List of Practical's

❑❑❑

UNIT - I

CHAPTER 1

Swing

Syllabus Topics

Need for swing components, Difference between AWT and swing, Components hierarchy, Panes.
Swing components : Jlabel, JTextField and JPasswordField, JTextArea, JButton, JCheckBox, JRadioButton, JComboBox and JList.

Syllabus Topic : Need for Swing Components

1.1 Need for Swing Components

Q. What is Swing ? Explain the need of Swing.
Q. Explain the role of Swing components in JAVA.
Q. Explain the features of Swing Components.

☞ **Swing :** Swing is a part the Java Foundation Classes (JFC). It is an extension library to AWT.

- GUI programming is simple and elegant with Swing. Java Swing provides platform-independent and lightweight components which is not the case with AWT.
- The javax.swing package provides classes for java swing API such as JLabel, JButton, JTextField, JTextArea, JColorChooser, JRadioButton, JCheckbox, JMenu, etc.

☞ **Need of Swing**

1. It has the uniform look and feel across platforms is an advantage.
2. It has ability to replace the objects on-the-fly.
3. It is Compatible for both standalone/GUI applications, Applets and Servlets.
4. Employs model/view design pattern.
5. It has the pluggable look and feel.
6. Its Light weight components.

Swing Features

1. Light Weight

Swing components are light weight meaning they are independent of native Operating System's API as Swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.

2. Rich Controls

It has rich set of advanced control such as Tree, TabbedPane, slider, colorpicker, and table controls.

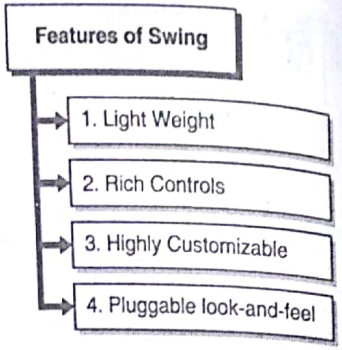


Fig. C1.1 : Features of swing

3. Highly Customizable

Swing controls can be easily customized as its visual appearance is independent of internal representation.

4. Pluggable look-and-feel

If your application is designed using SWING ,then its look and feel can be changed at run-time, based on available values.

1.2 Java Foundation Class (JFC)

Q. Explain the role of JFC in java.
Q. Write a short note on 'JFC'.

- The JFC includes a set of classes and interfaces that support the development of Graphical User Interface (GUI) in java based applications.
- JFC can be used to develop platform-independent user-interfaces for applets and applications.

JFC API (Application Program Interface)

The JFC APIs are :

1. Abstract Window Toolkit(AWT) : It is the preliminary package which consists of basic controls and layouts for designing and developing GUIs for the java platform. It also has the features of colors, graphics, images, events, *listeners*, layout managers, fonts, etc.

2. Swing : It consists of set of lightweight components build on top of the AWT component classes that has pluggable look and feel.

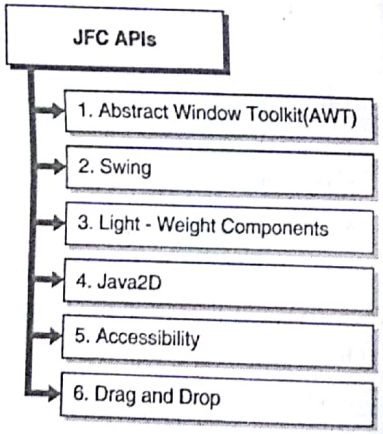


Fig. C1.2 : JFC APIs

3. Light - Weight Components : These are components that do not have a window of its own, that are drawn directly on the containers window.

4. Java2D : Java2D API include various painting styles and extended graphic features. It also support drawing complex shapes and controlling the rendering process. It supports enhanced font and drawing capabilities.

5. Accessibility : The Accessibility API allows assistive technologies to be included within a java applications. Technologies like screen readers, screen magnifiers, image and speech-recognition applications can be integrated with the java application.

6. Drag and Drop : Drag and Drop API provides interoperability between applications. Interoperability helps to move data between programs created with the different languages. It helps program created in another language to exchange data with java programs.

java.awt package

The AWT, Java2D, Drag and Drop APIs are contained in the core java java.awt package.

1. AWT APIs packages

(i) **java.awt.event :** It includes classes that support user events.

(ii) **java.awt.image :** It includes various classes for different color models and image filters for image processing.

(iii) **java.awt.peer :** It includes various classes that reduces the gap between the AWT classes and its corresponding implementations on the native operating system.

2. Java2D APIs packages

(i) **java.awt.color :** It includes various classes for the colors.

(ii) **java.awt.font :** It provides enhanced support for fonts.

(iii) **java.awt.geom :** It supports two-dimensional geometry operations.

(iv) **java.awt.print :** It includes various classes and interfaces for printing service.

3. Drag and Drop APIs packages

(i) **java.awt.datatransfer :** Include various classes that support interoperability i.e. data transfer between applications.

(ii) **java.awt.dnd :** Include various classes and interfaces for drag and drop operations.

4. javax.swing package

The Swing and Accessibility APIs are contained in the javax.swing package.

(i) **javax.swing :** It is the most basic swing package that includes various classes for the basic swing controls.

(ii) **javax.swing.event :** It includes various classes for event handling of swing components, that does not included in awt event handling.

