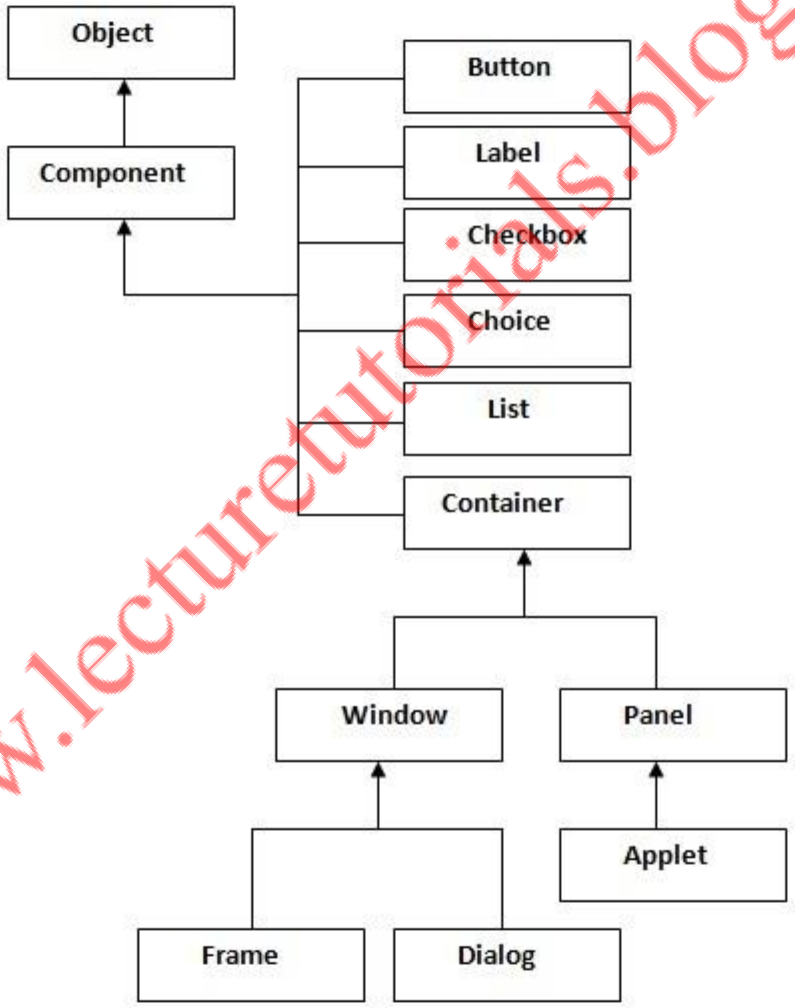
UNIT-VI

Java AWT:

- Java AWT (Abstract Windowing Toolkit) is an API to develop GUI or window-based application in java.
- Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components uses the resources of system.
- The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Java AWT Hierarchy

- The hierarchy of Java AWT classes are given below.



Container classes

- These are the predefined classes in java.awt package which can be used to display all non-container classes to the end user in the frame of window. container classes are; frame, panel.

non-container classes

- These are the predefined classes used to design the input field so that end user can provide input value to communicate with java program, these are also treated as GUI component. non-container classes are; Label, Button, List etc...
  non-container classes

- These are the predefined classes used to design the input field so that end user can provide input value to communicate with java program, these are also treated as GUI component.

Label

It can be any user defined name used to identify input field like textbox, textarea etc.

Example
Label l1=new Label("uname");
Label l2=new Label("password");

TextField

This class can be used to design textbox.

Example
TextField tf=new TextField();
Example
TextField tf=new TextField(40);
Button

This class can be used to create a push button.

Example
Button b1=new Button("submit");

Button b2=new Button("cancel");

TextArea

This class can be used to design a textarea, which will accept number of characters in rows and columns formate.

Example
TextArea ta=new TextArea(5, 10);
// here 5 is no. of rows and 10 is no. of column
Note: In above example at a time we can give the contains in textarea is in 5 rows and 9 column (n-1 columns).

