JAVA SWING

Introduction to GUI

Computer users today expect to interact with their computers using a graphical user interface (GUI)

Java can be used to write GUI programs ranging from simple applets which run on a Web page to sophisticated stand-alone applications.

One big difference is that GUI programs are event-driven

Event-driven Programming

1. Events are user actions
    - ✓ Clicking on a button
    - ✓ Pressing a key on the keyboard generate events
2. The program must respond to these events as they occur
3. Objects are everywhere in GUI programming
    - ✓ Events are objects
    - ✓ GUI components such as buttons and menus are objects

Java Swing is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

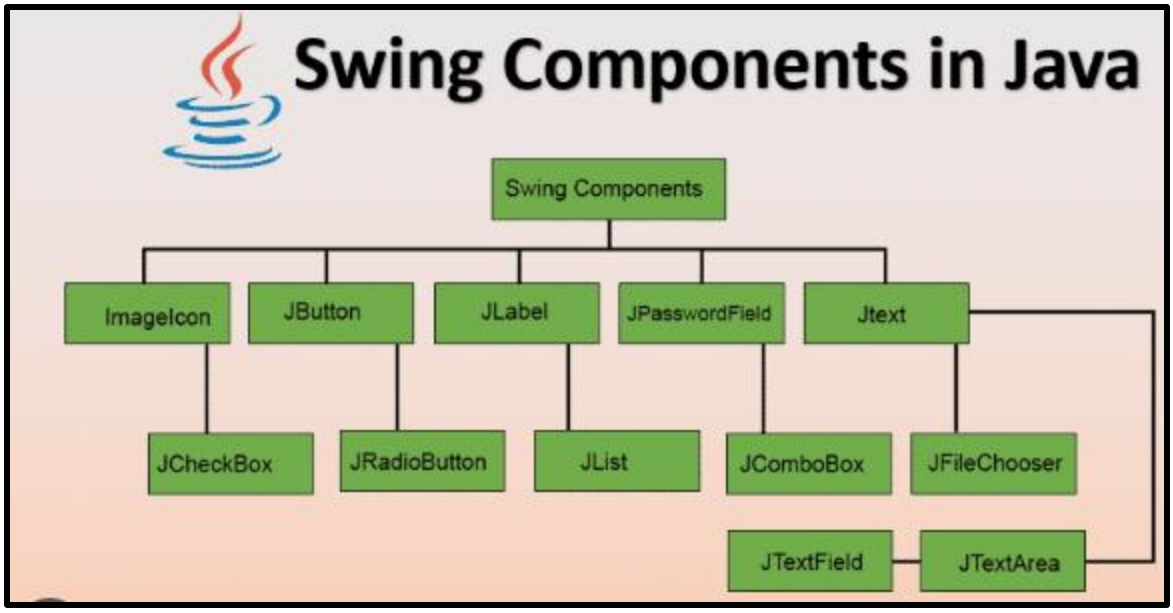
The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

Difference between AWT and Swing

No.	Java AWT	Java Swing
1)	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4)	AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
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.	Swing **follows MVC**.

What is JFC

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

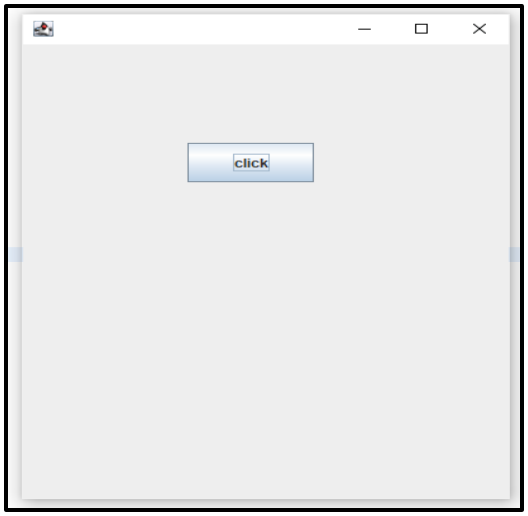


Simple Java Swing Example

Let's see a simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

*practice.java

```
1  import javax.swing.*;
2  public class practice {
3  public static void main(String[] args) {
4  JFrame f=new JFrame();//creating instance of JFrame
5
6  JButton b=new JButton("click");//creating instance of JButton
7  b.setBounds(130,100,100, 40);//x axis, y axis, width, height
8
9  f.add(b);//adding button in JFrame
10
11 f.setSize(400,500);//400 width and 500 height
12 f.setLayout(null);//using no layout managers
13 f.setVisible(true);//making the frame visible
14 }
15 }
16
17
```

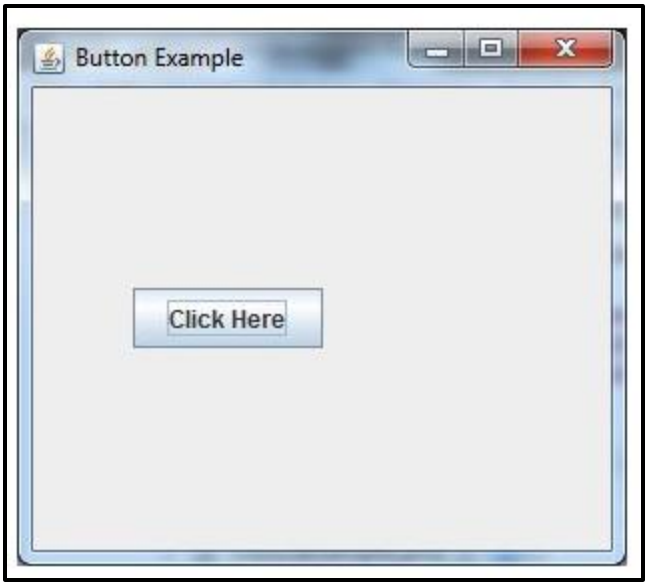


Java JButton

The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed.

