SWING - JSLIDER CLASS

 Copyright © tutorialspoint.com

Introduction

The class **JSlider** is a component which lets the user graphically select a value by sliding a knob within a bounded interval.

Class declaration

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

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

Field

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

- **protected ChangeEvent changeEvent** -- Only one ChangeEvent is needed per slider instance since the event's only *read − only* state is the source property.
- **protected ChangeListener changeListener** -- The changeListener *nosuffix* is the listener we add to the slider's model.
- **protected int majorTickSpacing** -- The number of values between the major tick marks -- the larger marks that break up the minor tick marks.
- **protected int minorTickSpacing** -- The number of values between the minor tick marks -- the smaller marks that occur between the major tick marks.
- **protected int orientation** -- Whether the slider is horizontal or vertical The default is horizontal.
- **protected BoundedRangeModel sliderModel** -- The data model that handles the numeric maximum value, minimum value, and current-position value for the slider.
- **protected boolean snapToTicks** -- If true, the knob *andthedatavalueitrepresents* resolve to the closest tick mark next to where the user positioned the knob.

Class constructors

S.N.	Constructor & Description
1	**JSlider** Creates a horizontal slider with the range 0 to 100 and an initial value of 50.
2	**JSlider*BoundedRangeModelbrm*** Creates a horizontal slider using the specified BoundedRangeModel.
3	**JSlider*intorientation*** Creates a slider using the specified orientation with the range 0 to 100 and an initial value of 50.
4	**JSlider*intmin, intmax***

Creates a horizontal slider using the specified min and max with an initial value equal to the average of the min plus max.

5 **JSlider*intmin, intmax, intvalue***

Creates a horizontal slider using the specified min, max and value.

6 **JSlider*intorientation, intmin, intmax, intvalue***

Creates a slider with the specified orientation and the specified minimum, maximum, and initial values.

Class methods

S.N.	Method & Description

1 **void addChangeListener*ChangeListenerl***

Adds a ChangeListener to the slider.

2 **protected ChangeListener createChangeListener**

Subclasses that want to handle ChangeEvents from the model differently can override this to return an instance of a custom ChangeListener implementation.

3 **Hashtable createStandardLabels*intincrement***

Creates a Hashtable of numerical text labels, starting at the slider minimum, and using the increment specified.

4 **Hashtable createStandardLabels*intincrement, intstart***

Creates a Hashtable of numerical text labels, starting at the starting point specified, and using the increment specified.

5 **protected void fireStateChanged**

Send a ChangeEvent, whose source is this JSlider, to all ChangeListeners that have registered interest in ChangeEvents.

6 **AccessibleContext getAccessibleContext**

Gets the AccessibleContext associated with this JSlider.

7 **ChangeListener[] getChangeListeners**

Returns an array of all the ChangeListeners added to this JSlider with addChangeListener.

8 **int getExtent**

Returns the "extent" from the BoundedRangeModel.

9 **boolean getInverted**

Returns true if the value-range shown for the slider is reversed.

10 **Dictionary getLabelTable**

Returns the dictionary of what labels to draw at which values.

11 **int getMajorTickSpacing**

This method returns the major tick spacing.

12 **int getMaximum**

Returns the maximum value supported by the slider from the BoundedRangeModel.

13 **int getMinimum**

Returns the minimum value supported by the slider from the BoundedRangeModel.

14 **int getMinorTickSpacing**

This method returns the minor tick spacing.

15 **BoundedRangeModel getModel**

Returns the BoundedRangeModel that handles the slider's three fundamental properties: minimum, maximum, value.

16 **int getOrientation**

Return this slider's vertical or horizontal orientation.

17 **boolean getPaintLabels**

Tells if labels are to be painted.

18 **boolean getPaintTicks**

Tells if tick marks are to be painted.

19 **boolean getPaintTrack**

Tells if the track *areatheslider slidesin* is to be painted.

20 **boolean getSnapToTicks**

Returns true if the knob *andthedatavalueitrepresents* resolve to the closest tick mark next to where the user positioned the knob.

21 **SliderUI getUI**

Gets the UI object which implements the L&F for this component.

22 **String getUIClassID**

Returns the name of the L&F class that renders this component.

23 **int getValue**

Returns the slider's current value from the BoundedRangeModel.

24 **boolean getValueIsAdjusting**

Returns the valueIsAdjusting property from the model.

25 **protected String paramString**

Returns a string representation of this JSlider.

26 **void removeChangeListener*ChangeListenerl***

Removes a ChangeListener from the slider.

27 **void setExtent*intextent***

Sets the size of the range "covered" by the knob.