Checkbox

This class can be used to design multi selection checkbox.

Example
Checkbox cb1=new Checkbox(".net");
Checkbox cb2=new Checkbox("Java");
Checkbox cb3=new Checkbox("html");
Checkbox cb4=new Checkbox("php");
CheckboxGroup

This class can be used to design radio button. Radio button is used for single selection.

Example
CheckboxGroup cbg=new CheckboxGroup();
Checkbox cb=new Checkbox("male", cbg, "true");
Checkbox cb=new Checkbox("female", cbg, "false");
Note: Under one group many number of radio button can exist but only one can be selected at a time.

Choice

This class can be used to design a drop down box with more options and supports single selection.

Example
Choice c=new Choice();

```
c.add(20);
c.add(21);
c.add(22);
c.add(23);
```

List

This class can be used to design a list box with multiple options and support multi selection.

Example

```
List l=new List(3);
l.add("goa");
l.add("delhi");
l.add("pune");
```

Note: By holding clt button multiple options can be selected.

Scrollbar

This class can be used to display a scroolbar on the window.

Example

Scrollbar sb=new Scrollbar(type of scrollbar, initialposition, thamb width, minmum value, maximum value);

Initial position represent very starting point or position of thumb.
Thumb width represent the size of thumb which is scroll on scrollbar
Minimum and maxium value represent boundries of scrollbar

Type of scrollbar

There are two types of scrollbar are available;

scrollbar.vertical
scrollbar.horizental
Example
Scrollbar sb=new Scrollbar(Scrollbar.HORIZENTAL, 10, 5, 0, 100);

Awt Frame

Frame

It is a predefined class used to create a container with the title bar and body part.

Example
Frame f=new Frame();
Mostly used methods

setTitle()

It is used to display user defined message on title bar.

Example
Frame f=new Frame();
f.setTitle("myframe");
setBackground()

It is used to set background or image of frame.

Example
Frame f=new Frame();
f.setBackground(Color.red);
Example
Frame f=new Frame();
f.setBackground("c:\image\myimage.png");

setForground()

It is used to set the foreground text color.

Example
Frame f=new Frame();
f.setForground(Color.red);

setSize()

It is used to set the width and height for frame.

Example

```
Frame f=new Frame();
f.setSize(400,300);
```

setVisible()

It is used to make the frame as visible to end user.

Example

```
Frame f=new Frame();
f.setVisible(true);
```

Note: You can write setVisible(true) or setVisible(false), if it is true then it visible otherwise not visible.

setLayout()

It is used to set any layout to the frame. You can also set null layout it means no any layout apply on frame.

Example

```
Frame f=new Frame();
f.setLayout(new FlowLayout());
```

Note: Layout is a logical container used to arrange the gui components in a specific order

add()

It is used to add non-container components (Button, List) to the frame.