Java JButton Example

```
import javax.swing.*;
public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

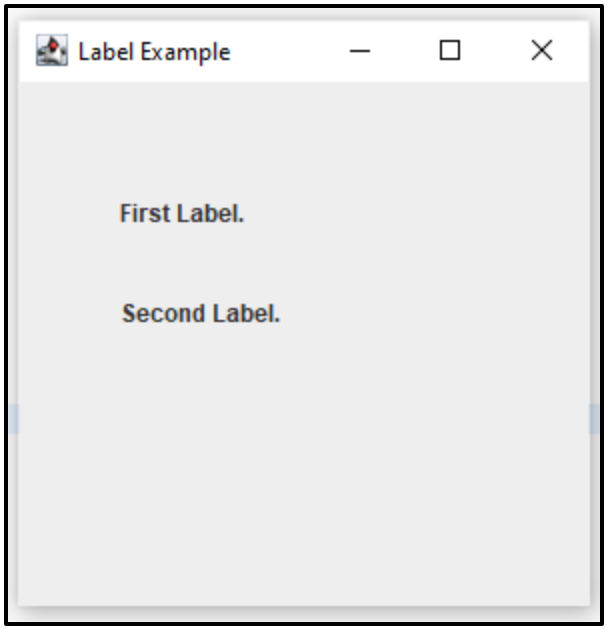


Java JLabel

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

*practice.java gopush.java

```java
import javax.swing.*;
class practice  {
public static void main(String args[])
    {
    JFrame f= new JFrame("Label Example");
    JLabel l1,l2;
    l1=new JLabel("First Label.");
    l1.setBounds(50,50, 100,30);
    l2=new JLabel("Second Label.");
    l2.setBounds(50,100, 100,30);
    f.add(l1); f.add(l2);
    f.setSize(300,300);
    f.setLayout(null);
    f.setVisible(true);
    }
    }
```

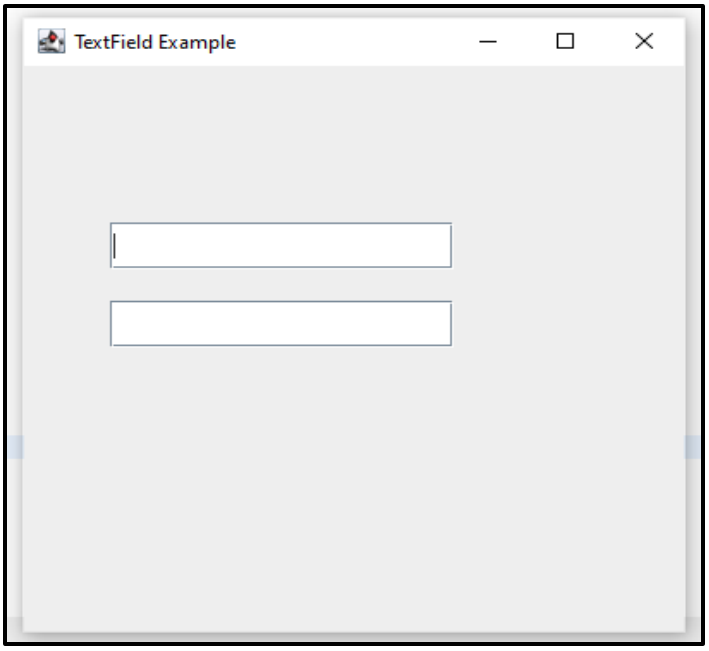


Java JTextField

The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.

practice.java gopush.java

```java
import javax.swing.*;
class practice
{
public static void main(String args[])
    {
    JFrame f= new JFrame("TextField Example");
    JTextField t1,t2;
    t1=new JTextField();
    t1.setBounds(50,100, 200,30);
    t2=new JTextField();
    t2.setBounds(50,150, 200,30);
    f.add(t1); f.add(t2);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
    }
    }
```



JMenu Class

It inherits the JMenuItem class, and is a pull down menu component which is displayed from the menu bar.

*practice.java gopush.java

```java
import javax.swing.*;
class practice{
JMenu menu;
JMenuItem a1,a2;
practice()
{
JFrame a = new JFrame("Example");
menu = new JMenu("options");
JMenuBar m1 = new JMenuBar();
a1 = new JMenuItem("example");
a2 = new JMenuItem("example1");
menu.add(a1);
menu.add(a2);
m1.add(menu);
a.setJMenuBar(m1);
a.setSize(400,400);
a.setLayout(null);
a.setVisible(true);
}
public static void main(String args[])
{
new practice();
}
}
```