Java Swing

Used to create window-based applications.

What is GUI in Java?

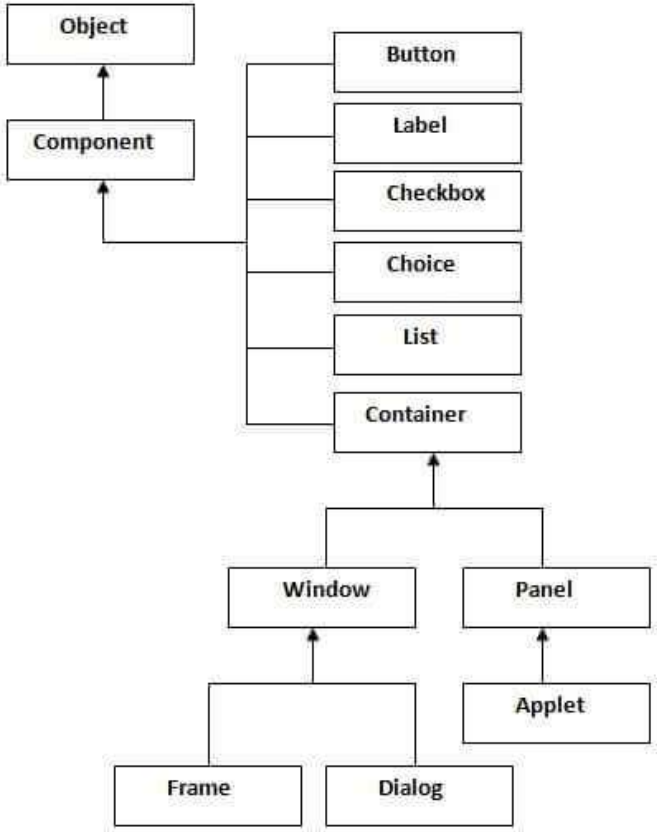
- GUI (Graphical User Interface) In Java gives programmers an easy-to-use visual experience to build Java applications.
- It is mainly made of graphical components like buttons, labels, windows, etc. through which the user can interact with the applications.
- Swing GUI in Java plays an important role in building easy interfaces.

Swing

- Swing in Java is a Graphical User Interface (GUI) toolkit that includes a rich set of widgets. It is a part of Java Foundation Classes(JFC), which is an API for Java programs that provide GUI.
- Swing includes packages that let you make a sophisticated set of GUI components for your Java applications and it is platform-independent.
- The Swing library is built on top of the Java Abstract Widget Toolkit (AWT), an older, platform dependent GUI toolkit.
- You can use the Java GUI components like button, textbox, etc. from the library and do not have to create the components from scratch.

AWT

- Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
- Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.



Difference between Swing and AWT

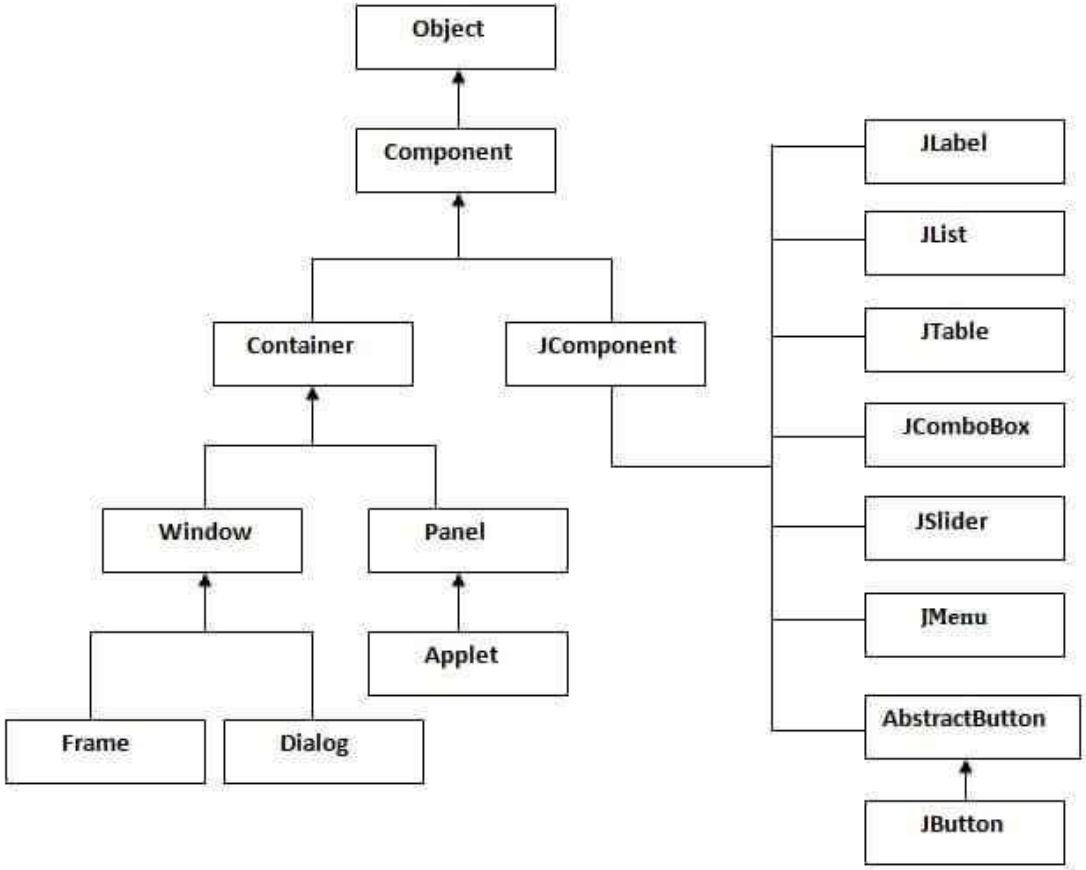
1. AWT components are platform-dependent.
2. AWT components are heavyweight.
3. AWT doesn't support pluggable look and feel.
4. AWT provides less components than Swing.
5. AWT doesn't follows MVC(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.