(iii) **javax.swing.table :** It can be used to implement JTable component, that can be used to display the data in two dimensional table format in rows and columns.

(iv) **javax.swing.tree :** It helps to design the JTree component, which shows hierarchy of data elements.

(v) **javax.swing.colorchooser :** It is used to access various colors and styles for graphics and drawing by JColorChooser objects.

(vi) **javax.swing.filechooser :** Used to access and store the files using JfileChooser objects.

5. Swing API Components

Each Swing Component :

(i) Can be drawn by itself.

Example : If Jlabel object is created and added into a container, the label gets fully rendered on the screen.

(ii) Provides support for GUI interactions.

Example : JButton generates the events when the user clicks on it.

Syllabus Topic : Difference between AWT and Swing

1.3 Difference between AWT and Swing

Q. Describe Abstract Window Toolkit (AWT) and Swing.

- To develop a Java program with Graphical User Interface (GUI) there are two basic set of components : Abstract Window Toolkit (AWT) and Swing.

➜ 1. Abstract Window Toolkit (AWT)

AWT is a basic set of GUI library for developing stand-alone applications and/or applets in java programming that provides the connection between the application and native operating system.

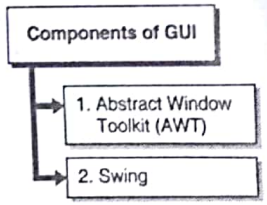


Fig. C1.3 : Components of GUI

☞ Features of AWT

AWT features are :

- It includes various set of basic user interface components.
- AWT helps event-handling model for handling the user interactions.
- It Supports graphics and imaging tools, such as shape, color, and font classes.
- It includes Layout managers, for laying out various controls on the top level container with alignment procedures.

➜ 2. Swing

- Swing consists of huge set of GUI components that build on AWT technology with pluggable look and feel.
- Swing components are pure Java versions of the corresponding AWT component set (like Button, Scrollbar, Label, etc.).
- Includes a set of higher-level components (such as tree view, list box, and tabbed panes) for creating an interactive user interfaces.
- Provides a Pluggable Look and Feel.

☞ Difference between AWT and Swing

Q. Discuss the difference between AWT components and swing components in java.

Sr. No.	AWT	Swing
1.	AWT is Abstract windows toolkit.	Another well know name for swing is JFC's (Java Foundation classes).
2.	Here AWT components are Heavy weight component.	Swing is light weight component because swing components are developed on top of the corresponding AWT components.
3.	java.awt package is needed for AWT components.	javax.swing package is needed for Swing components.
4.	AWT components are platform dependent.	Swing components are made in purely java and they are platform independent.
5.	AWT have single look and feel.	Swing provides different pluggable look and feel
6.	AWT does not support many advanced features that swing has	Swing support many advanced features that helps to design interactive user interfaces like : – **JTable :** Used to display data in two dimensional formats in rows and columns. – **JTabbedPane :** Tabs can be used to place more than one set of user interfaces within a single window with different tabs. – **JTree :** Used to create hierarchical structure for easier access of elements.
7.	AWT have 21 peers ; wherein one for each control and one for the dialog box. A peer is provided by the operating system, such as a button object, label, entry field.	Swing provides only one peer, the operating system's window object. All buttons, labels entry fields, etc. are basically drawn by the Swing package on the drawing surface provided by the window.
8.	AWT is a thin layer of code on top of the Operating System.	Swing is larger than AWT with advance functionality.
9.	AWT components are thread-safe	Swing components are not thread-safe.

Syllabus Topic : Components Hierarchy

1.4 Swing Components Hierarchy

Q. Discuss Swing Components hierarchy in java.

To develop a Java program with Graphical User Interface (GUI) there are two basic sets of components : **Abstract Window Toolkit (AWT)** and **Swing.**

1. AWT

AWT is a basic set of GUI library for developing stand-alone applications and/or applets in java programming that provides the connection between the application and native operating system.

2. Swing

Swing consists of huge set of GUI components that build on AWT technology with pluggable look and feel. The hierarchy of java swing API (Application Programming Interface) is given in Fig. 1.4.1.

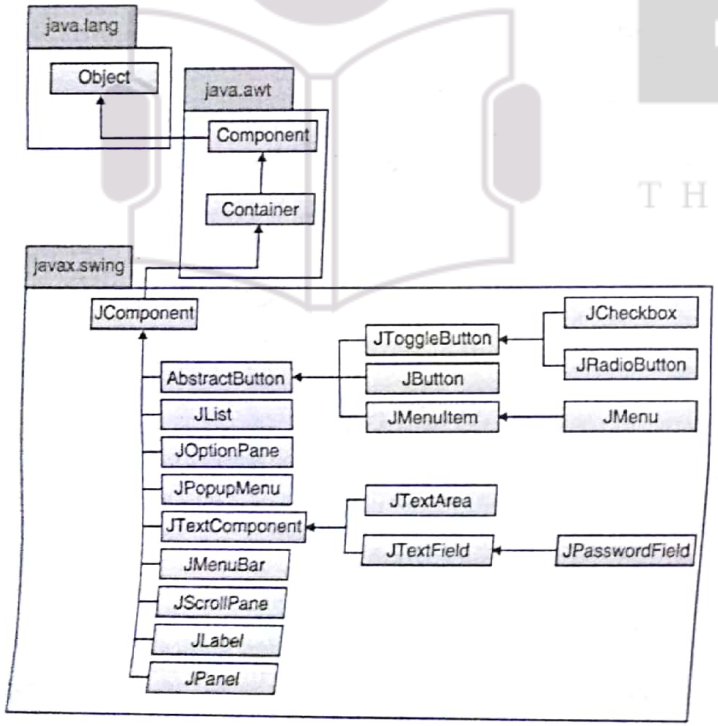


Fig. 1.4.1 : Java's Swing Components Hierarchy.

