Java Programming

Lecture-15

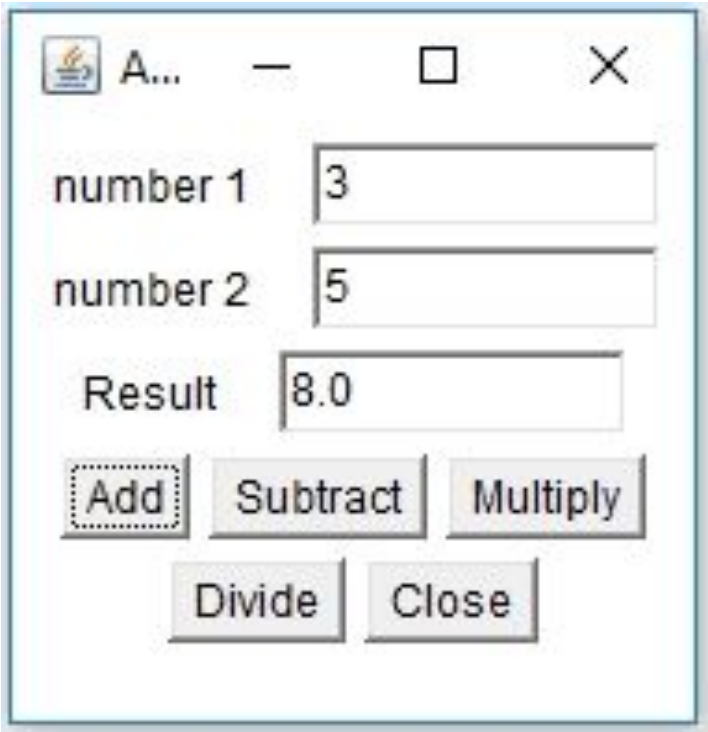
AWT & Swing

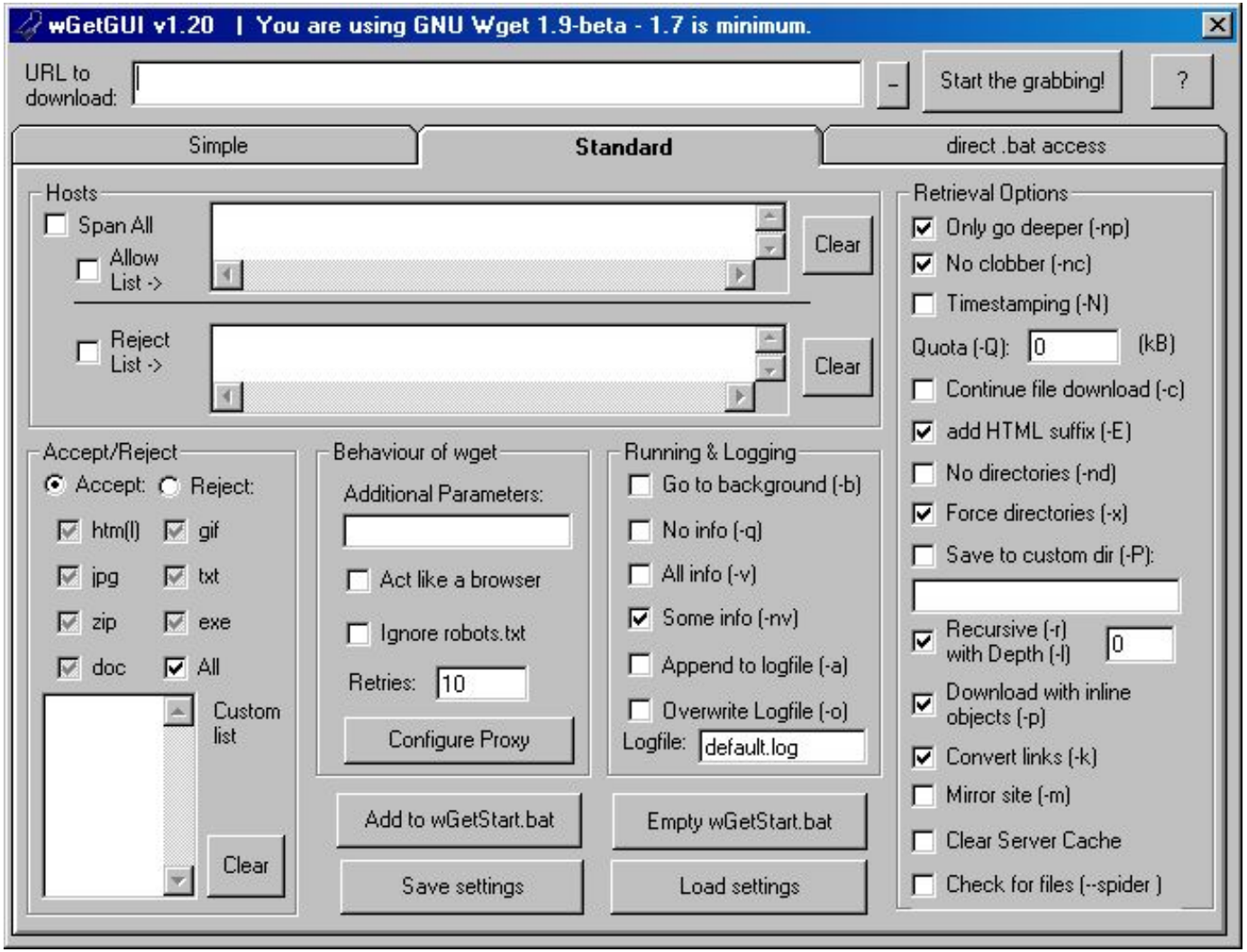
- ***Graphical User Interface***, is a user-friendly visual experience builder for Java applications.

