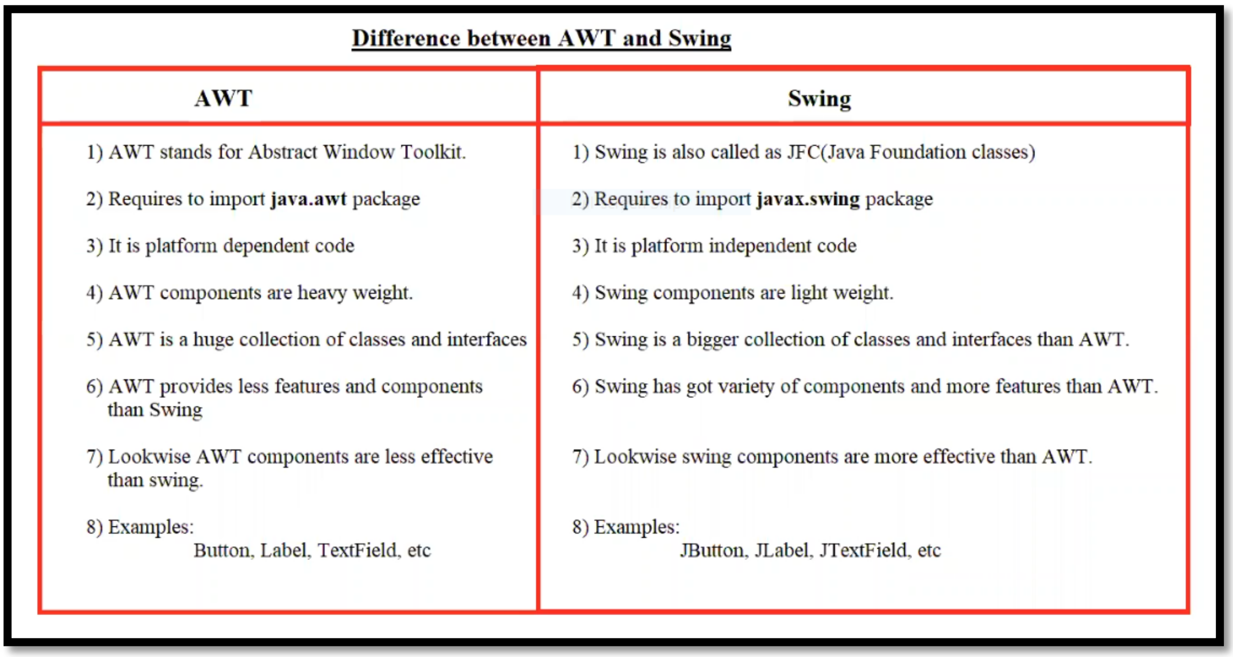
*******************Swing*******************



Difference between AWT and Swing

AWT	Swing
1) AWT stands for Abstract Window Toolkit.	1) Swing is also called as JFC(Java Foundation classes)
2) Requires to import **java.awt** package	2) Requires to import **javax.swing** package
3) It is platform dependent code	3) It is platform independent code
4) AWT components are heavy weight.	4) Swing components are light weight.
5) AWT is a huge collection of classes and interfaces	5) Swing is a bigger collection of classes and interfaces than AWT.
6) AWT provides less features and components than Swing	6) Swing has got variety of components and more features than AWT.
7) Lookwise AWT components are less effective than swing.	7) Lookwise swing components are more effective than AWT.
8) Examples: Button, Label, TextField, etc	8) Examples: JButton, JLabel, JTextField, etc

####JFrame Class####

- JFrame is a standard window which has title bar,menu-bar,borders, minimize,maximum button and resizing corner.

- It is present under javax.swing package.

*Constructor:

1) JFrame() - create window without title.

2) JFrame(String title) - create window with title.

*Methods:

1) void setVisible(true/false)

2) void setSize(width,height)

3) void setTitle(String title)

4) void setDefaultCloseOperation(int operation);

Where:

operation:- JFrame.EXIT_ON_CLOSE

JFrame.HIDE_ON_CLOSE

JFrame.DO_NOTHING_ON_CLOSE

- Above given three static constants of WindowConstants interface.

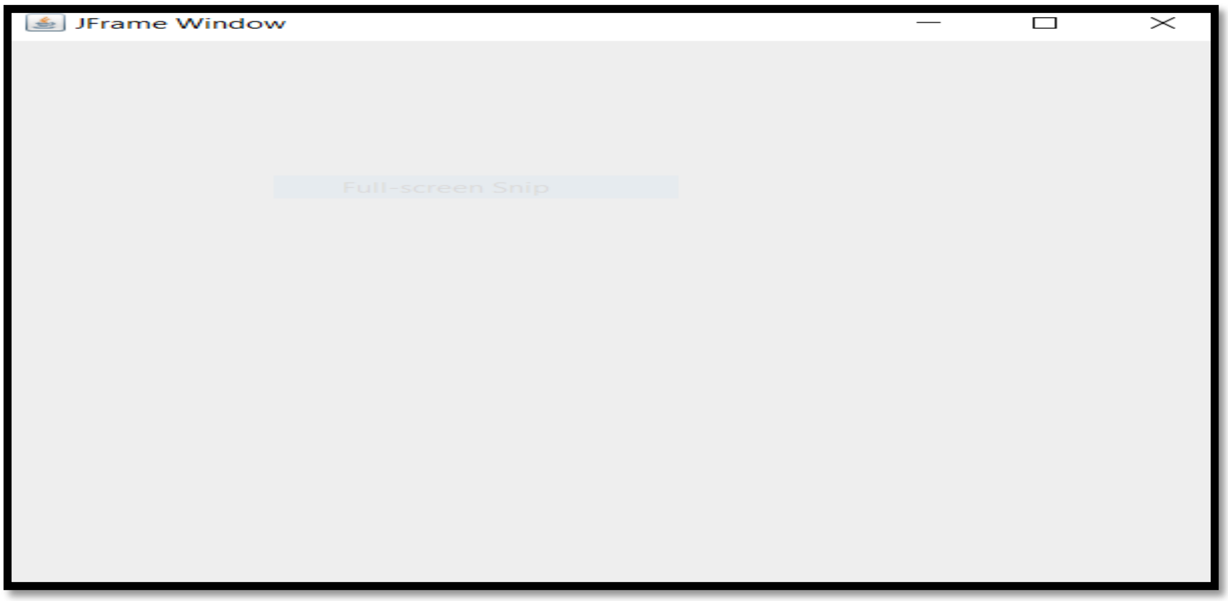
Code:-

1)

```
import javax.swing.*;
class JFrameDemo extends JFrame
{
 JFrameDemo()
 {
 }
 public static void main(String args[])
 {
      JFrameDemo jd=new JFrameDemo();
      jd.setVisible(true);
      jd.setTitle("JFrame Window");
```

```
        jd.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jd.setSize(500,500);
 }
}
```

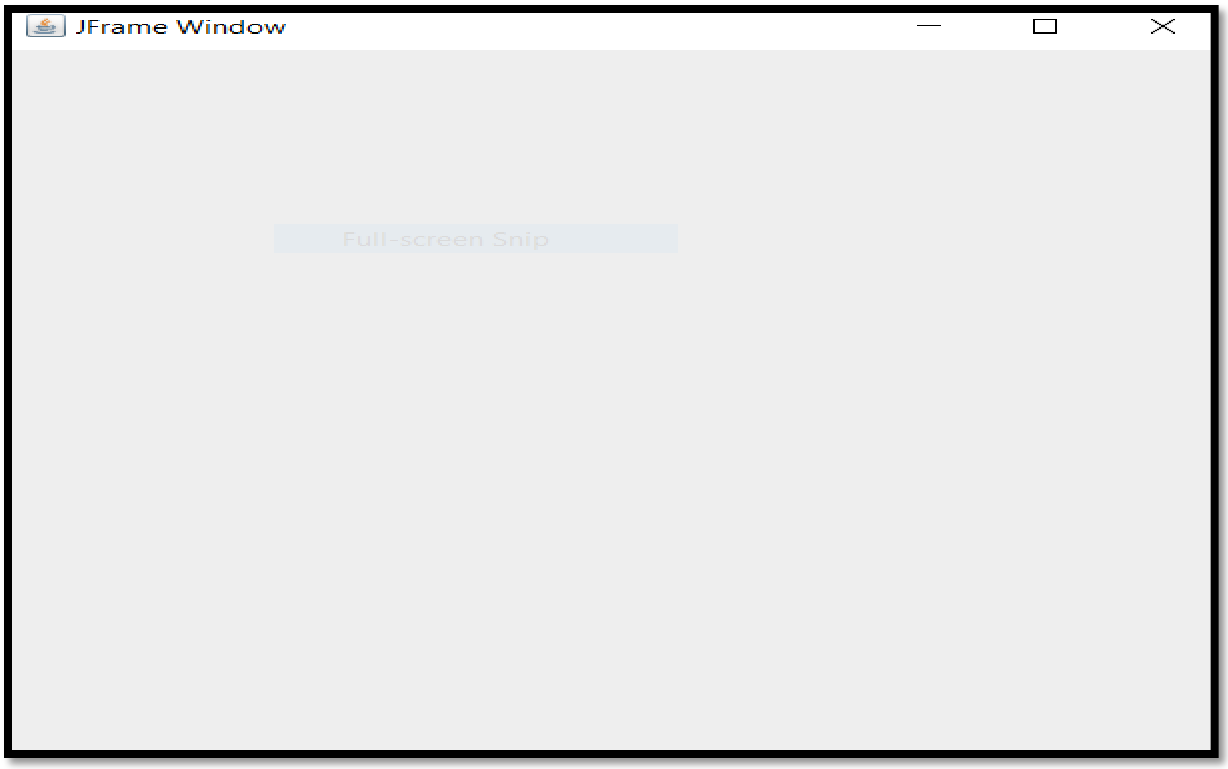
Output:-



2)

```
import javax.swing.*;
class JFrameDemo1 extends JFrame
{
 JFrameDemo1(String title)
 {
        super(title);
 }
 public static void main(String args[])
 {
   JFrameDemo1 jd=new JFrameDemo1("JFrame Window");
        jd.setVisible(true);
        jd.setSize(500,500);
 //jd.setTitle("JFrame Window");
 }
}
```