28 **void setFont*Fontfont***

Sets the font for this component.

29 **void setInverted*booleanb***

Specify true to reverse the value-range shown for the slider and false to put the value range in the normal order.

30 **void setLabelTable*Dictionarylabels***

Used to specify what label will be drawn at any given value.

31 **void setMajorTickSpacing*intn***

This method sets the major tick spacing.

32 **void setMaximum*intmaximum***

Sets the slider's maximum value to maximum.

33 **void setMinimum*intminimum***

Sets the slider's minimum value to minimum.

34 **void setMinorTickSpacing*intn***

This method sets the minor tick spacing.

35 **void setModel*BoundedRangeModelnewModel***

Sets the BoundedRangeModel that handles the slider's three fundamental properties: minimum, maximum, value.

36 **void setOrientation*intorientation***

Set the slider's orientation to either SwingConstants.VERTICAL or SwingConstants.HORIZONTAL.

37 **void setPaintLabels*booleanb***

Determines whether labels are painted on the slider.

38 **void setPaintTicks*booleanb***

Determines whether tick marks are painted on the slider.

39 **void setPaintTrack*booleanb***

Determines whether the track is painted on the slider.

40 **void setSnapToTicks*booleanb***

Specifying true makes the knob *andthedatavalueitrepresents* resolve to the closest tick mark next to where the user positioned the knob.

41 **void setUI*SliderUIui***

Sets the UI object which implements the L&F for this component.

42 **void setValue*intn* Sets the slider's current value to n.**

43 **void setValueIsAdjusting*booleanb***

Sets the model's valueIsAdjusting property.

44 **protected void updateLabelUIs**

Updates the UIs for the labels in the label table by calling updateUI on each label.

45 **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

JSlider 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.*;
```

```
import javax.swing.event.*;

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.showSliderDemo();
   }

   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 showSliderDemo(){
      headerLabel.setText("Control in action: JSlider");
      JSlider slider= new JSlider(JSlider.HORIZONTAL,0,100,10);
      slider.addChangeListener(new ChangeListener() {
         public void stateChanged(ChangeEvent e) {
            statusLabel.setText("Value : "
            + ((JSlider)e.getSource()).getValue());
         }
      });
      controlPanel.add(slider);
      mainFrame.setVisible(true);
   }
}
```

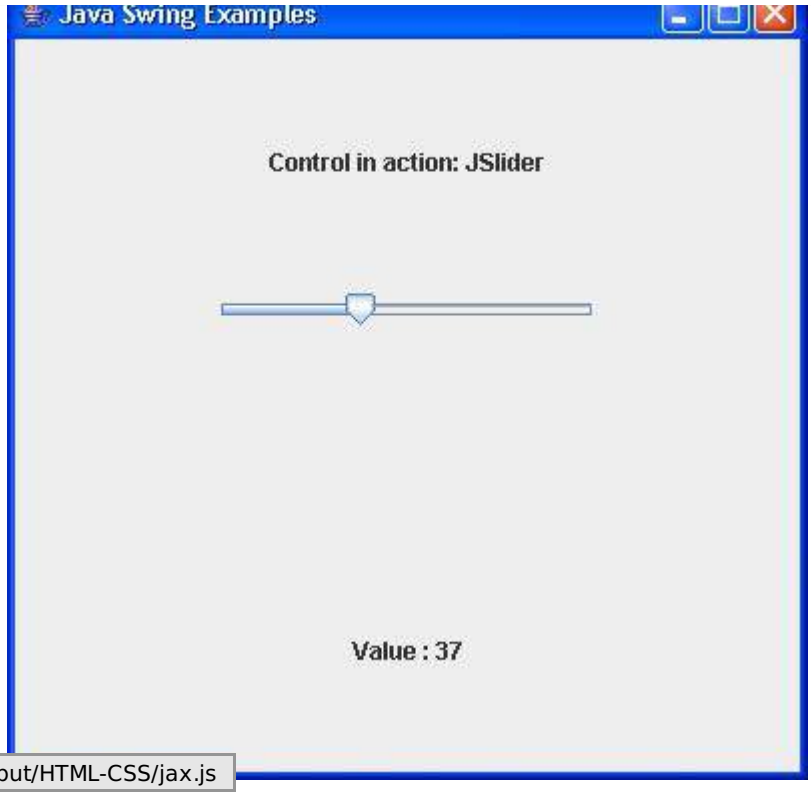
Compile the program using command prompt. Go to **D:/ > SWING** and type the following command.

```
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



Loading [MathJax]/jax/output/HTML-CSS/jax.js