SWING - JCOMBOBOX CLASS

Introduction

The class **JComboBox** is a component which combines a button or editable field and a drop-down list.

Class declaration

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

```
public class JComboBox
   extends JComponent
      implements ItemSelectable, ListDataListener,
         ActionListener, Accessible
```

Field

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

- **protected String actionCommand** − This protected field is implementation specific.
- **protected ComboBoxModel dataModel** − This protected field is implementation specific.
- **protected ComboBoxEditor editor** − This protected field is implementation specific.
- **protected boolean isEditable** − This protected field is implementation specific.
- **protected JComboBox.KeySelectionManager keySelectionManager** − This protected field is implementation specific.
- **protected boolean lightWeightPopupEnabled** − This protected field is implementation specific.
- **protected int maximumRowCount** − This protected field is implementation specific.
- **protected ListCellRenderer renderer** − This protected field is implementation specific.
- **protected Object selectedItemReminder** − This protected field is implementation specific.

Class constructors

S.N.	Constructor & Description
1	**JComboBox** Creates a JComboBox with a default data model.
2	**JComboBox*ComboBoxModelaModel*** Creates a JComboBox that takes its items from an existing ComboBoxModel.
3	**JComboBox*Object*[]*items*** Creates a JComboBox that contains the elements in the specified array.
4	**JComboBox*Vector* < ? > *items***

Creates a JComboBox that contains the elements in the specified Vector.

Class methods

S.N.	Method & Description
1	**void actionPerformed***ActionEvente* This method is public as an implementation side effect.
2	**protected void actionPropertyChanged***Actionaction, StringpropertyName* Updates the combobox's state in response to property changes in associated action.
3	**void addActionListener***ActionListenerl* Adds an ActionListener.
4	**void addItem***ObjectanObject* Adds an item to the item list.
5	**void addItemListener***ItemListeneraListener* Adds an ItemListener.
6	**void addPopupMenuListener***PopupMenuListenerl* Adds a PopupMenu listener which will listen to notification messages from the popup portion of the combo box.
7	**void configureEditor***ComboBoxEditoranEditor, ObjectanItem* Initializes the editor with the specified item.
8	**protected void configurePropertiesFromAction***Actiona* Sets the properties on this combobox to match those in the specified Action.
9	**void contentsChanged***ListDataEvente* This method is public as an implementation side effect.
10	**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.
11	**protected JComboBox.KeySelectionManager createDefaultKeySelectionManager** Returns an instance of the default key-selection manager.
12	**protected void fireActionEvent** Notifies all listeners that have registered interest for notification on this event type.

13 **protected void fireItemStateChanged***ItemEvente*

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

14 **void firePopupMenuCanceled**

Notifies PopupMenuListeners that the popup portion of the combo box has been canceled.

15 **void firePopupMenuWillBecomeInvisible**

Notifies PopupMenuListeners that the popup portion of the combo box has become invisible.

16 **void firePopupMenuWillBecomeVisible**

Notifies PopupMenuListeners that the popup portion of the combo box will become visible.

17 **AccessibleContext getAccessibleContext**

Gets the AccessibleContext associated with this JComboBox.

18 **Action getAction**

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

19 **String getActionCommand**

Returns the action command that is included in the event sent to action listeners.

20 **ActionListener[] getActionListeners**

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

21 **ComboBoxEditor getEditor**

Returns the editor used to paint and edit the selected item in the JComboBox field.

22 **Object getItemAt***intindex*

Returns the list item at the specified index.

23 **int getItemCount**

Returns the number of items in the list.

24 **ItemListener[] getItemListeners**

Returns an array of all the ItemListeners added to this JComboBox with addItemListener.

25 **JComboBox.KeySelectionManager getKeySelectionManager**

Returns the list's key-selection manager.

26 **int getMaximumRowCount**

Returns the maximum number of items the combo box can display without a scrollbar.

27 **ComboBoxModel getModel**

Returns the data model currently used by the JComboBox.

28 **PopupMenuListener[]getPopupMenuListeners**

Returns an array of all the PopupMenuListeners added to this JComboBox with addPopupMenuListener.

29 **Object getPrototypeDisplayValue**

Returns the "prototypical display" value - an Object used for the calculation of the display height and width.

30 **ListCellRenderer getRenderer**

Returns the renderer used to display the selected item in the JComboBox field.

31 **int getSelectedIndex**

Returns the first item in the list that matches the given item.

32 **Object getSelectedItem**

Returns the current selected item.

