SWING - JFRAME CLASS

Introduction

The class **JFrame** is an extended version of java.awt.Frame that adds support for the JFC/Swing component architecture.

Class declaration

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

```
public class JFrame
   extends Frame
      implements WindowConstants, Accessible, RootPaneContainer
```

Field

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

- **protected AccessibleContext accessibleContext** --The accessible context property.
- **static int EXIT_ON_CLOSE** -- The exit application default window close operation.
- **protected JRootPane rootPane** -- The JRootPane instance that manages the contentPane and optional menuBar for this frame, as well as the glassPane.
- **protected boolean rootPaneCheckingEnabled** -- If true then calls to add and setLayout will be forwarded to the contentPane.

Class constructors

S.N.	Constructor & Description
1	**JFrame** Constructs a new frame that is initially invisible.
2	**JFrame***GraphicsConfigurationgc* Creates a Frame in the specified GraphicsConfiguration of a screen device and a blank title.
3	**JFrame***Stringtitle* Creates a new, initially invisible Frame with the specified title.
4	**JFrame***Stringtitle, GraphicsConfigurationgc* Creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device.

Class methods

S.N.	Method & Description

1 **protected void addImpl***Componentcomp, Objectconstraints, intindex*

Adds the specified child Component.

2 **protected JRootPane createRootPane**

Called by the constructor methods to create the default rootPane.

3 **protected void frameInit**

Called by the constructors to init the JFrame properly.

4 **AccessibleContext getAccessibleContext**

Gets the AccessibleContext associated with this JFrame.

5 **Container getContentPane**

Returns the contentPane object for this frame.

6 **int getDefaultCloseOperation**

Returns the operation that occurs when the user initiates a "close" on this frame.

7 **Component getGlassPane**

Returns the glassPane object for this frame.

8 **Graphics getGraphics**

Creates a graphics context for this component.

9 **JMenuBar getJMenuBar**

Returns the menubar set on this frame.

10 **JLayeredPane getLayeredPane**

Returns the layeredPane object for this frame.

11 **JRootPane getRootPane**

Returns the rootPane object for this frame.

12 **TransferHandler getTransferHandler**

Gets the transferHandler property.

13 **static boolean isDefaultLookAndFeelDecorated**

Returns true if newly created JFrames should have their Windows decorations provided by the current look and feel.

14 **protected boolean isRootPaneCheckingEnabled**

Returns whether calls to add and setLayout are forwarded to the contentPane.

15 **protected String paramString**

Returns a string representation of this JFrame.

16 **protected void processWindowEvent***WindowEvente*

Processes window events occurring on this component.

17 **void remove***Componentcomp*

Removes the specified component from the container.

18 **void repaint***longtime, intx, inty, intwidth, intheight*

Repaints the specified rectangle of this component within time milliseconds.

19 **void setContentPane***ContainercontentPane*

Sets the contentPane property.

20 **void setDefaultCloseOperation***intoperation*

Sets the operation that will happen by default when the user initiates a "close" on this frame.

21 **static void setDefaultLookAndFeelDecorated***booleandefaultLookAndFeelDecorated*

Provides a hint as to whether or not newly created JFrames should have their Window decorations *suchasborders, widgetstoclosethewindow, title. . .* provided by the current look and feel.

22 **void setGlassPane***ComponentglassPane*

Sets the glassPane property.

23 **void setIconImage***Imageimage*

Sets the image to be displayed as the icon for this window.

24 **void setJMenuBar***JMenuBarmenubar*

Sets the menubar for this frame.

25 **void setLayeredPane***JLayeredPanelayeredPane*

Sets the layeredPane property.

26 **void setLayout***LayoutManagermanager*

Sets the LayoutManager.

27 **protected void setRootPane***JRootPaneroot*

Sets the rootPane property.

28 **protected void setRootPaneCheckingEnabled***booleanenabled*

Sets whether calls to add and setLayout are forwarded to the contentPane.

29 **void setTransferHandler***TransferHandlernewHandler*

Sets the transferHandler property, which is a mechanism to support transfer of data into this component.

30 **void update***Graphicsg*

Just calls paint*g*.

Methods inherited

This class inherits methods from the following classes:

- java.awt.Frame
- java.awt.Window
- java.awt.Container
- java.awt.Component
- java.lang.Object

JFrame Example

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

SwingContainerDemo.java

```
package com.tutorialspoint.gui;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

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

   public SwingContainerDemo(){
      prepareGUI();
   }

   public static void main(String[] args){
      SwingContainerDemo  swingContainerDemo = new SwingContainerDemo();
      swingContainerDemo.showJFrameDemo();
   }

   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);

      msglabel = new JLabel("Welcome to TutorialsPoint SWING Tutorial.", JLabel.CENTER);

      controlPanel = new JPanel();
      controlPanel.setLayout(new FlowLayout());

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

   private void showJFrameDemo(){
      headerLabel.setText("Container in action: JFrame");

      final JFrame frame = new JFrame();
      frame.setSize(300, 300);
      frame.setLayout(new FlowLayout());
      frame.add(msglabel);
      frame.addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent windowEvent){
            frame.dispose();
         }
      });
      JButton okButton = new JButton("Open a Frame");
      okButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            statusLabel.setText("A Frame shown to the user.");
            frame.setVisible(true);
         }
      });
      controlPanel.add(okButton);
      mainFrame.setVisible(true);
   }
}
```

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

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

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

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

Verify the following output