SWING - JLABEL CLASS

Introduction

The class **JLabel** can display either text, an image, or both. Label's contents are aligned by setting the vertical and horizontal alignment in its display area. By default, labels are vertically centered in their display area. Text-only labels are leading edge aligned, by default; image-only labels are horizontally centered, by default.

Class declaration

Following is the declaration for **javax.swing.JLabel** class:

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

Field

Following are the fields for **javax.swing.JLabel** class:

- **protected Component labelFor**

Class constructors

S.N.	Constructor & Description
1	**JLabel** Creates a JLabel instance with no image and with an empty string for the title.
2	**JLabel***Iconimage* Creates a JLabel instance with the specified image.
3	**JLabel***Iconimage, inthorizontalAlignment* Creates a JLabel instance with the specified image and horizontal alignment.
4	**JLabel***Stringtext* Creates a JLabel instance with the specified text.
5	**JLabel***Stringtext, Iconicon, inthorizontalAlignment* Creates a JLabel instance with the specified text, image, and horizontal alignment.
6	**JLabel***Stringtext, inthorizontalAlignment* Creates a JLabel instance with the specified text and horizontal alignment.

Class methods

S.N.	Method & Description

1 **protected int checkHorizontalKey***intkey, Stringmessage*

Verify that key is a legal value for the horizontalAlignment properties.

2 **protected int checkVerticalKey***intkey, Stringmessage*

Verify that key is a legal value for the verticalAlignment or verticalTextPosition properties.

3 **AccessibleContext getAccessibleContext**

Get the AccessibleContext of this object.

4 **Icon getDisabledIcon**

Returns the icon used by the label when it's disabled.

5 **int getDisplayedMnemonic**

Return the keycode that indicates a mnemonic key.

6 **int getDisplayedMnemonicIndex**

Returns the character, as an index, that the look and feel should provide decoration for as representing the mnemonic character.

7 **int getHorizontalAlignment**

Returns the alignment of the label's contents along the X axis.

8 **int getHorizontalTextPosition**

Returns the horizontal position of the label's text, relative to its image.

9 **Icon getIcon**

Returns the graphic image *glyph, icon* that the label displays.

10 **int getIconTextGap**

Returns the amount of space between the text and the icon displayed in this label.

11 **Component getLabelFor**

Get the component this is labelling.

12 **String getText**

Returns the text string that the label displays.

13 **LabelUI getUI**

Returns the L&F object that renders this component.

14 **String getUIClassID**

Returns a string that specifies the name of the l&f class that renders this component.

15 **int getVerticalAlignment**

Returns the alignment of the label's contents along the Y axis.

16 **int getVerticalTextPosition**

Returns the vertical position of the label's text, relative to its image.

17 **boolean imageUpdate***Imageimg, intinfoflags, intx, inty, intw, inth*

This is overridden to return false if the current Icon's Image is not equal to the passed in Image img.

18 **protected String paramString**

Returns a string representation of this JLabel.

19 **void setDisabledIcon***IcondisabledIcon*

Set the icon to be displayed if this JLabel is "disabled" *JLabel*. *setEnabled*(*false*).

20 **void setDisplayedMnemonic***charaChar*

Specifies the displayedMnemonic as a char value.

21 **void setDisplayedMnemonic***intkey*

Specify a keycode that indicates a mnemonic key.

22 **void setDisplayedMnemonicIndex***intindex*

Provides a hint to the look and feel as to which character in the text should be decorated to represent the mnemonic.

23 **void setHorizontalAlignment***intalignment*

Sets the alignment of the label's contents along the X axis.

24 **void setHorizontalTextPosition***inttextPosition*

Sets the horizontal position of the label's text, relative to its image.

25 **void setIcon***Iconicon*

Defines the icon this component will display.

26 **void setIconTextGap***inticonTextGap*

If both the icon and text properties are set, this property defines the space between them.

27 **void setLabelFor***Componentc*

Set the component this is labelling.

28 **void setText***Stringtext*

Defines the single line of text this component will display.

29 **void setUI***LabelUIui*

Sets the L&F object that renders this component.

30 **void setVerticalAlignment***intalignment*

Sets the alignment of the label's contents along the Y axis.

31 **void setVerticalTextPosition***inttextPosition*

Sets the vertical position of the label's text, relative to its image.

32 **void updateUI**

Resets the UI property to a value from the current look and feel.

Methods inherited

This class inherits methods from the following classes:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

JLabel Example

Create the following java program using any editor of your choice in say **D:/ > SWING > com > tutorialspoint > gui >**

SwingControlDemo.java

```
package com.tutorialspoint.gui;

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

public class SwingControlDemo {

   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;

   public SwingControlDemo(){
      prepareGUI();
   }

   public static void main(String[] args){
      SwingControlDemo  swingControlDemo = new SwingControlDemo();
      swingControlDemo.showLabelDemo();
   }

   private void prepareGUI(){
      mainFrame = new JFrame("Java Swing Examples");
```

```
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));
      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      headerLabel = new JLabel("", JLabel.CENTER);
      statusLabel = new JLabel("",JLabel.CENTER);

      statusLabel.setSize(350,100);

      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());

      mainFrame.add(headerLabel);
      mainFrame.add(controlPanel);
      mainFrame.add(statusLabel);
      mainFrame.setVisible(true);
   }

   private void showLabelDemo(){
      headerLabel.setText("Control in action: JLabel");

      JLabel label  = new JLabel("", JLabel.CENTER);
      label.setText("Welcome to TutorialsPoint Swing Tutorial.");
      label.setOpaque(true);
      label.setBackground(Color.GRAY);
      label.setForeground(Color.WHITE);
      controlPanel.add(label);

      mainFrame.setVisible(true);
   }
}
```

Compile the program using command prompt. Go to **D:/ > SWING** and type the following command.

```
D:\SWING>javac com\tutorialspoint\gui\SwingControlDemo.java
```

If no error comes that means compilation is successful. Run the program using following command.

```
D:\SWING>java com.tutorialspoint.gui.SwingControlDemo
```

Verify the following output