

Advanced Java Technology

Mr. Prashant Sahatiya, Assistant Professor
Information Technology Engineering



Teaching Scheme

- Lectures: 3 Hrs/Week
- Subject Credit: 3

PU



Examination Scheme

- External Exam:
 - Theory: 60 Marks
 - Practical: NA
- Internal Exam:
 - Theory: 20 Marks
 - Continuous Evaluation: 20 Marks
 - Practical: NA
- **Total Marks: 100**



CHAPTER-1

AWT & Swing



Outline

- Abstract Window Toolkit classes hierarchy, windows fundamentals,
- Creating a frame window in applet, canvas, creating windows program,
- Graphics-AWT Controls, Layout Managers, JApplet, JLabel,
- JTextField, JButton, JCheckBox, JRadioButton, JComboBox, Menus,
- MouseEvent Class , ActionEvent Class, WindowEventClass
- MouseListener, ActionListener, WindowListener and KeyListner

Java AWT

- Java AWT (Abstract Window Toolkit) is an API to develop **Graphical User Interface (GUI)** or windows-based applications in Java.
- Java AWT components are **platform-dependent** i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).



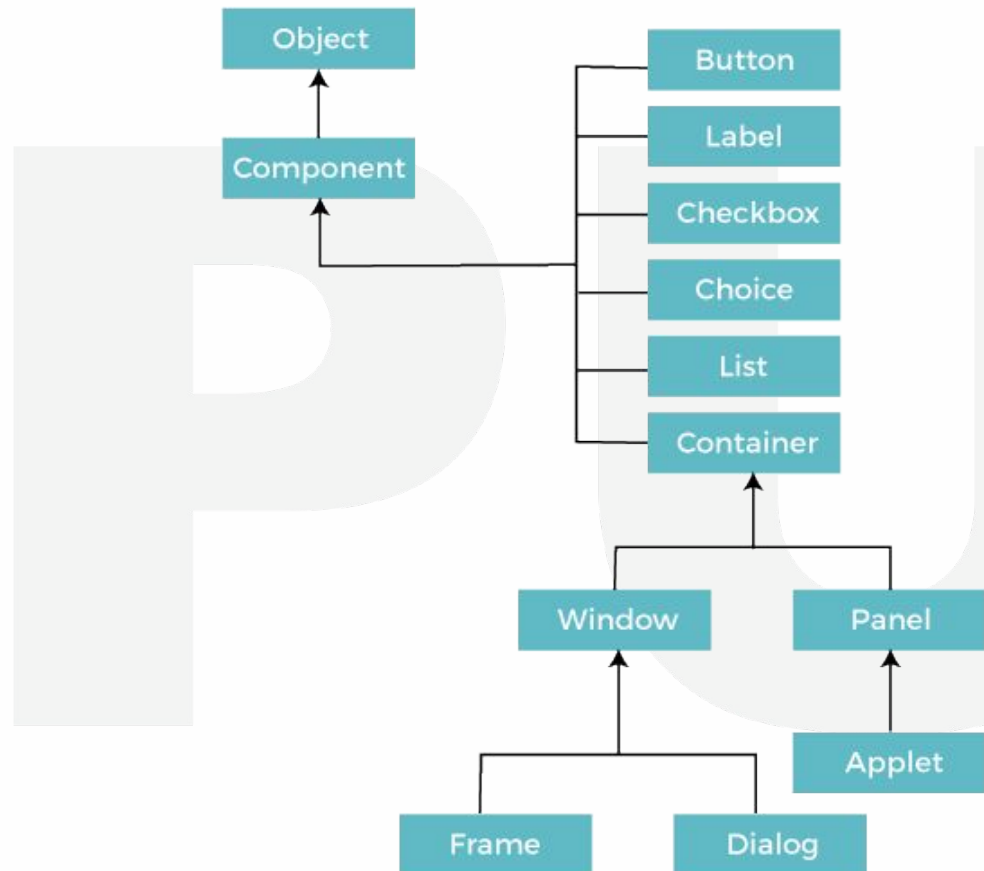
Java AWT

- The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.
- The AWT will help the user to understand Java GUI programming in simple and easy steps.

Why AWT is platform dependent?

- Java AWT calls the native platform (operating systems) subroutine for creating API components like TextField, CheckBox, button, etc.
- For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.
- In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

Java AWT Hierarchy





Components

- All the elements like the **button**, **text fields**, **scroll bars**, etc. are called components.
- In Java AWT, there are classes for each component as shown in above diagram. In order to place every component in a particular position on a screen, we need to add them to a **container**.

Containers

- The Container is a component in AWT 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**.
- It is basically a screen where the components are placed at their specific locations. Thus it contains and controls the layout of components.

Types of containers:

There are four types of containers in Java AWT:

1. Window
2. Panel
3. Frame
4. Dialog

PU



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. We need to create an instance of Window class to create this container.

Panel

- The Panel is the container that **doesn't contain title bar, border or menu bar**.
- It is generic container for holding the components. It can have other components like button, text field etc. An instance of Panel class creates a container, in which we can add components.

Frame

- The Frame is the container that **contain title bar** and **border** and can have **menu bars**. It can have other components like button, text field, scrollbar etc. Frame is most widely used container while developing an AWT application.



Useful Methods of Component Class

Sr No	Method	Description
1	<code>public void add(Component c)</code>	Inserts a component on this component.
2	<code>public void setSize(int width,int height)</code>	Sets the size (width and height) of the component.
3	<code>public void setLayout(LayoutManager m)</code>	Defines the layout manager for the component.
4	<code>public void setVisible(boolean status)</code>	Changes the visibility of the component, by default false.

