UNIT – V

Applets – Concepts of Applets, differences between applets and applications, life cycle of an applet, types of applets, creating applets, passing parameters to applets. Swing – Introduction, limitations of AWT, MVC architecture, components, containers, exploring swing- JApplet, JFrame and JComponent, Icons and Labels, text fields, buttons – The JButton class, Check boxes, Radio buttons, Combo boxes, Tabbed Panes, Scroll Panes, Trees, and Tables.

JAVA APPLET

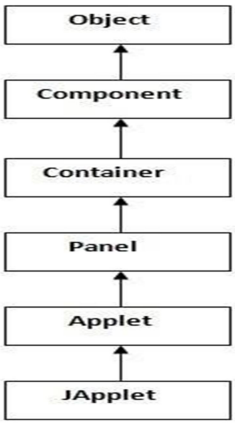
- Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.
- Applets are used to make the website more dynamic and entertaining.
- An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server.

Important points

- All applets are sub-classes of java.applet.Applet class.
- Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
- In general, execution of an applet does not begin at main() method.
- Output of an applet window is not performed by System.out.println(). Rather it is handled with various AWT methods, such as drawString().

HIERARCHY OF APPLET

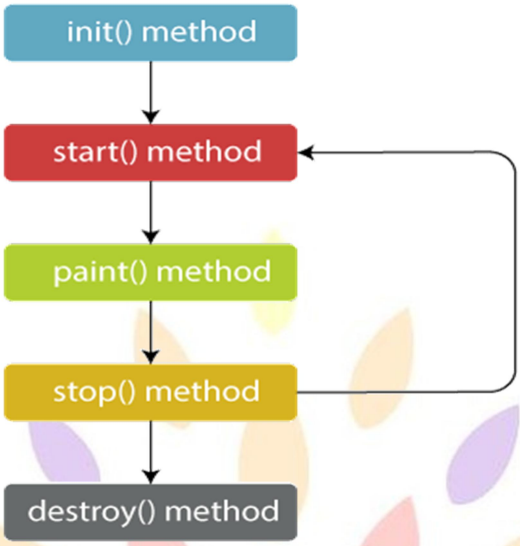
As displayed in the above diagram, Applet class extends Panel. Panel class extends Container which is the subclass of Component.



DIFFERENCES BETWEEN APPLETS AND APPLICATIONS

Java Application	Applet
Java application contains a main method	An applet does not contain a main method
Does not require internet connection to execute	Requires internet connection to execute
Is stand alone application	Is a part of web page
Can be run without a browser	Requires a Java compatible browser
Uses stream I/O classes	Use GUI interface provided by AWT or Swings
Entry point is main method	Entry point is init method
Generally used for console programs	Generally used for GUI interfaces

LIFE CYCLE OF AN APPLET



1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.

Lifecycle methods for Applet

The java.applet.Applet class provides 4 life cycle methods and java.awt.Component class provides 1 life cycle method for an applet.

java.applet.Applet class

For creating any applet, java.applet.Applet class must be inherited.

It provides 4 life cycle methods of

1. public void init(): is used to initialized the Applet. It is invoked only once.

2. public void start(): is invoked after the init() method or browser is maximized. It is used to start the Applet.

3. public void stop(): is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.

4. public void destroy(): is used to destroy the Applet. It is invoked only once.

java.awt.Component class

The Component class provides 1 life cycle method of applet.

1. public void paint(Graphics g): is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

CREATING APPLETS

How To Run an Applet?

There are two ways to run an applet

1. By html file.
2. By appletViewer tool (for testing purpose).

1. By html file

To execute the applet by html ile, create an applet and compile it. After that create an html ile and place the applet code in html ile. Now click the html ile.

Example

```
import java.applet.Applet;

import java.awt.Graphics;

public class  First extends Applet

{

public void paint(Graphics g)

{

g.drawString( "A simple Applet",20 20);

}

}
```


myapplet.html

```
<html>

<body>
```

<applet code="First.class" width="300" height="300">

</applet>

</body>

</html>

2. By appletViewer tool

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by: appletviewer First.java. Now Html ile is not required but it is for testing purpose only.