- It comprises graphical units like buttons, labels, windows, etc. via which users can connect with an application.

- Swing is commonly used application to create GUIs in Java.

GUI Examples




wGetGUI v1.20 | You are using GNU Wget 1.9-beta - 1.7 is minimum.
URL to download:
Start the grabbing!
?
Simple
Standard
direct .bat access
Hosts
Span All
Allow List ->
Clear
Reject List ->
Clear
Accept/Reject
Accept:
Reject:
htm(l)
gif
jpg
txt
zip
exe
doc
All
Custom list
Clear
Behaviour of wget
Additional Parameters:
Act like a browser
Ignore robots.txt
Retries:
10
Configure Proxy
Running & Logging
Go to background (-b)
No info (-q)
All info (-v)
Some info (-nv)
Append to logfile (-a)
Overwrite Logfile (-o)
Logfile:
default.log
Add to wGetStart.bat
Empty wGetStart.bat
Save settings
Load settings
Retrieval Options
Only go deeper (-np)
No clobber (-nc)
Timestamping (-N)
Quota (-Q):
0
(kB)
Continue file download (-c)
add HTML suffix (-E)
No directories (-nd)
Force directories (-x)
Save to custom dir (-P):
Recursive (-r) with Depth (-l)
0
Download with inline objects (-p)
Convert links (-k)
Mirror site (-m)
Clear Server Cache
Check for files (--spider)