Java AWT Example

- To create simple AWT example, you need a frame. There are two ways to create a GUI using Frame in AWT.
 1. By extending Frame class (**inheritance**)
 2. By creating the object of Frame class (**association**)



Java AWT Example

```
import java.awt.*;

public class AWTEExample1 extends Frame { // extending Frame class to our class AWTEExample1
    AWTEExample1() { // initializing using constructor
        Button b = new Button("Click Me!!"); // creating a button
        b.setBounds(30,100,80,30); // setting button position on screen
        add(b); // adding button into frame
        setSize(300,300); // frame size 300 width and 300 height
        setTitle("This is our basic AWT example"); // setting the title of Frame
        setLayout(null); // no layout manager
        setVisible(true); // now frame will be visible, by default it is not visible
        public static void main(String args[]) { // main method
            AWTEExample1 f = new AWTEExample1(); // creating instance of Frame class
        }
    }
}
```



Java AWT Example

```
import java.awt.*; // importing Java AWT class
class AWTEmployee2 { // class AWTEmployee2 directly creates instance of Frame class
    AWTEmployee2() { // initializing using constructor
        Frame f = new Frame(); // creating a Frame
        Label l = new Label("Employee id:"); // creating a Label
        Button b = new Button("Submit"); // creating a Button
        TextField t = new TextField(); // creating a TextField
        l.setBounds(20, 80, 80, 30);
        t.setBounds(20, 100, 80, 30);
        b.setBounds(100, 100, 80, 30);

        f.add(b);
        f.add(l);
        f.add(t);
        f.setSize(400,300);
        f.setTitle("Employee info");
        f.setLayout(null);
        f.setVisible(true);
    }

    public static void main(String args[]) {
        AWTEmployee2 awt_obj = new AWTEmployee2();
    }
}
```

Introduction to Java AWT Controls

- Java AWT controls are the controls that are used to design graphical user interfaces or web applications. To make an effective GUI, Java provides java.awt package that supports various AWT controls like Label, Button, CheckBox, CheckBox Group, List, Text Field, Text Area, Choice, Canvas, Image, Scrollbar, Dialog, File Dialog, etc that creates or draw various components on web and manage the GUI based application.

Structure of the Java AWT Controls

- The structure of the AWT is quite simple and it is used extensively in programs. Every AWT inherits controls from the Container Class.

Container Class

- It represents objects in graphical representation and it is an abstract class in the GUI interface.

1. Label

- A label is a user for placing text inside the container. A label is used only for inputting text. The label does not imply that the text can be altered or it can be used as a button which can be further worked upon.
 - *Label n=new Label("Name:",Label.CENTER);*

2. Button

- This command generates a button in the User Interface. Clicking on the button would move the command to another page or another web server which is used to show several other outputs in the user interface page.

- `a1=new Button("submit");`
- `a2=new Button("cancel");`

3. Checkbox

- There can be a certain question and the checkbox is used to determine the true or false nature of the question being asked. If the checkbox is ticked then it means that the said question is true which if it is unchecked it means that the said question is false. It is basically a true or false state in Java programming language.
 - *Checkbox checkbox1 = new Checkbox("Hello World");*

4. Checkbox Group

- As the name implies the checkbox group is a set of checkboxes that are being used in the programming language. There are many checkboxes that are being used and hence the group of checkboxes is known as the checkbox group.
- *CheckboxGroup cb = new CheckboxGroup();*
- *Checkbox checkBox1 = new Checkbox("Hello", cb, true);*
- *checkBox1.setBounds (100,100, 50,50);*

5. List

- The list gives a scrolling list of items for the user. The scrolling list of items is also being set by the user. The user sets the scrolling list of items such as Fruits, Vegetables, some questionnaire or other facts.
 - `List l1=new List(4);`
 - `l1.setBounds(100,100, 75,75);`

6. TextField

- A text field is used for the editing of a particular line of text which can be used within the programming concept.

```
na=new TextField(20);
```

7. TextArea

- A text area is used for the editing of multiple lines of text. The only difference between the Text field and Text area is that Text Field is used for editing a single line of text within the user interface while a Text Area is used for editing multiple lines of text.
 - `TextArea area=new TextArea("Welcome to the universe");`
 - `area.setBounds(10,30, 300,300);`

8. Choice

- A choice, as the name implies, shows the various options and the choice that is selected is shown in the top menu bar of the screen.

```
- Choice c=new Choice();  
- c.setBounds(100,100, 75,75);  
- c.add("Subject 1");  
- c.add("Subject 2");  
- c.add("Subject 3");  
- c.add("Subject 4");  
- c.add("Subject 5");
```

9. Canvas

- In the canvas space, there can be an input being given by the user or the user can draw something on the Canvas space being given.
 - *f.add(new MyCanvas());*
 - *f.setLayout(null);*
 - *f.setSize(400, 400);*
 - *f.setVisible(true);*