Example

```
import java.applet.Applet;

import java.awt.Graphics;

public class  First extends Applet

{

public void paint(Graphics g)

{

g.drawString("welcome applet",150,150);

 }

}

/*

<applet code="First.class" width="300" height="300">

</applet>

*/

);
```


To execute the applet by appletviewer tool, write in command prompt:

```
c:\>javac First.java
```

c:\>appletviewer First.java

TYPES OF APPLETS IN JAVA

- A special type of Java program that runs in a Web browser is referred to as Applet .
- It has less response time because it works on the client-side.
- It is much secured executed by the browser under any of the platforms such as Windows, Linux and Mac OS etc.
- There are two types of applets that a web page can contain.

1. Local Applet
2. Remote Applet

1. Local Applet

- Local Applet is written on our own, and then we will embed it into web pages.
- Local Applet is developed locally and stored in the local system.
- A web page doesn't need the get the information from the internet when it inds the local Applet in the system.
- It is specified or deined by the ile name or pathname.
- There are two attributes used in deining an applet, i.e., the codebase that speci ies the path name and code that deined the name of the ile that contains Applet's code.

Specifying Local applet

<applet

codebase ="tictactoe"

code = "FaceApplet.class"

width = 120

height =120 >

</applet>

Example

mport java.applet.Applet;

import java.awt.*;

public class GraphicsDemo extends Applet{

```
public void paint(Graphics g){
g.setColor(Color.red);
g.drawString("Welcome",50, 50);
g.drawLine(20,30,20,300);
g.drawRect(70,100,30,30);
g.fillRect(170,100,30,30);
g.drawOval(70,200,30,30);
g.setColor(Color.pink);
g.fillOval(170,200,30,30);
g.drawArc(90,150,30,30,30,270);
g.fillArc(270,150,30,30,0,180);
  }
}
```

myapplet.html

```
<html>
<body>
<applet code="GraphicsDemo.class" width="300" height="300">
</applet>
</body>
</html>
```


2. Remote Applet

- A remote applet is designed and developed by another developer.
- It is located or available on a remote computer that is connected to the internet.
- In order to run the applet stored in the remote computer, our system is connected to the internet then we can download run it.

- In order to locate and load a remote applet, we must know the applet's address on the web that is referred to as Uniform Recourse Locator(URL).

Specifying Remote applet

```
<applet
codebase  = "http://www.myconnect.com/applets/"
 code = "FaceApplet.class"
width = 120
 height =120>
 </applet>
```

PARAMETER IN APPLET

We can get any information from the HTML ile as a parameter. For this purpose, Applet class provides a method named getParameter().

Syntax:

```
public String getParameter(String parameterName)
```

Example of using parameter in Applet

```
import java.applet.Applet;
import java.awt.Graphics;
public class UseParam extends Applet{
  public void paint(Graphics g){
String str=getParameter("msg");
g.drawString(str,50, 50);
}
  }
```

myapplet.html

```
<html>
```

```
<body>

<applet code="UseParam.class" width="300" height="300">

<param name="msg" value="Welcome to applet">

</applet>

</body>

</html>
```

SWING INTRODUCTION

- Java Swing is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.
- Unlike AWT, Java Swing provides platform-independent and lightweight components.
- The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

LIMITATIONS OF AWT

- The buttons of AWT does not support pictures.
- It is heavyweight in nature.
- Two very important components trees and tables are not present.
- Extensibility is not possible as it is platform dependent

MVC ARCHITECTURE

The MVC design pattern consists of three modules model, view and controller.

Model

- The model represents the state (data) and business logic of the application.
- For example-in case of a check box, the model contains a ield which indicates whether the box is checked or unchecked.