Example

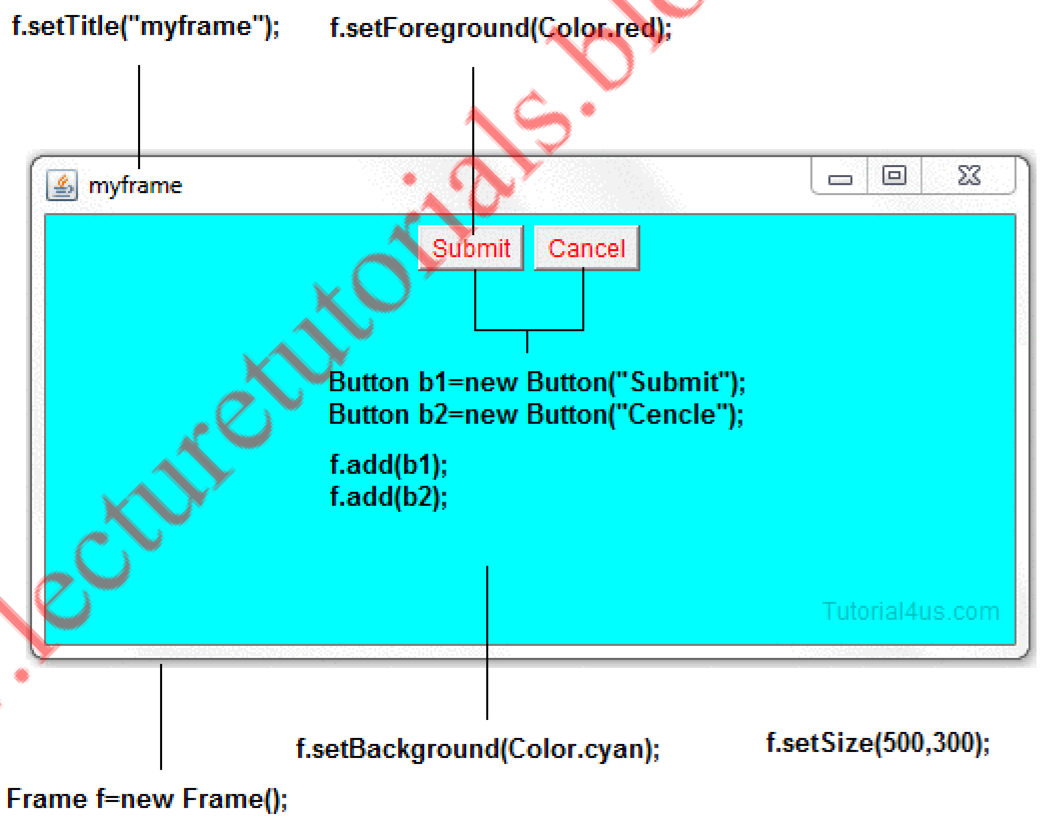
```
Frame f=new Frame();
Button b=new Button("Click");
f.add(b);
```

Explanation: In above code we add button on frame using f.add(b), here b is the object of Button class..

Example of Frame

```
import java.awt.*;

class FrameDemo
{
public static void main(String[] args)
{
Frame f=new Frame();
f.setTitle("myframe");
f.setBackground(Color.cyan);
f.setForeground(Color.red);
f.setLayout(new FlowLayout());
Button b1=new Button("Submit");
Button b2=new Button("Cancel");
f.add(b1);
f.add(b2);
f.setSize(500,300);
f.setVisible(true);
}
}
```


f.setTitle("myframe");
f.setForeground(Color.red);
myframe
Submit
Cancel
Button b1=new Button("Submit");
Button b2=new Button("Cencle");
f.add(b1);
f.add(b2);
Tutorial4us.com
f.setBackground(Color.cyan);
f.setSize(500,300);
Frame f=new Frame();

Introduction to swings

- Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT.
- In addition to the familiar components, such as buttons, check boxes, and labels, Swing supplies several exciting additions, including tabbed panes, scroll panes, trees, and tables.
- Even familiar components such as buttons have more capabilities in Swing.
- For example, a button may have both an image and a text string associated with it. Also, the image can be changed as the state of the button changes.
- Unlike AWT components, Swing components are not implemented by platform-specific code.
- Instead, they are written entirely in Java and, therefore, are platform-independent.
- The term lightweight is used to describe such elements

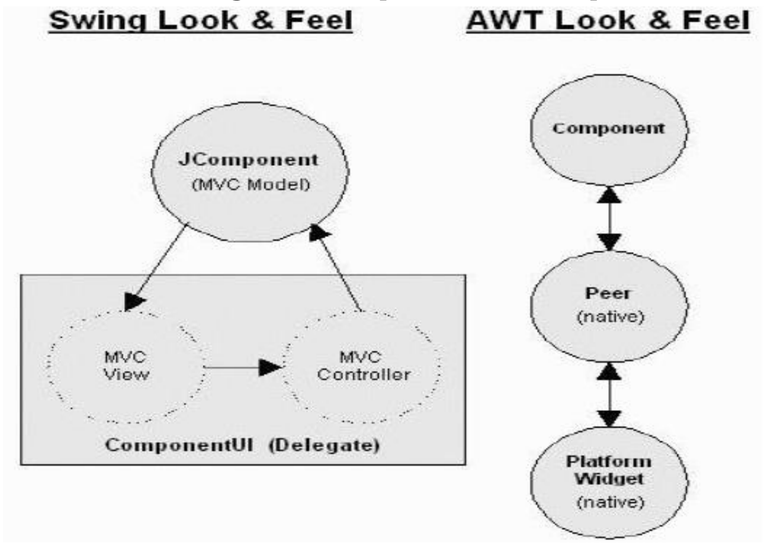
The Swing component are defined in javax.swing