JComponent class

- JComponent is the abstract superclass of all swing components like label (JLabel), button(JButton), checkbox(JCheckbox) etc.
- JComponent is the direct subclass of java.awt.Container class. Thus all swing components are containers
- java.awt.Container is the direct subclass of java.awt.Component class.
- JComponent class inherit various methods that provides the common behaviour among all the user interface components.

JComponent methods

1. **void addxxxListener(xxxListener object) :** used to register the listener for a particular event.
   - The listener can be used to handle the user-initiated events related to a specific component.
   - Different listeners are used to for different event handling.
2. **void repaint ( ) :** used to repaint the swing components. The repaint request is sent to the AWT.

   void repaint (long msec) : The repaint operation begins after msec (milliseconds).

   void repaint (int x, int y, int height, int width) : used to repaint a particular area of the component not all.

   void repaint (long msec, int x, int y, int height, int width) : used to repaint a particular area of the component, that begins after msec (milliseconds).
3. **void setBackground (Color c) :** used to set the background color.
4. **void setForeground (Color c) :** used to set the foreground color.
5. **void setBorder (Border b) :** used to add border to the component.
   - Border object can be used to provide an outer border for any component with an added title.
   - BorderFactory can be used to create various border styles.
6. **void setSize (Dimension d) :** Sets the size of the component in pixels.

   Dimension : Dimension object can be created with two int parameters for width and height.
7. **void setFont (Font f) :** used to set the font of text data in a component.
8. **void setEnabled (boolean b)**

   setEnabled (true) : Enables the component for user input and to generate events.

   If the argument is false, the component does not respond to events.
9. **void setVisible (boolean b) :** If the argument passed is true the component will be visible else hidden.

10. **void update (Graphics context) :** Update method calls paint() method to repaint the component.
11. **void setDoubleBuffered (Boolean b)**

    setDoubleBuffered (true) : Stores the component in an off-screen buffer are before it gets displayed on the screen. It can be used to eliminate the screen flicker.
12. **void setPreferredSize (Dimension d) :** Can be used to specify the ideal size of the component.

☞ **Jcomponent classes**

1. **AbstractButton**

   AbstractButton is the abstract super class for all the swing button component (like buttons, checkbox and radio button) and menu component that extends JComponent.

   (i) **JToggleButton :** There are two types of Jtoogle Button :

   (a) **JCheckBox :** JCheckBox can be used to create check box button. The JCheckBox object can be initialized with a label text, an icon and selected state.

   (b) **JRadioButton :** JRadioButton can be used to create radio button. The JRadioButton object can be initialized with a label text, an icon and selected state.

   (ii) **JButton :** JButton can be used to create a push button. A button can display text, an icon or both on its face.

   (iii) **JMenu and JmenuItem :** The Menu can create a pull-down menu component that can be deployed from a menu bar. Menu item represent individual menu element that user can select
2. **JList :** A JList can be used to display a group of items in one or more columns.
3. **JOptionPane :** JOptionPane can be used to create and show three types of dialog boxes as message box, confirmation and input dialogs.
4. **JPopupMenu :** A popup menu is a control that gets displayed while the user click the right mouse button.
5. **Edit Controls :** Edit controls are user entry component like JTextField, JTextArea and JPasswordField
6. **JMenuBar :** The JMenuBar can be used to create menu bar, which can display a set of menus
7. **JScrollPane :** JScrollBar is used for create scrollbar for scrolling various elements and displays
8. **JLabel :** Swing labels are used todisplay textual data for information purpose.
9. **JPanel :** JPanel is an intermediate swing container, used for grouping of components together before placing them in a top level container.

10. **JtextComponents :** It represent the user entry component in swing, JTextComponent is the abstract parent class of all the swing text objects.
11. **JTextField :** JTextField can be used to create a single line text control.
12. **JPasswordField :** JPasswordField is used to create a password entry control for accepting the password details. It is the sub-class of the JTextField, that the text entered will be replaced by an echo character.
13. **JtextArea :** A TextArea is used to create a multi-line area edit control where user can input textual data in more than one row. JTextArea object has to be placed inside a JScrollPane object to make the scrolling option.

