Jlbl = new JLabel("Enter your name: ");
        JTextField Jtf = new JTextField("Type name here");
        JCheckBox Jcb = new JCheckBox("Bold");
        JRadioButton Jrb = new JRadioButton("Red");
        JComboBox Jcmb = new JComboBox(new String[] { "Red","Green","Blue"});
        JPanel P = new JPanel();
        P.add(Jbt); P.add(Jlbl);
        P.add(Jtf); P.add(Jcb);
        P.add(Jrb); P.add(Jcmb);
        add(P);
        setSize(450,70);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Output



Comment

Here in the above code, a class named CustomFrame was extended from JFrame and in a constructor all the properties of the frame was initialized and all those was taken in a panel and that was added to the program with dimension. Finally in the main(), frame1 object was taken.

Theory (Frame Action)

To create a user interface(UI), it is needed to create a frame or an applet to hold the user interface components. In Frame action java.awt.* and java.swing.border.* these two packages need to be imported. Because in this scenario, different types of Layouts are being used like Flow Layout, Border Layout, Grid Layout.

Code

```
package frameaction;
import java.awt.*;
import javax.swing.border.*;
import javax.swing.*;

public class FrameAction extends JFrame {
    public FrameAction() {
        JPanel P1 = new JPanel(new FlowLayout(FlowLayout.LEFT,2,2));
        JButton Jbtleft = new JButton("Left");
        JButton Jbtcenter = new JButton("Center");
        JButton Jbtright = new JButton("Right");

        Jbtleft.setBackground(Color.YELLOW);
        Jbtcenter.setBackground(Color.GREEN);
        Jbtright.setBackground(Color.ORANGE);
        Jbtright.setToolTipText("This is the Right Button");
        Jbtleft.setToolTipText("This is the Left Button");
        Jbtcenter.setToolTipText("This is the Center Button");

        P1.add(Jbtleft);
        P1.add(Jbtcenter);
        P1.add(Jbtright);
        P1.setBorder(new TitledBorder("The Three Button"));

        Font longFont = new Font("Times Roman",Font.BOLD,18);
        Border lineBorder = new LineBorder(Color.BLACK,2);
        JPanel P2 = new JPanel(new GridLayout(1,2,3,4));
        JLabel Jlblred = new JLabel("Red");
        JLabel Jlblorange = new JLabel("Orange");
        Jlblred.setForeground(Color.RED);
        Jlblorange.setForeground(Color.ORANGE);
        Jlblred.setFont(longFont);
        Jlblorange.setFont(longFont);
        Jlblred.setBorder(lineBorder);
        Jlblorange.setBorder(lineBorder);

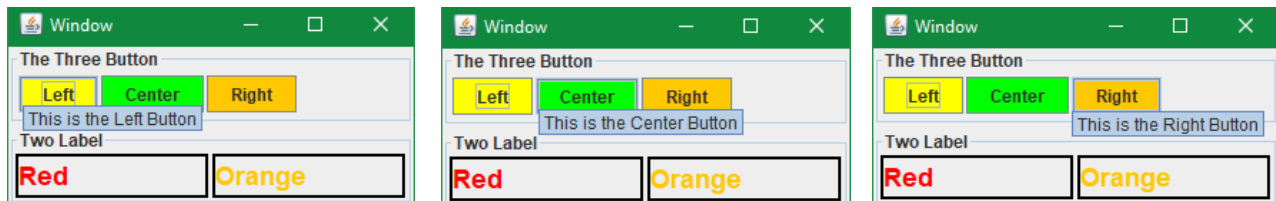
        P2.add(Jlblred);
        P2.add(Jlblorange);
        P2.setBorder(new TitledBorder("Two Label"));
```

```

        setLayout(new GridLayout(2,1,2,3));
        add(P1);
        add(P2);
    }
    public static void main(String[] args) {
        JFrame frame1 = new FrameAction();
        frame1.setTitle("Window");
        frame1.setLocation(400,50);
        frame1.setSize(300,150);
        frame1.setVisible(true);
    }
}

```

Output



Comment

Here in the above code, a class named FrameAction was extended from JFrame and in a constructor all the properties of the frame was initialized with FlowLayout, BorderLayout and GridLayout methods and all those was taken in a panel and that was added to the program with dimension. Finally in the main(), frame1 object was taken and again the dimation of the frame was taken by several methods like setSize, setTitle, SetLocation, setVisible etc.

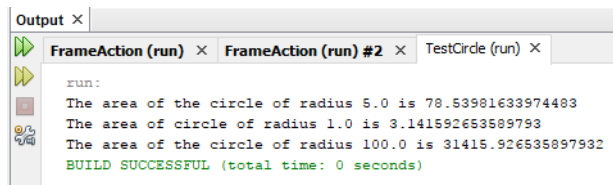
Theory (Circle)

Function overloading is a very common feature of object oriented programming language (OOP). Function overloading is a feature or property where functions having same return type and same function name but different arguments are appeared and when these functions are called in main(), the compiler understand which function is for which operation by checking the arguments.

Code

```
package testcircle;
public class TestCircle {
    public static void main(String[] args) {
        Circle mycircle = new Circle(5.0);
        System.out.println("The area of the circle of radius "+mycircle.radius+" is
"+mycircle.getArea());
        Circle yourcircle = new Circle();
        System.out.println("The area of circle of radius "+yourcircle.radius+" is
"+yourcircle.getArea());
        yourcircle.radius = 100;
        System.out.println("The area of the circle of radius "+yourcircle.radius+" is
"+yourcircle.getArea());
    }
}
class Circle {
    double radius;
    Circle() { radius = 1.0; }
    Circle(double r) { radius = r; }
    double getArea() { return radius*radius*Math.PI; }
}
```

Output



```
Output ×
FrameAction (run) × FrameAction (run) #2 × TestCircle (run) ×
run:
The area of the circle of radius 5.0 is 78.53981633974483
The area of circle of radius 1.0 is 3.141592653589793
The area of the circle of radius 100.0 is 31415.926535897932
BUILD SUCCESSFUL (total time: 0 seconds)
```

Comment

In the above program, a class named Circle was declared and in that class two constructor and a **getArea()** function were declared. In the main() , **mycircle** and **yourcircle** two objects of Circle class was declared. Though both of them are constructor but the program executed successfully by checking the parameter of the constructor and printed them in the output console.