SWING - JMENU CLASS

Introduction

The Menu class represents pull-down menu component which is deployed from a menu bar.

Class declaration

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

```
public class JMenu
   extends JMenuItem
      implements Accessible, MenuElement
```

Field

Following are the fields for **java.awt.Component** class:

- **protected JMenu.WinListener popupListener** -- The window-closing listener for the popup.

Class constructors

S.N.	Constructor & Description
1	**JMenu** Constructs a new JMenu with no text.
2	**JMenu***Actiona* Constructs a menu whose properties are taken from the Action supplied.
3	**JMenu***Strings* Constructs a new JMenu with the supplied string as its text.
4	**JMenu***Strings, booleanb* Constructs a new JMenu with the supplied string as its text and specified as a tear-off menu or not.

Class methods

S.N.	Method & Description
1	**JMenuItem add***Actiona* Creates a new menu item attached to the specified Action object and appends it to the end of this menu.
2	**Component add***Componentc* Appends a component to the end of this menu.

3 **Component add***Componentc, intindex*

Adds the specified component to this container at the given position.

4 **JMenuItem add***JMenuItemmenuItem*

Appends a menu item to the end of this menu.

5 **JMenuItem add***Strings*

Creates a new menu item with the specified text and appends it to the end of this menu.

6 **void addMenuListener***MenuListenerl*

Adds a listener for menu events.

7 **void addSeparator**

Appends a new separator to the end of the menu.

8 **void applyComponentOrientation***ComponentOrientationo*

Sets the ComponentOrientation property of this menu and all components contained within it.

9 **protected PropertyChangeListener createActionChangeListener***JMenuItemb*

Returns a properly configured PropertyChangeListener which updates the control as changes to the Action occur.

10 **protected JMenuItem createActionComponent***Actiona*

Factory method which creates the JMenuItem for Actions added to the JMenu.

11 **protected JMenu.WinListener createWinListener***JPopupMenup*

Creates a window-closing listener for the popup.

12 **void doClick***intpressTime*

Programmatically performs a "click".

13 **protected void fireMenuCanceled**

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

14 **protected void fireMenuDeselected**

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

15 **protected void fireMenuSelected**

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

16 **AccessibleContext getAccessibleContext**

Gets the AccessibleContext associated with this JMenu.

17 **Component getComponent**

Returns the java.awt.Component used to paint this MenuElement.

18 **int getDelay**

Returns the suggested delay, in milliseconds, before submenus are popped up or down.

19 **JMenuItem getItem*intpos***

Returns the JMenuItem at the specified position.

20 **int getItemCount**

Returns the number of items on the menu, including separators.

21 **Component getMenuComponent*intn***

Returns the component at position n.

22 **int getMenuComponentCount**

Returns the number of components on the menu.

23 **Component[] getMenuComponents**

Returns an array of Components of the menu's subcomponents.

24 **MenuListener[]getMenuListeners**

Returns an array of all the MenuListeners added to this JMenu with addMenuListener.

25 **JPopupMenu getPopupMenu**

Returns the popupmenu associated with this menu.

26 **protected Point getPopupMenuOrigin**

Computes the origin for the JMenu's popup menu.

27 **MenuElement[] getSubElements**

Returns an array of MenuElements containing the submenu for this menu component.

28 **String getUIClassID**

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

29 **JMenuItem insert*Actiona, intpos***

Inserts a new menu item attached to the specified Action object at a given position.

30 **JMenuItem insert*JMenuItemmi, intpos***

Inserts the specified JMenuitem at a given position.

31 **void insert*Strings, intpos***

Inserts a new menu item with the specified text at a given position.

32 **void insertSeparator*intindex***

Inserts a separator at the specified position.

33 **boolean isMenuComponent*Componentc***

Returns true if the specified component exists in the submenu hierarchy.

34 **boolean isPopupMenuVisible**

Returns true if the menu's popup window is visible.

35 **boolean isSelected**

Returns true if the menu is currently selected *highlighted*.

36 **boolean isTearOff**

Returns true if the menu can be torn off.

37 **boolean isTopLevelMenu**

Returns true if the menu is a 'top-level menu', that is, if it is the direct child of a menubar.

38 **void menuSelectionChanged*booleanisIncluded***

Messaged when the menubar selection changes to activate or deactivate this menu.

39 **protected String paramString**

Returns a string representation of this JMenu.

40 **protected void processKeyEvent*KeyEventevt***

Processes key stroke events such as mnemonics and accelerators.

41 **void remove*Componentc***

Removes the component c from this menu.

42 **void remove*intpos***

Removes the menu item at the specified index from this menu.

43 **void remove*JMenuItemitem***

Removes the specified menu item from this menu.

44 **void removeAll**

Removes all menu items from this menu.

45 **void removeMenuListener*MenuListenerl***

Removes a listener for menu events.

46 **void setAccelerator*KeyStrokekeyStroke***

setAccelerator is not defined for JMenu.

47 **void setComponentOrientation*ComponentOrientationo***

Sets the language-sensitive orientation that is to be used to order the elements or text within this component.

48 **void setDelay*intd***

Sets the suggested delay before the menu's PopupMenu is popped up or down.

