

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.

Advantage of Applet

There are many advantages of applet. They are as follows:

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

- Plugin is required at client browser to execute applet.

Difference between a Java Application and a Java Applet

Java Application is just like a Java program that runs on an underlying operating system with the support of a virtual machine. It is also known as an application program. The graphical user interface is not necessary to execute the java applications, it can be run with or without it.

Java Applet is a Java program that can be embedded into a web page. It runs inside the web browser and works on the client-side. An applet is embedded in an HTML page using the **APPLET** or **OBJECT** tag and hosted on a web server. Applets are used to make the website more dynamic and entertaining.

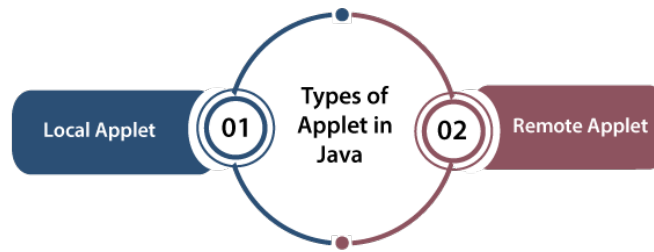
Parameters	Java Application	Java Applet
Definition	Applications are just like a Java program that can be executed independently without using the web browser.	Applets are small Java programs that are designed to be included with the HTML web document. They require a Java-enabled web browser for execution.
main () method	The application program requires a main() method for its execution.	The applet does not require the main() method for its execution instead init() method is required.
Compilation	The “javac” command is used to compile application programs, which are then executed using the “java” command.	Applet programs are compiled with the “javac” command and run using either the “appletviewer” command or the web browser.
File access	Java application programs have full access to the local file system and network.	Applets don’t have local disk and network access.

Parameters	Java Application	Java Applet
Access level	Applications can access all kinds of resources available on the system.	Applets can only access browser-specific services. They don't have access to the local system.
Installation	First and foremost, the installation of a Java application on the local computer is required.	The Java applet does not need to be installed beforehand.
Execution	Applications can execute the programs from the local system.	Applets cannot execute programs from the local machine.
Program	An application program is needed to perform some tasks directly for the user.	An applet program is needed to perform small tasks or part of them.
Run	It cannot run on its own; it needs JRE to execute.	It cannot start on its own, but it can be executed using a Java-enabled web browser.
Connection with servers	Connectivity with other servers is possible.	It is unable to connect to other servers.
Read and Write Operation	It supports the reading and writing of files on the local computer.	It does not support the reading and writing of files on the local computer.
Security	Application can access the system's data and resources without any security limitations.	Executed in a more restricted environment with tighter security. They can only use services that are exclusive to their browser.
Restrictions	Java applications are self-contained and require no additional security because they are trusted.	Applet programs cannot run on their own, necessitating the maximum level of security.

Types of Applets

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**

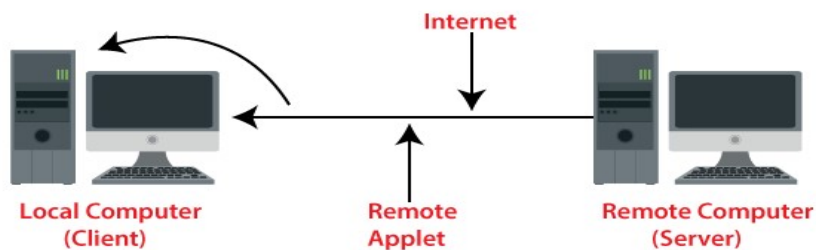


Local Applet

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 finds the local Applet in the system. It is specified or defined by the file name or pathname. There are two attributes used in defining an applet, i.e., the **codebase** that specifies the path name and **code** that defined the name of the file that contains Applet's code.

Remote Applet

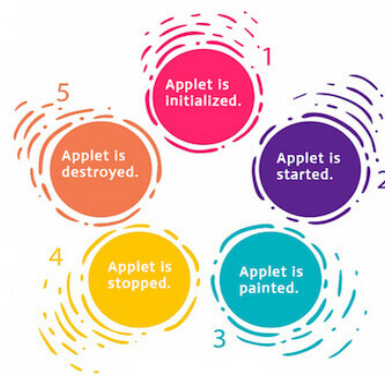
A remote applet is designed and developed by another developer. It is located or available on a remote computer that is connected to the internet. In order to run the applet stored in the remote computer, our system is connected to the internet then we can download run it. In order to locate and load a remote applet, we must know the applet's address on the web that is referred to as Uniform Recourse Locator(URL).



Difference Between Local Applet and Remote Applet

Local Applet	Remote Applet
There is no need to define the Applet's URL in Local Applet.	We need to define the Applet's URL in Remote Applet.
Local Applet is available on our computer.	Remote Applet is not available on our computer.
In order to use it or access it, we don't need Internet Connection.	In order to use it or access it on our computer, we need an Internet Connection.
It is written on our own and then embedded into the web pages.	It was written by another developer.
We don't need to download it.	It is available on a remote computer, so we need to download it to our system.

Applet Lifecycle



Lifecycle of Java 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 methods 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 applet.

1. **`public void init():`** is used to initialize 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.

