UNIT V **APPLET**

APPLETS:

An applet is a Java program that runs in a Web browser.(or) 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 small Java applications that can be accessed on an Internet server, transported over Internet, and can be automatically installed and run as apart of a web document.

After a user receives an applet, the applet can produce a graphical user interface. It has limited access to resources so that it can run complex computations without introducing the risk of viruses or breaching data integrity.

Any applet in Java is a class that extends the java.applet.Applet class.

An Applet class does not have any main() method. It is viewed using JVM. The JVM can use either a plug-in of the Web browser or a separate runtime environment to run an applet application.

JVM creates an instance of the applet class and invokes init() method to initialize an Applet.

Advantage of Applet:

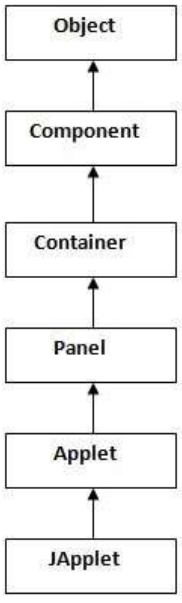
There are many advantages of applet. They are as follows:

•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.

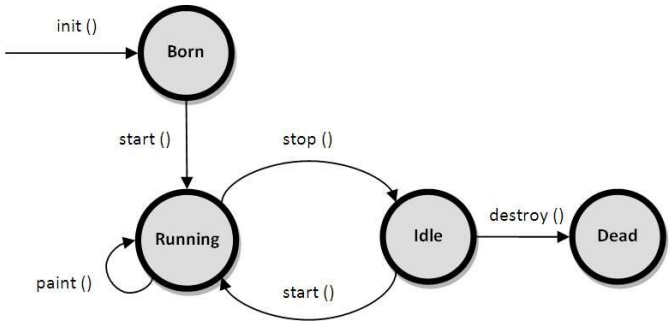
Hierarchy of Applet :



As displayed in the diagram, Applet class extends Panel. Panel class extends Container, which is the subclass of Component. Where Object class is base class for all the classes in java. JApplet class is extension of Applet class.

Applet Life Cycle

The life cycle of an applet is as shown in the figure below:



The life cycle of an applet starts with init() method and ends with destroy() method. Other life cycle methods are start(), stop() and paint(). The methods to execute only once in the applet life cycle are init() and destroy(). Other methods execute multiple times.

init(): The init() method is the first method to execute when the applet is executed. Variable declaration and initialization operations are performed in this method.

start(): The start() method contains the actual code of the applet that should run. The start() method executes immediately after the init() method. It also executes whenever the applet is restored, maximized or moving from one tab to another tab in the browser.

stop(): The stop() method stops the execution of the applet. The stop() method executes when the applet is minimized or when moving from one tab to another in the browser.

destroy(): The destroy() method executes when the applet window is closed or when the tab containing the webpage is closed. stop() method executes just before when destroy() method is invoked. The destroy() method removes the applet object from memory.

paint(): The paint() method is used to redraw the output on the applet display area. The paint() method executes after the execution of start() method and whenever the applet or browser is resized.

The method execution sequence when an applet is executed is:

- init()
- start()
- paint()

The method execution sequence when an applet is closed is:

- stop()
- destroy()

Applet HTML tag

The <applet> tag embeds a Java applet (mini Java applications) on the page. An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page.

Ex.

<applet code = demo.class width = 400 height = 200>

</applet>

Applet tag having three attribute

1. code – specifies name of the applet (.class file)
2. width – specifies width of the applet (in pixel)
3. height – specifies height of the applet (in pixel)

Simple example of Applet:

To execute an Applet, FirstCreate an applet and compile it just like a simple java program.

First.java

importjava.applet.Applet;

importjava.awt.Graphics;

publicclassFirstextendsApplet

{

publicvoidpaint(Graphicsg)

{

g.drawString(“Welcome to Applet",50,150);

}}

Compile:

D:\> javac First.java

After successful compilation, we get First.classfile.

After that create an html file and place the applet code in html file

First.html

<html>

<body>

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

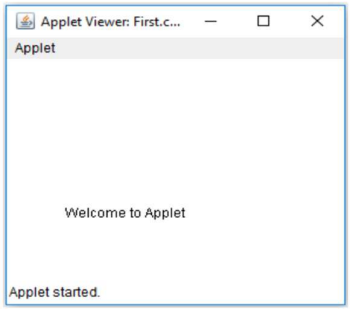
</applet>

</body>

</html>

Execute:

D:\> appletviewer First.html



How to run an Applet Program

An Applet program is compiled in the same way as you have been compiling your console programs. However there are two ways to run an applet.

