SWING - JTEXTFIELD CLASS

Introduction

The class **JTextField** is a component which allows the editing of a single line of text.

Class declaration

Following is the declaration for **javax.swing.JTextField** class −

```
public class JTextField
   extends JTextComponent
      implements SwingConstants
```

Field

Following are the fields for **javax.swing.JList** class −

- **static String notifyAction** − Name of the action to send notification that the contents of the field have been accepted.

Class constructors

S.N.	Constructor & Description
1	**JTextField** Constructs a new TextField.
2	**JTextField***Documentdoc, Stringtext, intcolumns* Constructs a new JTextField that uses the given text storage model and the given number of columns.
3	**JTextField***intcolumns* Constructs a new empty TextField with the specified number of columns.
4	**JTextField***Stringtext* Constructs a new TextField initialized with the specified text.
5	**JTextField***Stringtext, intcolumns* Constructs a new TextField initialized with the specified text and columns.

Class methods

S.N.	Method & Description
1	**protected void actionPropertyChanged***Actionaction, StringpropertyName* Updates the textfield's state in response to property changes in associated action.

2 **void addActionListener*ActionListenerl***

Adds the specified action listener to receive action events from this textfield.

3 **protected void configurePropertiesFromAction*Actiona***

Sets the properties on this textfield to match those in the specified Action.

4 **protected PropertyChangeListener createActionPropertyChangeListener*Actiona***

Creates and returns a PropertyChangeListener that is responsible for listening for changes from the specified Action and updating the appropriate properties.

5 **protected Document createDefaultModel**

Creates the default implementation of the model to be used at construction if one isn't explicitly given.

6 **protected void fireActionPerformed**

Notifies all listeners that have registered interest for notification on this event type.

7 **AccessibleContext getAccessibleContext**

Gets the AccessibleContext associated with this JTextField.

8 **Action getAction**

Returns the currently set Action for this ActionEvent source, or null if no Action is set.

9 **ActionListener[] getActionListeners**

Returns an array of all the ActionListeners added to this JTextField with addActionListener .

10 **Action[] getActions**

Fetches the command list for the editor.

11 **int getColumns**

Returns the number of columns in this TextField.

12 **protected int getColumnWidth**

Returns the column width.

13 **int getHorizontalAlignment**

Returns the horizontal alignment of the text.

14 **BoundedRangeModel getHorizontalVisibility**

Gets the visibility of the text field.

15 **Dimension getPreferredSize**

Returns the preferred size Dimensions needed for this TextField.

16 **int getScrollOffset**

Gets the scroll offset, in pixels.

17 **String getUIClassID**

Gets the class ID for a UI.

18 **boolean isValidateRoot**

Calls to revalidate that come from within the textfield itself will be handled by validating the textfield, unless the textfield is contained within a JViewport, in which case this returns false.

19 **protected String paramString**

Returns a string representation of this JTextField.

20 **void postActionEvent**

Processes action events occurring on this textfield by dispatching them to any registered ActionListener objects.

21 **void removeActionListener***ActionListenerl*

Removes the specified action listener so that it no longer receives action events from this textfield.

22 **void scrollRectToVisible***Rectangler*

Scrolls the field left or right.

23 **void setAction***Actiona*

Sets the Action for the ActionEvent source.

24 **void setActionCommand***Stringcommand*

Sets the command string used for action events.

25 **void setColumns***intcolumns*

Sets the number of columns in this TextField, and then invalidate the layout.

26 **void setDocument***Documentdoc*

Associates the editor with a text document.

27 **void setFont***Fontf*

Sets the current font.

28 **void setHorizontalAlignment***intalignment*

Sets the horizontal alignment of the text.

29	**void setScrollOffset***intscrollOffset* Sets the scroll offset, in pixels.

Methods inherited

This class inherits methods from the following classes:

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

JTextField 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.showTextFieldDemo();
   }

   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 showTextFieldDemo(){
      headerLabel.setText("Control in action: JTextField");

      JLabel  namelabel= new JLabel("User ID: ", JLabel.RIGHT);
      JLabel  passwordLabel = new JLabel("Password: ", JLabel.CENTER);
      final JTextField userText = new JTextField(6);
      final JPasswordField passwordText = new JPasswordField(6);

      JButton loginButton = new JButton("Login");
      loginButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String data = "Username " + userText.getText();
            data += ", Password: "
            + new String(passwordText.getPassword());
            statusLabel.setText(data);
         }
      });

      controlPanel.add(namelabel);
      controlPanel.add(userText);
      controlPanel.add(passwordLabel);
      controlPanel.add(passwordText);
      controlPanel.add(loginButton);
      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