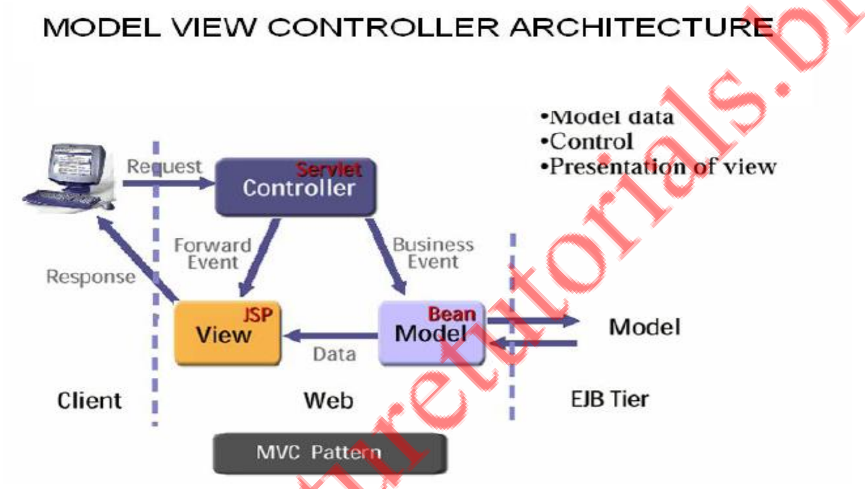
- AbstractButton: Abstract superclass for Swing buttons.
- .ButtonGroup: Encapsulates a mutually exclusive set of buttons.
- ImageIcon: Encapsulates an icon.
- JApplet: The Swing version of Applet.
- JButton: The Swing push button class.
- JCheckBox: The Swing check box class.
- JComboBox : Encapsulates a combo box (an combination of a drop-down list and text field).
- JLabel: The Swing version of a label.
- JRadioButton: The Swing version of a radio button.
- JScrollPane: Encapsulates a scrollable window.
- JTabbedPane: Encapsulates a tabbed window.
- JTable: Encapsulates a table-based control.
- JTextField: The Swing version of a text field.
- JTree: Encapsulates a tree-based control

Limitations of AWT

- AWT supports limited number of GUI components.
- AWT components are heavy weight components.
- AWT components are developed by using platform specific code.
- AWT components behaves differently in different operating systems.
- AWT component is converted by the native code of the operating system.
- Lowest Common Denominator
    - If not available natively on one Java platform, not available on any Java platform
- Simple Component Set
- Components Peer-Based

   - Platform controls component appearance
   - Inconsistencies in implementations
- Interfacing to native platform error-prone





Model

- Model consists of data and the functions that operate on data
- Java bean that we use to store data is a model component
- EJB can also be used as a model component

View

- View is the front end that user interact.
- View can be a
    - HTML
    - JSP
    - Struts ActionForm

