Chapter 32 Exploring Swing

The Swing component classes described in this chapter are shown here:

JButton	JCheckBox	JComboBox	JLabel
JList	JRadioButton	JScrollPane	JTabbedPane
JTable	JTextField	JToggleButton	JTree

These components are all lightweight, which means that they are all derived from JComponent.

1. JLabel and ImageIcon

JLabel is Swing's easiest-to-use component.

JLabel can be used to display text and/or an icon. It is a passive component in that ***it does not respond to user input.***

The object of JLabel class is a component for placing text in a container. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

JLabel class declaration

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

public class JLabel **extends** JComponent **implements** SwingConstants, Accessible

Commonly used Methods:

Methods	Description
String getText()	It returns the text string that a label displays.
void setText(String text)	It defines the single line of text this component will display.
void setHorizontalAlignment(int alignment)	It sets the alignment of the label's contents along the X axis.

Reference:- Herbert Schildt - Java_ The Complete Reference, Eleventh Edition 11(2019, McGraw-Hill Education)

Icon getIcon()	It returns the graphic image that the label displays.
int getHorizontalAlignment()	It returns the alignment of the label's contents along the X axis.

Commonly used Constructors:

Constructor	Description
JLabel()	Creates a JLabel instance with no image and with an empty string for the title.
JLabel(String s)	Creates a JLabel instance with the specified text.
JLabel(Icon i)	Creates a JLabel instance with the specified image.
JLabel(String s, Icon i, int horizontalAlignment)	Creates a JLabel instance with the specified text, image, and horizontal alignment.

The horizontalAlignment argument specifies the horizontal alignment of the text and/or icon within the dimensions of the label. It must be one of the following values: LEFT, RIGHT, CENTER, LEADING, or TRAILING. These constants are defined in the SwingConstants interface, along with several others used by the Swing classes.

Icons are specified by objects of type Icon, which is an interface defined by Swing. The easiest ***way to obtain an icon is to use the ImageIcon class.*** ImageIcon implements Icon and encapsulates an image. Thus, an object of type ImageIcon can be passed as an argument to the Icon parameter of JLabel's constructor.

Ways to provide the image

Reading it from a file or downloading it from a URL

ImageIcon constructor

- ImageIcon(String filename)---It obtains the image in the file named *filename*.

Reference:- Herbert Schildt - Java_ The Complete Reference, Eleventh Edition 11(2019, McGraw-Hill Education)

The icon and text associated with the label can be obtained by the following methods:

- Icon getIcon( )
- String getText( )

The icon and text associated with a label can be set by these methods:

- void setIcon(Icon icon)
- void setText(String str)

Example:-

The following program illustrates how to create and display a label containing both an icon and a string. It begins by creating an ImageIcon object for the file hourglass.png *(save this image named hourglass in the folder where*

```
import java.awt.*;
import javax.swing.*;

public class JLabelDemo {

   public JLabelDemo() {

     // Set up the JFrame.
     JFrame jfrm = new JFrame("JLabelDemo");
     jfrm.setLayout(new FlowLayout());
     jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     jfrm.setSize(260, 210);

     // Create an icon.
     ImageIcon ii = new ImageIcon("hourglass.png");

     // Create a label.
     JLabel jl = new JLabel("Hourglass", ii, JLabel.CENTER);

     // Add the label to the content pane.
     jfrm.add(jl);

     // Display the frame.
     jfrm.setVisible(true);
   }
```

Reference:- Herbert Schildt - Java_ The Complete Reference, Eleventh Edition 11(2019, McGraw-Hill Education)

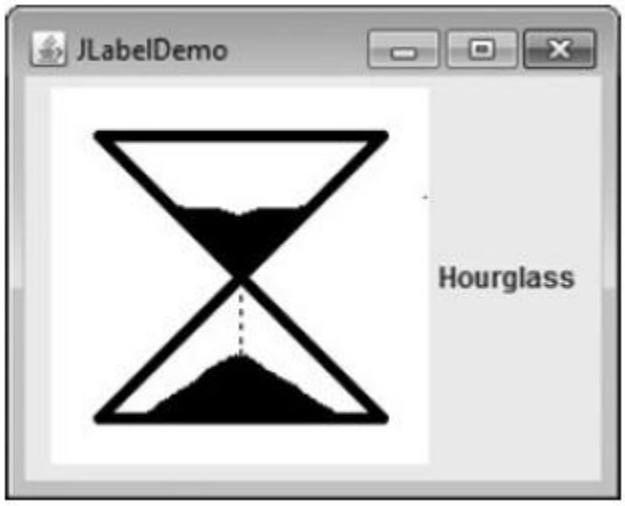
program resides), which depicts an hourglass. This is used as the second argument to the JLabel constructor. The first and last arguments for the JLabel constructor are the label text and the alignment. Finally, the label is added to the content pane.

```
  public static void main(String[] args) {
    // Create the frame on the event dispatching thread.

    SwingUtilities.invokeLater(
      new Runnable() {
        public void run() {
          new JLabelDemo();
        }
      }
    );

  }
}
```

Output:



2. JTextField

JTextField is the simplest Swing text component.

Reference:- Herbert Schildt - Java_ The Complete Reference, Eleventh Edition 11(2019, McGraw-Hill Education)

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.

JTextField class declaration

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

1. **public class** JTextField **extends** JTextComponent **implements** SwingConstants

Commonly used Constructors:

Constructor	Description
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

Commonly used Methods:

Methods	Description
void addActionListener(ActionListener l)	It is used to add the specified action listener to receive action events from this textfield.
Action getAction()	It returns the currently set Action for this ActionEvent source, or null if no Action is set.
void setFont(Font f)	It is used to set the current font.

Reference:- Herbert Schildt - Java_ The Complete Reference, Eleventh Edition 11(2019, McGraw-Hill Education)

void removeActionListener(ActionListener l)	It is used to remove the specified action listener so that it no longer receives action events from this textfield.

JTextField generates events in response to user interaction. For example, an ActionEvent is fired when the user presses enter. A CaretEvent is fired each time the caret (i.e., the cursor) changes position. **To obtain the text currently in the text field, call getText( ).**

The following example illustrates JTextField. It creates a JTextField and adds it to the content pane. When the user presses enter, an action event is generated. This is handled by displaying the text in a label.

Reference:- Herbert Schildt - Java_ The Complete Reference, Eleventh Edition 11(2019, McGraw-Hill Education)

```
// Demonstrate JTextField.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class JTextFieldDemo {

  public JTextFieldDemo() {

    // Set up the JFrame.
    JFrame jfrm = new JFrame("JTextFieldDemo");
    jfrm.setLayout(new FlowLayout());
    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    jfrm.setSize(260, 120);

    // Add a text field to content pane.
    JTextField jtf = new JTextField(15);
    jfrm.add(jtf);

    // Add a label.
    JLabel jlab = new JLabel();
    jfrm.add(jlab);

    // Handle action events.
    jtf.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ae) {
        // Show text when user presses ENTER.
        jlab.setText(jtf.getText());
      }
    });

    // Display the frame.
    jfrm.setVisible(true);
  }

  public static void main(String[] args) {
    // Create the frame on the event dispatching thread.

    SwingUtilities.invokeLater(
      new Runnable() {
        public void run() {
          new JTextFieldDemo();
        }
      }
    );

  }
}
```

Output:-

Reference:- Herbert Schildt - Java_ The Complete Reference, Eleventh Edition 11(2019, McGraw-Hill Education)