Ch – 5
Swing & GUI

Swing

- Swing is a set of classes that provides more powerful and flexible components than are possible with the AWT.

- Swing supplies several exciting additions, including tabbed panes, scroll panes, trees, and tables.

- The Swing component classes that are used in this are shown here:

JApplet class

- The Japplet class is a subclass of the Applet class. An applet must extend the Japplet class which uses a swing component. Japplet provide the support for various panes such as the content pane, the pane etc.

- To add a component in the Japplet, you have to call the add() mehhod by a content pane object.

- **Following methods is used :**
  - 1) Container getContentPane()
  - 2) void add(componentObj)

1) Buttons

- Swing buttons provide features that are not found in the **Button** class defined by the AWT.

- For example, you can associate an icon with a Swing button. Swing buttons are subclasses of the **AbstractButton** class, which extends **Jcomponent.**

2) The JButton Class

- 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 shown here:

  - **JButton(Icon *i*)**
  - **JButton(String *s*)**
  - **JButton(String *s*, Icon *i*)**

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/* <applet code="JButtonDemo" width=250 height=300>
</applet> */
public class JButtonDemo extends JApplet implements
    ActionListener {
JTextField jtf;
public void init() {Container contentPane =
    getContentPane();
contentPane.setLayout(new FlowLayout());
ImageIcon france = new ImageIcon("france.gif");
```

```
jb = new JButton(france);
jb.setActionCommand("France");
jb.addActionListener(this);
contentPane.add(jb);
ImageIcon germany = new ImageIcon("germany.gif");
jb = new JButton(germany);
jb.setActionCommand("Germany");
jb.addActionListener(this);
contentPane.add(jb);
ImageIcon italy = new ImageIcon("italy.gif");
jb = new JButton(italy);
jb.setActionCommand("Italy");
```

```
jb.addActionListener(this);
contentPane.add(jb);
ImageIcon japan = new ImageIcon("japan.gif");
jb = new JButton(japan);
jb.setActionCommand("Japan");
jb.addActionListener(this);
contentPane.add(jb);
jtf = new JTextField(15);
contentPane.add(jtf);  }
public void actionPerformed(ActionEvent ae) {
jtf.setText(ae.getActionCommand());
} }
```

3) Check Boxes

- The **JCheckBox** class, which provides the functionality of a check box, is a concrete implementation of **AbstractButton**. Its immediate superclass is **JToggleButton**, which provides support for two-state buttons.

- **Some of its constructors are shown here:**
  - **JCheckBox(Icon *i*)**
  - **JCheckBox(Icon *i*, boolean *state*)**
  - **JCheckBox(String *s*, Icon *i*, boolean *state*)**

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/* <applet code="JCheckBoxDemo" width=400
   height=50>
</applet> */
public class JCheckBoxDemo extends JApplet
   implements ItemListener {
JTextField jtf;
public void init() {
Container contentPane = getContentPane();
contentPane.setLayout(new FlowLayout());
```

```
ImageIcon normal = new ImageIcon("normal.gif");
ImageIcon rollover = new ImageIcon("rollover.gif");
ImageIcon selected = new ImageIcon("selected.gif");
JCheckBox cb = new JCheckBox("C", normal);
cb.setRolloverIcon(rollover);cb.setSelectedIcon(selected);
cb.addItemListener(this); contentPane.add(cb);
cb = new JCheckBox("C++", normal);
cb.setRolloverIcon(rollover);cb.setSelectedIcon(selected);
cb.addItemListener(this); contentPane.add(cb);
cb = new JCheckBox("Java", normal);
cb.setRolloverIcon(rollover);
cb.setSelectedIcon(selected);
```

```
cb.addItemListener(this);
contentPane.add(cb);
cb = new JCheckBox("Perl", normal);
cb.setRolloverIcon(rollover);
cb.setSelectedIcon(selected); cb.addItemListener(this);
contentPane.add(cb);
jtf = new JTextField(15);
contentPane.add(jtf); }
public void itemStateChanged(ItemEvent ie) {
JCheckBox cb = (JCheckBox)ie.getItem();
jtf.setText(cb.getText());
} }
```

4) Radio Buttons

- Radio buttons are supported by the **JRadioButton** class, which is a concrete implementation of **AbstractButton.** Its immediate superclass is **JToggleButton**, which provides support for two-state buttons.

- **Some of its constructors are shown here:**
  - JRadioButton(String *s*, Icon *i*, boolean *state*)
  - JRadioButton(String *s*, boolean *state*)

```
import java.awt.event.*;
import javax.swing.*;
/* <applet code="JRadioButtonDemo" width=300
   height=50>
</applet> */
public class JRadioButtonDemo extends JApplet
   implements ActionListener {
JTextField tf;
public void init() {
Container contentPane = getContentPane();
contentPane.setLayout(new FlowLayout());
JRadioButton b1 = new JRadioButton("A");
```

```
b1.addActionListener(this); contentPane.add(b1);
JRadioButton b2 = new JRadioButton("B");
b2.addActionListener(this);
contentPane.add(b2);
JRadioButton b3 = new JRadioButton("C");
b3.addActionListener(this); contentPane.add(b3);
ButtonGroup bg = new ButtonGroup();
bg.add(b1); bg.add(b2); bg.add(b3);
tf = new JTextField(5); contentPane.add(tf); }
public void actionPerformed(ActionEvent ae) {
tf.setText(ae.getActionCommand());
} }
```

5) 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.