Java AWT

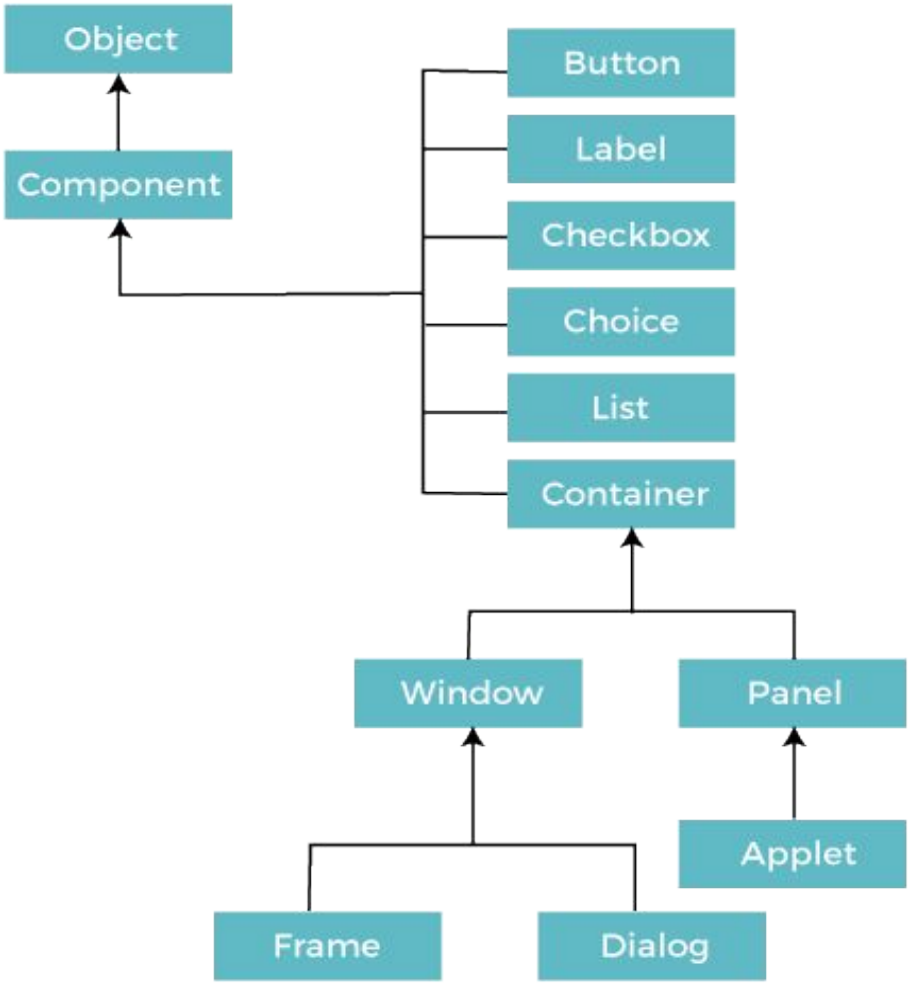
- **Java AWT** (Abstract Window Toolkit) is *an API to develop Graphical User Interface (GUI) or windows-based applications* in Java.

- Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system.

- AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

- The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Java AWT Hierarchy



Components

- All the elements like the button, text fields, scroll bars, etc. are called components.

- In Java AWT, there are classes for each component as shown in hierarchy diagram.

- In order to place every component in a particular position on a screen, we need to add them to a container.

Container

- The Container is a component in AWT that can contain another components like buttons, textfields, labels etc.

- The classes that extends Container class are known as container such as **Frame, Dialog** and **Panel**.

- It is basically a screen where the components are placed at their specific locations.

- Thus it contains and controls the layout of components.

Types of containers:

There are four types of containers in Java AWT:

1. Window
2. Panel
3. Frame
4. Dialog

Window

- The window is the container that have no borders and menu bars.
- You must use frame, dialog or another window for creating a window.
- We need to create an instance of Window class to create this container.

Panel

- The Panel is the container that doesn't contain title bar, border or menu bar.
- It is generic container for holding the components.
- It can have other components like button, text field etc.
- An instance of Panel class creates a container, in which we can add components.

Frame

- The Frame is the container that contain title bar and border and can have menu bars.

- It can have other components like button, text field, scrollbar etc.

- Frame is most widely used container while developing an AWT application.

Useful Methods of Component Class

Method	Description
public void add(Component c)	Inserts a component on this component.
public void setSize(int width,int height)	Sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	Defines the layout manager for the component.
public void setVisible(boolean status)	Changes the visibility of the component, by default false.

To create simple AWT example, you need a frame. There are two ways to create a GUI using Frame in AWT.