Run an Applet

There are two ways to run an applet

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

Parameter in Applet

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

Syntax

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

Example

```
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>
```

Java Swing

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

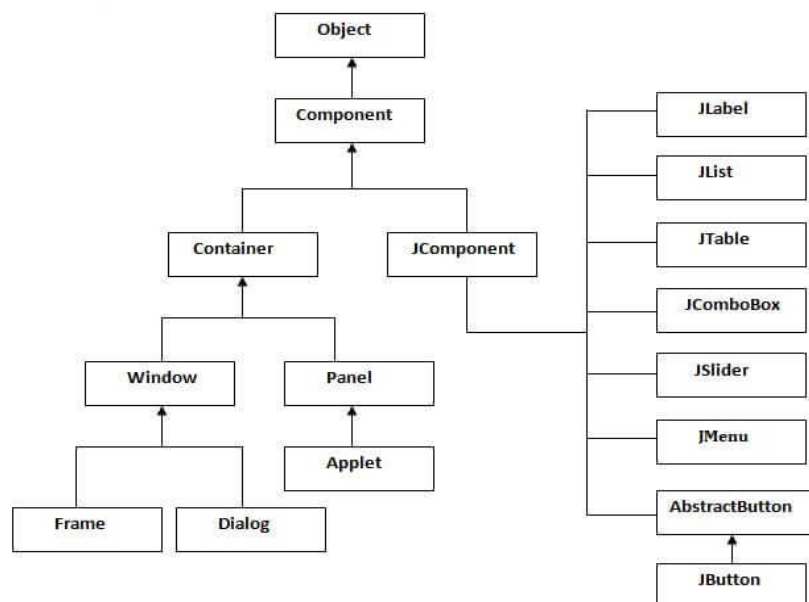
Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference between AWT and Swing

No.	Java AWT	Java Swing
1)	AWT components are platform-dependent .	Java swing components are platform-independent .
2)	AWT components are heavyweight .	Swing components are lightweight .
3)	AWT doesn't support pluggable look and feel .	Swing supports pluggable look and feel .
4)	AWT provides less components than Swing.	Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT doesn't follows MVC (Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing follows MVC .

Hierarchy of Java Swing classes



Commonly used Methods of Component class

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Java Swing Examples

There are two ways to create a frame:

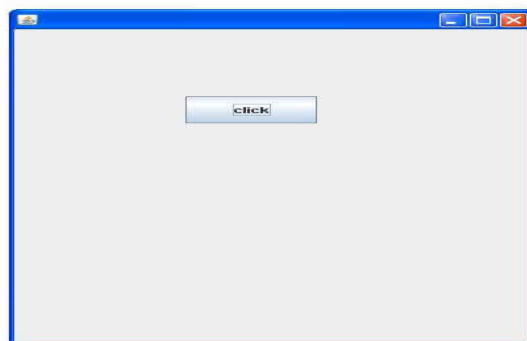
- By creating the object of Frame class (association)
- By extending Frame class (inheritance)

We can write the code of swing inside the main(), constructor or any other method.

Simple Java Swing Example

//Creating one button and adding it on the JFrame object inside the main() method.

```
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
    }
}
```



Java 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.

JLabel class declaration

public class JLabel extends JComponent implements SwingConstants, Accessible

Commonly used Constructors

Constructor	Description
JLabel()	Creates a JLabel instance with no image and with an empty string for the title.
JLabel(String s)	Creates a JLabel instance with the specified text.
JLabel(Icon i)	Creates a JLabel instance with the specified image.
JLabel(String s, Icon i, int horizontalAlignment)	Creates a JLabel instance with the specified text, image, and horizontal alignment.

Commonly used Methods

Methods	Description
String getText()	t returns the text string that a label displays.
void setText(String text)	It defines the single line of text this component will display.
void setHorizontalAlignment(int alignment)	It sets the alignment of the label's contents along the X axis.
Icon getIcon()	It returns the graphic image that the label displays.
int getHorizontalAlignment()	It returns the alignment of the label's contents along the X axis.

Java JLabel Example

```
import 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



Java JLabel Example with ActionListener

```
import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.*.*;
public class LabelExample extends Frame implements ActionListener
{
    JTextField tf; JLabel l; JButton b;
    LabelExample()
    {
        tf=new JTextField();
        tf.setBounds(50,50, 150,20);
        l=new JLabel();
        l.setBounds(50,100, 250,20);
        b=new JButton("Find IP");
        b.setBounds(50,150,95,30);
        b.addActionListener(this);
        add(b);add(tf);add(l);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
        try
        {
            String host=tf.getText();
            String ip=java.net.InetAddress.getByName(host).getHostAddress();
            l.setText("IP of "+host+" is: "+ip);
        }
        catch(Exception ex)      {      System.out.println(ex);      }
    }
    public static void main(String[] args)
    {      new LabelExample();      }
}
```

Output



Java JTextField

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

JTextField class declaration