Controller

Controller component responsibilities

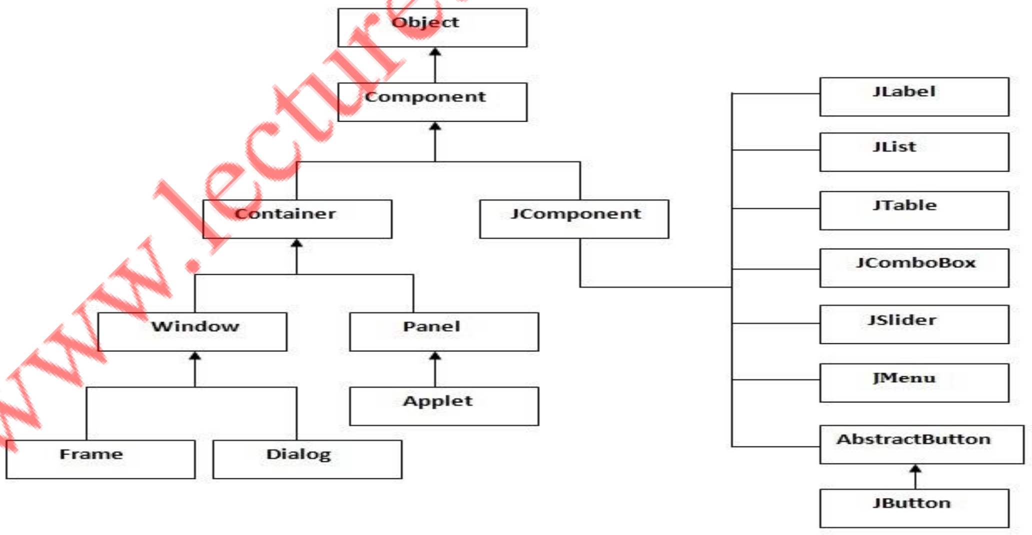
- Receive request from client
- Map request to specific business operation
- Determine the view to display based on the result of the business operation

Difference between AWT and Swing

There are many differences between java awt and swing that are given below.

Java AWT	Java Swing
AWT components are **platform-dependent**.	
AWT components are **heavyweight**.	Swing components are **lightweight**.
AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
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 swings



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

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 in main () method

Let's see a simple swing example where we are 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();
JButton b=new JButton("click");//creating instance of JButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height
f.add(b);
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

We can also write all the codes of creating JFrame, JButton and method call inside the java constructor.

```
import javax.swing.*;
public class Simple {
JFrame f;
Simple(){
f=new JFrame();
JButton b=new JButton("click");
b.setBounds(130,100,100, 40);
f.add(b);
f.setSize(400,500);//400 width and 500 height
f.setLayout(null);
f.setVisible(true);
}
public static void main(String[] args) {
new Simple();
}
}
```

Simple example of Swing by inheritance

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

```
import javax.swing.*;
public class Simple2 extends JFrame{
JFrame f;
Simple2(){
JButton b=new JButton("click");
b.setBounds(130,100,100, 40);
add(b);
setSize(400,500);
setLayout(null);
setVisible(true);
}
public static void main(String[] args) {
new Simple2();
}
}
```

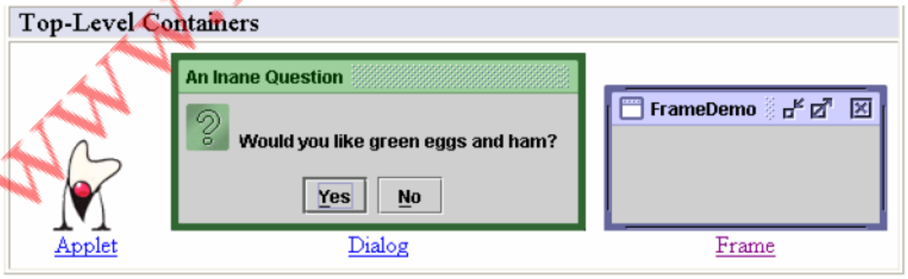
Components

- Container
  - JComponent
    - AbstractButton
  - JButton
  - JMenuItem
    - JCheckBoxMenuItem
    - JMenu
    - JRadioButtonMenuItem
  - JToggleButton
    - JCheckBox
    - JRadioButton
- JComponent
  - JTextComponent
  - JTextArea
  - JTextField
  - JPasswordField
  - JTextPane
  - JHTMLPane

- JComponent
  - JTextComponent
    - JTextArea
    - JTextField
      - ✓ JPasswordField
    - JTextPane
      - ✓ JHTMLPane