Java Swing

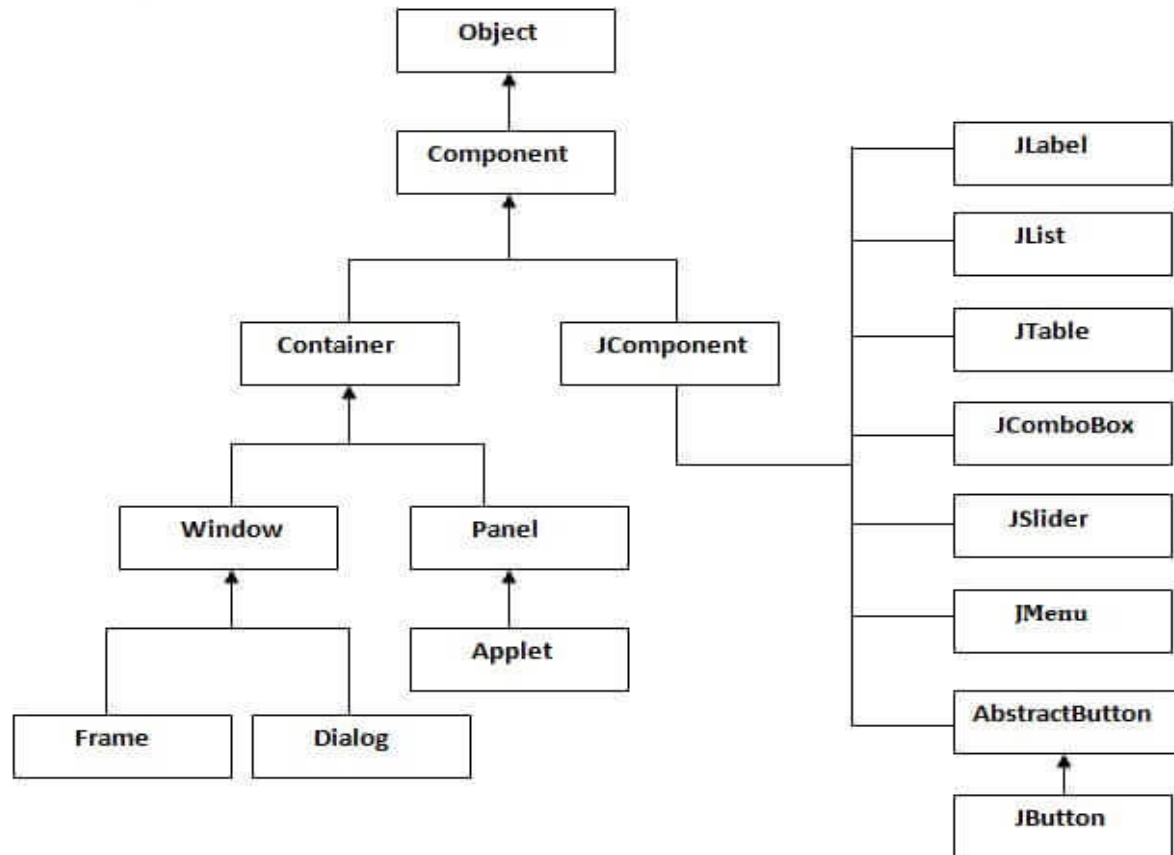
- Java Swing tutorial is a part of **Java Foundation Classes (JFC)** that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.
- Unlike AWT, Java Swing provides platform-independent and lightweight components.
- The 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 .

Java Swing Hierarchy



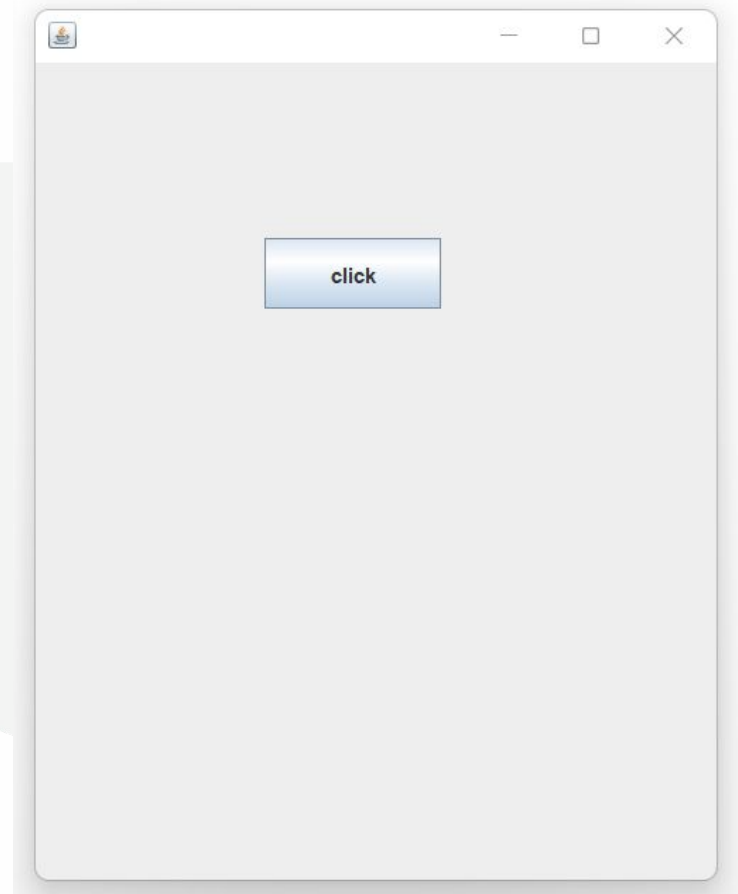


Useful Methods of Component Class

Sr No	Method	Description
1	<code>public void add(Component c)</code>	Inserts a component on this component.
2	<code>public void setSize(int width,int height)</code>	Sets the size (width and height) of the component.
3	<code>public void setLayout(LayoutManager m)</code>	Defines the layout manager for the component.
4	<code>public void setVisible(boolean status)</code>	Changes the visibility of the component, by default false.