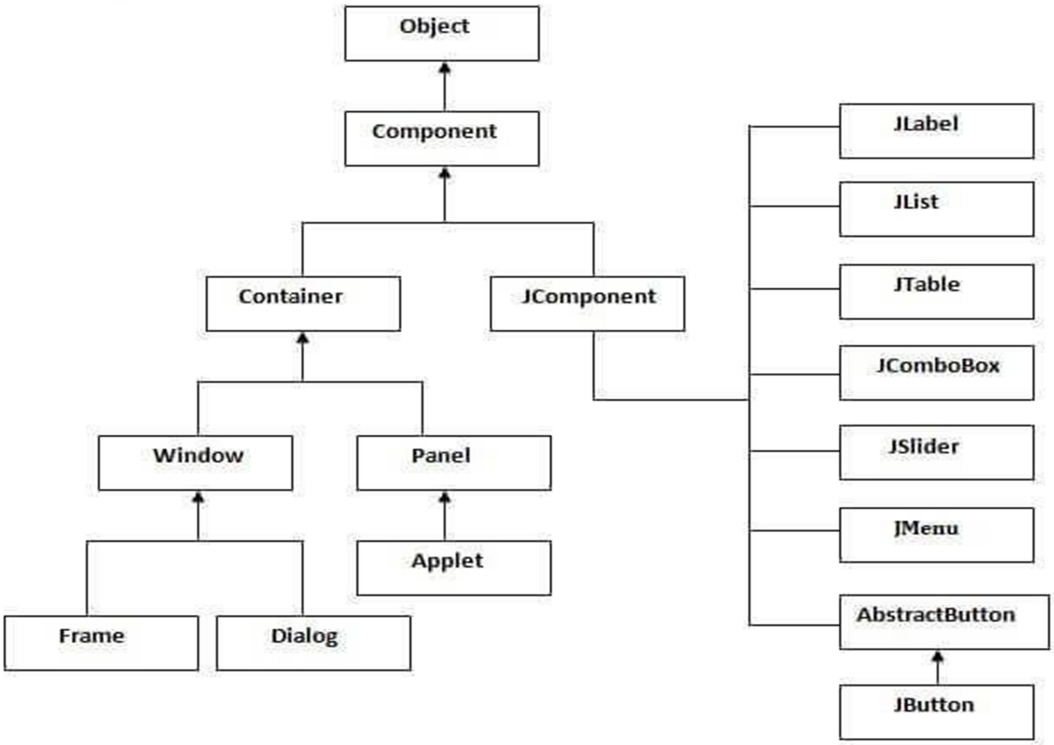
View

- The view module is responsible to display data i.e. it represents the presentation.
- The view determines how a component has displayed on the screen, including any aspects of view that are affected by the current state of the model.

Controller

- The controller determines how the component will react to the user.
- The controller module acts as an interface between view and model.
- It intercepts all the requests i.e. receives input and commands to Model / View to change accordingly.

HIERARCHY OF JAVA SWING CLASSES



COMPONENT CLASS

- A component is an object having a graphical representation that can be displayed on the screen and that can interact with the user.
- Examples of components are the buttons, checkboxes, and scrollbars of a typical graphical user interface.

The methods of Component class are widely used in java swing that are given below.

Method

public void add(Component c)

public void setSize(int width,int height)

public void setLayout(LayoutManager m)

public void setVisible(boolean b)

CONTAINER

- The Container is a component that can contain another components like buttons, textfields, labels etc.
- The classes that extends Container class are known as container such as Frame, Dialog and Panel.

WINDOW

- The window is the container that have no borders and menu bars.
- You must use frame, dialog or another window for creating a window.

JPANEL

- The Panel is the container that doesn't contain title bar and menu bars.
- It can have other components like button, textfield etc.

JDIALOG

- The JDialog control represents a top level window with a border and a title used to take some form of input from the user.
- It inherits the Dialog class.
- Unlike JFrame, it doesn't have maximize and minimize buttons.

JFrame

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

There are two ways to create a frame:

1. By creating the object of Frame class (association)

2. By extending Frame class (inheritance)

1. By creating the object of Frame class (association)

import javax.swing.*;

```
public class FirstSwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();//creating instance of JFrame
JButton b=new JButton("click");//creating instance of JButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height

f.add(b);//adding button in JFrame
f.setSize(400,500);//400 width and 500 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
}
);
```

Output:

2. By extending Frame class (inheritance)

We can also inherit the JFrame class, so there is no need to create the instance of JFrame class explicitly.

EXAMPLE

```
import javax.swing.*;

public class Simple2 extends JFrame{//inheriting JFrame

JFrame f;

Simple2(){

JButton b=new JButton("click");//create button

b.setBounds(130,100,100, 40);


add(b);//adding button on frame

setSize(400,500);

setLayout(null);

setVisible(true);

}

public static void main(String[] args) {

new Simple2();

}}
```

SWING COMPONENTS

JButton

- The JButton class is used to create a labeled button that has platform independent implementation.
- The application result in some action when the button is pushed.
- It inherits AbstractButton class.