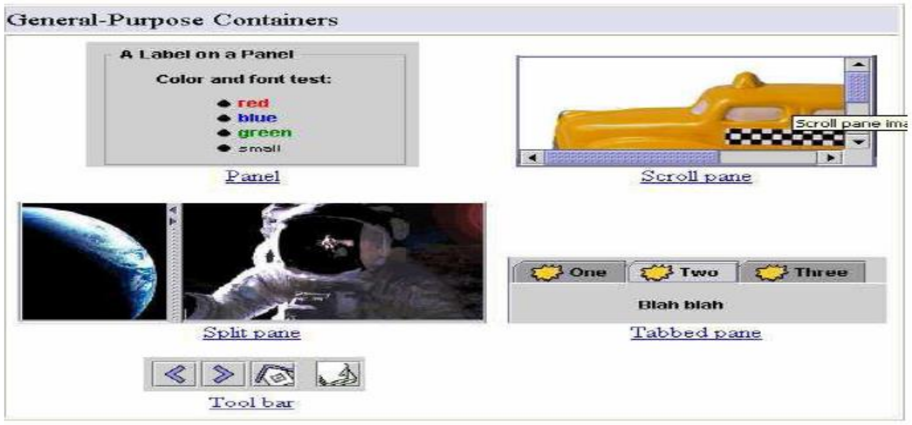
Containers

- Top-Level Containers
- The components at the top of any Swing containment hierarchy



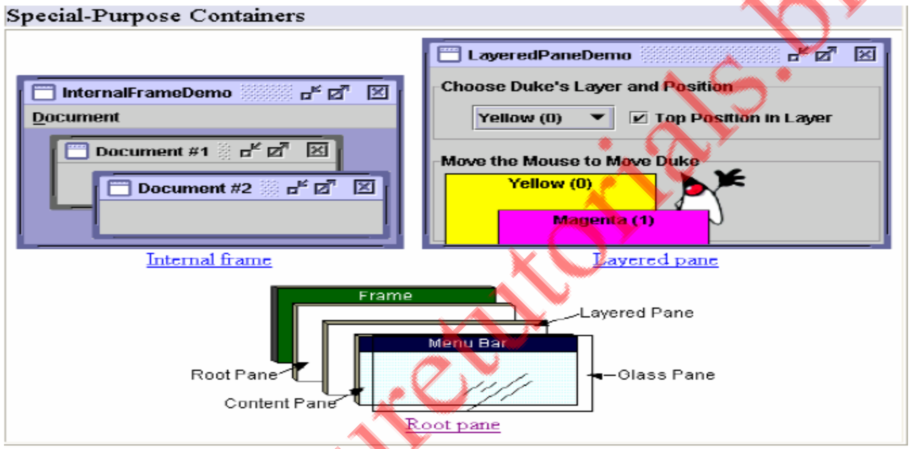
General Purpose Containers

Intermediate containers that can be used under many different circumstances.



Special Purpose Container

Intermediate containers that play specific roles in the UI.



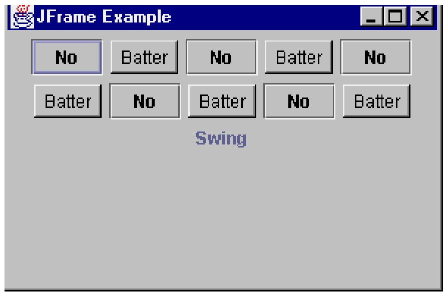
Exploring swing- JApplet,

If using Swing components in an applet, subclass JApplet, not Applet

- JApplet is a subclass of Applet
- Sets up special internal component event handling, among other things
- Can have a JMenuBar
- Default LayoutManager is BorderLayout

JFrame public class FrameTest {
public static void main (String args[]) {
JFrame f = new JFrame ("JFrame Example");
Container c = f.getContentPane();

```
c.setLayout (new FlowLayout());
for (int i = 0; i < 5; i++) {
c.add (new JButton ("No"));
c.add (new Button ("Batter"));
}
c.add (new JLabel ("Swing"));
f.setSize (300, 200);
f.show();
}
}
```



JComponent

•JComponent supports the following components.

