SWING - JPOPUPMENU CLASS

Introduction

Popup menu represents a menu which can be dynamically popped up at a specified position within a component.

Class declaration

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

```
public class JPopupMenu
   extends JComponent
      implements Accessible, MenuElement
```

Class constructors

S.N.	Constructor & Description
1	**JPopupMenu** Constructs a JPopupMenu without an "invoker".
2	**JPopupMenu*Stringlabel*** Constructs a JPopupMenu with the specified title.

Class methods

S.N.	Method & Description
1	**JMenuItem add*Actiona*** Appends a new menu item to the end of the menu which dispatches the specified Action object.
2	**JMenuItem add*JMenuItemmenuItem*** Appends the specified menu item to the end of this menu.
3	**JMenuItem add*Strings*** Creates a new menu item with the specified text and appends it to the end of this menu.
4	**void addMenuKeyListener*MenuKeyListenerl*** Adds a MenuKeyListener to the popup menu.
5	**void addPopupMenuListener*PopupMenuListenerl*** Adds a PopupMenu listener.
6	**void addSeparator**

Appends a new separator at the end of the menu.

7 **protected PropertyChangeListener createActionChangeListener***JMenuItemb*

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

8 **protected JMenuItem createActionComponent***Actiona*

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

9 **protected void firePopupMenuCanceled**

Notifies PopupMenuListeners that this popup menu is cancelled.

10 **protected void firePopupMenuWillBecomeInvisible**

Notifies PopupMenuListeners that this popup menu will become invisible.

11 **protected void firePopupMenuWillBecomeVisible**

Notifies PopupMenuListeners that this popup menu will become visible.

12 **AccessibleContext getAccessibleContext**

Gets the AccessibleContext associated with this JPopupMenu.

13 **Component getComponent**

Returns this JPopupMenu component.

14 **Component getComponentAtIndex***inti*

Deprecated. replaced by Container.getComponent*int*

15 **int getComponentIndex***Componentc*

Returns the index of the specified component.

16 **static boolean getDefaultLightWeightPopupEnabled**

Gets the defaultLightWeightPopupEnabled property, which by default is true.

17 **Component getInvoker**

Returns the component which is the 'invoker' of this popup menu.

18 **String getLabel**

Returns the popup menu's label

19 **Insets getMargin**

Returns the margin, in pixels, between the popup menu's border and its containees.

20 **MenuKeyListener[]getMenuKeyListeners**

Returns an array of all the MenuKeyListeners added to this JPopupMenu with addMenuKeyListener.

21 **PopupMenuListener[] getPopupMenuListeners**

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

22 **SingleSelectionModel getSelectionModel**

Returns the model object that handles single selections.

23 **MenuElement[] getSubElements**

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

24 **PopupMenuUI getUI**

Returns the look and feel L&F object that renders this component.

25 **String getUIClassID**

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

26 **void insert***Actiona, intindex*

Inserts a menu item for the specified Action object at a given position.

27 **void insert***Componentcomponent, intindex*

Inserts the specified component into the menu at a given position.

28 **boolean isBorderPainted**

Checks whether the border should be painted.

29 **boolean isLightWeightPopupEnabled**

Gets the lightWeightPopupEnabled property.

30 **boolean isPopupTrigger***MouseEvente*

Returns true if the MouseEvent is considered a popup trigger by the JPopupMenu's currently installed UI.

31 **boolean isVisible**

Returns true if the popup menu is visible *currentlybeingdisplayed*.

32 **void menuSelectionChanged***booleanisIncluded*

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

33 **void pack**

Lays out the container so that it uses the minimum space needed to display its contents.

34 **protected void paintBorder***Graphicsg*

Paints the popup menu's border if the borderPainted property is true.

35 **protected String paramString**

Returns a string representation of this JPopupMenu.

36 **protected void processFocusEvent***FocusEventevt*

Processes focus events occurring on this component by dispatching them to any registered FocusListener objects.

37 **protected void processKeyEvent***KeyEventevt*

Processes key stroke events such as mnemonics and accelerators.

38 **void processKeyEvent***KeyEvente, MenuElement***[]***path, MenuSelectionManagermanager*

Processes a key event forwarded from the MenuSelectionManager and changes the menu selection, if necessary, by using MenuSelectionManager's API.

39 **void processMouseEvent***MouseEventevent, MenuElement***[]***path, MenuSelectionManagermanager*

This method is required to conform to the MenuElement interface, but it not implemented.

40 **void remove***intpos*

Removes the component at the specified index from this popup menu.

41 **void removeMenuKeyListener***MenuKeyListenerl*

Removes a MenuKeyListener from the popup menu.

42 **void removePopupMenuListener***PopupMenuListenerl*

Removes a PopupMenu listener.

43 **void setBorderPainted***booleanb*

Sets whether the border should be painted.

44 **static void setDefaultLightWeightPopupEnabled***booleanaFlag*

Sets the default value of the lightWeightPopupEnabled property.

45 **void setInvoker***Componentinvoker*

Sets the invoker of this popup menu -- the component in which the popup menu menu is to be displayed.

46 **void setLabel***Stringlabel*

Sets the popup menu's label.

47 **void setLightWeightPopupEnabled***booleanaFlag*

Sets the value of the lightWeightPopupEnabled property, which by default is true.

48 **void setLocation***intx, inty*

Sets the location of the upper left corner of the popup menu using x, y coordinates.

49 **void setPopupSize***Dimensiond*

Sets the size of the Popup window using a Dimension object.

50 **void setPopupSize***intwidth, intheight*

Sets the size of the Popup window to the specified width and height.

51 **void setSelected***Componentsel*

Sets the currently selected component, This will result in a change to the selection model.

52 **void setSelectionModel***SingleSelectionModelmodel*

Sets the model object to handle single selections.

53 **void setUI***PopupMenuUIui*

Sets the L&F object that renders this component.

54 **void setVisible***booleanb*

Sets the visibility of the popup menu.

55 **void show***Componentinvoker, intx, inty*

Displays the popup menu at the position x,y in the coordinate space of the component invoker.

56 **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

JPopupMenu 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.showPopupMenuDemo();
   }

   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 showPopupMenuDemo(){
      final JPopupMenu editMenu = new JPopupMenu("Edit");

      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();

      cutMenuItem.addActionListener(menuItemListener);
      copyMenuItem.addActionListener(menuItemListener);
      pasteMenuItem.addActionListener(menuItemListener);

      editMenu.add(cutMenuItem);
      editMenu.add(copyMenuItem);
      editMenu.add(pasteMenuItem);

      mainFrame.addMouseListener(new MouseAdapter() {
         public void mouseClicked(MouseEvent e) {
            editMenu.show(mainFrame, e.getX(), e.getY());
```

```
            }
        });
        mainFrame.add(editMenu);
        mainFrame.setVisible(true);
    }
}
```

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. *Clickinthemiddleonthescreen.*