Syllabus Topic : Panes

1.5 Panes

- Lists, buttons, panels, and windows are the Component with basic user interface object and placed in it is found in all Java applications. Firstly container is used and then components are placed in it.
- A container is basically a component that holds and manages other components. Containers display components using a layout manager.
- Containers include JWindow, JFrame, JDialog, JApplet. JApplet is the root container for Swing applets and JFrame is the root container for a standalone GUI application.
- Once a **root** container is created, next step is to add **components** and other **containers** to it. Each top-level container consists of the following panes :

1. **Root pane :** We use a root pane to paint over multiple components or to catch input events. This is an intermediate container that manages the layered pane, content pane, and glass pane. It can also manage an optional menu bar.
2. **Layered pane :** It contains the content pane and the optional menu bar. It enables us to add pop-up menus to applications. It also provides six functional layers in which we can place the components we add to it.
3. **Content pane :** It covers the visible section of the JFrame or JWindow and we use it to add components to the display area. It holds all the visible components of the root pane, except the menu bar. Java automatically creates a content pane when we create a JFrame or JWindow but we can create our own content pane.
4. **Glass pane :** This is invisible by default but we can make it visible. When glass pane is visible, it covers the components of the content pane, stops all input events from reading these components, and can paint over an existing area containing one or more components.

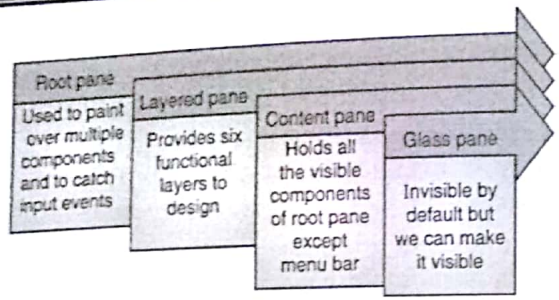


Fig. 1.5.1 : Various panes in swing top level container

Syllabus Topic : Swing Components - JLabel, JTextField, JPasswordField, JTextArea, JButton, JCheckBox, JRadioButton, JComboBox, JList

1.6 Swing Components

- As discussed in above topic, a component is an independent visual control. Swing Framework contains a large set of components which provide rich functionalities and allow for customization.
- Java's all component classes are derived from JComponent class. All these components are lightweight components. This class provides some common functionality like pluggable look and feel, support for accessibility, drag and drop, layout, etc.
- A container holds a group of components. It provides a area where a component can be managed and displayed.
- **Containers are of two types :**

Sr. No.	Top level Containers	Lightweight Containers
1.	It inherits Component and Container of AWT.	It is a general purpose container.
2.	It cannot be contained within other containers.	It inherits JComponent class.
3.	Heavyweight	It can be used to organize related components together.
4.	**Example :** JFrame, JDialog, JApplet	**Example :** JPanel

1.6.1 JLabel

- Swing labels are used to display textual data for informational purpose.
- It can display text, image or both.
- JLabel object does not generate events.

☞ **Constructors**

1. JLabel( )
2. JLabel(Icon image)
3. JLabel(Icon image, int halign)
4. JLabel(String text)
5. JLabel(String text, int halign)
6. JLabel(String text, Icon image, int halign)

Create a JLabel object with a face text, image icon and specific horizontal alignment.

- The horizontal alignment can be :
  - (i) SwingConstants.LEFT
  - (ii) SwingConstants.CENTER
  - (iii) SwingConstants.RIGHT
  - (iv) SwingConstants.LEADING
  - (v) SwingConstants.TRAILING

☞ **Alignment Methods**

(1) int getHorizontalAlignment( )

```
void setHorizontalAlignment( int halign)
```

Used to get and set the horizontal alignment of label object.

(2) int getVerticalAlignment( )

```
void setVerticalAlignment( int valign)
```

Used to get and set the vertical alignment of label object. It can be :

(i) SwingConstants.TOP
(ii) SwingConstants.CENTER
(iii) SwingConstants.BOTTOM

☞ **Icon and Text method**

(1) Icon getIcon( )

```
void setIcon( Icon i)
```

Used to return and change the default icon for the label.

(2) String getText( )

```
void setText( String text )
```

Used to return and change the text display for the label.

1.6.2 JTextComponent, JTextArea, JPasswordField and JTextField

Q. Describe the three types of text controls in Swing.

(A) Text Component (JTextComponent)

- It represents the user entry component in swing : JTextComponent is the abstract parent class of all the swing text objects.
- It is defined in the package javax.swing.text.

☞ **JTextComponent subclasses**

1. JTextField
2. JTextArea
3. JPasswordField

☞ **Text Methods**

1. **String getText( ) :** returns the text contained in the JTextComponent.
2. **void setText (String text) :** change the text contained in the JTextComponent.

☞ **Copy, Cut and Paste Methods**

1. **void copy( ) :** copy the selected text into the clipboard.
2. **void cut( ) :** delete the selected text and store into clipboard.
3. **void paste( ) :** append the system clipboard content into the JTextComponent

☞ **I/O Method**

1. **void read(Reader inputstream, Object description) :** initializes the JTextComponent object by the data read from inputstream.
2. **void write(Writer outputstream) :** writes the JTextComponent content to the ouputstream.

☞ **JTextComponent Properties Method**

1. **char getFocusAccelerator( )**

void setFocusAccelerator(char key)

Used to return and set the accelerator key for the text component.

2. **boolean isEditable( )**

void setEditable(boolean editable)

Used to return or change the editable state of the JTextComponent object.

(B) JTextField

Q. Write a short note on JTextField.

☞ **Use :** JTextField can be used to create a single line text control.

☞ **JTextField() Constructor**

1. JTextField()
2. JTextField(int columns)
3. JTextField(String text)
4. JTextField(String text, int columns)
5. JTextField(Document doc, text, int columns)

During the creation of a text field particular number of columns, an initial data to be displayed within the text field and a document model can be used.