Java Swing Examples

```
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  
    }  
}
```





Example of Swing by Association inside constructor

```
import javax.swing.*;

public class Simple {
    JFrame f;

    Simple(){
        f=new JFrame();//creating instance of JFrame

        JButton b=new JButton("click");//creating instance of JButton
        b.setBounds(130,100,100, 40);

        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
    }

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



Simple example of Swing by inheritance

```
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();
    }
}
```

Java ActionListener Interface

- The Java ActionListener is notified whenever you click on the button or menu item.
- It is notified against ActionEvent.
- The ActionListener interface is found in java.awt.event package.
- It has only one method: **actionPerformed()**.



How to write ActionListener

- The common approach is to implement the ActionListener. If you implement the ActionListener class, you need to follow 3 steps:

1) Implement the ActionListener interface in the class

```
public class ActionListenerExample Implements ActionListener
```

2) Register the component with the Listener

```
component.addActionListener(instanceOfListenerclass);
```

3) Override the actionPerformed() method

```
public void actionPerformed(ActionEvent e){  
    //Write the code here  
}
```



How to write ActionListener

```
1.import java.awt.*;
2.import java.awt.event.*;
3.//1st step
4.public class ActionListenerExample implements ActionListener{
5.public static void main(String[] args) {
6.    Frame f=new Frame("ActionListener Example");
7.    final TextField tf=new TextField();
8.    tf.setBounds(50,50, 150,20);
9.    Button b=new Button("Click Here");
10.   b.setBounds(50,100,60,30);
11.   //2nd step
12.   b.addActionListener(this);
13.   f.add(b);f.add(tf);
14.   f.setSize(400,400);
15.   f.setLayout(null);
16.   f.setVisible(true);
17.}
18.//3rd step
19.public void actionPerformed(ActionEvent e){
20.    tf.setText("Welcome to Javatpoint.");
21.}
22.}
```


Java MouseListener Interface

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

1. **public abstract void** mouseClicked(MouseEvent e);
2. **public abstract void** mouseEntered(MouseEvent e);
3. **public abstract void** mouseExited(MouseEvent e);
4. **public abstract void** mousePressed(MouseEvent e);
5. **public abstract void** mouseReleased(MouseEvent e);



Java KeyListener Interface

- The **Java KeyListener** is notified whenever you change the state of **key**. It is notified against **KeyEvent**. The **KeyListener** interface is found in **java.awt.event** package, and it has three methods.

Sr. no.	Method name	Description
1.	<code>public abstract void keyPressed (KeyEvent e);</code>	It is invoked when a key has been pressed.
2.	<code>public abstract void keyReleased (KeyEvent e);</code>	It is invoked when a key has been released.
3.	<code>public abstract void keyTyped (KeyEvent e);</code>	It is invoked when a key has been typed.