```
public class JTextField extends JTextComponent implements SwingConstants
```

Commonly used Constructors

Constructor	Description
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

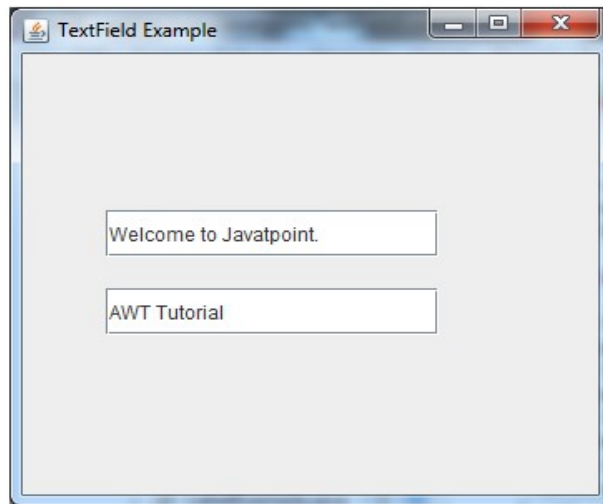
Commonly used Methods

Methods	Description
void addActionListener(ActionListener l)	It is used to add the specified action listener to receive action events from this textfield.
Action getAction()	It returns the currently set Action for this ActionEvent source, or null if no Action is set.
void setFont(Font f)	It is used to set the current font.
void removeActionListener(ActionListener l)	It is used to remove the specified action listener so that it no longer receives action events from this textfield.

Java JTextField 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 Javatpoint.");
        t1.setBounds(50,100, 200,30);
        t2=new JTextField("AWT Tutorial");
        t2.setBounds(50,150, 200,30);
        f.add(t1); f.add(t2);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

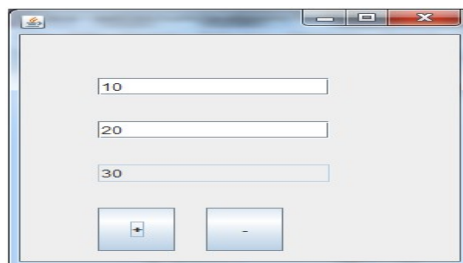
Output



Java JTextField Example with ActionListener

```
import javax.swing.*.*;
import java.awt.event.*;
public class TextFieldExample implements ActionListener
{
    JTextField tf1,tf2,tf3;
    JButton b1,b2;
    TextFieldExample()
    {
        JFrame f= new JFrame();
        tf1=new JTextField();      tf1.setBounds(50,50,150,20);
        tf2=new JTextField();      tf2.setBounds(50,100,150,20);
        tf3=new JTextField();      tf3.setBounds(50,150,150,20);
        tf3.setEditable(false);
        b1=new JButton("+");      b1.setBounds(50,200,50,50);
        b2=new JButton("-");      b2.setBounds(120,200,50,50);
        b1.addActionListener(this);
        b2.addActionListener(this);
        f.add(tf1);f.add(tf2);f.add(tf3);f.add(b1);f.add(b2);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
        String s1=tf1.getText();
        String s2=tf2.getText();
        int a=Integer.parseInt(s1);
        int b=Integer.parseInt(s2);
        int c=0;
        if(e.getSource()==b1) {      c=a+b;      }
        else if(e.getSource()==b2) {      c=a-b;      }
        String result=String.valueOf(c);
        tf3.setText(result);
    }
    public static void main(String[] args) {      new TextFieldExample();  }
}
```

Output



Java JTextArea

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

JTextArea class declaration

```
public class JTextArea extends JTextComponent
```

Commonly used Constructors

Constructor	Description
JTextArea()	Creates a text area that displays no text initially.
JTextArea(String s)	Creates a text area that displays specified text initially.
JTextArea(int row, int column)	Creates a text area with the specified number of rows and columns that displays no text initially.
JTextArea(String s, int row, int column)	Creates a text area with the specified number of rows and columns that displays specified text.

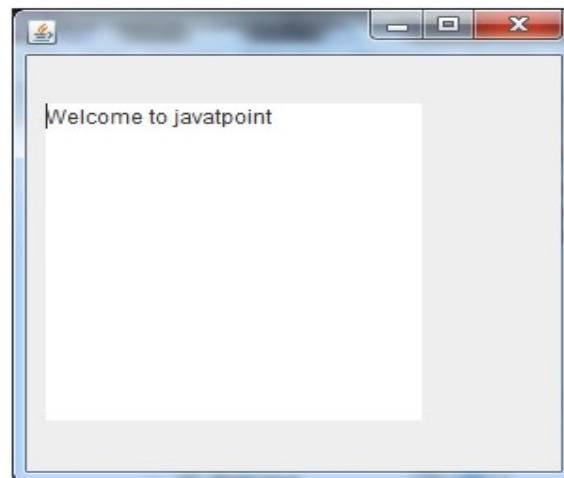
Commonly used Methods

Methods	Description
void setRows(int rows)	It is used to set specified number of rows.
void setColumns(int cols)	It is used to set specified number of columns.
void setFont(Font f)	It is used to set the specified font.
void insert(String s, int position)	It is used to insert the specified text on the specified position.
void append(String s)	It is used to append the given text to the end of the document.

Java JTextArea 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();
    }
}
```