(C) JPasswordField

Q. Write a short note on JPasswordField.

☞ **Use :** JPasswordField is used to create a password entry control for accepting the password details. It is the sub-class of the JTextField that the text entered will be replaced by an echo character.

☞ **JPasswordField() Constructor**

1. JPasswordField( )
2. JPasswordField(int columns)
3. JPasswordField(String text)
4. JPasswordField(String text, int columns)
5. JPasswordField(Document doc, String txt, int columns)

During the creation of a password field particular number of columns, an initial password and a document model can be used.

☞ **Echo Character Method**

1. **boolean echoCharIsSet( ) :** Returns true if the echo character is set else false.
2. **char getEchoChar( ) :** Gets the character that is to be used for echoing.
3. **void setEchoChar(char echocharacter) :** Sets the echo character for the text field.
4. **char[] getPassword( ) :** Returns the text in the JPasswordField object.

(D) JTextArea

Q. Write a short note on JTextArea.

☞ **Use :** A TextArea is used to create a multi-line area edit control where user can input textual data in more than one row.

– JTextArea object has to be placed inside a JScrollPane object to make the scrolling option.

☞ **JTextArea( ) constructors**

1. JTextArea( )
2. JTextArea(int rows, int columns)
3. JTextArea(String text)
4. JTextArea(String text, int rows, int columns)
5. JTextArea(Document doc)
6. JTextArea(Document doc, String text, int rows, int columns)

During the creation of a text-area number of rows and columns, an initial data to be displayed and a document model can be used.

☞ **Method to add or replace text**

1. **void append(String str) :** Appends the text at the end of the document.
2. **void insert(String str, int pos) :** Inserts the text at the specified position.

☞ **Method for row column and tab**

1. **int getColumns()**

void setColumns(int columns)

Used to return and set the number of columns for the TextArea.

2. **int getRows()**

void setRows(int rows)

Used to return and set the number of rows for the TextArea.

3. **int getTabSize()**

void setTabSize(int size)

Used to return and set the tab size.

Program 1.6.1 : *Write a program to create a login form (for entering the user name and password).On click of "Ok" button check the username and password is correct or not and report with proper messages.*

Soln.: :

Program creating a login form

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



```
import javax.swing.*;
public class login extends JFrame implements ActionListener
{
   JTextField jtf1;
   JPasswordField jpf1;
   JLabel lbl, lb2;
   JButton b1;
   public  login(String title)
   {
         super(title);
         Container con = getContentPane();
         lbl = new JLabel(" User Name ");
         jtf1 = new JTextField(10);
         lb2 = new JLabel(" Password ");
         jpf1 = new JPasswordField(10);
         b1 = new JButton(" OK ");
         b1.addActionListener(this);

         JPanel p = new JPanel();
         p.add(lbl);
         p.add(jtf1);
         p.add(lb2);
         p.add(jpf1);
         p.add(b1);
         con.add(p);
   }
   public void actionPerformed(ActionEvent ae)
   {
         JButton bb=(JButton)ae.getSource();
         if(bb.equals(b1))
         {
            String name=jtf1.getText();
            String pass=String.valueOf(jpf1.getPassword());
            if(name.equals("mumbai") && pass.equals("love"))
            JOptionPane.showMessageDialog(this, "Sucessful login");
            else
            JOptionPane.showMessageDialog(this, "Invalid user detail");
            jtf1.setText("");
```

```
            jpf1.setText("");
        }
    }
public static void main(String args[])
    {
        login ob =  new login("password eg");
        ob.setSize(500,200);
        ob.setVisible(true);
        ob.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

1.6.3 JButton, JCheckBox and JRadioButton

(A) AbstractButton class

Q. Explain the function of AbstractButton.

- AbstractButton is the abstract super class for all the swing button component (like buttons, checkbox and radio button) and menu component that extends JComponent.

☞ **Difference between the swing buttons and java.awt.Button are**

- The swing button can be used to display both icons and text on its face but AWT buttons allow only textual display.
- Different Icons can be displayed on the swing buttons, while it is disabled, pressed, selected or rollover.

☞ **Icon methods**

Method to set and get the Icons for default, disabled, pressed, selected and rollover for the AbstractButton are :

1. **void setIcon(Icon default)**

```
Icon getIcon()
```

2. **void setDisabledIcon(Icon di)**

```
Icon getDisabledIcon()
```

3. **void setPressedIcon(Icon pi)**

```
Icon getPressedIcon()
```

4. **void setSelectedIcon(Icon si)**

```
Icon getSelectedIcon()
```


5. **void setRolloverIcon(Icon ri)**

```
Icon getRolloverIcon()
```

☞ **Methods for AbstractButton Properties**

1. **void setText(String label)**

```
String getText( )
```

Used to set or get the face content of the button.

2. **void setMargin(Insets newinsets)**

```
Insets getMargin()
```

Used to set and get the distance between the borders of the AbstractButton and its content.

☞ **Methods for AbstractButton States**

1. **void setEnabled(boolean enabled) :** Enables (or disables) the button.
2. **void setSelected(boolean selected) :** Set the state of the button.
3. **boolean isSelected() :** Returns the state of the button.

☞ **Mnemonic Method for AbstractButton**

A mnemonic is a key that when pressed with the Alt-key generate the same effect as clicking the button.

1. **void setMnemonic(char c)**