Java WindowListener Interface

- The Java WindowListener is notified whenever you change the state of window. It is notified against WindowEvent. The WindowListener interface is found in java.awt.event package. It has three methods.

PU



Java WindowListener Interface

Sr. no.	Method signature	Description
1.	public abstract void windowActivated(WindowEvent e);	It is called when the Window is set to be an active Window.
2.	public abstract void windowClosed(WindowEvent e);	It is called when a window has been closed as the result of calling dispose on the window.
3.	public abstract void windowClosing(WindowEvent e);	It is called when the user attempts to close the window from the system menu of the window.
4.	public abstract void windowDeactivated(WindowEvent e);	It is called when a Window is not an active Window anymore.
5.	public abstract void windowDeiconified(WindowEvent e);	It is called when a window is changed from a minimized to a normal state.
6.	public abstract void windowIconified(WindowEvent e);	It is called when a window is changed from a normal to a minimized state.
7.	public abstract void windowOpened(WindowEvent e);	It is called when window is made visible for the first time.

JAVA Swing Components - 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.

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.



JAVA Swing Components - Java JButton

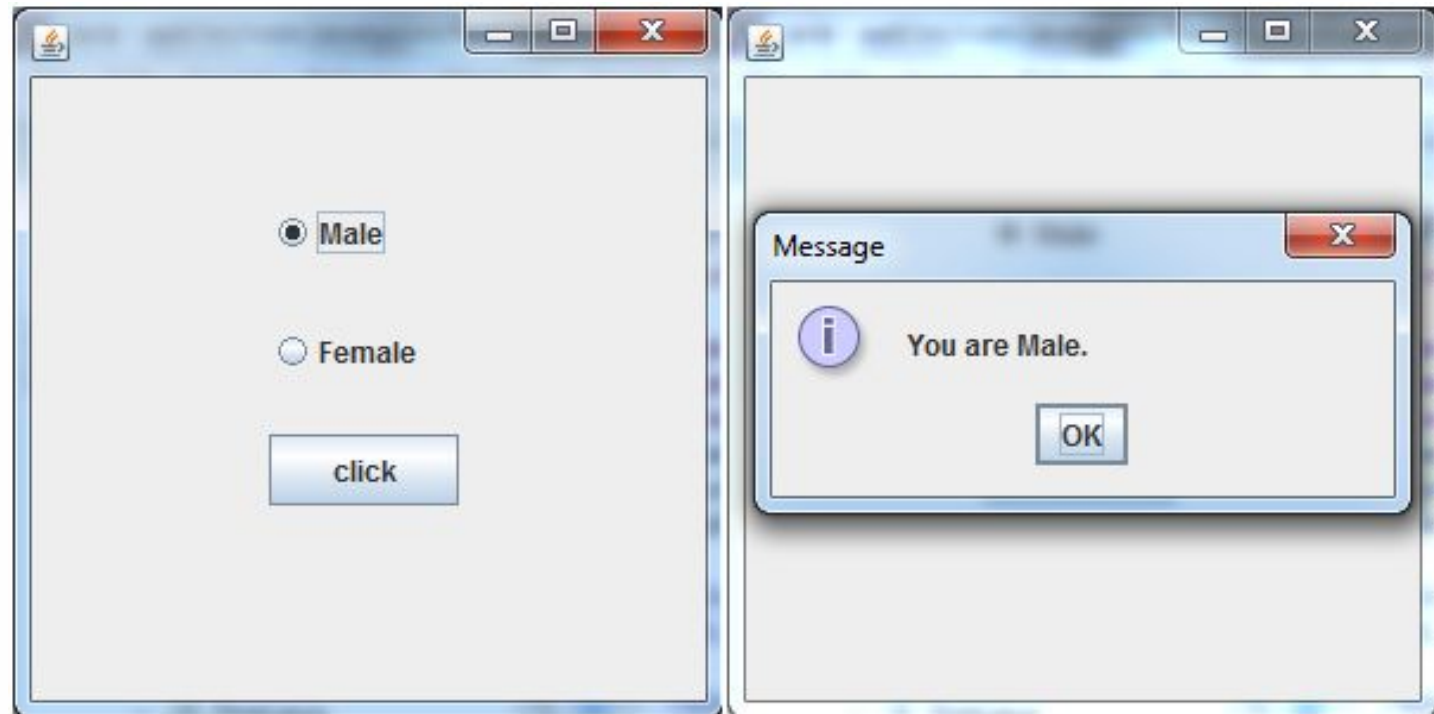
```
1.import javax.swing.*;  
2.public class ButtonExample {  
3.public static void main(String[] args) {  