Output:-

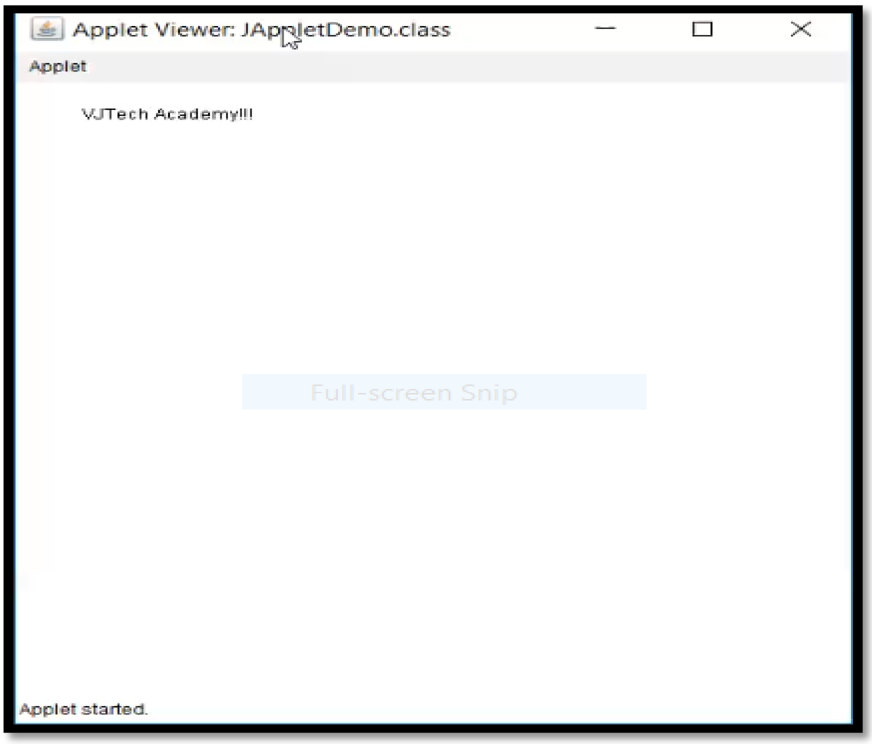


3)

```
import java.applet.*;
import java.awt.*;
import javax.swing.*;
public class JAppletDemo extends JApplet
```

```
{
        public void paint(Graphics g)
        {
                g.drawString("VJTech Academy!!!",40,30);
        }
}
/*<applet code="JAppletDemo.class" width=500 height=500>
</applet>*/
```

Output:-



****

##### ImageIcon Class

-If we want to apply image on any supported components then we can use ImageIcon class.

*****Constructor:**

1) ImageIcon(String filename);

2) ImageIcon(URL url);

****Methods:**

1) int getIconHeight();

2) int getIconWidth();

3) void paintIcon(Component obj, Graphics g,int x,int y);

****

#### JLabel Class

*****Constructor:**

1) JLabel(ImageIcon ii);

2) JLabel(String str);

3) JLabel(String str, ImageIcon ii, int alignment); //alignment -> JLabel.RIGHT, JLabel.LEFT, JLabel.CENTER, JLabel.LEADING, JLabel.TRAILING

4) JLabel();

*****Methods:**

1) void setText(String str)

2) String getText();

3) void setAlignment(int alignment);

4) int getAlignment();

5) void setIcon(ImageIcon ii);

6) ImageIcon getIcon();

Code:-

```
import java.awt.*;

import javax.swing.*;

class JLabelDemo extends JFrame

{

 JLabelDemo()

 {

        Container c=getContentPane();

        FlowLayout f1=new FlowLayout();

        c.setLayout(f1);

         ImageIcon ii=new ImageIcon("vjtech.jpg");

        JLabel L1=new JLabel(ii);

        c.add(L1);

 }

 public static void main(String args[])

 {

        JLabelDemo jld=new JLabelDemo();

        jld.setVisible(true);

        jld.setTitle("JLabel Demo");

        jld.setSize(500,500);

 }

}
```

Output:-

****

#### JTextField ####

- Single line edit control.

- JTextField class derived from JTextComponent class.

- This is most popular component of swing.

***Constructor:**

1) JTextField();

2) JTextField(String str);

3) JTextField(int max_chars);

4) JTextField(String str,int max_chars);

***Methods:**

1) void setText(String str)

2) String getText();

3) void setEchoChar(char ch);

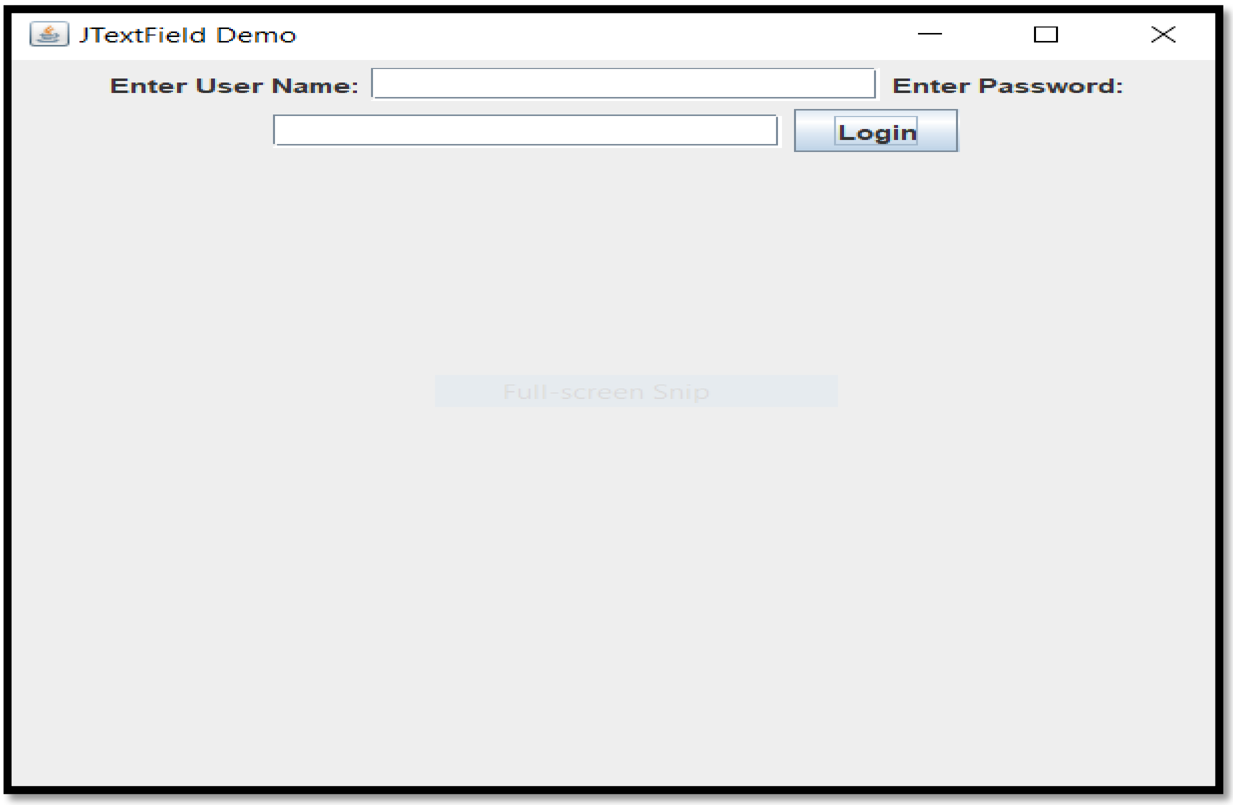
4) Boolean isEdiatble();

5) char getEchoChar();

Code:-

```
import java.awt.*;

import javax.swing.*;

class JTextFieldDemo extends JFrame

{

 JTextFieldDemo()

 {

          Container c1=getContentPane();

        FlowLayout f1=new FlowLayout();

        c1.setLayout(f1);

        JLabel L1=new JLabel("Enter User Name:");

        JLabel L2=new JLabel("Enter Password:");

        JTextField tf1=new JTextField(20);

        JTextField tf2=new JTextField(20);

        JButton b1=new JButton("Login");


        c1.add(L1);

        c1.add(tf1);

        c1.add(L2);
        c1.add(tf2);

        c1.add(b1);

 }

 public static void main(String args[])

 {

        JTextFieldDemo jfd=new JTextFieldDemo();

        jfd.setVisible(true);

        jfd.setTitle("JTextField Demo");

        jfd.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        jfd.setSize(500,500);

 }

}
```

Output:-



#### JTextArea #######

- JTextArea is a multi-line Textfield.

- Here, we can provide multiple lines as input.

- JTextArea is a predefined class which is present under javax.swing package.

-Constructor.**

1) JTextArea();

2) JTextArea(int rows,int cols);

3) JTextArea(String str);

4) JTextArea(String str,int rows,int cols);

-Methods:**

1) String getText();

2) void setText(String str);

3) boolean isEditable();

4) void setEditable(boolean flag);

5) String getSelectedText();

6) void append(String str);

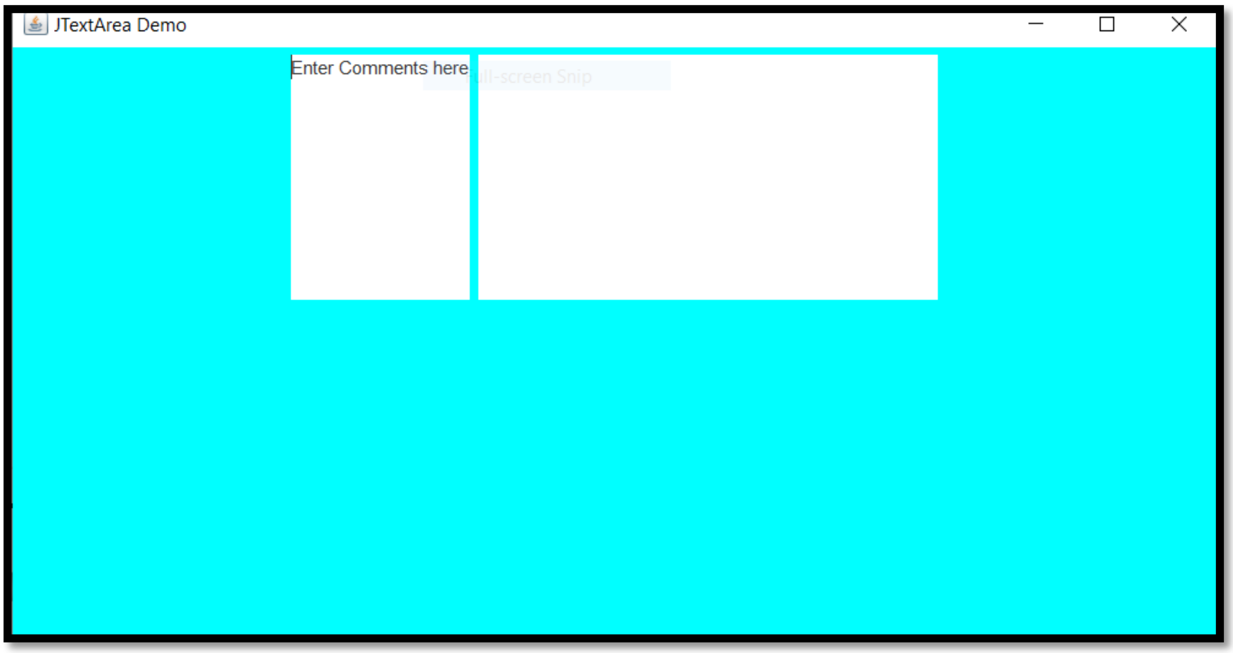
7) void insert(String str, int index);

8) void replaceRange(String str, int startindex, int endindex);

Code:-

```
import java.awt.*;

import javax.swing.*;

class JTextAreaDemo extends JFrame

{
 JTextAreaDemo()
 {
            Container c=getContentPane();
          FlowLayout f1=new FlowLayout();
          c.setLayout(f1);
          c.setBackground(Color.cyan);

          JTextArea ta1=new JTextArea("Enter Comments here",10,10);
          JTextArea ta2=new JTextArea(10,30);

          c.add(ta1);
          c.add(ta2);

 }
 public static void main(String args[])
 {
          JTextAreaDemo j1=new JTextAreaDemo();
          j1.setVisible(true);
          j1.setTitle("JTextArea Demo");
          j1.setSize(800,800);
          j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
}
```

Output:-



****

###### JComboBox

- Swing provides JComboBox class that creates a combo box.