Example

```
import javax.swing.*;
public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

Output:

JLabel

- The object of JLabel class is a component for placing text in a container.

- It is used to display a single line of read only text.
- The text can be changed by an application but a user cannot edit it directly.
- It inherits JComponent class.

Example

```
mport javax.swing.*;

class LabelExample

{

public static void main(String args[])

    {

    JFrame f= new JFrame("Label Example");

    JLabel l1,l2;

    l1=new JLabel("First Label.");

    l1.setBounds(50,50, 100,30);

    l2=new JLabel("Second Label.");

    l2.setBounds(50,100, 100,30);

    f.add(l1); f.add(l2);

    f.setSize(300,300);

    f.setLayout(null);

    f.setVisible(true);

    }

    }
```

Output :

JTextField

- The object of a JTextField class is a text component that allows the editing of a single line text.
- It inherits JTextComponent class.

Example

```
import javax.swing.*;

class TextFieldExample

{

public static void main(String args[])

{

JFrame f= new JFrame("TextField Example");

JTextField t1,t2;

t1=new JTextField("Welcome  to Javatutorial ");

t1.setBounds(50,100, 200,30);

t2=new JTextField("Swing Tutorial");

t2.setBounds(50,150, 200,30);

f.add(t1);
```

```
f.add(t2);

f.setSize(400,400);

f.setLayout(null);

f.setVisible(true);

}

}
```

JTextArea

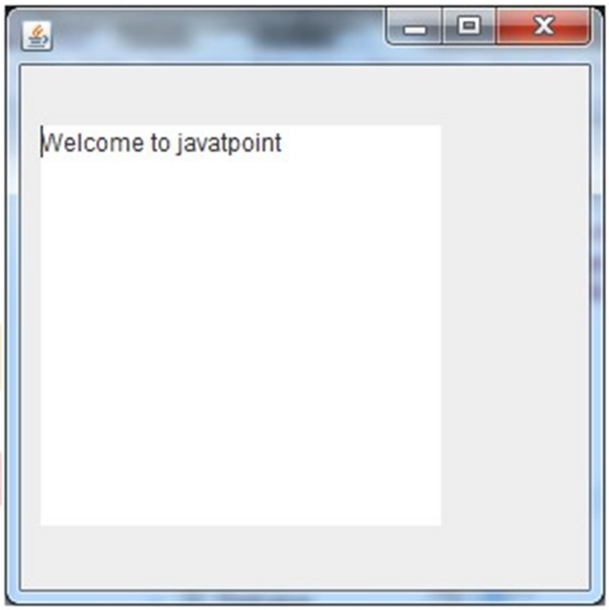
The object of a JTextArea class is a multiline region that displays text. It allows the editing of multiple line text. It inherits JTextComponent class.

Example

```
import javax.swing.*;

public class TextAreaExample

{

    TextAreaExample(){

       JFrame f= new JFrame();

       JTextArea area=new JTextArea("Welcome to javatpoint");

       area.setBounds(10,30, 200,200);

       f.add(area);

       f.setSize(300,300);

       f.setLayout(null);

       f.setVisible(true);

    }

public static void main(String args[])

   {

  new TextAreaExample();
```

}}

Output:



JCheckBox

- The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false).
- Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

Example

```
import javax.swing.*;

public class CheckBoxExample

{

    CheckBoxExample(){

        JFrame f= new JFrame("CheckBox Example");

        JCheckBox checkBox1 = new JCheckBox("C++");

        checkBox1.setBounds(100,100, 50,50);
```

```
        JCheckBox checkBox2 = new JCheckBox("Java", true);
        checkBox2.setBounds(100,150, 50,50);
        f.add(checkBox1);
        f.add(checkBox2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
public static void main(String args[])
    {
    new CheckBoxExample();
    }}
```

JRadioButton

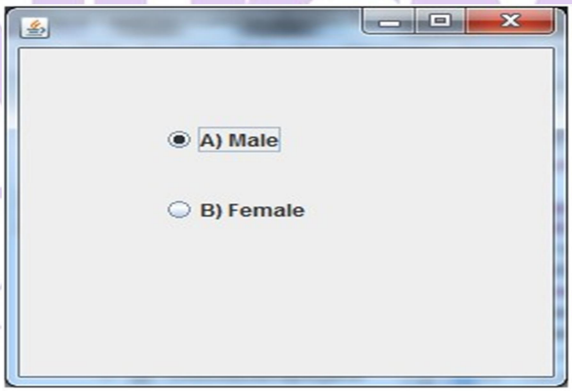
- The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.
- It should be added in ButtonGroup to select one radio button only.

Example