4.    JFrame f=new JFrame("Button Example");  
5.    JButton b=new JButton("Click Here");  
6.    b.setBounds(50,100,95,30);  
7.    f.add(b);  
8.    f.setSize(400,400);  
9.    f.setLayout(null);  
10.   f.setVisible(true);  
11.}  
12.}
```



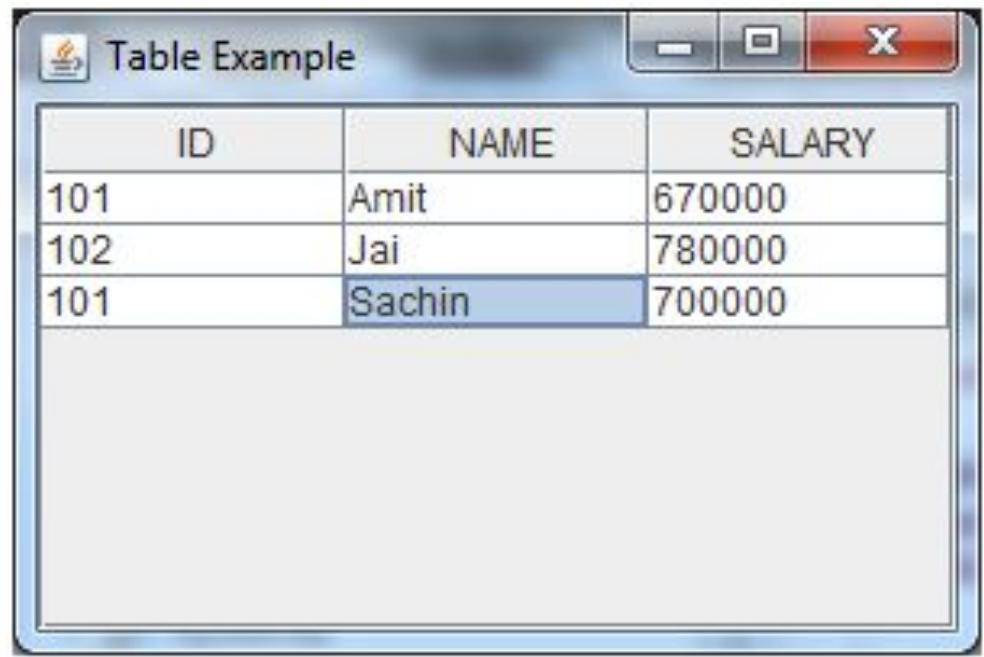
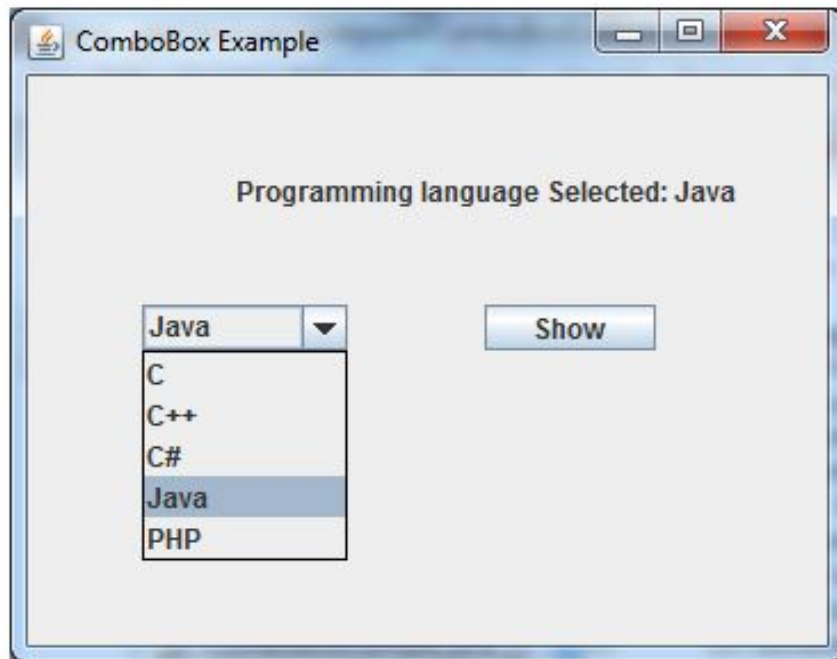
JAVA Swing Components

- JLabel, JTextField, JTextArea
- JPasswordField: The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.
- JCheckBox
- JRadioButton
- 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.
- JTable: The JTable class is used to display data in tabular form. It is composed of rows and columns.

JAVA Swing Components



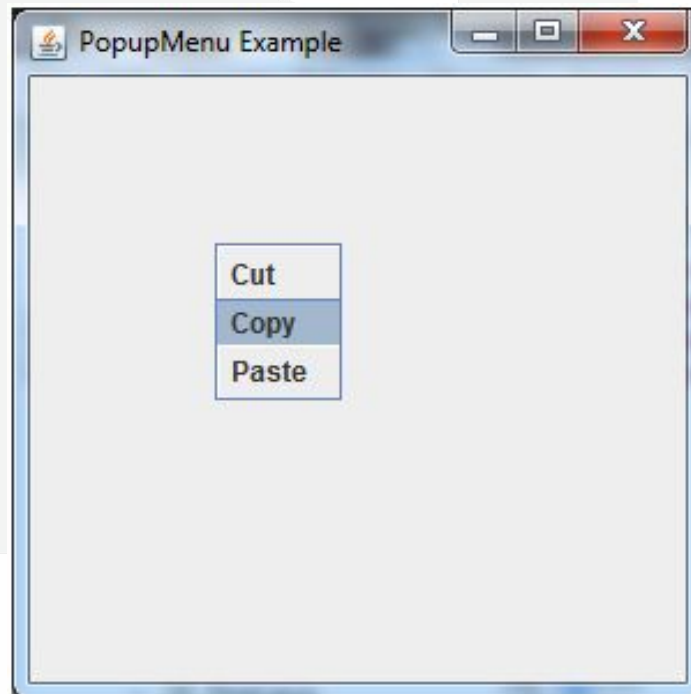
JAVA Swing Components

A screenshot of a Java Swing window titled "Table Example". The window has a standard Mac OS-style title bar with minimize, maximize, and close buttons. The main content area displays a table with three columns: "ID", "NAME", and "SALARY". The table contains three rows of data. The first row has ID "101", NAME "Amit", and SALARY "670000". The second row has ID "102", NAME "Jai", and SALARY "780000". The third row has ID "101", NAME "Sachin", and SALARY "700000". The "Sachin" row is highlighted with a blue background.

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

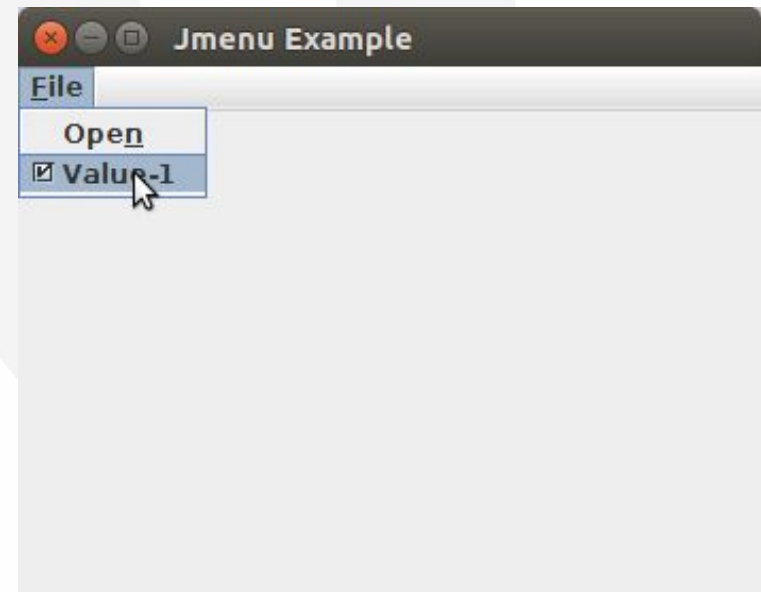
JPopupMenu

- JPopupMenu can be dynamically popped up at specific position within a component. It inherits the JComponent class.



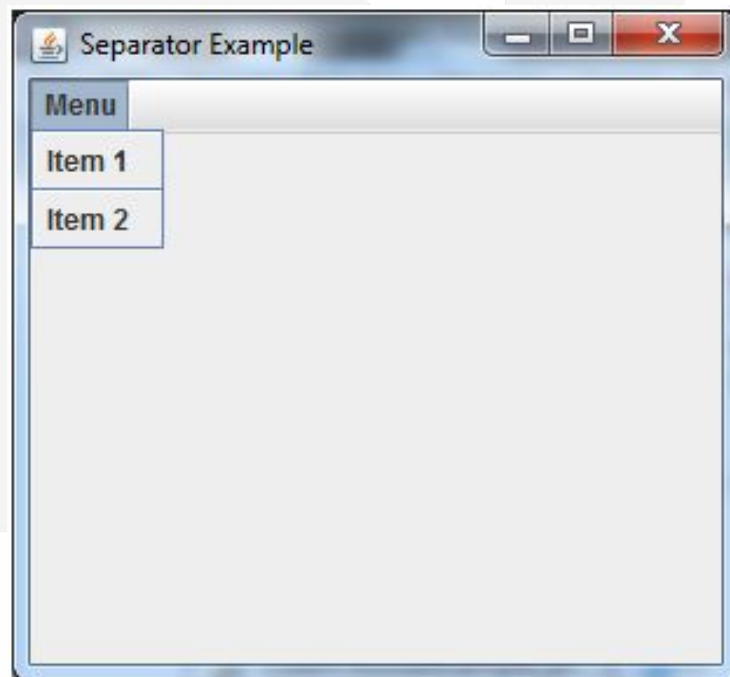