Executing the Applet within Java-compatible web browser.

Using an Applet viewer, such as the standard tool, applet viewer. An applet viewer executes your applet in a window

For executing an Applet in an web browser, create short HTML file in the same directory.
Inside body tag of the file, include the following code. (applet tag loads the Applet class)

Running Applet using Applet Viewer

To execute an Applet with an applet viewer, write short HTML file as discussed above. If name it as run.htm, then the following command will run your applet program.

Security issues in Java applets include:

Code Execution: Applets could run arbitrary code on a user's computer, potentially leading to malware infection or unauthorized access to system resources.

Outdated Java Versions: Users might have outdated Java installations with known vulnerabilities that attackers could exploit.

Unsigned Applets: Even unsigned applets had limited access to system resources, raising the possibility of malicious actions within the user's browser or system.

Certificate Problems: Signed applets were considered safer, but compromised or stolen certificates could be used to distribute malicious applets.

Phishing: Malicious applets could be disguised as legitimate ones to trick users into running them.

Cross-Site Scripting (XSS): Applets could be exploited to facilitate cross-site scripting attacks, compromising user data.

Browser Support: Major browsers phased out support for applets due to their security concerns.

Performance and Compatibility: Applets often suffered from slow loading times, performance issues, and compatibility problems.

FEATURES OF JAVA APPLET:

The main features of the applet are discussed below.

Small size
Applets are typically small, making them easy to download and embed in web pages.
Browser-based
Applets run inside a web browser, providing a convenient way to add interactivity and dynamic content to web pages.
Restricted access to resources
Applets are run in a sandboxed environment, which restricts their access to system resources such as memory, processing power, and storage.
Limited user interaction
Applets typically provide limited user interaction, usually through HTML controls or UI components provided by the applet itself.
Deployment
Applets are typically embedded in HTML pages using the <applet> tag or the newer <object> tag.
Platform-independent
Applets are written in Java and can run on any platform that supports a Java Virtual Machine (JVM), which makes them highly portable.
Security
Applets provide a high level of security and cannot access system resources or sensitive information.

JAVA APPLICATION FEATURES

The main features of the application are discussed below.

Standalone
Applications are programs installed on a computer or other device and run independently of a web browser or other software.
Access to Resources
Applications have access to system resources allowing them to perform complex tasks and manipulate data.
User interaction
Applications provide user interaction through a GUI or CLI.
Deployment
They are generally distributed as executable files or installers. It can be downloaded and installed on a computer or other device.
Platform-specific
They are usually developed for specific operating systems or hardware platforms.
Extensibility
Applications can be extended through plug-ins, add-ons, or other third-party software components.

DIFFERENCE BETWEEN APPLICATION AND APPLET

JAVA APPLICATION	JAVA APPLET
An application is a standalone Java program that can be run independently on a client/server without the need for a web browser.	An applet is a form of Java program which is embedded with an HTML page and loaded by a web server to be run on a web browser.
The execution of the program starts from the main() method.	There is no requirement of main() method for the execution of the program.
The application can access local disk files/ folders and the network system.	The applet doesn't have access to the local network files and folders.

It doesn't require any Graphical User Interface (GUI).	It must run within a GUI.
It is a trusted application and doesn't require much security.	It requires high-security constraints as applets are untrusted.
It requires Java Runtime Environment (JRE) for its successful execution.	It requires a web browser like Chrome, Firefox, etc for its successful execution.
An application can perform read and write operations on files stored on the local disk.	An applet doesn't have access to local files so you cannot perform read and write operations on files stored on the local disk.

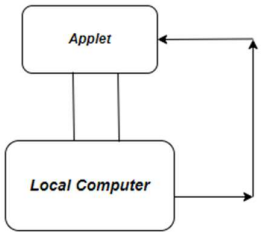
Types of applets

A web page can contain two types of applets:

- Local applet
- Remote applet

Local applets

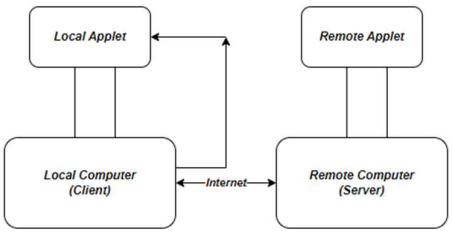
Local applets are developed and stored locally, and therefore do not require an Internet connection to be located because the directory is located on the local system.