- It will create drop down list(pop-up list).

- JComboBox is a predefined class

- We can select only one item from the given list.

- When user click on combo box control then list of items are displayed and user can select one of the item among the list.

*****Constructor:**

1) JComboBox();

2) JComboBox(Vector obj);

*****Method:**

1) void addItem(Object obj);

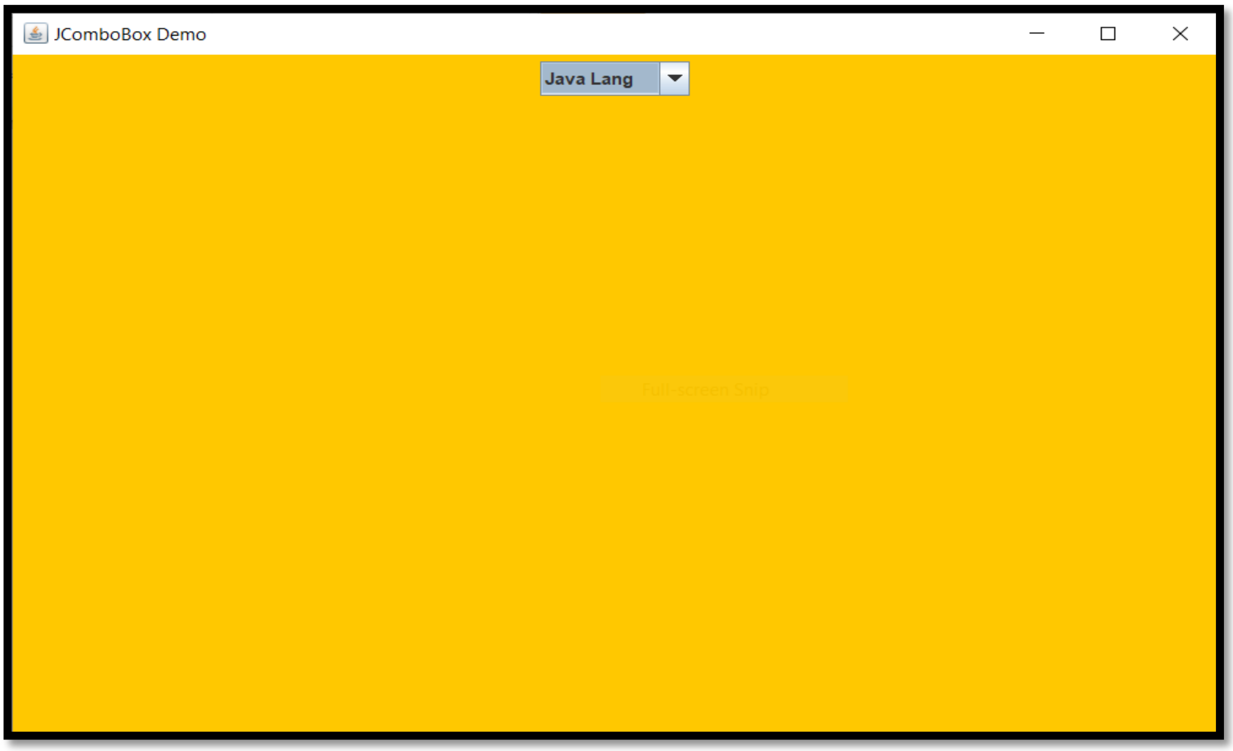
2) void removeItem(Object obj);

3) void removeAllItems();

Code:-

```
import java.awt.*;

import javax.swing.*;

class JComboBoxDemo extends JFrame

{

 JComboBoxDemo()

 {

        Container c=getContentPane();

        FlowLayout f1=new FlowLayout();

        c.setLayout(f1);

        c.setBackground(Color.orange);


        JComboBox jcb=new JComboBox();

        jcb.addItem("C Lang");

        jcb.addItem("C++ Lang");

        jcb.addItem("Java Lang");

        jcb.addItem("Python Lang");

         jcb.addItem("PHP Lang");


        c.add(jcb);

 }
 public static void main(String args[])
 {
        JComboBoxDemo j1=new JComboBoxDemo();
        j1.setVisible(true);
        j1.setTitle("JComboBox Demo");
        j1.setSize(800,800);
        j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
}
```

Output:-



****

#### JButton Class

- It provides Push Button.

- We can set Label or Image on Button.

- Swing provides three types of buttons which is derived from AbstractButton class.

- They are JButton, JToggleButton,JRadioButton

***Constructor:

1) JButton(String str)

2) JButton(ImageIcon ii)

3) JButton(String str,ImageIcon ii);

4) JButton();

***Methods:

1) void setLabel(String str)

2) String getLabel();

3) void setIcon(ImageIcon ii);

4) ImageIcon getIcon();

5) void setDisabledIcon(ImageIcon ii);

6) void setPressedIcon(ImageIcon ii);

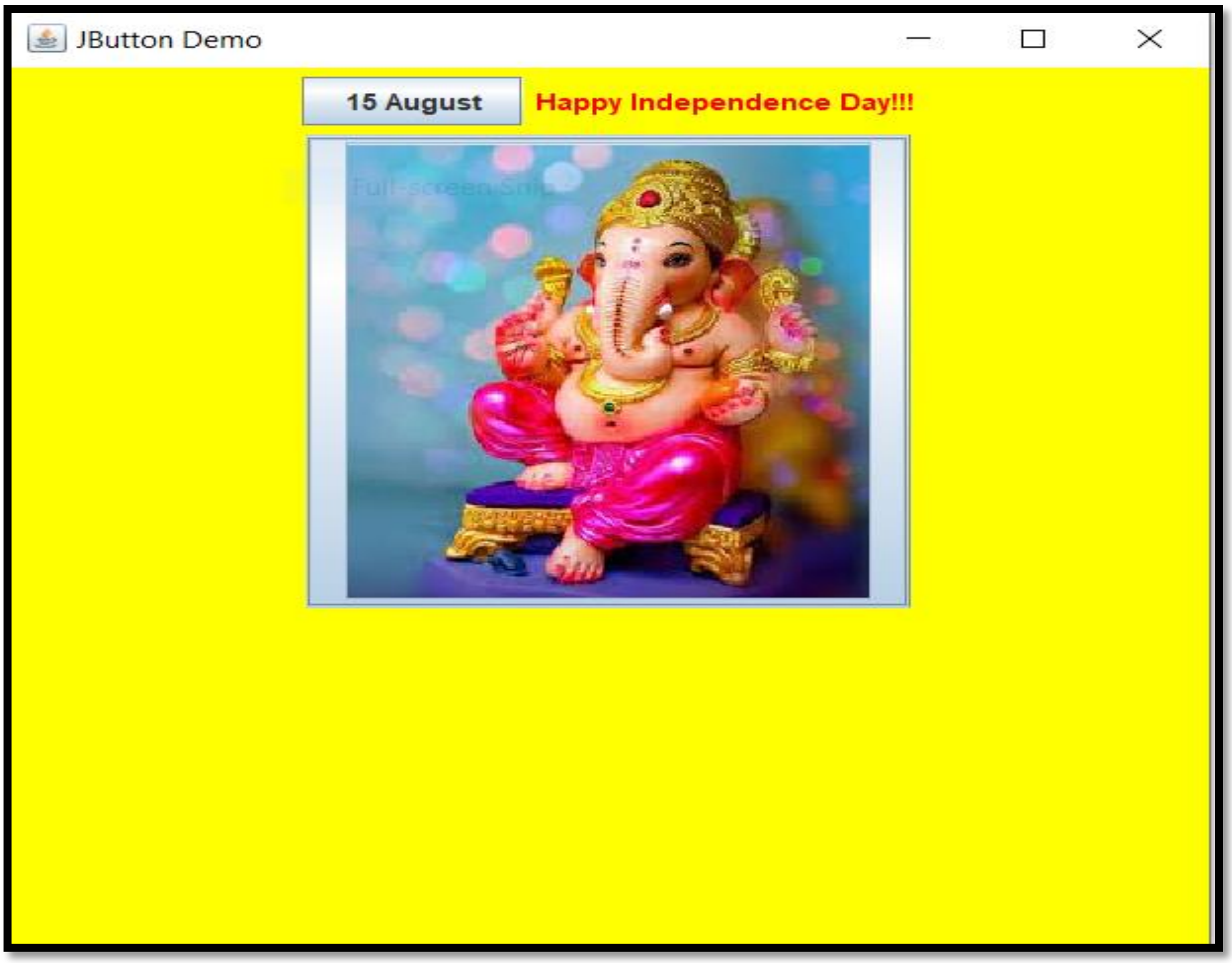
7) void setSelectedIcon(ImageIcon ii);

8) void setRolloverIcon(ImageIcon ii);

Code:-

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

class JButtonDemo extends JFrame implements ActionListener

{

 JButton b1,b2;

 JLabel L1;

 JButtonDemo()

 {

        Container c=getContentPane();

        FlowLayout f1=new FlowLayout();

        c.setLayout(f1);

        c.setBackground(Color.yellow);

        b1=new JButton("15 August");

        ImageIcon ii=new ImageIcon("Morya.jpg");

        b2=new JButton(ii);

        L1=new JLabel("                              ");