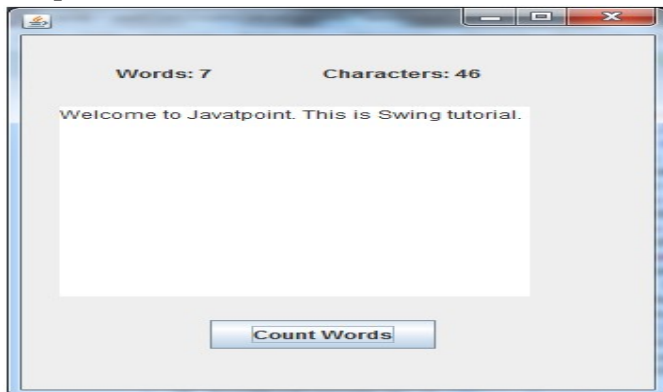
Output



Java JTextArea Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class TextAreaExample implements ActionListener
{
    JLabel l1,l2;
    JTextArea area;
    JButton b;
    TextAreaExample()
    {
        JFrame f= new JFrame();
        l1=new JLabel();           l1.setBounds(50,25,100,30);
        l2=new JLabel();           l2.setBounds(160,25,100,30);
        area=new JTextArea();      area.setBounds(20,75,250,200);
        b=new JButton("Count Words");
        b.setBounds(100,300,120,30);
        b.addActionListener(this);
        f.add(l1);f.add(l2);f.add(area);f.add(b);
        f.setSize(450,450);
        f.setLayout(null);
        f.setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
        String text=area.getText();
        String words[]=text.split("\\s");
        l1.setText("Words: "+words.length);
        l2.setText("Characters: "+text.length());
    }
    public static void main(String[] args) {    new TextAreaExample(); }
}
```

Output



Java 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.

JButton class declaration

```
public class JButton extends AbstractButton implements Accessible
```

Commonly used Constructors

Constructor	Description
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.
JButton(Icon i)	It creates a button with the specified icon object.

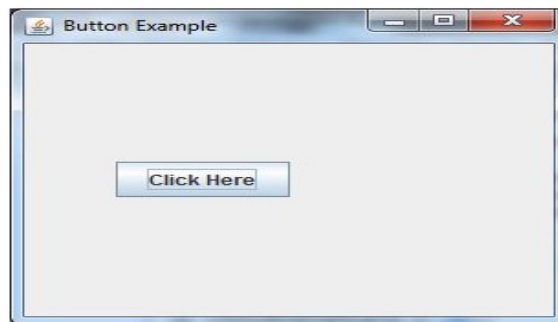
Commonly used Methods of AbstractButton class:

Methods	Description
void setText(String s)	It is used to set specified text on button
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the <u>action listener</u> to this object.

Java JButton 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);
    }
}
```

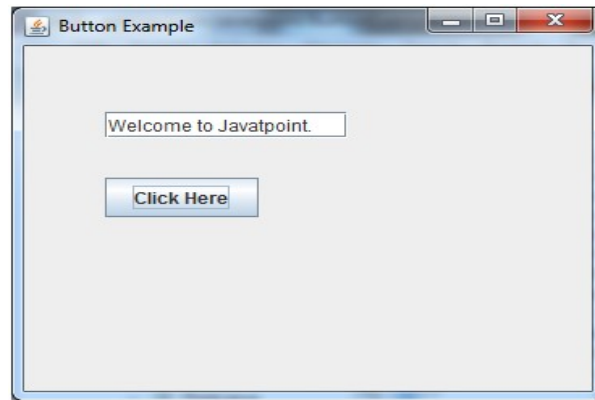
Output



Java JButton Example with ActionListener

```
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("Welcome to Javatpoint."); } });
        f.add(b);f.add(tf);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

Output

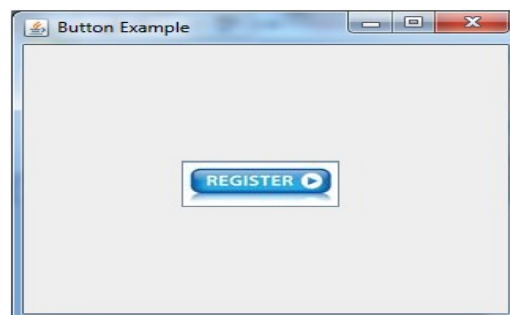


Example of displaying image on the button

```
import javax.swing.*;
public class ButtonExample
{
    ButtonExample()
    {
        JFrame f=new JFrame("Button Example");
        JButton b=new JButton(new ImageIcon("D:\\icon.png"));
        b.setBounds(100,100,100, 40);
        f.add(b);
        f.setSize(300,400);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args)
    {
        new ButtonExample();
    }
}
```

Output



Java 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.

JCheckBox class declaration

public class JCheckBox extends JToggleButton implements Accessible

Commonly used Constructors

Constructor	Description
JCheckBox()	Creates an initially unselected check box button with no text, no icon.
JChechBox(String s)	Creates an initially unselected check box with text.
JCheckBox(String text, boolean selected)	Creates a check box with text and specifies whether or not it is initially selected.
JCheckBox(Action a)	Creates a check box where properties are taken from the Action supplied.

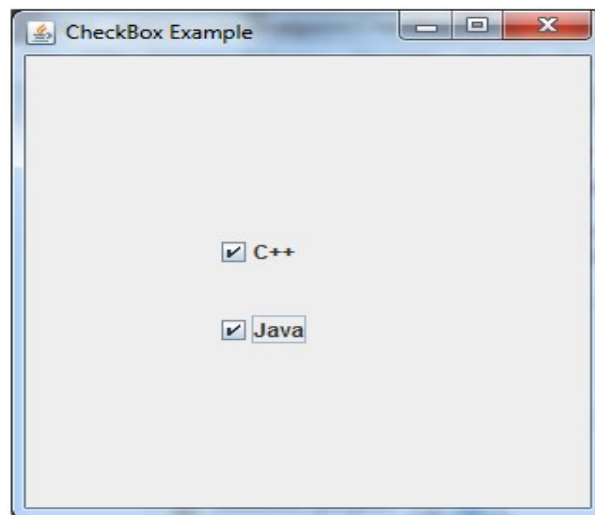
Commonly used Methods

Methods	Description
AccessibleContext getAccessibleContext()	It is used to get the AccessibleContext associated with this JCheckBox.
protected String paramString()	It returns a <u>string</u> representation of this JCheckBox.

Java JCheckBox 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();
    }
}
```