A local applet

Remote applets

Remote applets are stored in a remote computer and an Internet connection is needed to access them. The remote applet is designed and developed by other developers. To find and load a remote applet, you need to know the address of the network applet, i.e., the Uniform Resource Locator (URL).



A remote applet

Introduction to Java GUI Programming with Swing:

Java Swing is a GUI toolkit and a part of JFC (Java Foundation Class) helpful in developing window-based applications.

Java Swing is lightweight and platform-independent that contains various components and container classes.

It provides a wide range of components, such as buttons, text fields, and menus, that can be used to create attractive and functional GUIs.

In Java Swing, there are several components like a scroll bar, button, text field, text area, checkbox, radio button, etc. These components jointly form a GUI that offers a rich set of functionalities and allows high-level customization.

What is Java Swing?

Java Swing is a set of graphical user interface (GUI) components that are part of the Java platform. Swing components are built on top of the Abstract Window Toolkit (AWT), but they provide a number of advantages over AWT components. For example, Swing components are more lightweight and efficient, and they are platform-independent.

How to install Java Swing

Java Swing is included in the Java Development Kit (JDK).

To install the JDK, go to the Oracle website and download the latest version of the JDK for your operating system. Once the JDK is installed, you can start creating Swing GUIs.

Main Features of Swing Toolkit

- ✓ Platform Independent
- ✓ Customizable
- ✓ Extensible
- ✓ Configurable
- ✓ Lightweight

Swing and JFC

JFC is an abbreviation for Java Foundation classes, which encompass a group of features for building Graphical User Interfaces(GUI) and adding rich graphical functionalities and interactivity to Java applications.

Features of JFC

- Swing GUI components.
- Look and Feel support.
- Java 2D.

Introduction to Swing Classes

JPanel : JPanel is Swing's version of AWT class Panel and uses the same default layout, FlowLayout. JPanel is descended directly from JComponent.

JFrame : JFrame is Swing's version of Frame and is descended directly from Frame class. The component which is added to the Frame, is refered as its Content.

JWindow : This is Swing's version of Window and hass descended directly from Window class. Like Window it uses BorderLayout by default.

JLabel : JLabel descended from Jcomponent, and is used to create text labels.

JButton : JButton class provides the functioning of push button. JButton allows an icon, string or both associated with a button.

JTextField : JTextFields allow editing of a single line of text.

Creating a JFrame

There are two way to create a JFrame Window.

- By instantiating JFrame class.
- By extending JFrame class.

Limitations of AWT:

The AWT defines a basic set of controls, windows, and dialog boxes that support a usable, but limited graphical interface. One reason for the limited nature of the AWT is that it translates its various visual components into their corresponding, platform-specific equivalents or peers. This means that the look and feel of a component is defined by the platform, not by java. Because the AWT components use native code resources, they are referred to as heavy weight.

The use of native peers led to several problems. First, because of variations between operating systems, a component might look, or even act, differently on different platforms. This variability threatened java's philosophy: write once, run anywhere. Second, the look and feel of each component was fixed and could not be changed. Third, the use of heavyweight components caused some frustrating restrictions.

MODEL-VIEW-CONTROLLER (MVC)

The Model-View-Controller (MVC) architecture in Java is a design pattern that provides a structured approach for developing applications. It separates the application's concerns into three main components: the model, the view, and the controller. Each component has a specific role and responsibility within the architecture.

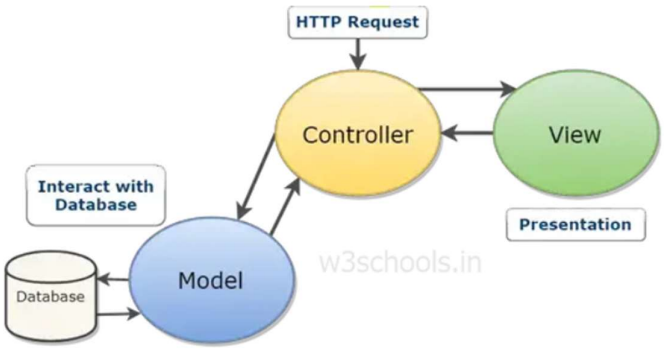


Fig: MVC Architecture

Model:
The model represents the data and business logic of the application. It encapsulates the application's data and provides methods for accessing, manipulating, and updating that data. The model component is independent of the user interface and focuses solely on the application's functionality.