        L1.setForeground(Color.red);

        b1.addActionListener(this);

        c.add(b1);

        c.add(L1);

        c.add(b2);
}
 public void actionPerformed(ActionEvent ae)
 {
           L1.setText("Happy Independence Day!!!");
 }
 public static void main(String args[])
 {
         JButtonDemo j1=new JButtonDemo();
         j1.setVisible(true);
         j1.setTitle("JButton Demo");
         j1.setSize(500,500);
         j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
}
```

Output:-



****

### JCheckBox Class

- It will creates checkbox which has rectangular box and its associated label.

- JCheckBox class is a predefined class which is present under javax.swing package.

- JCheckBox class is derived from JToggleButton class.

- JCheckBox has two states ON and OFF.

- Here, we can set Images to the checkbox.

*****Constructor:**

1) JCheckBox(String str);

2) JCheckBox(String str,boolean state);

3) JCheckBox(ImageIcon ii)

4) JCheckBox(ImageIcon ii, boolean state);

5) JCheckBox(String str,ImageIcon ii);

6) JCheckBox(String str,ImageIcon ii,boolean state);

***Methods

1) void setText(String str);

2) String getText();

3) void setIcon(ImageIcon ii);

4) ImageIcon getIcon();

5) boolean getState();

6) boolean isSelected();

IMP-Check Box Always Off.

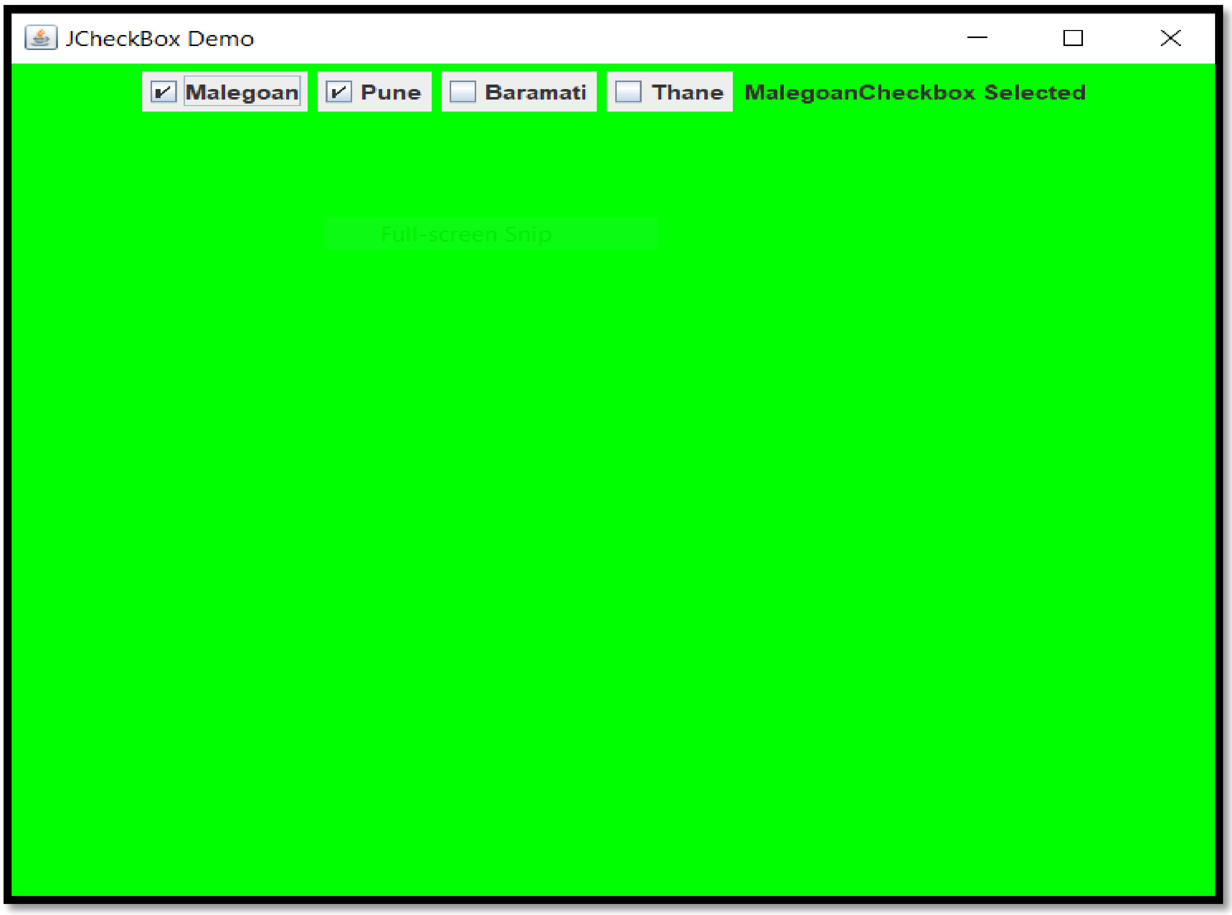
Code:-

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

class JCheckBoxDemo extends JFrame implements ItemListener

{

  JCheckBox jc1,jc2,jc3,jc4;

  JLabel L1;

  JCheckBoxDemo()

  {
```

```
            Container c=getContentPane();

            FlowLayout f1=new FlowLayout();

           c.setLayout(f1);

           c.setBackground(Color.green);


            jc1=new JCheckBox("Malegoan",true);

            jc2=new JCheckBox("Pune");

            jc3=new JCheckBox("Baramati");

            jc4=new JCheckBox("Thane");

            L1=new JLabel("                                        ");


           jc1.addItemListener(this);

           jc2.addItemListener(this);

           jc3.addItemListener(this);

           jc4.addItemListener(this);


           c.add(jc1);

           c.add(jc2);

           c.add(jc3);

           c.add(jc4);

           c.add(L1);

 }

 public void itemStateChanged(ItemEvent ie)
```

```
{
        if(jc1.isSelected())
        {
                L1.setText(jc1.getText()+"Checkbox Selected");
        }
        else if(jc2.isSelected())
        {
                L1.setText(jc2.getText()+"Checkbox Selected");
        }
        else if(jc3.isSelected())
        {
                L1.setText(jc3.getText()+"Checkbox Selected");
        }
        else if(jc4.isSelected())
        {
                L1.setText(jc4.getText()+"Checkbox Selected");
        }
}
public static void main(String args[])
{
        JCheckBoxDemo j1=new JCheckBoxDemo();
        j1.setVisible(true);
        j1.setTitle("JCheckBox Demo");
        j1.setSize(600,600);

        j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 }

}
```

Output:-



MalegoanCheckbox Selected

****

#### JRadioButton Class

- It will create radio button.

- It is used to select one option from multiple options.

- It is widely used in exam systems or quiz.

- JRadioButton class is derived from JToggleButton class.

- It is present under javax.swing package.

- Here, we can set image to the radio button.

- If user selected radio button then automatically deselected previous radio button. For this you need to create ButtonGroup class object and use add methods to add all radio button in this group.

void add(JRadioButton obj);

*****Constructor:**

1) JRadioButton(String str)

2) JRadioButton(String str, boolean state);

3) JRadioButton(ImageIcon ii);

4) JRadioButton(ImageIcon ii, boolean state);

5) JRadioButton(String str, ImageIcon ii);

6) JRadioButton(String str, ImageIcon ii, boolean state);

*****Methods:**

1) void setText(String str);

2) String getText()

3) void setIcon(ImageIcon ii);

4) ImageIcon getIcon();

5) void setEnabled(boolean state);

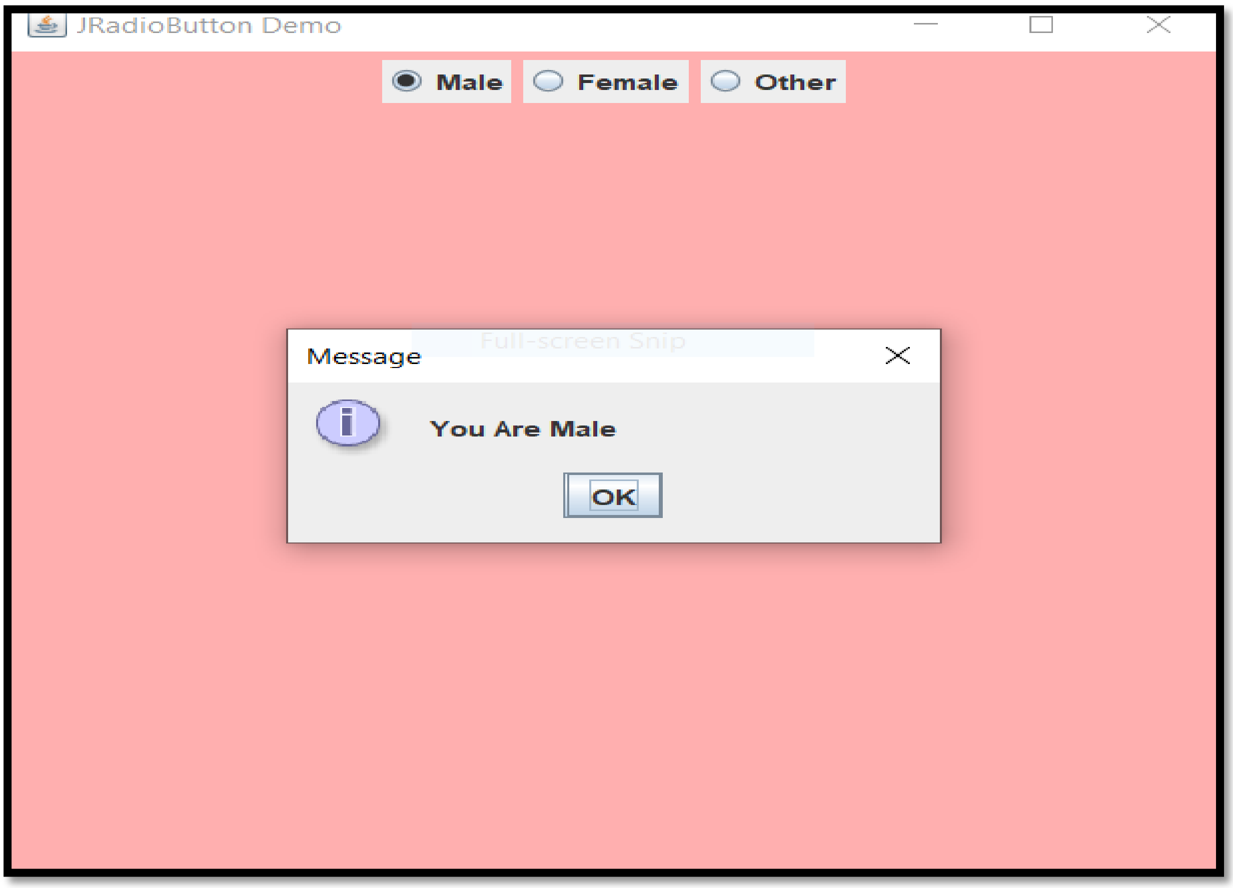
6) boolean isSelected();

Code:-

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;
```

```
class JRadioButtonDemo extends JFrame implements ActionListener
{
 JRadioButton b1,b2,b3;
 JRadioButtonDemo()
 {
        Container c=getContentPane();
        FlowLayout f1=new FlowLayout();
        c.setLayout(f1);
        c.setBackground(Color.pink);

         b1=new JRadioButton("Male");
         b2=new JRadioButton("Female");
         b3=new JRadioButton("Other");
         ButtonGroup bg=new ButtonGroup();

        bg.add(b1);
        bg.add(b2);
        bg.add(b3);

        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        c.add(b1);

        c.add(b2);

        c.add(b3);


 }

 public void actionPerformed(ActionEvent ae)

 {

          if(b1.isSelected())

          {

                    JOptionPane.showMessageDialog(this,"You Are Male");//This
        Method  is used to display the Message on Current Frame.