Output



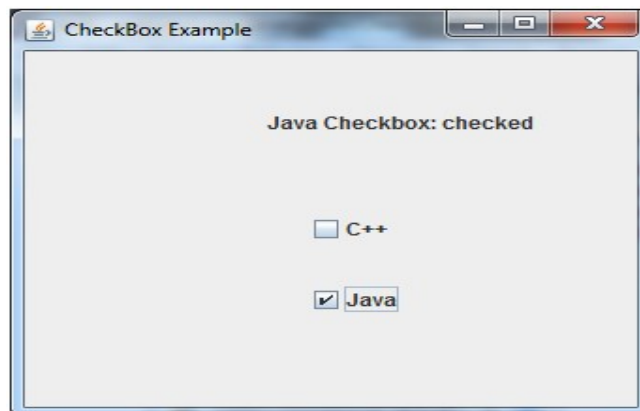
Java JCheckBox Example with ItemListener

```
import javax.swing.*;
import java.awt.event.*;
public class CheckBoxExample
{
    CheckBoxExample()
    {
        JFrame f= new JFrame("CheckBox Example");
        final JLabel label = new JLabel();
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setSize(400,100);
        JCheckBox checkbox1 = new JCheckBox("C++");
        checkbox1.setBounds(150,100, 50,50);
        JCheckBox checkbox2 = new JCheckBox("Java");
        checkbox2.setBounds(150,150, 50,50);
        f.add(checkbox1); f.add(checkbox2); f.add(label);
        checkbox1.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                label.setText("C++ Checkbox: "
                    + (e.getStateChange()==1?"checked":"unchecked")); } });

        checkbox2.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                label.setText("Java Checkbox: "
                    + (e.getStateChange()==1?"checked":"unchecked")); } });

        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String args[]) { new CheckBoxExample(); }
}
```

Output



Java JCheckBox Example: Food Order

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

public class CheckBoxExample extends JFrame implements ActionListener
{
    JLabel l;
    JCheckBox cb1,cb2,cb3;
    JButton b;
    CheckBoxExample()
    {
        l=new JLabel("Food Ordering System");           l.setBounds(50,50,300,20);
        cb1=new JCheckBox("Pizza @ 100");               cb1.setBounds(100,100,150,20);
        cb2=new JCheckBox("Burger @ 30");               cb2.setBounds(100,150,150,20);
        cb3=new JCheckBox("Tea @ 10");                   cb3.setBounds(100,200,150,20);
        b=new JButton("Order");                         b.setBounds(100,250,80,30);
        b.addActionListener(this);
        add(l);add(cb1);add(cb2);add(cb3);add(b);
        setSize(400,400);
        setLayout(null);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    public void actionPerformed(ActionEvent e)
    {
        float amount=0;
        String msg="";
        if(cb1.isSelected()) { amount+=100; msg="Pizza: 100\n"; }
        if(cb2.isSelected()) { amount+=30; msg+="Burger: 30\n"; }
        if(cb3.isSelected()) { amount+=10; msg+="Tea: 10\n"; }
        msg+="-----\n";
        JOptionPane.showMessageDialog(this,msg+"Total: "+amount);
    }
    public static void main(String[] args)
    {
        new CheckBoxExample();
    }
}
```

Output

Food Ordering System

☒ Pizza @ 100

☒ Burger @ 30

☒ Tea @ 10

Order

Message

i

Pizza: 100

Burger: 30

Tea: 10

Total: 140.0

OK

Java 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.

JRadioButton class declaration

public class JRadioButton extends JToggleButton implements Accessible

Commonly used Constructors

Constructor	Description
JRadioButton()	Creates an unselected radio button with no text.
JRadioButton(String s)	Creates an unselected radio button with specified text.
JRadioButton(String s, boolean selected)	Creates a radio button with the specified text and selected status.

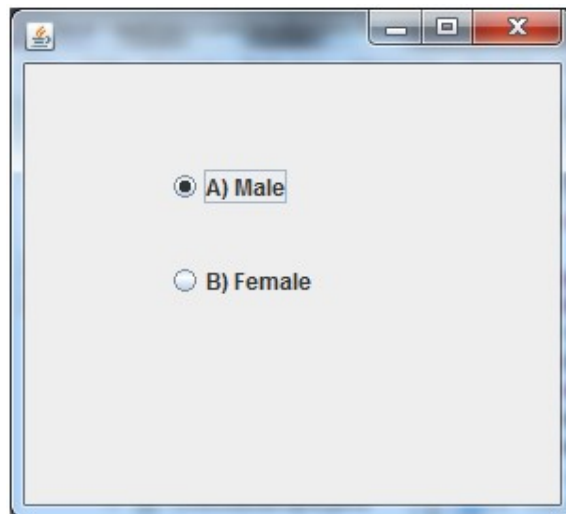
Commonly used Methods

Methods	Description
void setText(String s)	It is used to set specified text on button.
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

Java JRadioButton 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);
        r2.setBounds(75,100,100,30);
        ButtonGroup bg=new ButtonGroup();
        bg.add(r1);bg.add(r2);
        f.add(r1);f.add(r2);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
    }
    public static void main(String[] args)
    {
        new RadioButtonExample();
    }
}
```