49 **void setMenuLocation*intx, inty***

Sets the location of the popup component.

50 **void setModel*ButtonModelnewModel***

Sets the data model for the "menu button" -- the label that the user clicks to open or close the menu.

51 **void setPopupMenuVisible*booleanb***

Sets the visibility of the menu's popup.

52 **void setSelected*booleanb***

Sets the selection status of the menu.

53 **void updateUI**

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

Methods inherited

This class inherits methods from the following classes:

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

JMenu Example

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

SwingMenuDemo.java

```
package com.tutorialspoint.gui;

import java.awt.*;
import java.awt.event.*;

public class SwingMenuDemo {
   private JFrame mainFrame;
   private JLabel headerLabel;
   private JLabel statusLabel;
   private JPanel controlPanel;

   public SwingMenuDemo(){
      prepareGUI();
   }

   public static void main(String[] args){
      SwingMenuDemo  swingMenuDemo = new SwingMenuDemo();
      swingMenuDemo.showMenuDemo();
   }

   private void prepareGUI(){
      mainFrame = new JFrame("Java SWING Examples");
      mainFrame.setSize(400,400);
      mainFrame.setLayout(new GridLayout(3, 1));

      headerLabel = new JLabel("",JLabel.CENTER );
      statusLabel = new JLabel("",JLabel.CENTER);

      statusLabel.setSize(350,100);
      mainFrame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            System.exit(0);
         }
      });
      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());

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

   private void showMenuDemo(){
      //create a menu bar
      final JMenuBar menuBar = new JMenuBar();

      //create menus
      JMenu fileMenu = new JMenu("File");
      JMenu editMenu = new JMenu("Edit");
      final JMenu aboutMenu = new JMenu("About");
      final JMenu linkMenu = new JMenu("Links");

      //create menu items
      JMenuItem newMenuItem = new JMenuItem("New");
      newMenuItem.setMnemonic(KeyEvent.VK_N);
      newMenuItem.setActionCommand("New");

      JMenuItem openMenuItem = new JMenuItem("Open");
      openMenuItem.setActionCommand("Open");

      JMenuItem saveMenuItem = new JMenuItem("Save");
      saveMenuItem.setActionCommand("Save");

      JMenuItem exitMenuItem = new JMenuItem("Exit");
      exitMenuItem.setActionCommand("Exit");
```

```
      JMenuItem cutMenuItem = new JMenuItem("Cut");
      cutMenuItem.setActionCommand("Cut");

      JMenuItem copyMenuItem = new JMenuItem("Copy");
      copyMenuItem.setActionCommand("Copy");

      JMenuItem pasteMenuItem = new JMenuItem("Paste");
      pasteMenuItem.setActionCommand("Paste");

      MenuItemListener menuItemListener = new MenuItemListener();

      newMenuItem.addActionListener(menuItemListener);
      openMenuItem.addActionListener(menuItemListener);
      saveMenuItem.addActionListener(menuItemListener);
      exitMenuItem.addActionListener(menuItemListener);
      cutMenuItem.addActionListener(menuItemListener);
      copyMenuItem.addActionListener(menuItemListener);
      pasteMenuItem.addActionListener(menuItemListener);

      final JCheckBoxMenuItem showWindowMenu =
         new JCheckBoxMenuItem("Show About", true);
      showWindowMenu.addItemListener(new ItemListener() {
         public void itemStateChanged(ItemEvent e) {
            if(showWindowMenu.getState()){
               menuBar.add(aboutMenu);
            }else{
               menuBar.remove(aboutMenu);
            }
         }
      });

      final JRadioButtonMenuItem showLinksMenu =
         new JRadioButtonMenuItem("Show Links", true);
      showLinksMenu.addItemListener(new ItemListener() {
         public void itemStateChanged(ItemEvent e) {
            if(menuBar.getMenu(3)!= null){
               menuBar.remove(linkMenu);
               mainFrame.repaint();
            }else{
               menuBar.add(linkMenu);
               mainFrame.repaint();
            }
         }
      });

      //add menu items to menus
      fileMenu.add(newMenuItem);
      fileMenu.add(openMenuItem);
      fileMenu.add(saveMenuItem);
      fileMenu.addSeparator();
      fileMenu.add(showWindowMenu);
      fileMenu.addSeparator();
      fileMenu.add(showLinksMenu);
      fileMenu.addSeparator();
      fileMenu.add(exitMenuItem);
      editMenu.add(cutMenuItem);
      editMenu.add(copyMenuItem);
      editMenu.add(pasteMenuItem);

      //add menu to menubar
      menuBar.add(fileMenu);
      menuBar.add(editMenu);
      menuBar.add(aboutMenu);
      menuBar.add(linkMenu);

      //add menubar to the frame
      mainFrame.setJMenuBar(menuBar);
      mainFrame.setVisible(true);
   }
```

```
   class MenuItemListener implements ActionListener {
      public void actionPerformed(ActionEvent e) {
         statusLabel.setText(e.getActionCommand()
         + " JMenuItem clicked.");
      }
   }
}
```

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

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

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

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

Verify the following output. *ClickonFileMenu.*