View:
The view is responsible for rendering the user interface and displaying the data to the user. It presents the data from the model to the user in a visually appealing and understandable way. The view component does not contain any business logic but instead relies on the model for data.

Controller:
The controller acts as an intermediary between the model and the view. It handles user input, processes user actions, and updates the model or view accordingly. The controller interprets user

actions and triggers the appropriate methods in the model or view. It ensures the separation of concerns by keeping the view and model independent of each other.

Swing components

Swing provides a wide range of components that can be used to create GUIs. Swing Framework contains a large set of components which provide rich functionalities and allow high level of customization. All these components are lightweight components. They all are derived from JComponent class.Some of the most commonly used components include:

- Buttons
- Text fields
- Labels
- Menus
- Panels
- Layout managers

JButton

JButton class provides functionality of a button. The button can include some display text or images. It yields an event when clicked and double-clicked. JButton class has three constuctors,

JButton(Icon ic)

JButton(String str)

JButton(String str, Icon ic)

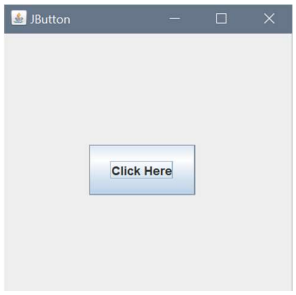
It allows a button to be created using icon, a string or both. JButton supports ActionEvent. When a button is pressed an ActionEvent is generated.

Syntax:

JButton okBtn = new JButton(“Click”);

EXAMPLE:

```
import javax.swing.*;

public class Jbutton_Class{

public static void main(String[] args) {

   JFrame f=new JFrame("JButton ");

   JButton b=new JButton("Click Here");

   b.setBounds(85,110,105,50);

   f.add(b);

   f.setSize(300,300);

   f.setLayout(null);

   f.setVisible(true);

} }
```



JLabel

JLabel is a Java Swing component. JLabel can show text, images, or a combination of the two. JLabel does not respond to input events like mouse or keyboard focus.

Syntax:

JLabel textLabel = new JLabel("This is 1st L...");

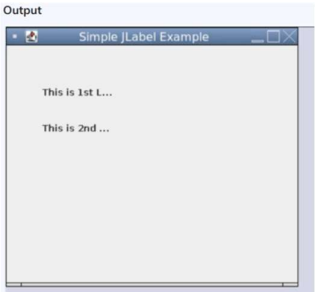
The JLabel Contains four constructors. They are as follows:

JLabel(): It generates a blank label with no text or image.
JLabel(String s): It generates a new label with the provided string.
JLabel(Icon I): This produces a new label with an image.
JLabel(String s, Icon i, int align): Creates a new label containing a string, an image, and a given horizontal alignment using this.
JLabel(Icon i, int align): Creates a JLabel instance with the picture and horizontal alignment that you provide.
JLabel(String s, int align): Creates a JLabel instance with the text and horizontal alignment that you provide.

Example:

```
import javax.swing.*;

class Main

{

public static void main(String args[])

  {

  JFrame myframe= new JFrame("Simple JLabel Example");

  JLabel label1,label2;

  label1=new JLabel("This is 1st Label");

  label1.setBounds(50,50, 100,30);

  label2=new JLabel("This is 2nd Label");

  label2.setBounds(50,100, 100,30);

  myframe.add(label1); myframe.add(label2);

  myframe.setSize(300,300);

  myframe.setLayout(null);
```

myframe.setVisible(true); } }



JTextField

The JTextField renders an editable single-line text box. Users can input non-formatted text in the box. We can initialize the text field by calling its constructor and passing an optional integer parameter. This parameter sets the box width measured by the number of columns. Also, it does not limit the number of characters that can be input into the box. It has three constructors:

JTextField(): It generates a new TextField Constructor.
JTextField(int columns): It generates a new empty TextField with the given number of columns.
JTextField(String text): Constructor for creating a new empty text field with the specified string as its first value.
JTextField(String text, int columns): Constructor that generates a new empty textField with a specified number of columns and the given string.
JTextField(Document doc, String text, int columns): Constructor for a textfield with the specified text storage model and a number of columns.