```
int getMnemonic( )
```

Used to set and get the keyboard mnemonic value.

☞ **Alignment Methods for AbstractButton States**

1. **void setHorizontalAlignment(int alignment)**

```
int getHorizontalAlignment()
```

Used to set and get the horizontal alignment of the icon and text.

2. **void setHorizontalTextPosition(int textPosition)**

```
int getHorizontalTextPosition()
```

Used to set and get the horizontal position of the text relative to the icon.

3. **void setVerticalAlignment(int alignment).**

```
int getVerticalAlignment()
```

Used to set and get the vertical alignment of the icon and text.

4. **void setVerticalTextPosition(int textPosition)**

int getVerticalTextPosition()

- Used to set and get the vertical position of the text relative to the icon.
- **Values for the horizontal alignment are :**
  - (i) SwingConstants.LEFT
  - (ii) SwingConstants.CENTER
  - (iii) SwingConstants.RIGHT
  - (iv) SwingConstants.LEADING
  - (v) SwingConstants.TRAILING
- **Values for the vertical alignment are :**
  - (i) SwingConstants.TOP
  - (ii) SwingConstants.CENTER
  - (iii) SwingConstants.BOTTOM

☞ **doClick( ) method**

1. **void doClick( ) :** It initiates that button being pressed and create an ActionEvent.
2. **void doClick (int milliseconds) :** Time indicate how long the button will appear as pressed.

☞ **ActionEvent**

- ActionEvent objects are generated when the swing button is pressed.

Program 1.6.2 : A program to create a button that shows an image and text on its face and makes it as the default button.Close the application on click of the button.

Soln.: :

Program performing an ActionEvent when button is pressed

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class buttonwithimage extends JFrame implements ActionListener
{
   JButton b1;
   public buttonwithimage(String title)
   {
         super(title);
         Container con = getContentPane();
         JRootPane jroot = getRootPane();
```



```
         ImageIcon img = new ImageIcon("bullet.gif");
         b1 = new JButton("quit", img);
         b1.addActionListener(this);
         JPanel p = new JPanel();
         p.add(b1);
         con.add(p);
         jroot.setDefaultButton(b1);
   }
   public void actionPerformed(ActionEvent ae)
   {
         if (ae.getActionCommand().equals("quit"))
         System.exit(0);
   }
   public static void main(String args[])
   {
    buttonwithimage bb = new buttonwithimage("button with image");
    bb.setSize(600,600);
    bb.setVisible(true);
    bb.setDefaultCloseOperation(EXIT_ON_CLOSE);
   }
}
```

(B) JCheckBox

Q. Discuss the use of JCheckbox in swing program.

☞ **Use :** JCheckBox can be used to create check box button for selecting a choice item. The JCheckBox object can be initialized with a label text, an icon and selected state.

☞ **JCheckBox ( ) constructors**

(i) JCheckBox( )
(ii) JCheckBox(Action action)
(iii) JCheckBox(Icon icon)
(iv) JCheckBox(Icon icon, boolean selected)
(v) JCheckBox(String text)
(vi) JCheckBox(String text, boolean selected)
(vii) JCheckBox(String text, Icon icon)
(viii) JCheckBox(String text, Icon icon, boolean selected)

Where,

(i) **action :** creates a checkbox object with the specified Action object.

(ii) **icon :** icon can display on the default box.

(iii) **text :** specifies the checkbox label.

(iv) **selected :** used to select or deselect the checkbox initially. By default the selecte state is false

- JCheckBox generates ItemEvent as the checkbox is selected or deselected.

Program 1.6.3 : Create an applet that shows three checkbox and a text box. On select of th check box display it selected or deselected in the text box.

Soln.: :

Program performing the ItemEvent using CheckBox

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
//<applet code="tcheckbox.class" width=500 height=500 ></applet>
public class tcheckbox extends JApplet implements ItemListener
{
   JCheckBox ch1=new JCheckBox("ch1");
   JCheckBox ch2=new JCheckBox("ch2");
   JCheckBox ch3=new JCheckBox("ch3");
   JTextField t1 = new JTextField(15);
   public void init()
   {
        Container con = getContentPane();
        con.setLayout(new FlowLayout());
        ch1.addItemListener(this);
        ch2.addItemListener(this);
        ch3.addItemListener(this);
        con.add(ch1);
        con.add(ch2);
        con.add(ch3);
        con.add(t1);
   }
   public void itemStateChanged(ItemEvent ie)
   {
        JCheckBox jcb = (JCheckBox) ie.getItem();
```



