Topics to cover:

1. GUI programming with Swing
2. Programming practice

1. GUI programming with Swing

1.1 What is GUI?

A graphical user interface (GUI) is a digital interface in which a user interacts with graphical components such as icons, buttons, and menus. In a GUI, the visuals displayed in the user interface convey information relevant to the user, as well as actions that they can take.

Today, it's hard to imagine computers without GUIs. But, there was a time when we didn't even have a mouse cursor.

So far, we have covered the basic programming constructs (such as variables, data types, decision, loop, array and method) and introduced the important concept of Object-Oriented Programming (OOP). As discussed, OOP permits higher level of abstraction than traditional Procedural-Oriented Languages (such as C). You can create high-level abstract data types called *classes* to mimic real-life things. These classes are self-contained and are *reusable*.

In this article, we will show how you can *reuse* the graphics classes provided in JDK for constructing your own GUI applications. Writing your own graphics classes (and re-inventing the wheels) is mission impossible! These graphics classes, developed by expert programmers, are highly complex and involve many advanced *design patterns*. However, re-using them are not so difficult, if you follow the API documentation,

samples and templates provided. There are ways of learning GUI programming in Java through different APIs such as AWT, Swing, JavaFX etc. However, AWT is pretty much outdated these days (although a part commonly used in only some cases, we will see that later). We will learn Swing in this tutorial.

1.2 What is Swing in Java?

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

1.3 Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.

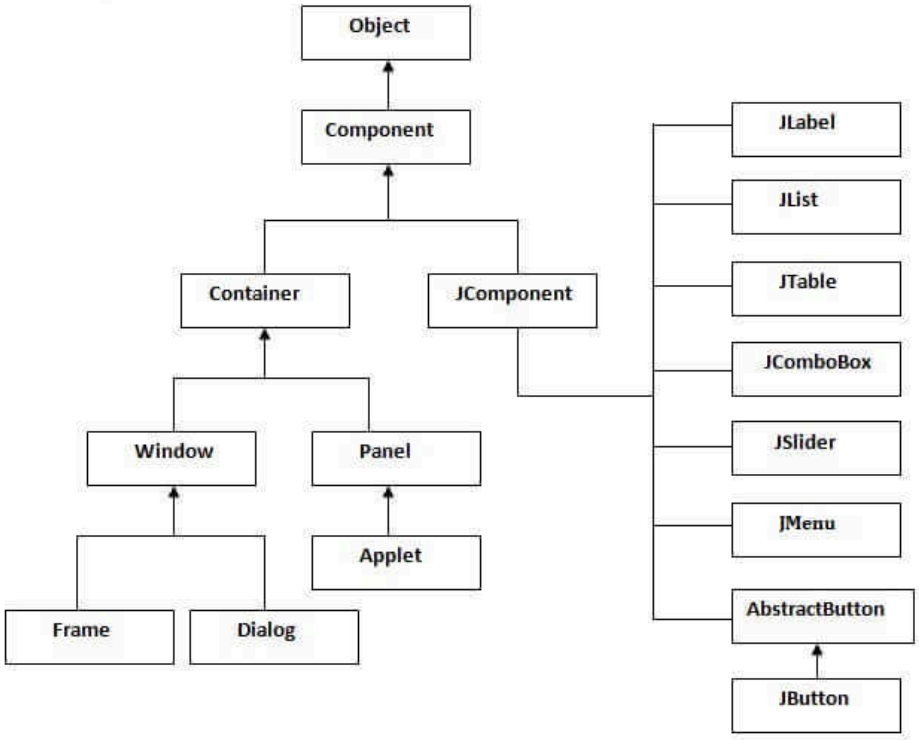

Fig 1: Hierarchy of Swing API

1.4 Commonly used methods of component class

The methods of Component class are widely used in java swing that are given below.

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false.

Table 1: Most commonly used methods of Swing API

2. Programming practice

There are two ways to create a frame:

- By creating the object of Frame class (association)
- By extending Frame class (inheritance)

We can write the code of swing inside the main(), constructor or any other method.

Before starting programming with Swing in shell, we need to first install the package called “openjdk-11-jre” or similar from the terminal in Ubuntu. If you are using virtual box, it can be done by first entering root access through typing “su root” and then entering the password. Afterwards, you need to type something like follows:

```
apt-get install openjdk-11-jre
```

Once the above step is done (i.e., successfully installed), you can simply exit from the root user and start writing the program. If you are using Ubuntu linux directly, you can simply type the above command without entering the root. Note that the steps may be different for Windows systems. Please explore steps for Windows systems.