- JComponent
  - JComboBox
  - JLabel
  - JList
  - JMenuBar
  - JPanel
  - JPopupMenu
  - JScrollBar
  - JScrollPane
  - JTextComponent
- JTextArea
- JTextField
  - JPasswordField
- JTextPane
  - JHTMLPane

Icons and Labels

- In Swing, icons are encapsulated by the ImageIcon class, which paints an icon from an image.
- constructors are:
  - ImageIcon(String filename)
  - ImageIcon(URL url)
- The ImageIcon class implements the Icon interface that declares the methods

        - int getIconHeight( )
        - int getIconWidth( )
        - void paintIcon(Component comp,Graphics g,int x, int y)
- Swing labels are instances of the JLabel class, which extends JComponent.
- It can display text and/or an icon.
- Constructors are:
    - JLabel(Icon i)
    - Label(String s)
    - JLabel(String s, Icon i, int align)
- Here, s and i are the text and icon used for the label. The align argument is either LEFT, RIGHT, or CENTER. These constants are defined in the SwingConstants interface,
- Methods are:
    - Icon getIcon( )
    - String getText( )
    - void setIcon(Icon i)
    - void setText(String s)
        - Here, i and s are the icon and text, respectively.

Text fields

- The Swing text field is encapsulated by the JTextComponent class, which extendsJComponent.
- It provides functionality that is common to Swing text components.
- One of its subclasses is JTextField, which allows you to edit one line of text.
- Constructors are:
    - JTextField( )
    - JTextField(int cols)
    - JTextField(String s, int cols)
    - JTextField(String s)
        - Here, s is the string to be presented, and cols is the number of columns in the text field

Buttons

- Swing buttons provide features that are not found in the Button class defined by the AWT.
- Swing buttons are subclasses of the AbstractButton class, which extends JComponent.
- AbstractButton contains many methods that allow you to control the behavior of buttons, check boxes, and radio buttons.
- Methods are:
    - void setDisabledIcon(Icon di)
    - void setPressedIcon(Icon pi)
    - void setSelectedIcon(Icon si)
    - void setRolloverIcon(Icon ri)
        - Here, di, pi, si, and ri are the icons to be used for these different conditions.
- The text associated with a button can be read and written via the following methods:

    - String getText( )
    - void setText(String s)
        - Here, s is the text to be associated with the button.

JButton

- The JButton class provides the functionality of a push button.
- JButton allows an icon, a string, or both to be associated with the push button.
- Some of its constructors are :
    - JButton(Icon i)
    - JButton(String s)
    - JButton(String s, Icon i)
        - Here, s and i are the string and icon used for the button.

Check boxes

- The JCheckBox class, which provides the functionality of a check box, is a concrete implementation of AbstractButton.
- Some of its constructors are shown here:
    - JCheckBox(Icon i)
    - JCheckBox(Icon i, boolean state)
    - JCheckBox(String s)
    - JCheckBox(String s, boolean state)
    - JCheckBox(String s, Icon i)
    - JCheckBox(String s, Icon i, boolean state)
        - Here, i is the icon for the button. The text is specified by s. If state is true, the check box is initially selected.
        - Otherwise, it is not.
- The state of the check box can be changed via the following method:
    - void setSelected(boolean state)
        - Here, state is true if the check box should be checked.

Combo boxes

- Swing provides a *combo box* (a combination of a text field and a drop-down list) through the **JComboBox** class, which extends **JComponent**.
- A combo box normally displays one entry. However, it can also display a drop-down list that allows a user to select a different entry. You can also type your selection into the text field.
- Two of **JComboBox**'s constructors are :
    - JComboBox( )
    - JComboBox(Vector v)
        - Here, *v* is a vector that initializes the combo box.
- Items are added to the list of choices via the **addItem( )** method, whose signature is:
    - void addItem(Object obj)
        - Here, *obj* is the object to be added to the combo box.