JCheckBoxMenuItem

- JCheckBoxMenuItem class represents checkbox which can be included on a menu.
- A JCheckBoxMenuItem can have text or a graphic icon or both, associated with it. MenuItem can be selected or deselected. MenuItems can be configured and controlled by actions.



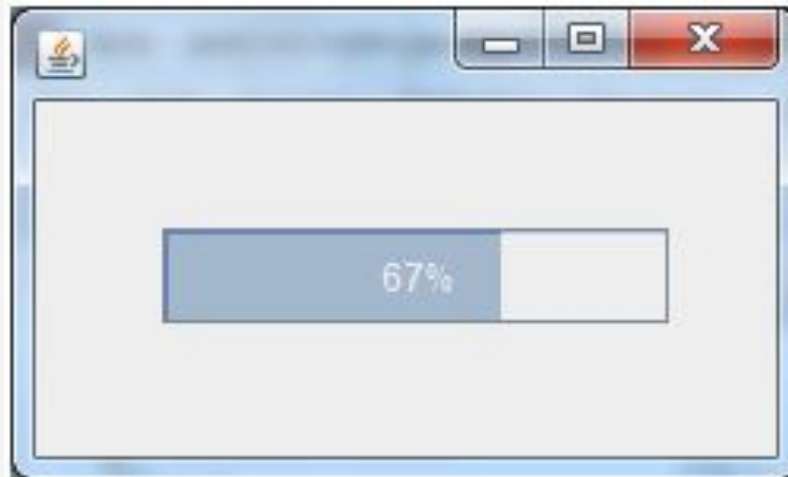
JSeparator

- The object of JSeparator class is used to provide a general purpose component for implementing divider lines. It is used to draw a line to separate widgets in a Layout. It inherits JComponent class.



JProgressBar

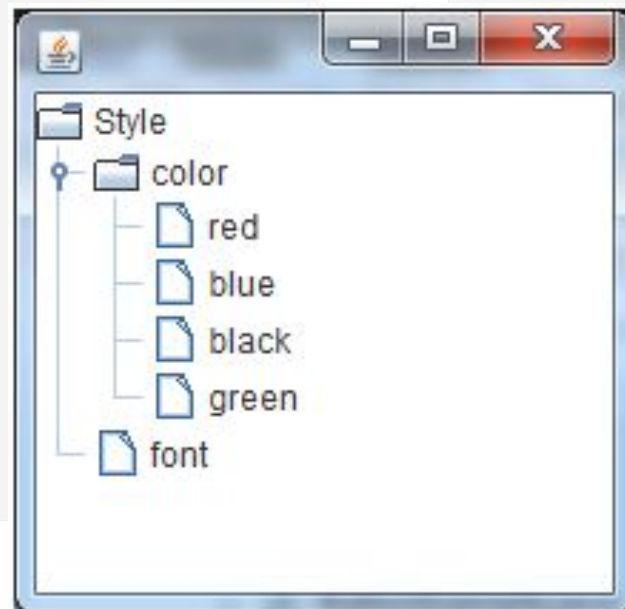
- The JProgressBar class is used to display the progress of the task. It inherits JComponent class.





Java JTree

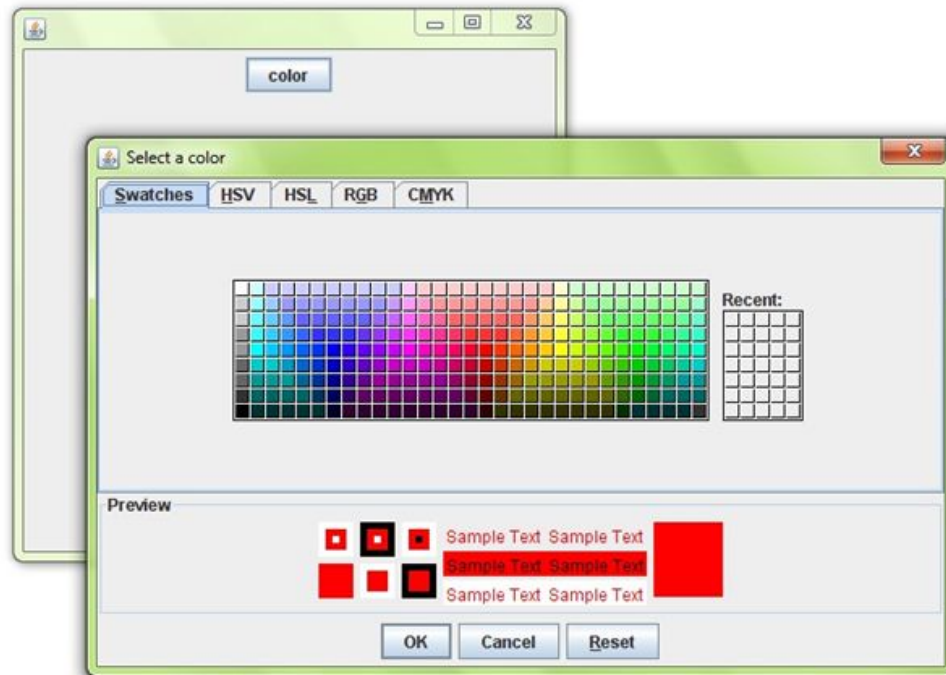
- The JTree class is used to display the tree structured data or hierarchical data. JTree is a complex component. It has a 'root node' at the top most which is a parent for all nodes in the tree. It inherits JComponent class.





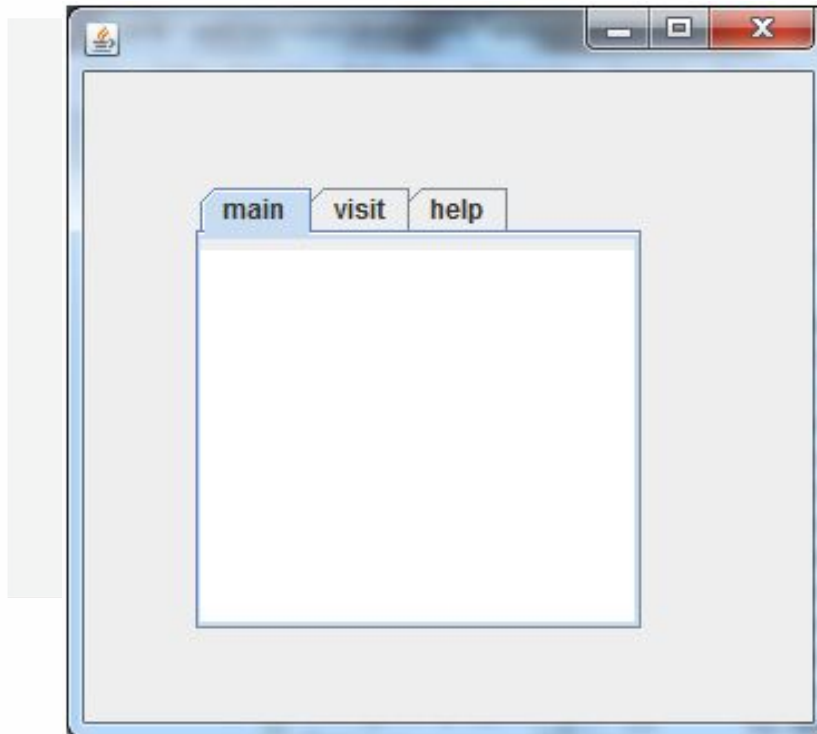
JColorChooser

- The JColorChooser class is used to create a color chooser dialog box so that user can select any color. It inherits JComponent class.



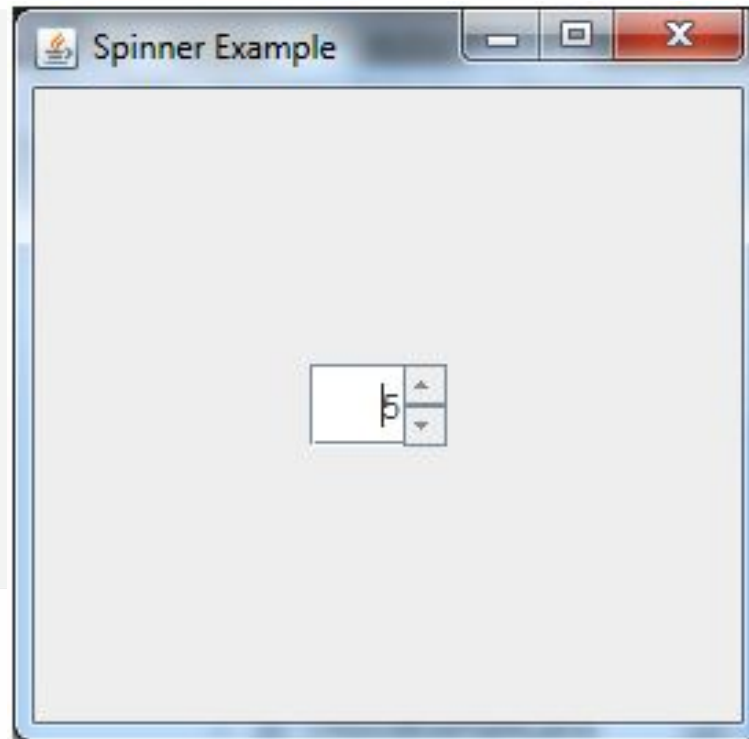
JTabbedPane

- 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.



JSpinner

- The object of JSpinner class is a single line input field that allows the user to select a number or an object value from an ordered sequence.



× ○ DIGITAL LEARNING CONTENT



Parul[®] University



www.paruluniversity.ac.in