2.1 Examples

Program 1:

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

```
import javax.swing.*;

    public class FirstSwingExample {

    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
}
}
```

Output:

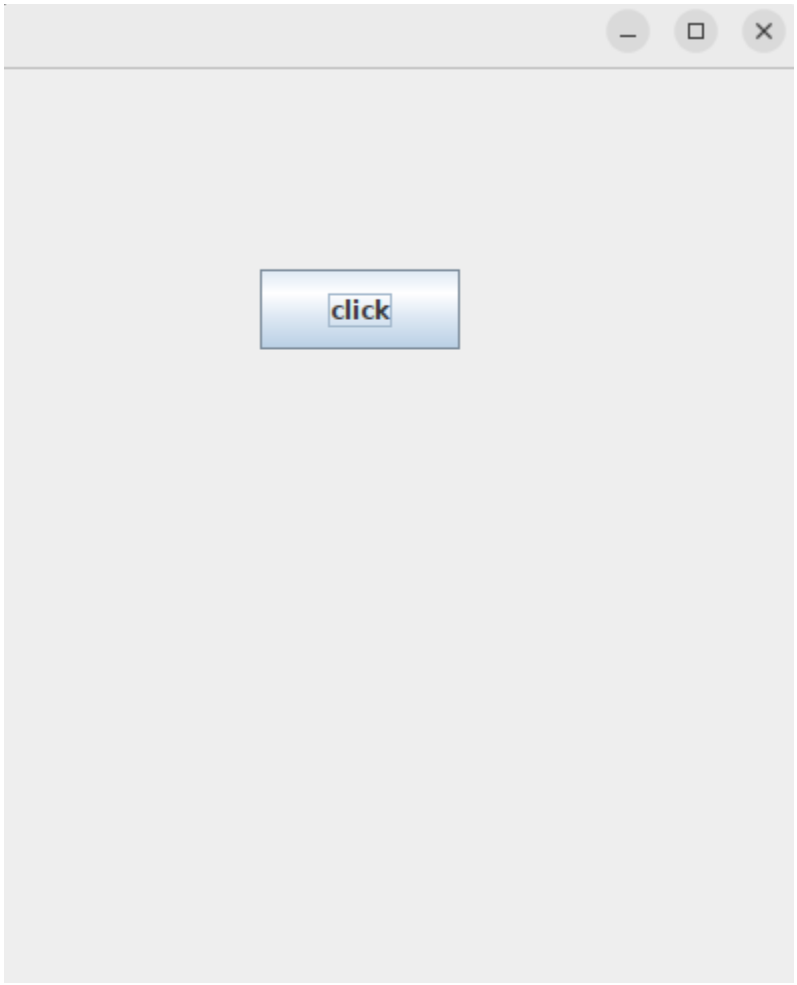

Fig 2: Our first window using swing programming

Program 2:

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

```
import javax.swing.*;

    public class SwingExampleConstructor {

    JFrame f;

    SwingExampleConstructor(){
```

```
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 SwingExampleConstructor();
}
}
```

Output:

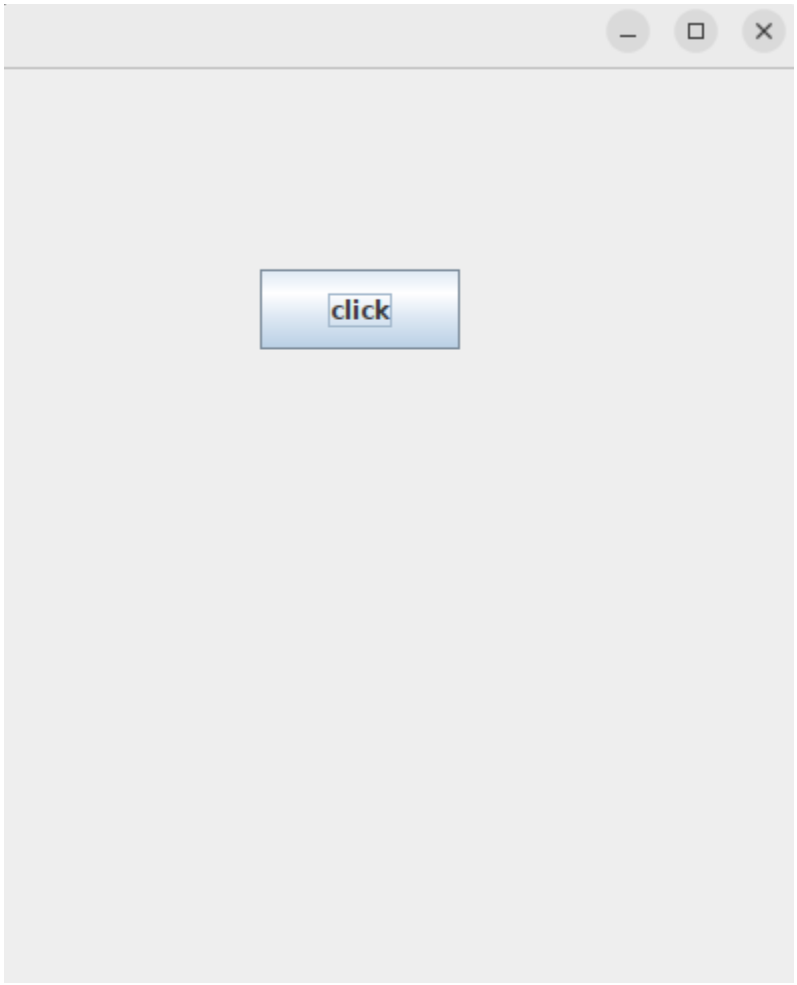

Fig 3: Same window as program 1 using constructor

Program 3:

We can also write the same code using inheritance. We can simply inherit the JFrame class, so there is no need to create the instance of JFrame class explicitly.

```
import javax.swing.*;


public class JFrameExtend extends JFrame{//inheriting JFrame
```

```
JFrame f;
JFrameExtend(){
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 JFrameExtend();
}}
```

Output:

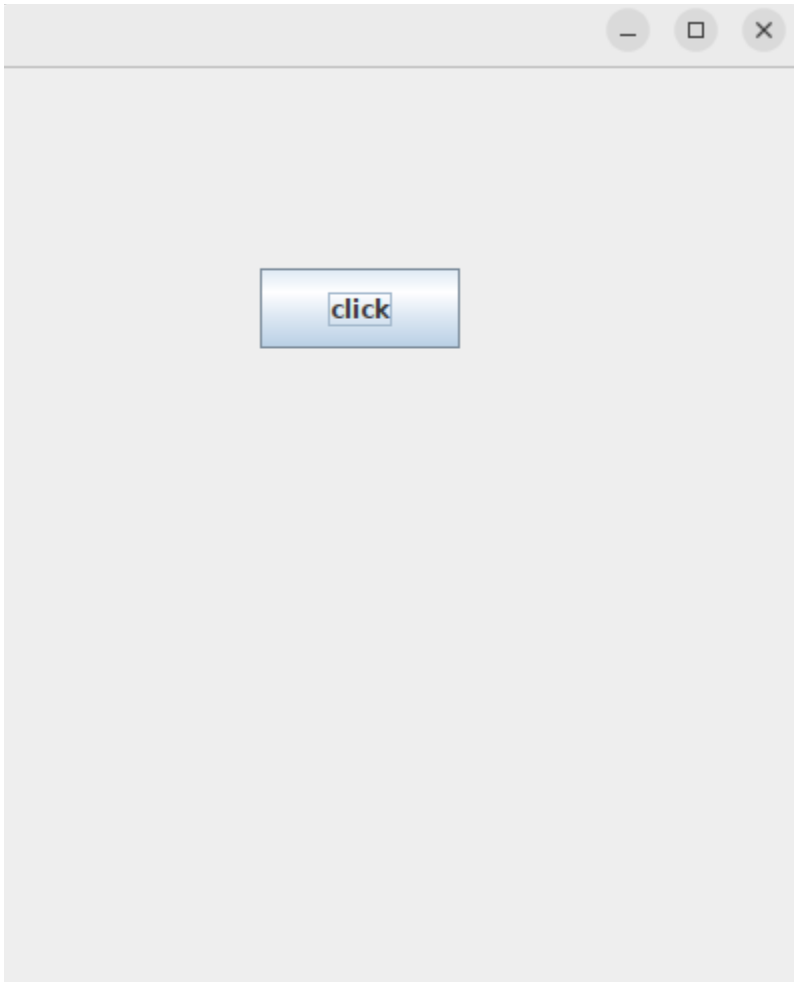

Fig 4: Same window as program 1 using inheritance