33 **Object[] getSelectedObjects**

Returns an array containing the selected item.

34 **ComboBoxUI getUI**

Returns the L&F object that renders this component.

35 **String getUIClassID**

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

36 **void hidePopup**

Causes the combo box to close its popup window.

37 **void insertItemAt***ObjectanObject, intindex*

Inserts an item into the item list at a given index.

38 **protected void installAncestorListener**

39 **void intervalAdded***ListDataEvente*

This method is public as an implementation side effect.

40 **void intervalRemoved***ListDataEvente*

This method is public as an implementation side effect.

41 **boolean isEditable**

Returns true if the JComboBox is editable.

42 **boolean isLightWeightPopupEnabled**

Gets the value of the lightWeightPopupEnabled property.

43 **boolean isPopupVisible**

Determines the visibility of the popup.

44 **protected String paramString**

Returns a string representation of this JComboBox.

45 **void processKeyEvent***KeyEvente*

Handles KeyEvents, looking for the Tab key.

46 **void removeActionListener***ActionListenerl*

Removes an ActionListener.

47 **void removeAllItems**

Removes all items from the item list.

48 **void removeItem***ObjectanObject*

Removes an item from the item list.

49 **void removeItemAt***intanIndex*

Removes the item at anIndex This method works only if the JComboBox uses a mutable data model.

50 **void removeItemListener***ItemListeneraListener*

Removes an ItemListener.

51 **void removePopupMenuListener***PopupMenuListenerl*

Removes a PopupMenuListener.

52 **protected void selectedItemChanged**

This protected method is implementation specific.

53 **boolean selectWithKeyChar***charkeyChar*

Selects the list item that corresponds to the specified keyboard character and returns true, if there is an item corresponding to that character.

54 **void setAction***Actiona*

Sets the Action for the ActionEvent source.

55 **void setActionCommand***StringaCommand*

Sets the action command that should be included in the event sent to action listeners.

56 **void setEditable***booleanaFlag*

Determines whether the JComboBox field is editable.

57 **void setEditor***ComboBoxEditoranEditor*

Sets the editor used to paint and edit the selected item in the JComboBox field.

58 **void setEnabled***booleanb*

Enables the combo box so that items can be selected.

59 **void setKeySelectionManager***JComboBox. KeySelectionManageraManager*

Sets the object that translates a keyboard character into a list selection.

60 **void setLightWeightPopupEnabled***booleanaFlag*

Sets the lightWeightPopupEnabled property, which provides a hint as to whether or not a lightweight Component should be used to contain the JComboBox, versus a heavyweight Component such as a Panel or a Window.

61 **void setMaximumRowCount***intcount*

Sets the maximum number of rows the JComboBox displays.

62 **void setModel***ComboBoxModelaModel*

Sets the data model that the JComboBox uses to obtain the list of items.

63 **void setPopupVisible***booleanv*

Sets the visibility of the popup.

64 **void setPrototypeDisplayValue***ObjectprototypeDisplayValue*

Sets the prototype display value used to calculate the size of the display for the UI portion.

65 **void setRenderer***ListCellRendereraRenderer*

Sets the renderer that paints the list items and the item selected from the list in the JComboBox field.

66 **void setSelectedIndex***intanIndex*

Selects the item at index anIndex.

67 **void setSelectedItem***ObjectanObject*

Sets the selected item in the combo box display area to the object in the argument.

68 **void setUI*ComboBoxUIui***

Sets the L&F object that renders this component.

69 **void showPopup**

Causes the combo box to display its popup window.

70 **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

JComboBox 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.showComboboxDemo();
   }

   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 showComboboxDemo(){
      headerLabel.setText("Control in action: JComboBox");

      final DefaultComboBoxModel fruitsName = new DefaultComboBoxModel();

      fruitsName.addElement("Apple");
      fruitsName.addElement("Grapes");
      fruitsName.addElement("Mango");
      fruitsName.addElement("Peer");

      final JComboBox fruitCombo = new JComboBox(fruitsName);
      fruitCombo.setSelectedIndex(0);

      JScrollPane fruitListScrollPane = new JScrollPane(fruitCombo);

      JButton showButton = new JButton("Show");

      showButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String data = "";
            if (fruitCombo.getSelectedIndex() != -1) {
               data = "Fruits Selected: "
                  + fruitCombo.getItemAt
                    (fruitCombo.getSelectedIndex());
            }
            statusLabel.setText(data);
         }
      });
      controlPanel.add(fruitListScrollPane);
      controlPanel.add(showButton);
      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