          }

          else if(b2.isSelected())

          {

                    JOptionPane.showMessageDialog(this,"You Are Female");

          }

          else

          {

                    JOptionPane.showMessageDialog(this,"You have Selected
                           Other");

          }

 }

 public static void main(String args[])
 {
        JRadioButtonDemo j1=new JRadioButtonDemo();
        j1.setVisible(true);
        j1.setTitle("JRadioButton Demo");
        j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        j1.setSize(500,500);
 }
}
```

Output:-



You Are Male

************Swing Advance Component************

#### JTabbedPane Class

- JTabbedPane is a predefined class which is present under javax.swing package.

- It will creates multiple tabs and each tab has its own title and related components.

- We can see only one tab at a time. We can not open multiple tabs at same time.

- We use JTabbedPane for providing configuration related options.

***** Constructor:**

1) JTabbedPane();

***** Method:**

1) void addTab(String title,Component obj);

***** Procedure:**

1) Define required no of Panel classes which contains different components.

2) Create JTabbedPane object

3) Use addTab() method to add tabs into JTabbedPane(title and component object)

4) Add JTabbedPane object into Container.

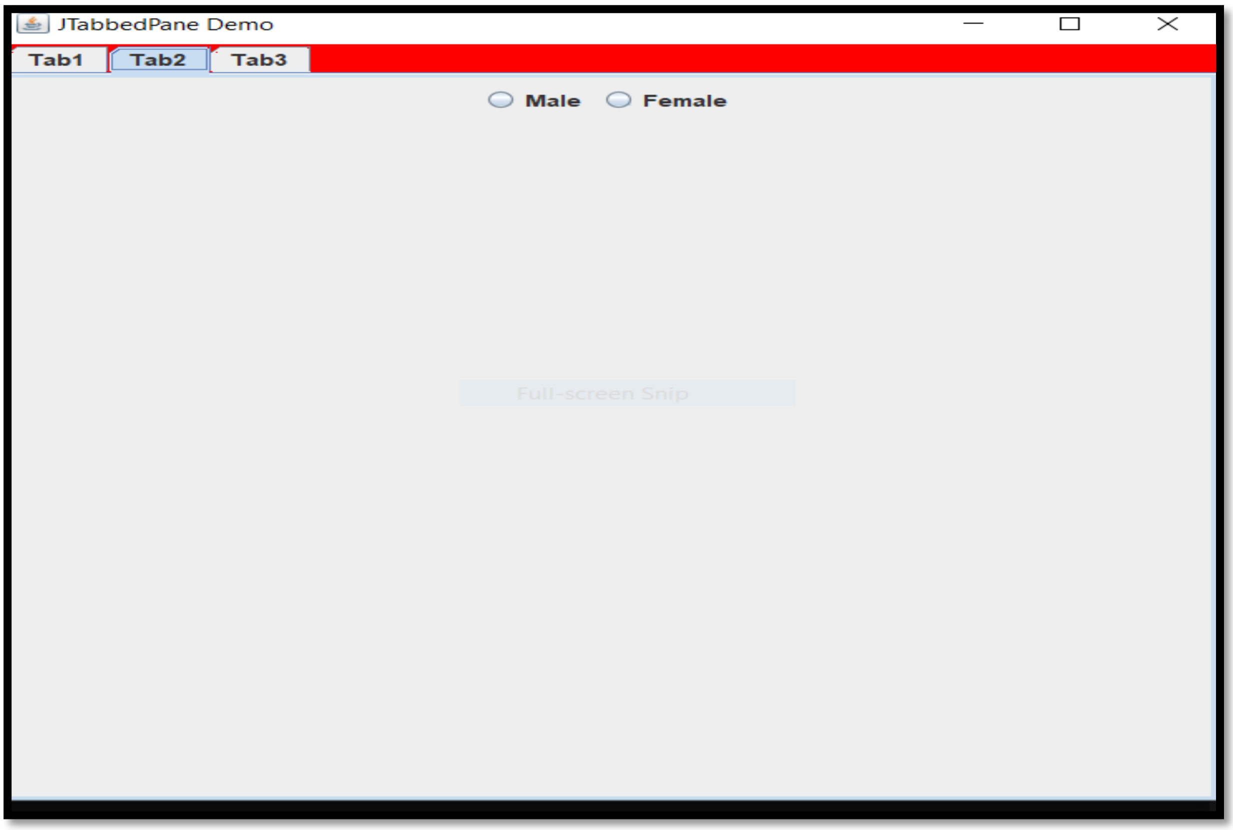
Code:-

```
import java.awt.*;
import javax.swing.*;
class JPanel1 extends JPanel
{
  JPanel1()
 {
          JButton b1=new JButton("Ok");
         JButton b2=new JButton("Cancel");
         JButton b3=new JButton("Retry");

         add(b1);
         add(b2);
         add(b3);
 }
}
class JPanel2 extends JPanel
{
 JPanel2()
 {
          JRadioButton r1=new JRadioButton("Male");
         JRadioButton r2=new JRadioButton("Female");
         ButtonGroup bg=new ButtonGroup();
```

```
            bg.add(r1);

            bg.add(r2);

            add(r1);

            add(r2);

  }

}

class JPanel3 extends JPanel

{

  JPanel3()

  {

             JComboBox cb=new JComboBox();

            cb.addItem("India");

            cb.addItem("US");

            cb.addItem("UK");


            add(cb);

  }

}

class JTabbedPaneDemo extends JFrame

{

  JTabbedPaneDemo()

  {
```

```
            Container c=getContentPane();

            c.setBackground(Color.red);

            JTabbedPane jtp=new JTabbedPane();

            JPanel1 p1=new JPanel1();

            JPanel2 p2=new JPanel2();

            JPanel3 p3=new JPanel3();


            jtp.addTab("Tab1",p1);

            jtp.addTab("Tab2",p2);

            jtp.addTab("Tab3",p3);


            c.add(jtp);

  }

  public static void main(String args[])

  {

             JTabbedPaneDemo j1=new JTabbedPaneDemo();

             j1.setVisible(true);

             j1.setTitle("JTabbedPane Demo");

             j1.setSize(600,600);

             j1.setDefaultCloseOperation(EXIT_ON_CLOSE);

  }

}
```

Output:-



#### JScrollPane Class ####

- JScrollPane is a predefined class which is present under javax.swing package

- JScrollPane class provides rectangular scrollable area in which components can be viewed.

***** Constructor:**

1) JScrollPane(Component obj)

2) JScrollPane(int vertical, int horizontal)

3) JScrollPane(Component obj,int vertical,int horizontal)

Where:

vertical,horizontal=> this is integer constants(defined in ScrollPaneConstants interface) which defines vertical and horizontal scrollbar.

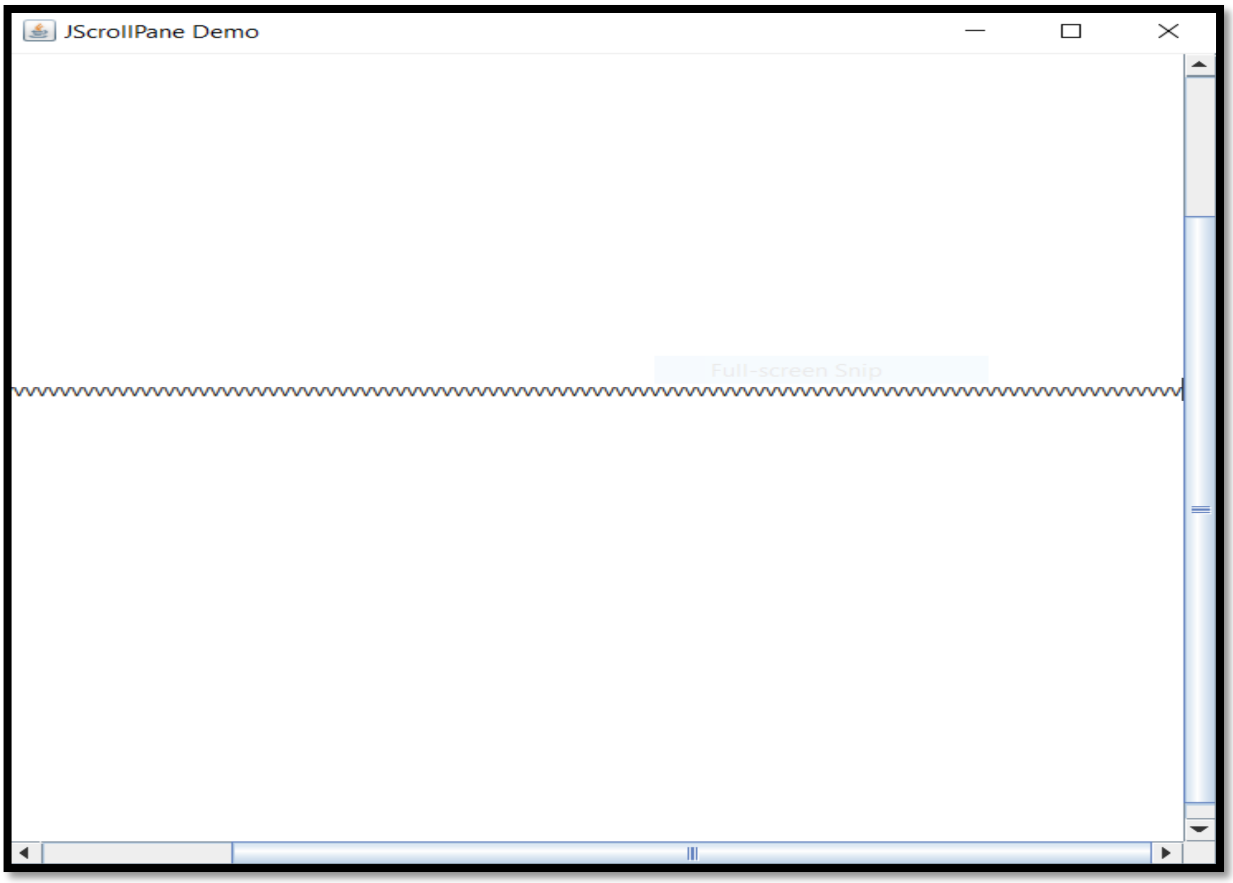
HORIZONTAL_SCROLLBAR_ALWAYS
VERTICAL_SCROLLBAR_ALWAYS
HORIZONTAL_SCROLLBAR_AS_NEEDED
VERTICAL_SCROLLBAR_AS_NEEDED

Hint:-JScrollPane Class will Provide One Rectangular Scrollable Area In which we can display Our Components.

Code:-