- Its constructors are shown here:
  - **1) JComboBox( )**
  - **2) JComboBox(Vector *v*)**

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
/* <applet code="JComboBoxDemo" width=300
   height=100>
</applet> */
public class JComboBoxDemo extends JApplet
   implements ItemListener {
JLabel jl;
ImageIcon france, germany, italy, japan;
public void init() {
Container contentPane = getContentPane();
```

```
contentPane.setLayout(new FlowLayout());
JComboBox jc = new JComboBox();
jc.addItem("France");
 jc.addItem("Germany");
jc.addItem("Italy");
 jc.addItem("Japan");
jc.addItemListener(this); contentPane.add(jc);
jl = new JLabel(new ImageIcon("france.gif"));
contentPane.add(jl); }
public void itemStateChanged(ItemEvent ie) {
String s = (String)ie.getItem();
jl.setIcon(new ImageIcon(s + ".gif")); } }
```

6) JLabel

- The Jlabel class is used to display a label on the window. This label can have a text as well as an image also.

- **Its constructor :**
  - JLabel(String label)
  - Jlabel(Icon i)
  - Jlabel(String label, Icon I, int align)

7) JTextField

- JTextFields class creates a text field which can accept the user input.

- **Its constructor :**
  - JTextField()
  - JTextField(String str)
  - JTextField(int column)
  - JTextField(String str, int column)

8) JTextArea

- JTextArea class is used to create a multi line text input. This text area does not display any scrollbars by default but you should use scroll panes if need it.

- **Its constructor :**
  - JTextField(String str, int row, int column)

- **Its method :**
  - **1) String getText() :** It reaturns the text written in the text area.
  - **2) Void setEditable(boolean edit)**

9) JPasswordField

- JPassworField creates a text field which display the disks(dots) instead of actual characters.

- **Its constructor :**
  - JPasswordField(int length)
  - JPasswordField(String str)
  - JPasswordField(String str, int length)

- **Its method:**
  - **String getPassword() :** return password

10) JScrollPane

- JScrollPane is used to display other components or an image n a rectangular area. This pane can have horizontal as well as vertical scroll bars.

- **Its constructor :**

  - JScrollPane(Component obj)
  - JScrollPane(int vertScrollbar, int horScrollBar)
  - JScrollPane(int vertScrollbar, int horScrollBar, int vertScrollbar)

- **Following Constants :**
  - **HORIZONTAL_SCROLLBAR_ALWAYS**
  - **HORIZONTAL_SCROLLBAR_AS_NEEDED**
  - **VERTICAL_SCROLLBAR_ALWAYS**
  - **VERTICAL_SCROLLBAR_AS_NEEDED**

Example

```
import java.awt.*;
import javax.swing.*;
/* <applet code="JScrollPaneDemo" width=300
   height=250> </applet> */
public class JScrollPaneDemo extends JApplet {
   public void init() {
Container contentPane = getContentPane();
contentPane.setLayout(new BorderLayout());
JPanel jp = new JPanel();
jp.setLayout(new GridLayout(20, 20));
int b = 0;
```

Example