1. Java swing components are platform-independent.
2. Swing components are lightweight.
3. Swing supports pluggable look and feel.
4. Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5. Swing follows MVC.

Swing Features

- Light Weight – Swing components are independent of native Operating System's API as Swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.
- Rich Controls – Swing provides a rich set of advanced controls like Tree, TabbedPane, slider, colorpicker, and table controls.
- Highly Customizable – Swing controls can be customized in a very easy way as visual apperance is independent of internal representation.
- Pluggable look-and-feel – SWING based GUI Application look and feel can be changed at run-time, based on available values.

Swing Hierarchy



Container

Container classes are classes that can have other components on it. So for creating a GUI, we need at least one container object. There are 3 types of containers.

1. Panel: It is a pure container and is not a window in itself. The sole purpose of a Panel is to organize the components on to a window.
2. Frame: It is a fully functioning window with its title and icons.
3. Dialog: It can be thought of like a pop-up window that pops out when a message has to be displayed. It is not a fully functioning window like the Frame.

Component Class

A Component is the abstract base class for the non menu user-interface controls of SWING. Component represents an object with graphical representation

Container Class

A Container is a component that can contain other SWING components

JComponent Class

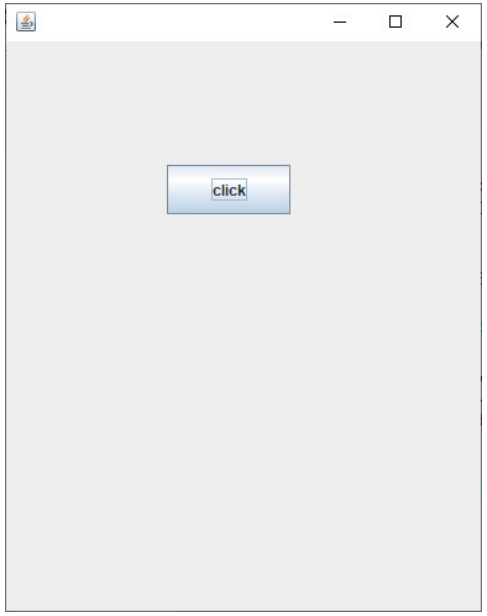
A JComponent is a base class for all SWING UI components. In order to use a SWING component that inherits from JComponent, the component must be in a containment hierarchy whose root is a top-level SWING container

```java
import javax.swing.*;
public class Driver {
    public static void main(String[] args) {
        JFrame f=new JFrame();//creating instance of JFrame

        JButton b=new JButton("click");//creating instance of JButton
        b.setBounds(130,100,100, 40);//x axis, y axis, width, height

        f.add(b);//adding button in JFrame

        f.setSize(400,500);//400 width and 500 height
        f.setLayout(null);//using no layout managers
        f.setVisible(true);//making the frame visible
    }
}
```



A simple Java Swing-I

We are creating one button and adding it on the JFrame object inside the main() method.

```java
import javax.swing.*;
public class Driver {
    JFrame f;
    Driver(){
    f=new JFrame();//creating instance of JFrame

    JButton b=new JButton("click");//creating instance of JButton
    b.setBounds(130,100,100, 40);

    f.add(b);//adding button in JFrame

    f.setSize(400,500);//400 width and 500 height
    f.setLayout(null);//using no layout managers
    f.setVisible(true);//making the frame visible
    }

    public static void main(String[] args) {
    new Driver();
    }
}
```

A simple Java Swing-II

We can also write all the codes of creating JFrame, JButton and method call inside the java constructor.

```java
import javax.swing.*;
public class Driver extends JFrame{
//inheriting JFrame
    JFrame f;
    Driver(){
        JButton b=new JButton("click");//create button
        b.setBounds(130,100,100, 40);

        add(b);//adding button on frame
        setSize(400,500);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String[] args) {
        new Driver();
    }
}
```

A simple Java Swing-III

We can also inherit the JFrame class, so there is no need to create the instance of JFrame class explicitly.

Next
Layout Manager