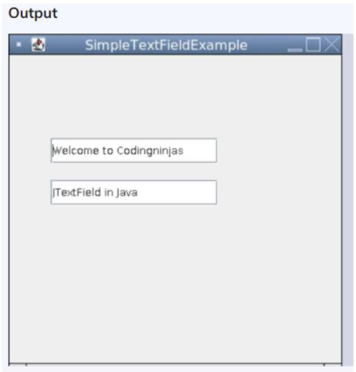
Syntax:

JTextField txtBox = new JTextField(50);

Example:

```
import javax.swing.*;

class Main

{

public static void main(String args[])

  {

  JFrame myframe= new JFrame("SimpleTextFieldExample");

  JTextField text1,text2;

  text1=new JTextField("Welcome to Codingninjas");

  text1.setBounds(50,100, 200,30);

  text2=new JTextField("JTextField in Java");

  text2.setBounds(50,150, 200,30);

  myframe.add(text1); myframe.add(text2);

  myframe.setSize(400,400);
```

```
myframe.setLayout(null);
myframe.setVisible(true);
} }
```



JCheckBox:

The JCheckBox renders a check-box with a label. The check-box has two states, i.e., on and off. On selecting, the state is set to "on," and a small tick is displayed inside the box.

Syntax:

```
CheckBox chkBox = new JCheckBox(“Java Swing”, true);
```

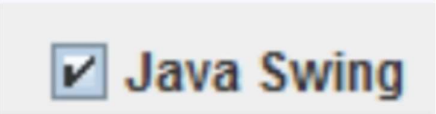
Constructors of JCheckBox

JCheckBox() - Initially, an unselected check box button is created with no text or icon.
JCheckBox(Action a) - This constructor creates a checkbox where the properties are taken from the Action supplied.
JCheckBox(Icon icon) - This constructor initially creates an unselected checkbox with an icon.
JCheckBox(Icon icon, boolean selected) - It creates a checkbox with an icon that specifies if it is initially selected or unselected.
JCheckBox(String text) - This constructor creates an unselected checkbox with the text.
JCheckBox(String text, boolean selected) - A checkbox with text is created and specifies if it is initially selected or unselected.
JCheckBox(String text, Icon icon) - An unselected checkbox is created with the specified icon and text.
JCheckBox(String text, Icon icon, boolean selected) - A checkbox is created with text and icon and specifies if it is initially selected or unselected.

Example:

```
import java.awt.*;
import javax.swing.*;
class solve extends JFrame
{
 static JFrame f;
  public static void main(String[] args)
```

```
{
    f = new JFrame("Coding Ninjas");
    f.setLayout(new FlowLayout());
    JCheckBox c1 = new JCheckBox("Ninja 1");
    JCheckBox c2 = new JCheckBox("Ninja 2");
    JPanel p = new JPanel();
    p.add(c1);
    p.add(c2);
    f.add(p);
    f.setSize(400, 400);
    f.show();
}}
```



JRadioButton

Radio button is a group of related button in which only one can be selected. JRadioButton class is used to create a radio button in Frames.

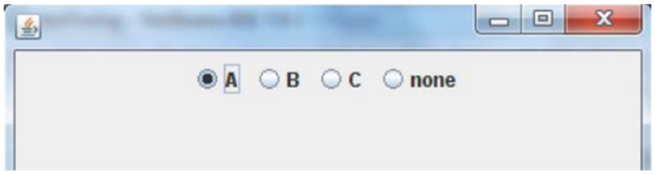
Syntax:

```
JRadioButton jrb = new JRadioButton("Easy");
```

Example:

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class Test extends JFrame
{
 public Test()
 {
  JRadioButton jcb = new JRadioButton("A");    //creating JRadioButton.
  add(jcb);                                    //adding JRadioButton to frame.
  jcb = new JRadioButton("B");                 //creating JRadioButton.
  add(jcb);                                    //adding JRadioButton to frame.
  jcb = new JRadioButton("C");                 //creating JRadioButton.
  add(jcb);                                    //adding JRadioButton to frame.
```