```
for(int i = 0; i < 20; i++) {
for(int j = 0; j < 20; j++) {
jp.add(new JButton("Button " + b)); ++b; } }
int v = ScrollPaneConstants.
   VERTICAL_SCROLLBAR_AS_NEEDED;
int h = ScrollPaneConstants.
   HORIZONTAL_SCROLLBAR_AS_NEEDED;
JScrollPane jsp = new JScrollPane(jp, v, h);
contentPane.add(jsp, BorderLayout.CENTER);
}
}
```

11) JList

- JList class creates a list box which can display more than one items in the combo box at a time.

- **Its constructor :**

  - **JList()**
  - **JList(Vector v)**
  - **JList(Object[] obj)**

- **Some methods of the JList class are:**
  - **1) int getSelectedIndex() :** This methods returns the index of the selected item.
  - **2) Object getSelectedValue() :** It returns the selected item.
  - **3) int getFirstVisibleIndex()**
  - **4) int getVisibleRowCount()**
  - **5) void addListSelectionListener(ListSelectionListener obj)**
  - **6) void setSelectionMode(int selectionMode)**

- **Selection mode can be specified by the following constants defined by the ListSelectionModel:**
  - **1) ListSelectionModel.SINGLE_SELECTION**
  - **2) ListSelectionModel.SINGLE_INTERVAL_SELECTION**
  - **3) ListSelectionModel.MULTIPLE_INTERVAL_SELECTION**

12) JPanel

- Jpanel class is used to create a panel, we can add our components to a panel amd the panel should be added to a container such as frame.

- Now to draw the applet window, following method to be used instead of the paint() method.

  - **Public void paintComponent(Graphics g)**

13) JFrame

- JFrame class create frame which is window. This window has a border, a title, and button for closing minimizing and maximizing the window.

- To set default closing behavior following method :
- **Void setDefaultClosingOperation(int op)**

- **Its Constants :**
  - **HIDE_ON_CLOSE**
  - **DESPOSE_ON_CLOSE**
  - **EXIT_ON_CLOSE**
  - **DO_NOTHING_ON_CLOSE**

14) Jmenu, JMenuBar and JMenuItem

- **1)** The **JMenuBar** calss create a mennu bar which contains the menus. Its constructor :
  - JMenuBar()

- **2)** The **Jmenu** class is a container class menus. Its constructor :
  - JMenu()
  - Jmenu(String name)

- **3)** The **JMenuItem** class is used to create menu items. This items will be added to the menu.

AWT Controls

- AWT control are much similar to that of the swing but having some features.

- The following controls are defined in the java.awt package.

- **1) Label :** A label is a control that display a string as a label on it. It does not support any events.

- **Its constructor are :**
  - 1) Label()
  - 2) Label(String label)
  - 3) Label(String label, int align)

- **Its constant are :**
  - Label.LEFT, Label.RIGHT, Label.CENTER

- **Following methods :**
  - 1) String getLabel()
  - 2) void setLabel(String label)

- **2) Button : B**utton class creates a button.

- **Its constructor :**
  - 1) Button()
  - 2) Button(String caption)

- **3) Checkbox and CheckboxGroup :** A The checkbox class creates a checkbox.

- **Its constructor :**
  - 1) Checkbox()
  - 2) Checkbox(String label)

- 3) Checkbox(Strig label, boolean on)
- 4) Checkbox(String label, boolean on, ccheckboxgroup on)

- **Its methods :**
  - 1) String getLabel() : get label of the checkbox
  - 2) void setLabel(String label) : set the label.
  - 3) void setState(boolean state) : set state of the checkbox
  - 4) boolean getState() : get state

- **4) Choice :** The Choice class creates a drop-down list from the user can select an item.

- **Its constructor :**
  - 1) Choice()

- **Following methods :**
  - **1) void add(String item)**
  - **2) void addItem(String item) :** It crates and add items in the list

  - **3) String getSelectedItem() :** It returns the selected item.

- **4) int getSelectedIndex() :** It returns index of selected item.

- **5) String getItem(int index) :** It returns the item whose index is specified by the index.

- **6) int getItemCount() :** It returns the number of items in the list.

- **5) List :** The List class creates a list which contains items but it shows more than one item at a time.

- **Its constructor :**
  - 1) List()
  - 2) List(int numOfRows)
  - 3) List(int numOfRows, boolean multSelect)

- **Following methods :**

  - **1) void add(String item) : It** and add items to the list

- **2) String getSelectedItem() :** It returns the selected item.

- **4) int getSelectedIndex() :** It returns index of selected item.

- **5) String[] getSelectedIndex() :** It returns array of indexes of the selected items.

- **6) String[] getSelectedItem() :** It returns array of the selected items.

- **6) String getItem(int index)**

- **6) TextField :** The TextField class creates a text field.

- **Its constructor :**
  - 1) TextField()
  - 2) TetField(String str)
  - 3) TextField(int numOfChars)
  - 4) TextField(String str,int numOfChars)

- **Following methods :**

  - **1) String getText() : It** returns the text written in the text field

- **2) void setText(String text) : It** returns the text into the text field.

- **3) String getSelectedText() :** It returns the selected text in the text field.

- **4) void select(int strat, int end) :** It sets the text starting from the start to the end as selected.

- **5) boolean isEditable() :** It returns true if the text in the text field is editable, else return false.

- **6) void setEchoChar(char ch) :** It sets the character ch as the echo character. In case of password field you can display any character here.

- **7) boolean echoCharIsSet() :** It returns true if any echo character is set, else returns false.

- **8) char getEchoChar() :** It returns the echo character.

- **7) TextArea :** The TextArea class creates a multi line input text field.

- **Its constructor :**
  - 1) TextArea()
  - 2) TextArea(String str)
  - 3) TextArea(int numOfLines, int numOfChars)
  - 4) TextArea(String str,int numOfLines, int numOfChars, int scrollbar)

- **Following methods :**
  - **1) String getText() : It** returns the text written in the text area.

- **2) void setText() : It** sets the selected text of the text area.

- **3) String getSelectedText()** : It returns the selected text of the text area.

- **4) void append() :** It appends the string str to the text area string.

- **5) void insert(String str, int index) :** It inserts the string str at the specified index.