Radio Buttons

- Radio buttons are supported by the JRadioButton class, which is a concrete implementation of AbstractButton.
- Some of its constructors are :
    - JRadioButton(Icon i)

- JRadioButton(Icon i, boolean state)
- JRadioButton(String s)
- JRadioButton(String s, boolean state)
- JRadioButton(String s, Icon i)
- JRadioButton(String s, Icon i, boolean state)
  - Here, i is the icon for the button.
  - the text is specified by s.
  - If state is true, the button is initially selected.
  - Otherwise, it is not.

➢ Elements are then added to the button group via the following method:
- void add(AbstractButton ab)
  - Here, ab is a reference to the button to be added to the group.

Tabbed Panes

➢ A tabbed pane is a component that appears as a group of folders in a file cabinet.
➢ Each folder has a title. When a user selects a folder, its contents become visible. Only one of the folders may be selected at a time.
➢ Tabbed panes are commonly used for setting configuration options.
➢ Tabbed panes are encapsulated by the JTabbedPane class, which extends JComponent. We will use its default constructor. Tabs are defined via the following method:
- void addTab(String str, Component comp)
  - Here, str is the title for the tab, and
  - comp is the component that should be added to the tab.
  - Typically, a JPanel or a subclass of it is added.

➢ The general procedure to use a tabbed pane in an applet is outlined here:
- Create a JTabbedPane object.

➢ Call addTab( ) to add a tab to the pane.
➢ (The arguments to this method define the title of the tab and the component it contains.)
➢ Repeat step 2 for each tab.
➢ Add the tabbed pane to the content pane of the applet.

Scroll Panes

➢ A scroll pane is a component that presents a rectangular area in which a component may be viewed. Horizontal and/or vertical scroll bars may be provided if necessary.
➢ Scroll panes are implemented in Swing by the JScrollPane class, which extends JComponent. Some of its constructors are :
- JScrollPane(Component comp)
- JScrollPane(int vsb, int hsb)
- JScrollPane(Component comp, int vsb, int hsb)
  - ✓ Here, comp is the component to be added to the scroll pane.
  - ✓ vsb and hsb are int constants that define when vertical and horizontal scroll bars for this scroll pane are shown.
  - ✓ These constants are defined by the ScrollPaneConstants interface.
  - ✓ HORIZONTAL_SCROLLBAR_ALWAYS
  - ✓ .HORIZONTAL_SCROLLBAR_AS_NEEDED

        - ✓ VERTICAL_SCROLLBAR_ALWAYS
        - ✓ .VERTICAL_SCROLLBAR_AS_NEEDED
- ➢ Here are the steps to follow to use a scroll pane in an applet:
    - ❖ Create a JComponent object.
    - ❖ Create a JScrollPane object.
    - ❖ The arguments to the constructor specify thecomponent and the policies for vertical and horizontal scroll bars.
    - ❖ Add the scroll pane to the content pane of the applet.

Trees

- ➢ Data Model - TreeModel
    - ❖ default: DefaultTreeModel
    - ❖ getChild, getChildCount, getIndexOfChild, getRoot, isLeaf
- ➢ Selection Model - TreeSelectionModel
- ➢ View - TreeCellRenderer
    - ❖ getTreeCellRendererComponent
- ➢ Node – DefaultMutableTreeNode

Tables

- ➢ A table is a component that displays rows and columns of data. You can drag the cursor on column boundaries to resize columns. You can also drag a column to a new position.
- ➢ Tables are implemented by the JTable class, which extends JComponent.
- ➢ One of its constructors is :
    - ❖ JTable(Object data[ ][ ], Object colHeads[ ])
        - ▪ Here, data is a two-dimensional array of the information to be presented, and
        - ▪ colHeads is a one-dimensional array with the column headings.
- ➢ Here are the steps for using a table in an applet:
    - ❖ Create a JTable object.
    - ❖ Create a JScrollPane object.
    - ❖ The arguments to the constructor specify the table and the policies for vertical and horizontal scroll barsAdd the table to the scroll pane.
    - ❖ Add the scroll pane to the content pane of the applet