```
import javax.swing.*;
public class RadioButtonExample
{
JFrame f;
RadioButtonExample()
{
f=new JFrame();
JRadioButton r1=new JRadioButton("A) Male");
```

```
JRadioButton r2=new JRadioButton("B) Female");
r1.setBounds(75,50,100,30);
ButtonGroup bg=new ButtonGroup();
bg.add(r1);
bg.add(r2);
f.add(r1);
f.add(r2);
f.setSize(300,300);
f.setVisible(true);
}
public static void main(String[] args)
{
newRadioButtonExample();
}
}
```

Output:



JComboBox

- The object of Choice class is used to show popup menu of choices.
- Choice selected by user is shown on the top of a menu .

- It inherits JComponent class.

Example

```
import javax.swing.*;
public class ComboBoxExample
{
JFrame f;
ComboBoxExample()
{
f=new JFrame("ComboBox Example");
 String country[]={"India","Aus","U.S.A","England","Newzealand"};
 JComboBox cb=new JComboBox(country);
 f.add(cb);
 f.setSize(400,500);
 f.setVisible(true);
}
public static void main(String[] args)
{
newComboBoxExample();
}
}
```

Output :

JTable

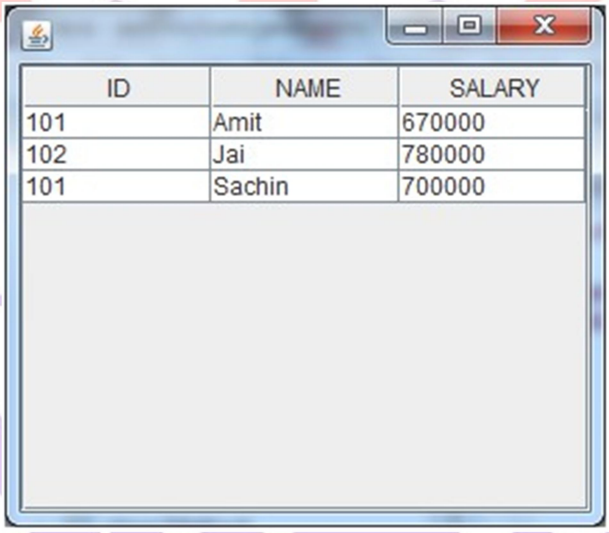
- The JTable class is used to display data in tabular form.
- It is composed of rows and columns.

Example

```
import javax.swing.*;

public class TableExample

{

JFrame f;

TableExample()

    {

f=new JFrame();

String data[][]={ {"101","Amit","670000"}, {"102","Jai","780000"},
{"101","Sachin","700000"}};

String column[]={"ID","NAME","SALARY"};

JTable jt=new JTable(data,column);

JScrollPane sp=new JScrollPane(jt);
```

```
f.add(sp);

f.setSize(300,400);

f.setVisible(true);

}

public static void main(String[] args)

{

newTableExample();

}

}
```

Output



JTree

- The JTree class is used to display the tree structured data or hierarchical data.
- JTree is a complex component. It has a 'root node' at the top most which is a parent for all nodes in the tree. It inherits JComponent class.

Example