Output

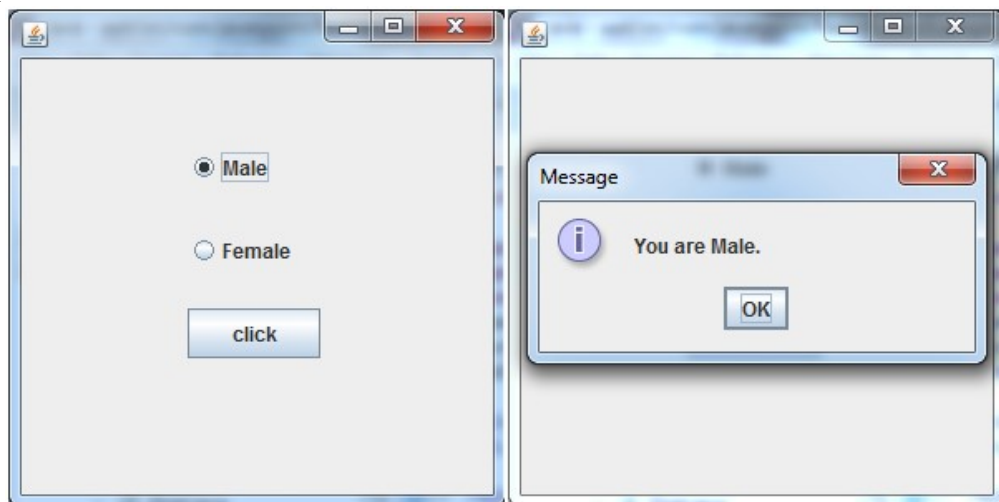


Java JRadioButton Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
class RadioButtonExample extends JFrame implements ActionListener
{
    JRadioButton rb1,rb2;
    JButton b;
    RadioButtonExample()
    {
        rb1=new JRadioButton("Male");    rb1.setBounds(100,50,100,30);
        rb2=new JRadioButton("Female");  rb2.setBounds(100,100,100,30);
        ButtonGroup bg=new ButtonGroup(); bg.add(rb1);    bg.add(rb2);
        b=new JButton("click");          b.setBounds(100,150,80,30);
        b.addActionListener(this);    add(rb1);    add(rb2);    add(b);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public void actionPerformed(ActionEvent e)
    {
        if(rb1.isSelected())
        {    JOptionPane.showMessageDialog(this,"You are Male.");    }
        if(rb2.isSelected())
        {    JOptionPane.showMessageDialog(this,"You are Female.");    }
    }

    public static void main(String args[])
    {    new RadioButtonExample();    }
}
```

Output



Java 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.

JComboBox class declaration

```
public class JComboBox extends JComponent implements ItemSelectable,  
ListDataListener, ActionListener, Accessible
```

Commonly used Constructors

Constructor	Description
JComboBox()	Creates a JComboBox with a default data model.
JComboBox(Object[] items)	Creates a JComboBox that contains the elements in the specified <u>array</u> .
JComboBox(Vector<?> items)	Creates a JComboBox that contains the elements in the specified <u>Vector</u> .

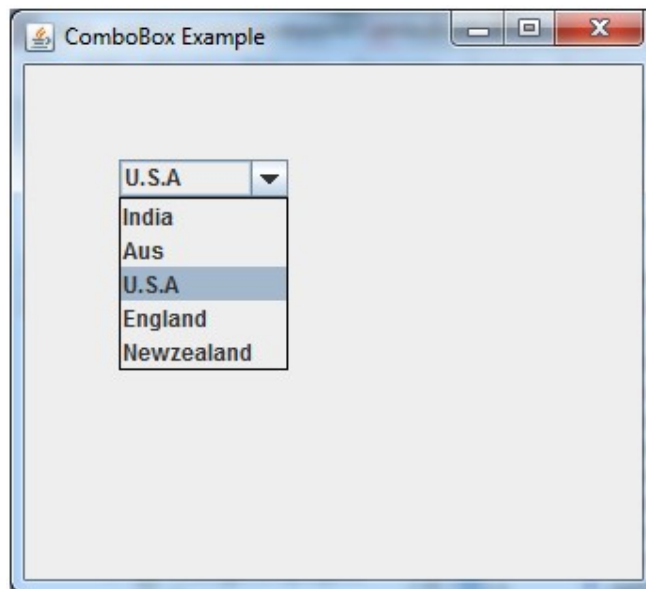
Commonly used Methods

Methods	Description
void addItem(Object anObject)	It is used to add an item to the item list.
void removeItem(Object anObject)	It is used to delete an item to the item list.
void removeAllItems()	It is used to remove all the items from the list.
void setEditable(boolean b)	It is used to determine whether the JComboBox is editable.
void addActionListener(ActionListener a)	It is used to add the <u>ActionListener</u> .
void addItemListener(ItemListener i)	It is used to add the <u>ItemListener</u> .

Java JComboBox 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);
        cb.setBounds(50, 50,90,20);
        f.add(cb);
        f.setLayout(null);
        f.setSize(400,500);
        f.setVisible(true);
    }
    public static void main(String[] args)
    {
        new ComboBoxExample();
    }
}
```