```
import java.awt.*;

import javax.swing.*;

class JScrollPaneDemo extends JFrame

{

        JScrollPaneDemo()

        {

          Container c=getContentPane();

          c.setBackground(Color.red);

          JTextArea ta1=new JTextArea("Good Morning Friends Have A Nice
        Day");


          int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;

          int h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;

          JScrollPane jsp=new JScrollPane(ta1,v,h);


          c.add(jsp);
        }
   public static void main(String args[])
        {
                JScrollPaneDemo j1=new JScrollPaneDemo();
                j1.setVisible(true);
                j1.setTitle("JScrollPane Demo");
                j1.setSize(600,600);
                j1.setDefaultCloseOperation(EXIT_ON_CLOSE);
        }
}
```

Output:-



##### JTree Class

- JTree is a predefined class which is present under javax.swing package.

- Hierarchical view of data is represented by using JTree.

- This component allows use to expand or collapse the individual subtree.

- JTree will derived properties of JComponent.

*** Constructor:

1) JTree(Hashtable ht)

2) JTree(Object obj[])

3) JTree(TreeNode tn)

4) JTree(Vector v);

- DefaultMutableTreeNode class implements MutableTreeNode interface.

1) DefaultMutableTreeNode(Object obj)

**Method:

void add(MutableTreeNode child);

***Procedure:

1) Create JTree Object

2) Create JscrollPane

3) Add tree to the scrollpane.

4) Add ScrollPane to the container.

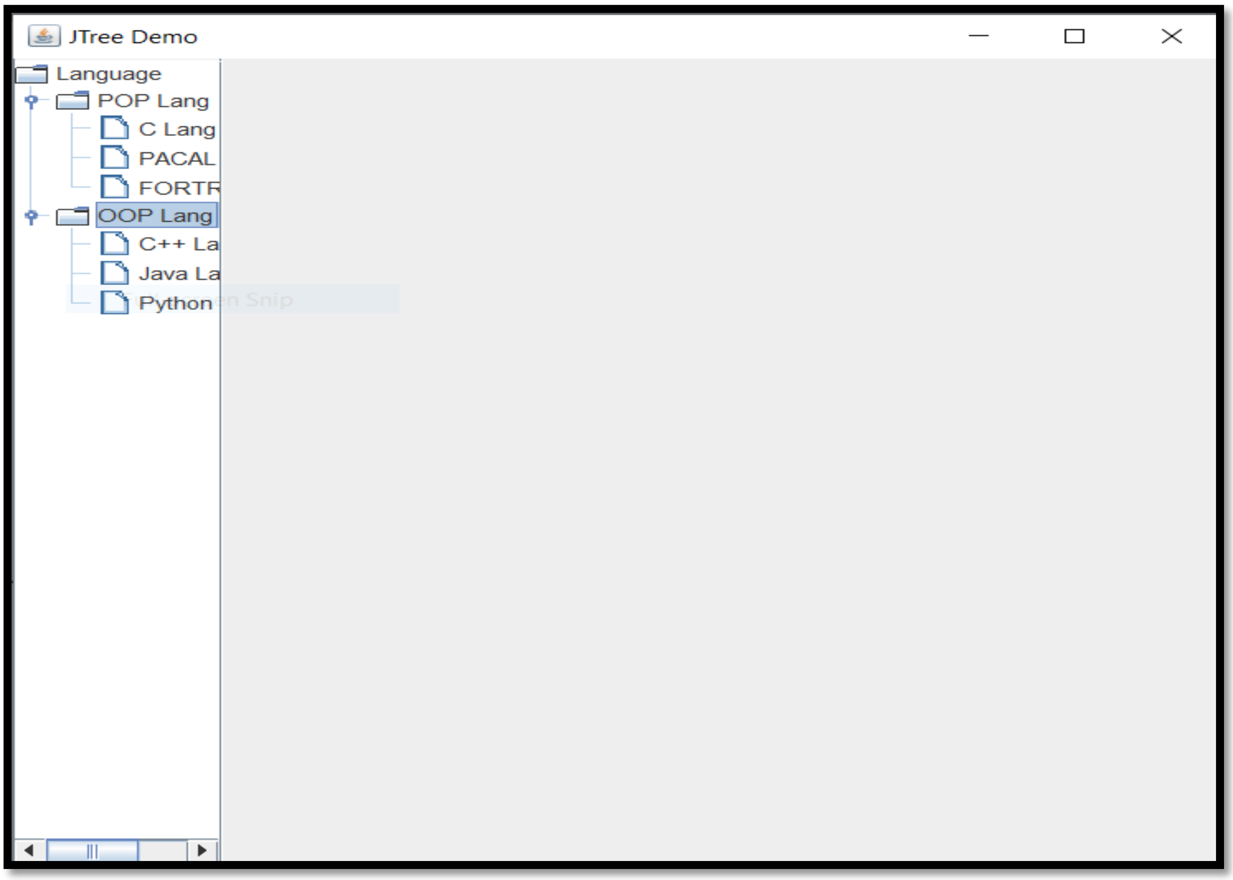
Code:-

```
import javax.swing.*;

import javax.swing.tree.*;

import java.awt.*;

class JTreeNodeDemo extends JFrame

{

		JTreeNodeDemo()

		{

			Container c=getContentPane();

			BorderLayout b1=new BorderLayout();

			c.setLayout(b1);


			DefaultMutableTreeNode root=new
	DefaultMutableTreeNode("Language");

			DefaultMutableTreeNode r1=new
	DefaultMutableTreeNode("POP Lang");

			DefaultMutableTreeNode r2=new
	DefaultMutableTreeNode("OOP Lang");

			root.add(r1);

			root.add(r2);


			DefaultMutableTreeNode r11=new
	DefaultMutableTreeNode("C Lang");

			DefaultMutableTreeNode r12=new
	DefaultMutableTreeNode("PACAL Lang");
```

```
                    DefaultMutableTreeNode r13=new
          DefaultMutableTreeNode("FORTRAN Lang");

                    r1.add(r11);

                    r1.add(r12);

                    r1.add(r13);


                    DefaultMutableTreeNode r21=new
          DefaultMutableTreeNode("C++ Lang");

                    DefaultMutableTreeNode r22=new
          DefaultMutableTreeNode("Java Lang");

                    DefaultMutableTreeNode r23=new
DefaultMutableTreeNode("Python Lang");

                    r2.add(r21);

                    r2.add(r22);

                    r2.add(r23);


                    JTree jt=new JTree(root);


                    int
          v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;

                    int
          h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;


                    JScrollPane jsp=new JScrollPane(jt,v,h);

                    c.add(jsp,BorderLayout.WEST);
```

```
            }
            public static void main(String args[])
            {
                        JTreeNodeDemo j1=new JTreeNodeDemo();
                        j1.setVisible(true);
                        j1.setTitle("JTree Demo");
                        j1.setSize(600,600);
                       j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            }
}
```

Output:-



##### JTable Class

- JTable is a predefined class which is present under javax.swing package.

- JTable is a component which displays rows and column of data.

- If we want to display the data in tabular format then we can use JTable class.

***Constructor:

1) JTable(Object data[][],Object ColHeads[])

***Procedure:

1) Create JTable object with data and column heading.

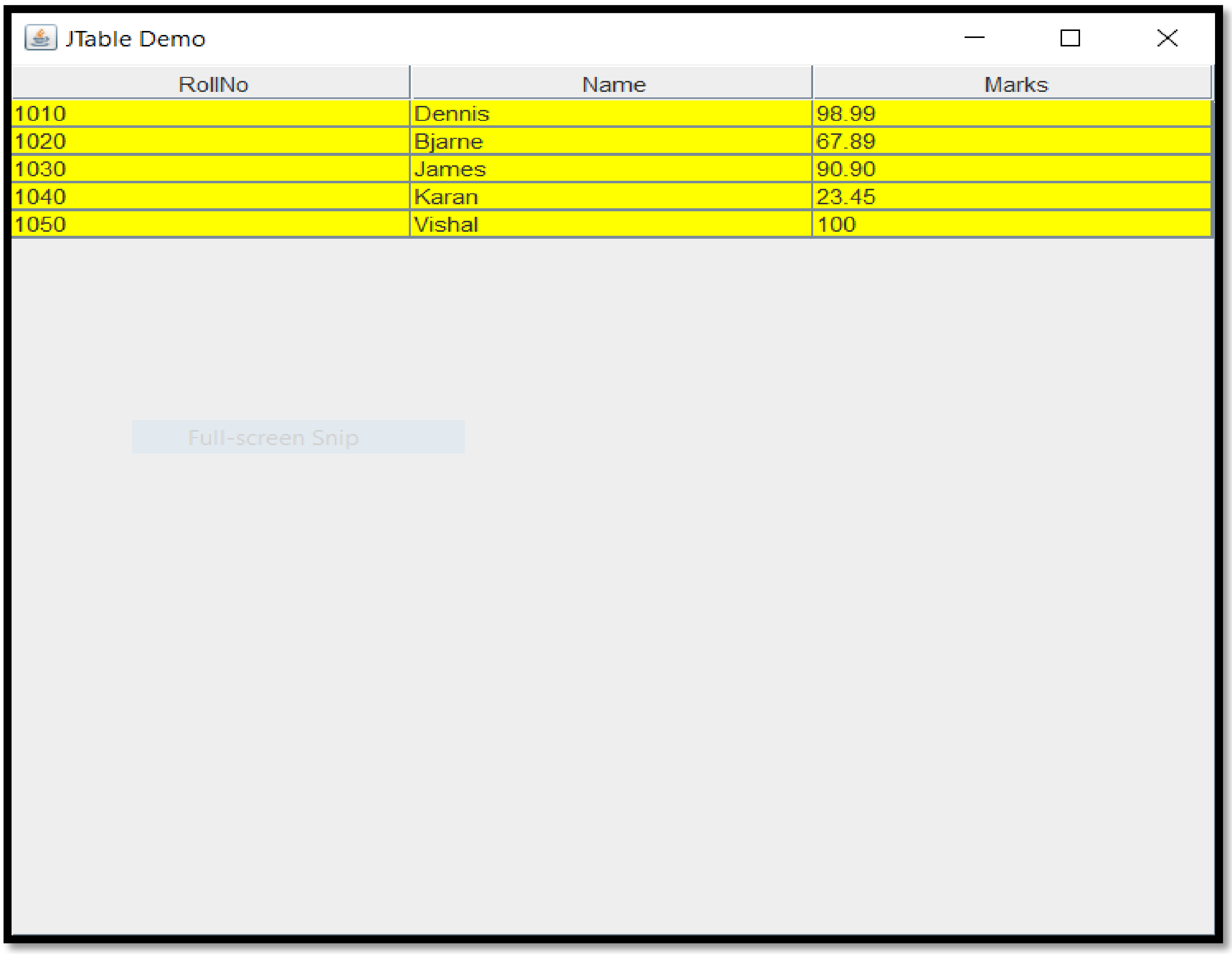
2) Create JScrollPane object and add JTable object in it.

3) Add Scrollpane into the container.

Code:-

