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:

```java
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

<table>
<thead>
<tr>
<th>S.N.</th>
<th>Constructor &amp; Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>JFrame</td>
</tr>
<tr>
<td></td>
<td>Constructs a new frame that is initially invisible.</td>
</tr>
<tr>
<td>2</td>
<td>JFrame.GraphicsConfigurationgc</td>
</tr>
<tr>
<td></td>
<td>Creates a Frame in the specified GraphicsConfiguration of a screen device and a blank title.</td>
</tr>
<tr>
<td>3</td>
<td>JFrame.StringTitle</td>
</tr>
<tr>
<td></td>
<td>Creates a new, initially invisible Frame with the specified title.</td>
</tr>
<tr>
<td>4</td>
<td>JFrame.StringTitle, GraphicsConfigurationgc</td>
</tr>
<tr>
<td></td>
<td>Creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device.</td>
</tr>
</tbody>
</table>

Class methods

<table>
<thead>
<tr>
<th>S.N.</th>
<th>Method &amp; Description</th>
</tr>
</thead>
</table>
protected void addImplComponent comp, Object constraints, int index

Adds the specified child Component.

protected JRootPane createRootPane

Called by the constructor methods to create the default rootPane.

protected void frameInit

Called by the constructors to init the JFrame properly.

AccessibleContext getAccessibleContext

Gets the AccessibleContext associated with this JFrame.

Container getContentPane

Returns the contentPane object for this frame.

int getDefaultCloseOperation

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

Component getGlassPane

Returns the glassPane object for this frame.

Graphics getGraphics

Creates a graphics context for this component.

JMenuBar getJMenuBar

Returns the menubar set on this frame.

JLayeredPane getLayeredPane

Returns the layeredPane object for this frame.

JRootPane getRootPane

Returns the rootPane object for this frame.

TransferHandler getTransferHandler

Gets the transferHandler property.

static boolean isDefaultLookAndFeelDecorated

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

protected boolean isRootPaneCheckingEnabled

Returns whether calls to add and setLayout are forwarded to the contentPane.
protected String paramString
Returns a string representation of this JFrame.

protected void processWindowEvent
Processes window events occurring on this component.

void remove
Removes the specified component from the container.

void repaint
Repaints the specified rectangle of this component within time milliseconds.

void setContentPane
Sets the contentPane property.

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

static void setDefaultLookAndFeelDecorated
Provides a hint as to whether or not newly created JFrames should have their Window decorations such as borders, widgetstoclosethewindow, title... provided by the current look and feel.

void setGlassPane
Sets the glassPane property.

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

void setJMenuBar
Sets the menubar for this frame.

void setLayeredPane
Sets the layeredPane property.

void setLayout
Sets the LayoutManager.

protected void setRootPane
Sets the rootPane property.

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

29  void setTransferHandler

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

30  void updateGraphics

Just calls paint.

Methods inherited

This class inherits methods from the following classes:

- java.awt.Frame
- java.awt.Window
- java.awt.Container
- java.awt.Componen
- 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

```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
A Frame shown to the user.