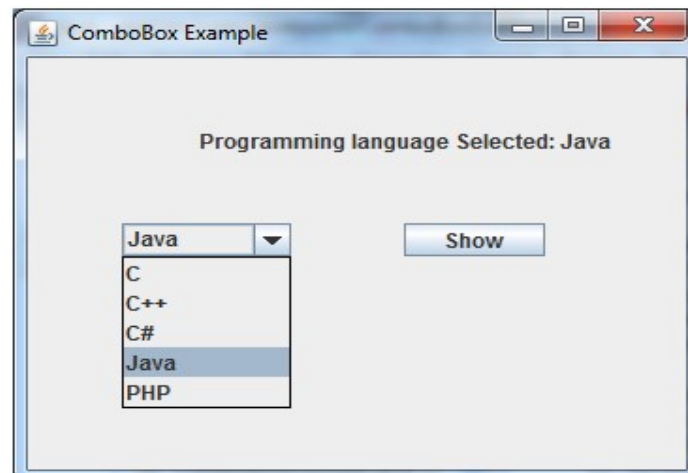
Output



Java JComboBox Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class ComboBoxExample
{
    JFrame f;
    ComboBoxExample()
    {
        f=new JFrame("ComboBox Example");
        final JLabel label = new JLabel();
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setSize(400,100);
        JButton b=new JButton("Show");
        b.setBounds(200,100,75,20);
        String languages[]={"C","C++","C#","Java","PHP"};
        final JComboBox cb=new JComboBox(languages);
        cb.setBounds(50, 100,90,20);
        f.add(cb);  f.add(label);      f.add(b);
        f.setLayout(null);
        f.setSize(350,350);
        f.setVisible(true);
        b.addActionListener (new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String data = "Programming language Selected: "
                    + cb.getItemAt(cb.getSelectedIndex());
                label.setText(data);    } } );
    }
    public static void main(String[] args)
    {
        new ComboBoxExample();
    }
}
```

Output



Java 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.

JTabbedPane class declaration

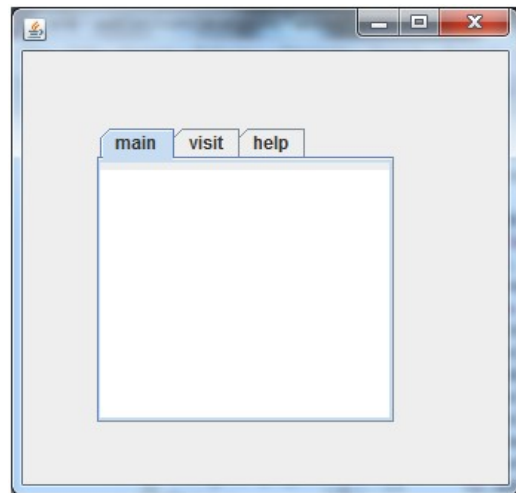
public class JTabbedPane **extends** JComponent **implements** Serializable, Accessible, SwingConstants

Commonly used Constructors

Constructor	Description
JTabbedPane()	Creates an empty TabbedPane with a default tab placement of JTabbedPane.Top.
JTabbedPane(int tabPlacement)	Creates an empty TabbedPane with a specified tab placement.
JTabbedPane(int tabPlacement, int tabLayoutPolicy)	Creates an empty TabbedPane with a specified tab placement and tab layout policy.

Example Program

```
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)
    {
        new TabbedPaneExample();
    }
}
```



Java 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.

Constructors

Constructor	Purpose
JScrollPane()	It creates a scroll pane. The Component parameter, when present, sets the scroll pane's client. The two int parameters, when present, set the vertical and horizontal scroll bar policies (respectively).
JScrollPane(Component)	
JScrollPane(int, int)	
JScrollPane(Component, int, int)	

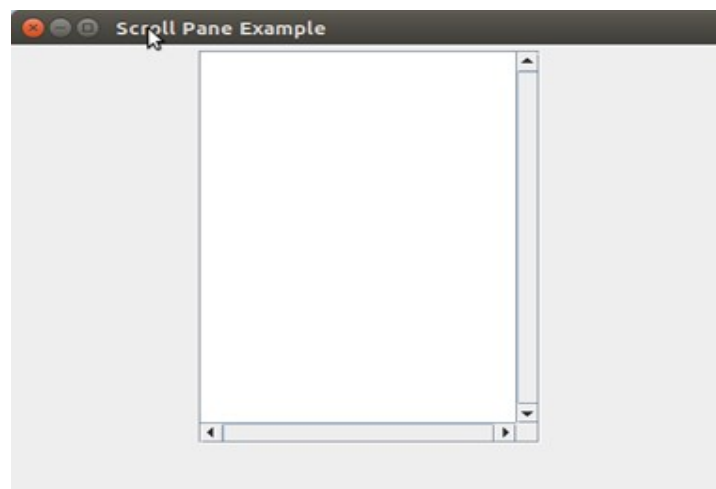
Useful Methods

Modifier	Method	Description
void	setColumnHeaderView(Component)	It sets the column header for the scroll pane.
void	setRowHeaderView(Component)	It sets the row header for the scroll pane.
void	setCorner(String, Component)	It sets or gets the specified corner. The int parameter specifies which corner and must be one of the following constants defined in ScrollPaneConstants: UPPER_LEFT_CORNER, UPPER_RIGHT_CORNER, LOWER_LEFT_CORNER, LOWER_RIGHT_CORNER, LOWER_LEADING_CORNER, LOWER_TRAILING_CORNER, UPPER_LEADING_CORNER, UPPER_TRAILING_CORNER.
Component	getCorner(String)	
void	setViewportView(Component)	Set the scroll pane's client.

JScrollPane 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()
    {
        final JFrame frame = new JFrame("Scroll Pane Example");
        frame.setSize(500, 500);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(new FlowLayout());
        JTextArea textArea = new JTextArea(20, 20);
        JScrollPane scrollableTextArea = new JScrollPane(textArea);
        scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_S
        CROLLBAR_ALWAYS);
        scrollableTextArea.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROL
        LBAR_ALWAYS);
        frame.getContentPane().add(scrollableTextArea);
    }
    public static void main(String[] args)
    {
        javax.swing.SwingUtilities.invokeLater(new Runnable()
        {
            public void run() { createAndShowGUI(); }
        });
    }
}
```