```
import javax.swing.*;

import java.awt.*;

class JTableDemo extends JFrame

{

 JTableDemo()

 {

        Container c=getContentPane();

        BorderLayout b1=new BorderLayout();

        c.setLayout(b1);

        String colHeads[]={"RollNo","Name","Marks"};

        String data[][]={

                        {"1010","Dennis","98.99"},
                                            {"1020","Bjarne","67.89"},
                                            {"1030","James","90.90"},
                                            {"1040","Karan","23.45"},
                                             {"1050","Vishal","100"},
                                       };

        JTable jt=new JTable(data,colHeads);
        jt.setBackground(Color.yellow);
        int v=ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED;
        int h=ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED;
        JScrollPane jsp=new JScrollPane(jt,v,h);
        c.add(jsp,BorderLayout.CENTER);
 }
 public static void main(String args[])
 {
        JTableDemo j1=new JTableDemo();
        j1.setVisible(true);
        j1.setTitle("JTable Demo");
        j1.setSize(600,600);
        j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 }
}
```

Output:-



RollNo	Name	Marks
1010	Dennis	98.99
1020	Bjarne	67.89
1030	James	90.90
1040	Karan	23.45
1050	Vishal	100

****

#### JProgressBar Class

- JProgressBar is a predefined class which is present under javax.swing package.

- JProgressBar is a component that graphically shows the progress of task.

- In this progress bar, we can display the task completion percentage.

******Constructor:**

1) JProgressBar() - create horizontal progress bar.

2) JProgressBar(int orient) - orient => SwingConstants.VERTICAL, SwingConstants.HORIZONTAL

3) JProgressBar(int orient,int min,int max)

4) JProgressBar(BoundedRangeModel model)

5) JProgressBar(int min,int max)

*****Methods:**

1) void setValue(int val);

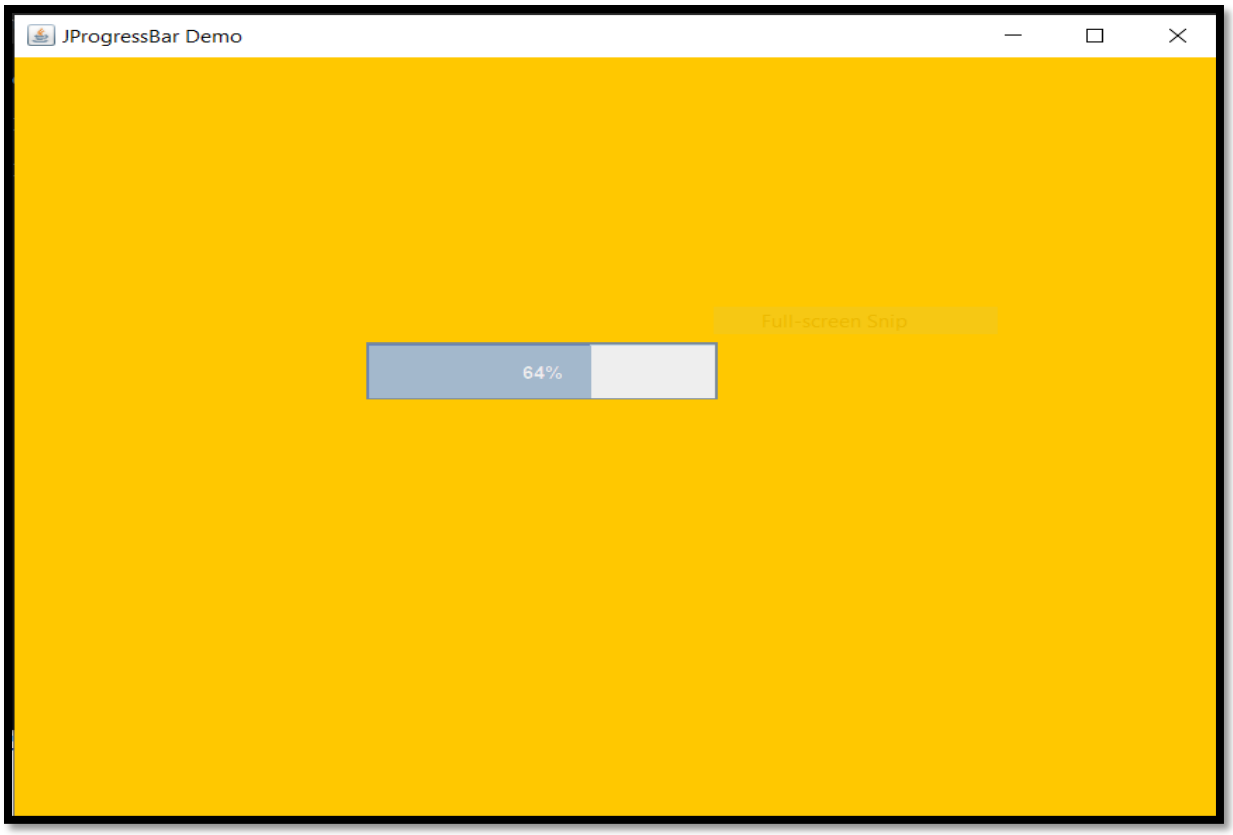
2) void setStringPainted(boolean flag);

Code:-

```
import java.awt.*;

import javax.swing.*;

class JProgressBarDemo extends JFrame

{

  JProgressBar jpb;

  JProgressBarDemo()

  {

              Container c=getContentPane();

              c.setLayout(null);

              c.setBackground(Color.orange);


              jpb=new JProgressBar(0,3000);

              jpb.setBounds(200,200,200,40);

              jpb.setValue(0);

              jpb.setStringPainted(true);
```

```
                c.add(jpb);


  }
  public void ChangeProgressBarValue()
  {
          int i=0;
          while(i<3000)
          {
                  jpb.setValue(i);
                  i=i+20;
                  try
                  {
                          Thread.sleep(150);
                  }
                  catch(Exception e)
                  {
                  }
          }
  }


  public static void main(String args[])
  {
        JProgressBarDemo j1=new JProgressBarDemo();
        j1.setVisible(true);
        j1.setTitle("JProgressBar Demo");
        j1.setSize(700,700);
        j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        j1.ChangeProgressBarValue();

  }
}
```

Output:-



#### JToolTip Class

- It is used to display one pop-up message when your mouse pointer stays on a component.

- This pop-up message can be a short description or labelled information of any component.

- We can add tooltip text to almost all the components of Java Swing.

- To add tool tip to a component object. we have to use following method.

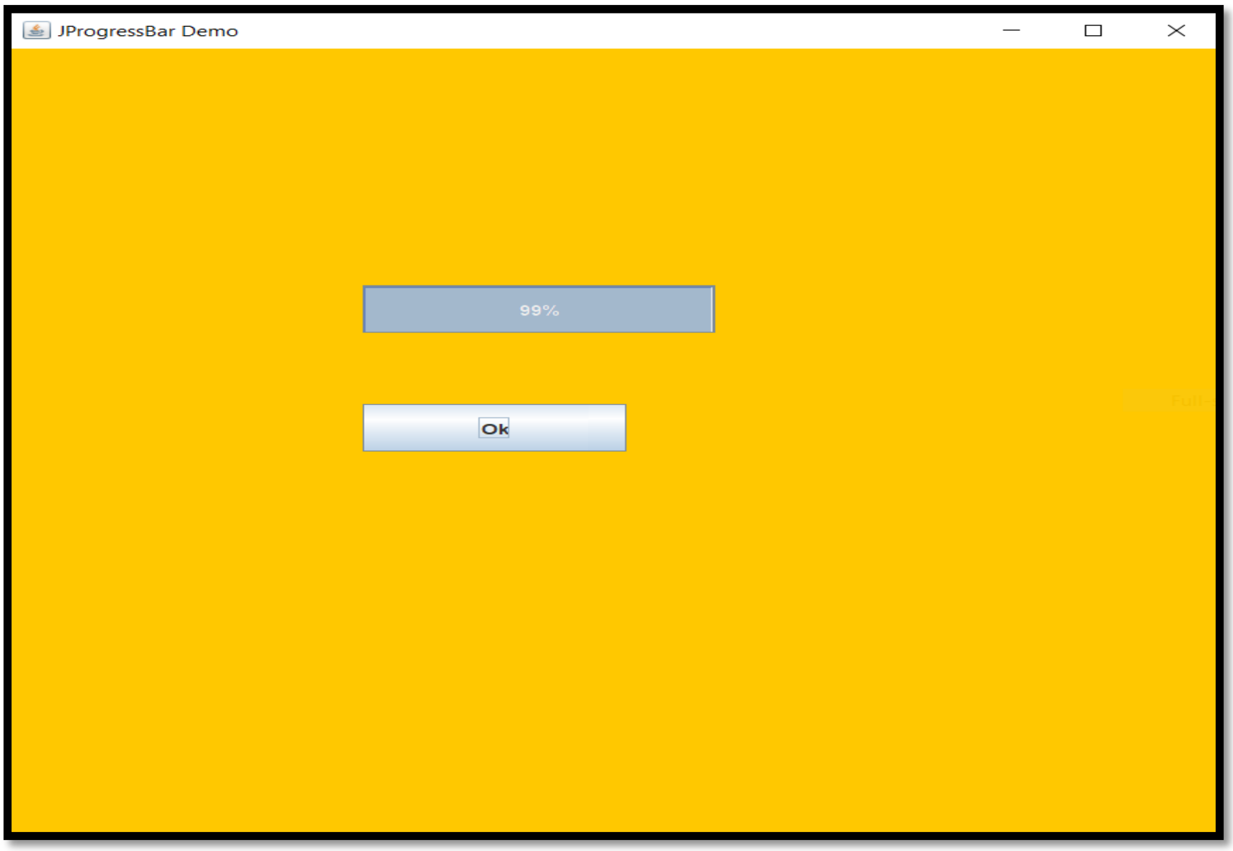
- void setToolTipText(String str);

Code:-

```
import java.awt.*;

import javax.swing.*;

class JToolTipClass extends JFrame

{

  JProgressBar jpb;

  JToolTipClass()

  {

                Container c=getContentPane();

                c.setLayout(null);

                c.setBackground(Color.orange);

                JButton b1=new JButton("Ok");

                b1.setToolTipText("This is JButton Component Which is present
                Under Javax.swing Package");

                jpb=new JProgressBar(0,3000);

                jpb.setBounds(200,200,200,40);
                b1.setBounds(200,300,150,40);

                jpb.setValue(0);

                jpb.setStringPainted(true);

                c.add(jpb);

                c.add(b1);