```
import javax.swing.*;
```

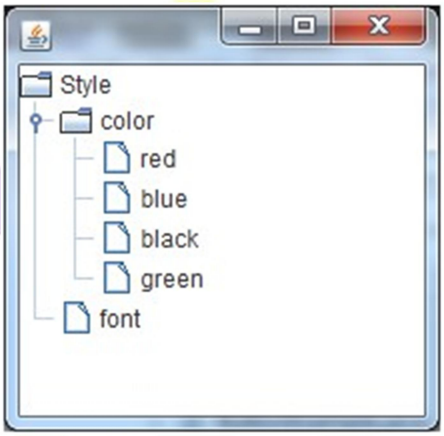
```
import javax.swing.tree.DefaultMutableTreeNode;
public class TreeExample
{
JFrame f;
TreeExample()
{
f=new JFrame();
DefaultMutableTreeNode style=new DefaultMutableTreeNode("Style");
DefaultMutableTreeNode color=new DefaultMutableTreeNode("color");
DefaultMutableTreeNode font=new DefaultMutableTreeNode("font");
style.add(color);
style.add(font);
DefaultMutableTreeNode red=new DefaultMutableTreeNode("red");
DefaultMutableTreeNode blue=new DefaultMutableTreeNode("blue");
DefaultMutableTreeNode black=new DefaultMutableTreeNode("black");
DefaultMutableTreeNode green=new DefaultMutableTreeNode("green");
color.add(red); color.add(blue); color.add(black); color.add(green);
JTree jt=new JTree(style);
f.add(jt);
f.setSize(200,200);
f.setVisible(true);
}
public static void main(String[] args)
{
```

newTreeExample();

}

}

Output:



JTabbedPane

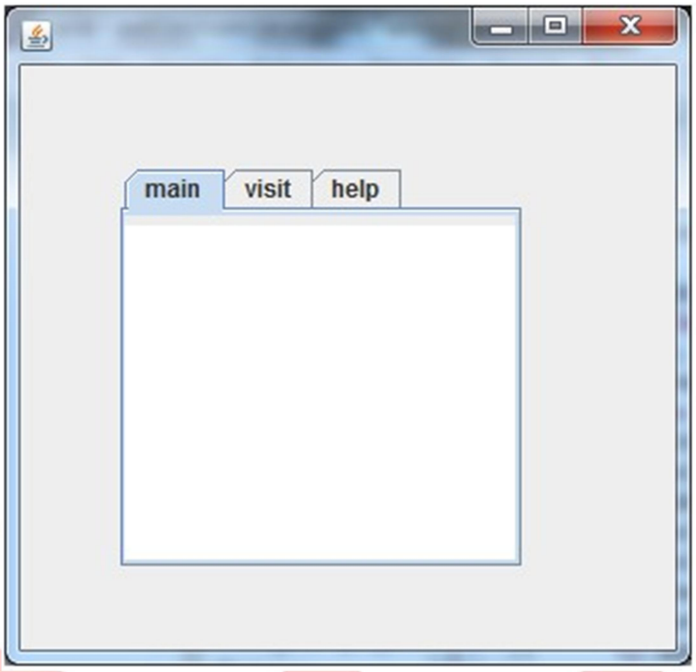
The JTabbedPane class is used to switch between a group of components by clicking on a tab with a given title or icon. It inherits JComponent class.

Example

```
import javax.swing.*;

public class TabbedPaneExample

{

JFrame f;

TabbedPaneExample()

{

f=new JFrame();

JTextArea ta=new JTextArea(200,200);

JPanel p1=new JPanel();

p1.add(ta);
```

```
 JPanel p2=new JPanel();
JPanel p3=new JPanel();
JTabbedPane tp=new JTabbedPane();
tp.setBounds(50,50,200,200);
tp.add("main",p1);
tp.add("visit",p2);
 tp.add("help",p3);
f.add(tp);
f.setSize(400,400);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String[] args)
{
newTabbedPaneExample();
}
}
```

Output:



JScrollPane

A JscrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically.

Example

```
import java.awt.FlowLayout;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JtextArea;

public class JScrollPaneExample {

    private static final long serialVersionUID = 1L;

    private static void createAndShowGUI() {

        // Create and set up the window.

        final JFrame frame = new JFrame("Scroll Pane Example");
```

```
        // Display the window.
        frame.setSize(500, 500);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // set flow layout for the frame
        frame.getContentPane().setLayout(new FlowLayout());
        JTextArea textArea = new JTextArea(20, 20);
        JScrollPane scrollableTextArea = new JScrollPane(textArea);

scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_
ALWAYS);

scrollableTextArea.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWA
YS);
        frame.getContentPane().add(scrollableTextArea);
    }
    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Output: