SWING - JPROGRESSBAR CLASS

Introduction

The class **JProgressBar** is a component which visually displays the progress of some task.

Class declaration

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

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

Field

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

- **protected ChangeEvent changeEvent** -- Only one ChangeEvent is needed per instance since the event's only interesting property is the immutable source, which is the progress bar.
- **protected ChangeListener changeListener** -- Listens for change events sent by the progress bar's model, redispatching them to change-event listeners registered upon this progress bar.
- **protected BoundedRangeModel model** -- The object that holds the data for the progress bar.
- **protected int orientation** -- Whether the progress bar is horizontal or vertical.
- **protected boolean paintBorder** -- Whether to display a border around the progress bar.
- **protected boolean paintString** -- Whether to display a string of text on the progress bar.
- **protected String progressString** -- An optional string that can be displayed on the progress bar.

Class constructors

S.N.	Constructor & Description
1	**JProgressBar** Creates a horizontal progress bar that displays a border but no progress string.
2	**JProgressBar*BoundedRangeModelnewModel*** Creates a horizontal progress bar that uses the specified model to hold the progress bar's data.
3	**JProgressBar*intorient*** Creates a progress bar with the specified orientation, which can be either SwingConstants. VERTICAL or SwingConstants.HORIZONTAL.
4	**JProgressBar*intmin, intmax*** Creates a horizontal progress bar with the specified minimum and maximum.

5	**JProgressBar***intorient, intmin, intmax* Creates a progress bar using the specified orientation, minimum, and maximum.

Class methods

S.N.	Method & Description
1	**void addChangeListener***ChangeListenerl* Adds the specified ChangeListener to the progress bar.
2	**protected ChangeListener createChangeListener** Subclasses that want to handle change events from the model differently can override this to return an instance of a custom ChangeListener implementation.
3	**protected void fireStateChanged** Send a ChangeEvent, whose source is this JProgressBar, to all ChangeListeners that have registered interest in ChangeEvents.
4	**AccessibleContext getAccessibleContext** Gets the AccessibleContext associated with this JProgressBar.
5	**ChangeListener[] getChangeListeners** Returns an array of all the ChangeListeners added to this progress bar with addChangeListener.
6	**int getMaximum** Returns the progress bar's maximum value from the BoundedRangeModel.
7	**int getMinimum** Returns the progress bar's minimum value from the BoundedRangeModel.
8	**BoundedRangeModel getModel** Returns the data model used by this progress bar.
9	**int getOrientation** Returns SwingConstants.VERTICAL or SwingConstants.HORIZONTAL, depending on the orientation of the progress bar.
10	**double getPercentComplete** Returns the percent complete for the progress bar.
11	**String getString**

Returns a String representation of the current progress.

12 **ProgressBarUI getUI**

Returns the look-and-feel object that renders this component.

13 **String getUIClassID**

Returns the name of the look-and-feel class that renders this component.

14 **int getValue**

Returns the progress bar's current value from the BoundedRangeModel.

15 **boolean isBorderPainted**

Returns the borderPainted property.

16 **boolean isIndeterminate**

Returns the value of the indeterminate property.

17 **boolean isStringPainted**

Returns the value of the stringPainted property.

18 **protected void paintBorder***Graphicsg*

Paints the progress bar's border if the borderPainted property is true.

19 **protected String paramString**

Returns a string representation of this JProgressBar.

20 **void removeChangeListener***ChangeListenerl*

Removes a ChangeListener from the progress bar.

21 **void setBorderPainted***booleanb*

Sets the borderPainted property, which is true if the progress bar should paint its border.

22 **void setIndeterminate***booleannewValue*

Sets the indeterminate property of the progress bar, which determines whether the progress bar is in determinate or indeterminate mode.

23 **void setMaximum***intn*

Sets the progress bar's maximum value *storedintheprogressbar'sdatamodel* to n.

24 **void setMinimum***intn*

Sets the progress bar's minimum value *storedintheprogressbar'sdatamodel* to n.

25 **void setModel***BoundedRangeModelnewModel*

Sets the data model used by the JProgressBar.

26 **void setOrientation*intnewOrientation***

Sets the progress bar's orientation to newOrientation, which must be SwingConstants.VERTICAL or SwingConstants.HORIZONTAL.

27 **void setString*Strings***

Sets the value of the progress string.

28 **void setStringPainted*booleanb***

Sets the value of the stringPainted property, which determines whether the progress bar should render a progress string.

29 **void setUI*ProgressBarUIui***

Sets the look-and-feel object that renders this component.

30 **void setValue*intn***

Sets the progress bar's current value to n.

31 **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

JProgressBar 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.showProgressBarDemo();
   }

   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 JProgressBar progressBar;
   private Task task;
   private JButton startButton;
   private JTextArea outputTextArea;

   private void showProgressBarDemo(){
      headerLabel.setText("Control in action: JProgressBar");

      progressBar = new JProgressBar(0, 100);
      progressBar.setValue(0);
      progressBar.setStringPainted(true);
      startButton = new JButton("Start");

      outputTextArea = new JTextArea("",5,20);

      JScrollPane scrollPane = new JScrollPane(outputTextArea);
         startButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            task = new Task();
            task.start();
         }});

      controlPanel.add(startButton);
      controlPanel.add(progressBar);
      controlPanel.add(scrollPane);
      mainFrame.setVisible(true);
   }

   private class Task extends Thread {
      public Task(){
      }

      public void run(){
         for(int i =0; i<= 100; i+=10){
            final int progress = i;
            SwingUtilities.invokeLater(new Runnable() {
               public void run() {
```

```
                    progressBar.setValue(progress);
                    outputTextArea.setText(outputTextArea.getText()
                    + String.format("Completed %d%% of task.\n", progress));
                }
            });
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {}
        }
    }
}
```

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