}

public void ChangeProgressBarValue()

{

        int i=0;

        while(i<3000)

        {

                jpb.setValue(i);

                i=i+20;

                try

                {

                        Thread.sleep(150);

                }

                catch(Exception e)

                {

                }

        }

}
```

```
  public static void main(String args[])
  {
        JToolTipClass j1=new JToolTipClass();
        j1.setVisible(true);
        j1.setTitle("JProgressBar Demo");
        j1.setSize(700,700);
        j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        j1.ChangeProgressBarValue();
  }
}
```

Output:-



##### JSlider Class

- JSlider class is a predefined class which is present under javax.swing package.

- JSlider class is a component that allows the user to select a value by moving the knob.

- The knob is always positioned at the points that match the integer values within specified interval.

***Constructor:

1) JSlider() - it will create HORIZONTAL JSlider with 0 to 100 values.

2) JSlider(BoundedRangeMode model) - It will create HORIZONTAL slider with specified model.

3) JSlider(int orient)- It will create HORIZONTAL slider with range value 0 to 100. Initila value 50.

4) JSlider(int min,int max) - HORIZONTAL slider with specified min and max value

5) JSlider(int min,int max,int value); -

6) JSlider(int orient,int min,int max,int value);

Where:

orient = HORIZONTAL,VERTICAL

***Methods:

1) public void setMinorTickSpacingl(int n)

2) public void setMajorTickSpacing(int n)

3) public void setPaintTicks(boolean flag)

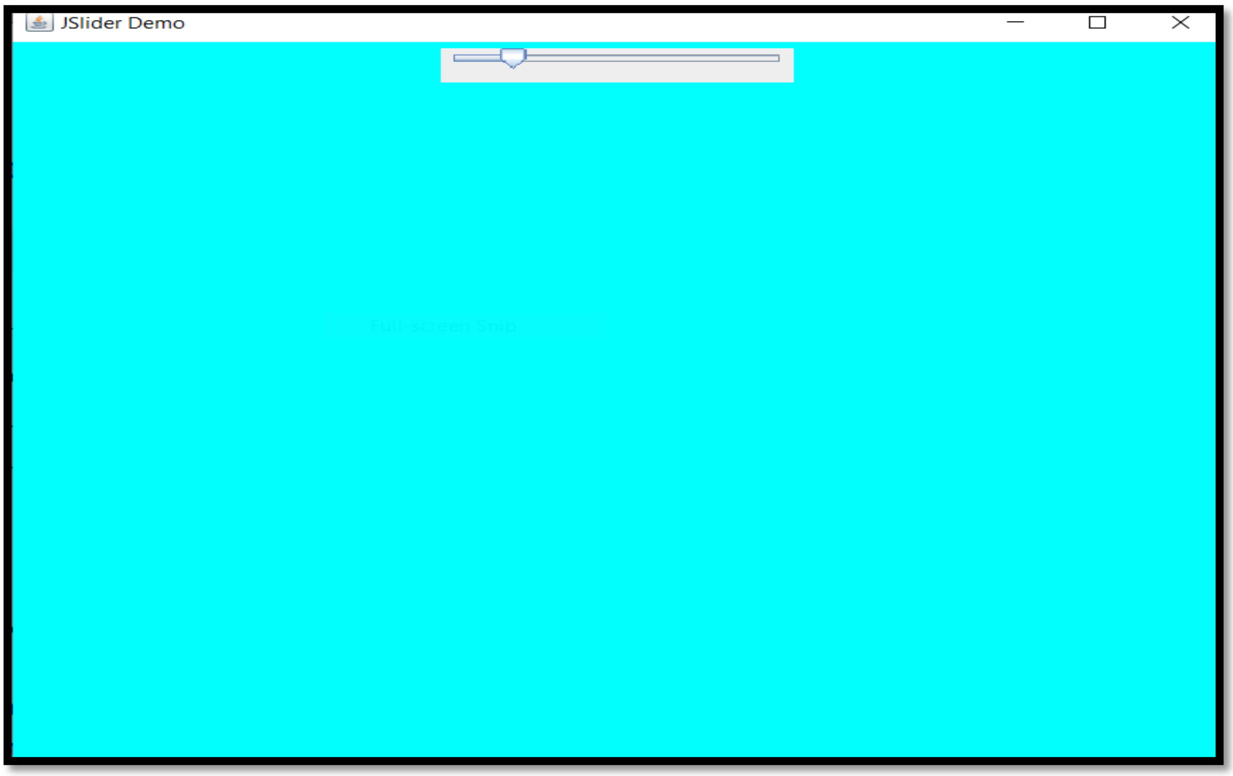
4) public void setPaintLabels(boolean flag)

5) public void setPaintTracks(boolean flag);

Code:-

```
import java.awt.*;

import javax.swing.*;

class JSliderDemo extends JFrame

{

 JSliderDemo()

 {

      Container c=getContentPane();

      c.setLayout(new FlowLayout());

      c.setBackground(Color.cyan);

      JSlider js=new JSlider(0,200,50);

      c.add(js);


 }

 public static void main(String args[])

 {

   JSliderDemo j1=new JSliderDemo();

       j1.setVisible(true);

       j1.setTitle("JSlider Demo");

       j1.setSize(700,700);

       j1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 }

}
```

Output:-



###### MVC Architecture

- This Architecture contains three stages/Phases (Model-View-Controller)

1) Information/Properties associated with that components.

2) The way that the component looks when submitted on the screen.

3) The way that the component react to the user.

In MVC Architecture:

1) Model:

- It defined the state information about the component i.e the event and its graphical change when even occurs on a component.

2) View:

- It defined how that component displayed on the screen.

3) Controller:

- It controls how the component react to the user.

#### JDialog Class

- JDialog class is a predefined class which is present under javax.swing package.

- JDialog class is used for creating dialog window.

- It is always appear as child window.

- It is used to create any type of custom dialog boxes with the use of JOptionPane.

***Constructor:

1) JDialog() - it will create model less dialog without title and without parent window object.

2) JDialog(Frame obj) - It will create model less dialog without title and with parent window Frame object.

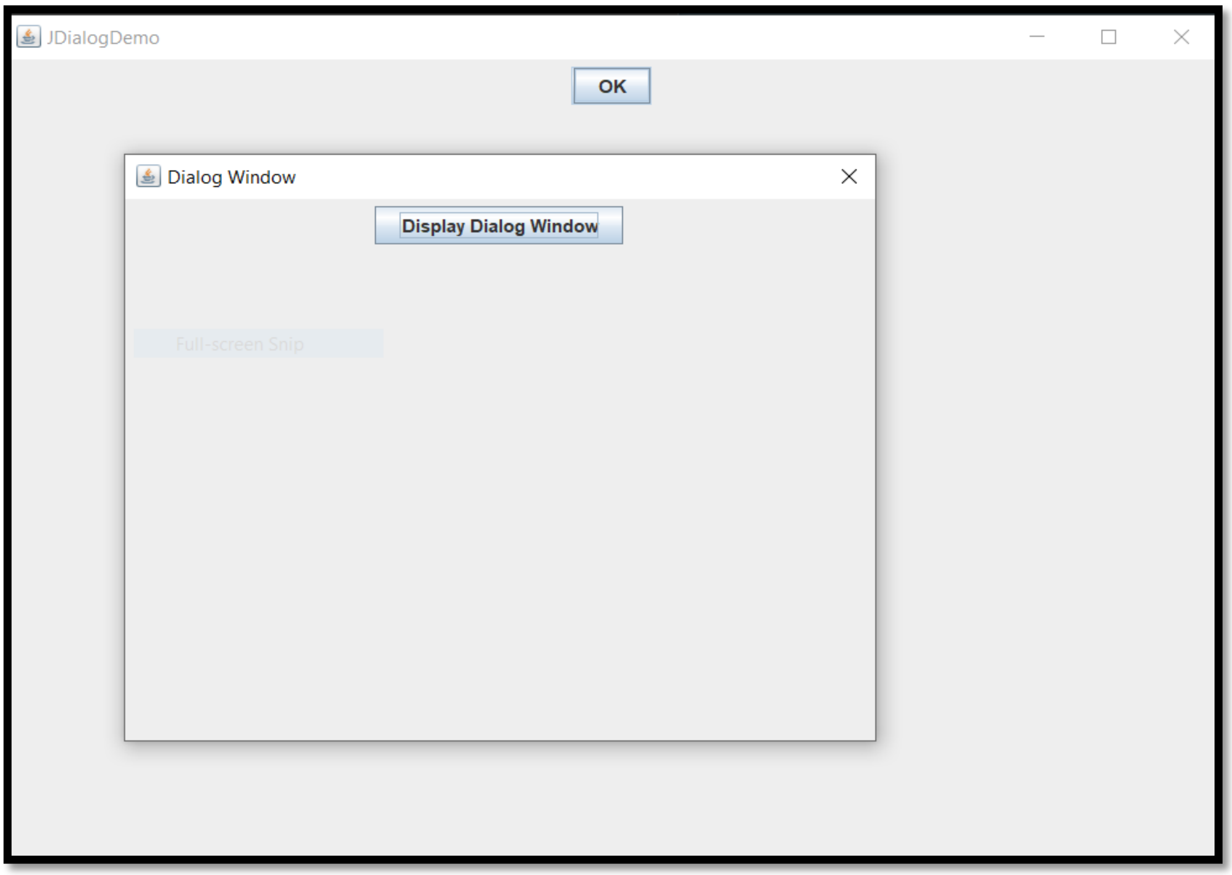
3) JDialog(Frame obj, String title, boolean flag) flag=> TRUE then model and flag=>FALSE then model less.

Code:-

```
import javax.swing.*;

import java.awt.event.*;

import java.awt.*;

class JDialogDemo extends JFrame implements ActionListener

{

 JDialog j1;

 JDialogDemo()

 {

        Container c=getContentPane();
```

```
            c.setLayout(new FlowLayout());
            JButton b2=new JButton("OK");
            c.add(b2);
            b2.addActionListener(this);
             j1=new JDialog(this,"Dialog Window",true);
           j1.setLayout(new FlowLayout());
           JButton b1=new JButton("Display Dialog Window");
           j1.add(b1);
            j1.setSize(500,500);
 }
 public void actionPerformed(ActionEvent ae)
 {
            j1.setVisible(true);
 }
 public static void main(String args[])
 {
           JDialogDemo jdd=new JDialogDemo();
           jdd.setVisible(true);
           jdd.setTitle("JDialogDemo");
           jdd.setSize(800,800);
           jdd.setDefaultCloseOperation(EXIT_ON_CLOSE);
 }
}
```

Output:-


JDialogDemo
OK
Dialog Window
Display Dialog Window
Full-screen Snip

Inspiring Your Success

VJTech Academy…