```
        if (jcb.isSelected())
            t1.setText(jcb.getText() + " selected");
        if (!jcb.isSelected())
            t1.setText(jcb.getText() + " de-selected");
   }
}
```

(C) JRadioButton

Q. Describe the use of JRadioButton.

☞ **Use :** JRadioButton can be used to create a radio button for displaying a mutually exclusive item for selection. The JRadioButton object can be initialized with a label text, an icon and selected state.

☞ **JRadioButton ( ) constructors**

(i) JRadioButton( )

(ii) JRadioButton(Action action)

(iii) JRadioButton(Icon icon)

(iv) JRadioButton(Icon icon, boolean selected)

(v) JRadioButton(String text)

(vi) JRadioButton(String text, boolean selected)

(vii) JRadioButton(String text, Icon icon)

(viii) JRadioButton(String text, Icon icon, boolean selected)

Where,

(i) **action :** Creates a radio button object with the Action object.

(ii) **icon :** Icon can display on the default radio box.

(iii) **text :** Specifies the radio button label.

(iv) **selected :** Used to select or deselect the radio button initially. By default the selected state is false.

- JRadioButton generates ItemEvent as the radio button is selected or deselected.
- JRadioButton objects can be grouped together using ButtonGroup object, to make the radio button mutually exclusive (only one has to be selected from a group).

☞ ButtonGroup

- A ButtonGroup object is used to group radio buttons together, that only one from the group can be selected at a time.
- Once a button is selected, ButtonGroup make sure that always one button is selected. It is not possible to unselect all the buttons.

☞ **ButtonGroup( ) constructor**

(i) **ButtonGroup( ) :** Creates a button group object.

☞ **Methods to add or remove the buttons**

(i) **void add (AbstractButton b) :** Used to add the AbstractButton to the ButtonGroup

(ii) **void remove(AbstractButton b) :** Used to remove the AbstractButton to the ButtonGroup

☞ **get Methods**

1. **int getButtonCount( ) :** Returns the number of buttons in the ButtonGroup.
2. **Enumeration getElements( ) :** Returns all buttons in the ButtonGroup object.

Program 1.6.4 : Create a frame that shows three radio button and a text box. On select of the radio button display its text value in the text box.

Soln.: :

Program performing the event when radio button is pressed

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
public class tradiobutton extends JFrame implements ItemListener
{
   JRadioButton r1,r2,r3;
   ButtonGroup bgrp;
   JTextField t1;
   public  tradiobutton(String title)
   {
         super(title);
         Container con = getContentPane();
         con.setLayout(new FlowLayout());
         r1 = new JRadioButton("r1");
         r2 = new JRadioButton("r2");
         r3 = new JRadioButton("r3");
         bgrp = new ButtonGroup();
         bgrp.add(r1);
         bgrp.add(r2);
         bgrp.add(r3);
         t1 = new JTextField(15);
         r1.addItemListener(this);
```



```
         r2.addItemListener(this);
         r3.addItemListener(this);
         con.add(r1);
         con.add(r2);
         con.add(r3);
         con.add(t1);
   }
   public void itemStateChanged(ItemEvent ie)
   {
         JRadioButton jrb = (JRadioButton) ie.getItem();
         t1.setText(jrb.getText());
   }
   public static void main(String args[])
   {
   tradiobutton ob =  new tradiobutton("example ");
   ob.setSize(600,600);
   ob.setVisible(true);
   ob.setDefaultCloseOperation(EXIT_ON_CLOSE);
   }
}
```

1.6.4 JComboBox and JList

(A) JComboBox

Q. Describe the concept of swing ComboBox control with code specification.

- JComboBox can be used to create a combo-box control that displays a drop-down list of choices from which the user can select an item.
- JComboBox object is a combination of a JList and JTextField.
- It can be used to list a small set of item list not a large set of values.
- JComboBox generates an ItemEvent when the selected combobox item changes and ActionEvent when the item gets clicked.

☞ **JComobBox ( ) constructors**

1. **JComboBox() :** Creates a JComboBox with a default data model.
2. **JComboBox(ComboBoxModel aModel) :** Creates a JComboBox with specific ComboBoxModel.

3. **JComboBox(Object[] items) :** Creates a JComboBox with the initial contents from the specified array.
4. **JComboBox(Vector items) :** Creates a JComboBox with the initial content from the specified Vector.

☞ **Methods to Add and Remove Item**

1. **void addItem(Object item) :** Adds an item to the item list.
2. **void removeItem(Object item) :** Removes an item from the item list.
3. **void insertItemAt(Object item, int index) :** Inserts an item into the item list at a particular index.
4. **void removeItemAt(int index) :** Removes the item from an Index.
5. **void removeAllItems() :** Removes all items from the item list.

☞ **Selection Methods**

1. **Object getItemAt(int index) :** Returns the list item from the specified index.
2. **int getSelectedIndex( ) :** Returns the index of the selected item.
3. **void setSelectedIndex(int index) :** Selects the item at the particular index value.
4. **Object getSelectedItem() :** Returns the currently selected item.
5. **void setSelectedItem(Object obj) :** Select the specific item.

Program 1.6.5 : Write a swing program to create a combobox and a text field. Load the Language names in the combobox. On select of the item from the combobox, display the item name in the text field.