```
 jcb = new JRadioButton("none");
 add(jcb);
 setLayout(new FlowLayout());
 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 setSize(400, 400);
 setVisible(true);
}
public static void main(String[] args)
{
 new Test();
}}
```



JComboBox

Combo box is a combination of text fields and drop-down list.JComboBox component is used to create a combo box in Swing.

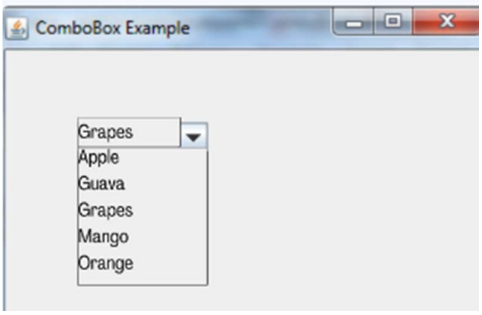
Syntax:

```
JComboBox jcb = new JComboBox(name);
```

Example:

```
import javax.swing.*;
public class ComboBoxExample {
    JFrame f;
    ComboBoxExample(){
        f = new JFrame("ComboBox Example");
        String country[]={“Apple”, “Guava”, “Grapes”, “Mango”, “Orange”};
        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();
}}
```



JTabbedPane

The JTabbedPane is another beneficial component that lets the user switch between tabs in an application. It is a handy utility as it allows users to browse more content without navigating to different pages.

Syntax:

JTabbedPane jtabbedPane = new JTabbedPane();

jtabbedPane.addTab("Tab_1", new JPanel());

jtabbedPane.addTab("Tab_2", new JPanel());

The above code creates a two-tabbed panel with headings Tab_1 and Tab_2.

JMenuBar, JMenu and JMenuItem

In Java, the Swing toolkit contains a JMenuBar, JMenu, and JMenuItem class. It is under package javax.swing.JMenuBar, javax.swing.JMenu and javax.swing.JMenuItem class. The JMenuBar class is used for displaying menubar on the frame. The JMenu Object is used to pull down the menu bar's components. The JMenuItem Object is used for adding the labeled menu item.

JMenuBar, JMenu and JMenuItem Declarations:

public class JMenuBar extends JComponent implements MenuElement, Accessible

public class JMenu extends JMenuItem implements MenuElement, Accessible

public class JMenuItem extends AbstractButton implements Accessible, MenuElement

JToggleButton

A JToggleButton is a button with two states. Selected and unselected are the two statuses. This class is subclassed by the JRadioButton and JCheckBox classes. The toggle button toggles

between being pressed and unpressed when the user presses it. JToggleButton is a button that allows you to choose from a list of options.

Constructors

JToggleButton(): Creates a toggle button with no text or image that is initially unselected.

JToggleButton(Action a): This method creates a toggle button with properties based on the Action.

JToggleButton(Icon icon): Creates a toggle button with the provided image but no text that is initially unselected.

JToggleButton(Icon icon, boolean selected): Creates a toggle button with the provided image and selection status, but no text.

JToggleButton(String text): Creates an unselected toggle button with the provided text.

JToggleButton(String text, boolean selected): Creates a toggle button with the supplied text and selection status.

JToggleButton(String text, Icon icon): This method creates an unselected toggle button with the provided text and image.

JToggleButton(String text, Icon icon, boolean selected): Creates a toggle button with the supplied text, image, and selection status.

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

JScrollPane()

JScrollPane(Component)

JScrollPane(int, int)

JScrollPane(Component, int, int)

JList

JList in Java is one of the Java Swing components. These components include buttons, sliders, list boxes, checkboxes, etc. JList is defined in Java in the 'java.swing' package. JList is a component that shows a list of objects and allows the user to choose one or more of them. JList is inherited from the JComponent class. JComponent is an abstract class that serves as the foundation for all Swing components.

Class Declaration

public class JList extends JComponent implements Scrollable, Accessible

JList()	Creates a JList with an empty, read-only, model.
JList(ary[] listData)	Creates a JList that displays the elements in the specified array.
JList(ListModel<ary> dataModel)	Creates a JList that displays elements from the specified, non-null, model.

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.

JDialog class declaration

public class JDialog extends Dialog implements WindowConstants, Accessible, RootPaneContainer

Constructors:

JDialog() - It is used to create a modeless dialog without a title and without a specified Frame owner.

JDialog(Frame owner) - It is used to create a modeless dialog with specified Frame as its owner and an empty title.

JDialog(Frame owner, String title, boolean modal) - It is used to create a dialog with the specified title, owner Frame and modality.

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

Let's see the declaration for javax.swing.JTree class.

public class JTree extends JComponent implements Scrollable, Accessible

JTable

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

JTable class declaration

Let's see the declaration for javax.swing.JTable class.