Output



Java 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.

JTree class declaration

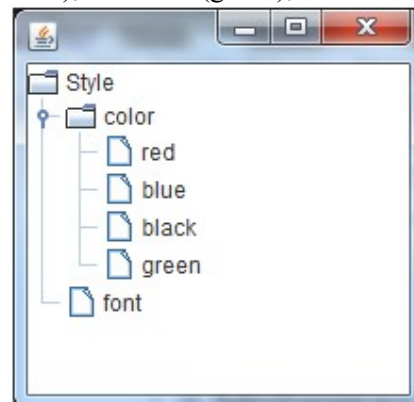
public class JTree extends JComponent implements Scrollable, Accessible

Commonly used Constructors

Constructor	Description
JTree()	Creates a JTree with a sample model.
JTree(Object[] value)	Creates a JTree with every element of the specified array as the child of a new root node.
JTree(TreeNode root)	Creates a JTree with the specified TreeNode as its root, which displays the root node.

Java JTree Example Program

```
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)
    {
        new TreeExample();
    }
}
```



Java JTable

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

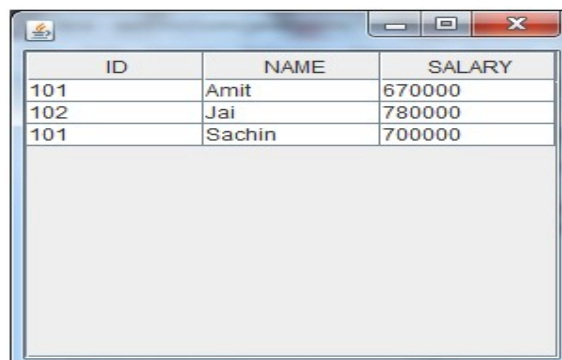
Commonly used Constructors

Constructor	Description
JTable()	Creates a table with empty cells.
JTable(Object[][] rows, Object[] columns)	Creates a table with the specified data.

Example Program

```
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);
        jt.setBounds(30,40,200,300);
        JScrollPane sp=new JScrollPane(jt);
        f.add(sp);
        f.setSize(300,400);
        f.setVisible(true);
    }
    public static void main(String[] args)
    {
        new TableExample();
    }
}
```

Output

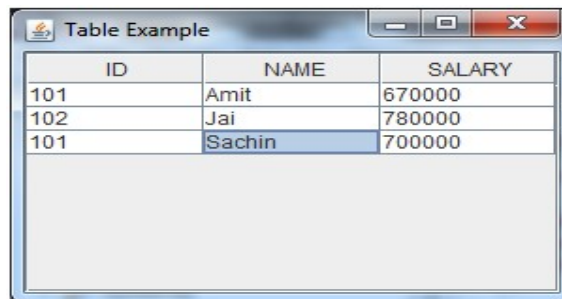


ID	NAME	SALARY
101	Amit	670000
102	Jai	780000
101	Sachin	700000

Java JTable Example with ListSelectionListener

```
import javax.swing.*.*;
import javax.swing.event.*;
public class TableExample
{
    public static void main(String[] a)
    {
        JFrame f = new JFrame("Table Example");
        String data[][]={ {"101","Amit","670000"}, {"102","Jai","780000"},
                           {"101","Sachin","700000"} };
        String column[]={"ID","NAME","SALARY"};
        final JTable jt=new JTable(data,column);
        jt.setCellSelectionEnabled(true);
        ListSelectionModel select= jt.getSelectionModel();
        select.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        select.addListSelectionListener ( new ListSelectionListener()
        { public void valueChanged(ListSelectionEvent e)
          {
              String Data = null;
              int[] row = jt.getSelectedRows();
              int[] columns = jt.getSelectedColumns();
              for (int i = 0; i < row.length; i++)
                  for (int j = 0; j < columns.length; j++)
                      Data = (String) jt.getValueAt(row[i], columns[j]);
              System.out.println("Table element selected is: " + Data);
          } } );
        JScrollPane sp=new JScrollPane(jt);
        f.add(sp);
        f.setSize(300, 200);
        f.setVisible(true);
    }
}
```

Output



ID	NAME	SALARY
101	Amit	670000
102	Jai	780000
101	Sachin	700000

If you select an element in column **NAME**, name of the element will be displayed on the console:

Table element selected is: Sachin