Soln.: :

Program performing an event on ComboBox

```
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
public class combobox extends JFrame implements ItemListener
{
    JComboBox jcb;
    JTextField jtf;
    public combobox(String title)
    {
        super(title);
        Container con=getContentPane();
        con.setLayout(new FlowLayout());
```



```
        jcb = new JComboBox();
        jcb.addItem("Hindi");
        jcb.addItem("Marathi");
        jcb.addItem("English");
        jcb.setSelectedIndex(-1);
        jcb.addItemListener(this);
        jtf = new JTextField(20);

        con.add(jcb);
        con.add(jtf);
    }
    public void itemStateChanged(ItemEvent ie)
    {
        if (ie.getSource() == jcb)
        {
            Object s = (Object) jcb.getSelectedItem();
            jtf.setText("Selection is : " + s);
        }
    }
    public static void main(String args[])
    {
     combobox obj = new combobox(" Combo Box");
     obj.setSize(400,400);
     obj.setVisible(true);
     obj.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

(B) JList

Q. Discuss the various ways of using JList.

- A JList can be used to create a list box that can be used to display a group of items in one or more columns.
- Lists can have many items. Thus scroll panes can be attached with the JList to scroll all the options.

Initializing a List

Code to create a list with initial data :

```
list = new JList(data);          //data has type Object[]
```

Methods

1. **void setModel(ListModel) :** Set or get the model of the list. It is used to display the data in multiple columns
2. **ListModel getModel() :** List's model can be associated with the DefaultListModel object to dynamically change the list item.
3. **void setLayoutOrientation(int) :** Sets the way list cells are laid out.

   int getLayoutOrientation() : Gets the way list cells are laid out.

   The layout formats of JList are :

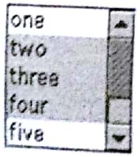
   (i) **VERTICAL :** is the default layout that display a single column of cells.

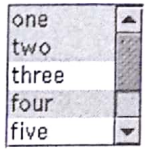
   (ii) **HORIZONTAL_WRAP :** content flowing horizontally then vertically.

   (iii) **VERTICAL_WRAP :** content flowing vertically then horizontally.
4. **int getSelectedIndex() :** Get information about the current selection.

ListSelectionModel

- ListSelectionModel can be used with a list box to manage its selection. By default, a list selection model allows any combination of items to be selected at a time.
- setSelectionMode() method can be used to change the list's selection mode. Three list selection modes are :

Sr. No.	Mode	Description
1.	SINGLE_SELECTION 	– Only one item can be selected at a time. – When user selects an item, previously selected item is deselected first.
2.	SINGLE_INTERVAL_SELECTION 	– Multiple, contiguous items can be selected. – When user begins a new selection range, previously selected items are deselected first.

Sr. No.	Mode	Description
3.	MULTIPLE_INTERVAL_SELECTION 	– Default selection format. – Any combination of items can be selected. – The user must explicitly deselect items.

Adding, Removing Items - JList

- DefaultListModel() and its method addElement() can be used to add an item into the JList.

Example

```
listModel = new DefaultListModel();
listModel.addElement("Mumbai");
listModel.addElement("Chennai");
list = new JList(listModel);
```

- remove() method of DefaultListModel can be used to delete an item from the JList.

Example : Delete the selected item from the JList.

```
public void actionPerformed(ActionEvent e) {
int index = list.getSelectedIndex();
listModel.remove(index); }
```

Program 1.6.6 : Write a program to create a list box with associated scrollpane object.

Soln.: :

Program of list box with scrollpane object

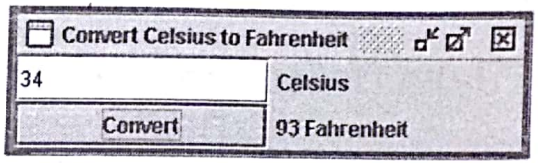
```
import java.awt.*;
import javax.swing.*;
public class listwithscrollpane extends JFrame
{
   JList list;
   JScrollPane jsp;
   public listwithscrollpane(String title)
```

```
{
        super(title);
        String[] lang = { "Marati","Hindi","English","French"};
        list = new JList(lang);
        jsp = new JScrollPane(list, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
        JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        JPanel p = new JPanel();
        p.add(jsp);
        getContentPane().add(p, BorderLayout.CENTER);
        setSize(400,400);
        setVisible(true);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
    public static void main(String args[])
    {
        listwithscrollpane fr = new listwithscrollpane(" list box with scrollpane");
    }
}
```

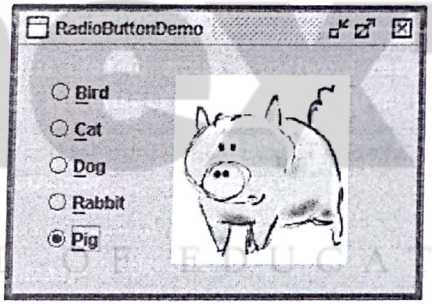
1.7 Programming Exercise

1. Design a Login Form titled as "My Login Form" containing User Name and password. The password should contain max 10 characters. When user name and password matches with preset values, a Greeting message should be displayed otherwise error message should be displayed. Maximum 3 attempts should be allowed for wrong passwords otherwise the application should end.
2. Design a java program that accepts Principle Amount, No. of Years (period) & Rate of Interest (ROI), it should also contain "Calculate Interest" button, on clicking this button the data is sent to a method that returns the simple interest, also design "Final Amount" button, which will display the final amount by adding principle amount and interest.

3. Develop GUI program for following :



4. Design GUI application for PizzaHub, pizza is available in three different sizes small, medium and large having different rate. It also has various topping options such as plain, sausage, mushroom and pepperoni. Customers are allowed to place different order. Calculate bill.
5. Design a GUI application to demonstrate working of RadioButton with images. Place a five RadioButton name it with different pet names. When user clicks on one of the option it should display corresponding image.



6. Solve above question no : 5 using ComboBox instead of RadioButton.
7. Create a GUI applications that include basic listbox operations.

❑❑❑