1. By extending Frame class (**inheritance**)

2. By creating the object of Frame class (**association**)

AWT Example by Inheritance

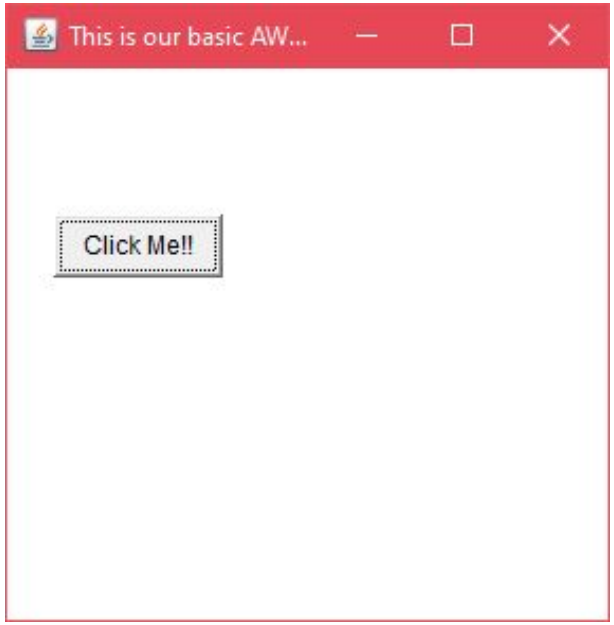
Let's see a simple example of AWT where we are inheriting Frame class. Here, we are showing Button component on the Frame.

Example 1:

```
// importing Java AWT class
import java.awt.*;

// extending Frame class to our class AWTExample1
public class AWTExample1 extends Frame {

  // initializing using constructor
  AWTExample1() {

    // creating a button
    Button b = new Button("Click Me!!");

    // setting button position on screen
    b.setBounds(30,100,80,30);

    // adding button into frame
    add(b);
```

```
    // frame size 300 width and 300 height
    setSize(300,300);

    // setting the title of Frame
    setTitle("This is our basic AWT example");

    // no layout manager
    setLayout(null);

    // now frame will be visible, by default it is not visible
    setVisible(true);
}

// main method
public static void main(String args[]) {

// creating instance of Frame class
AWTExample1 f = new AWTExample1();

}

}
```

Output:

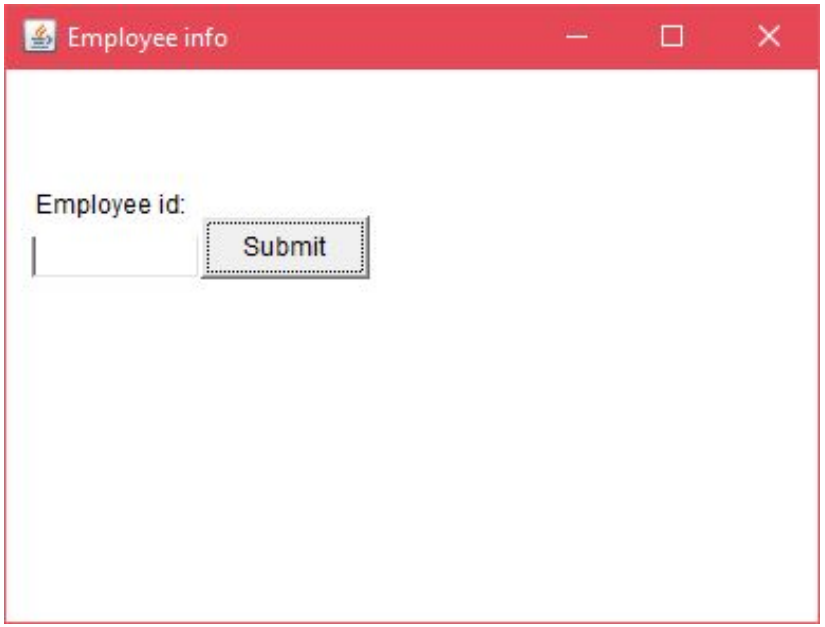


Example2

```
// importing Java AWT class
import java.awt.*;

// class AWTExample2 directly creates instance of Frame class
class AWTExample2 {

  // initializing using constructor
  AWTExample2() {

    // creating a Frame
    Frame f = new Frame();

    // creating a Label
    Label l = new Label("Employee id:");

    // creating a Button
    Button b = new Button("Submit");

    // creating a TextField
    TextField t = new TextField();
```

```
    // adding components into frame
    f.add(b);
    f.add(l);
    f.add(t);

    // frame size 300 width and 300 height
    f.setSize(400,300);

    // setting the title of frame
    f.setTitle("Employee info");

    // no layout
    f.setLayout(null);

    // setting visibility of frame
    f.setVisible(true);
}

// main method
public static void main(String args[]) {

// creating instance of Frame class
AWTExample2 awt_obj = new AWTExample2();

}

}
```

Output:



Java Swing

- **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.

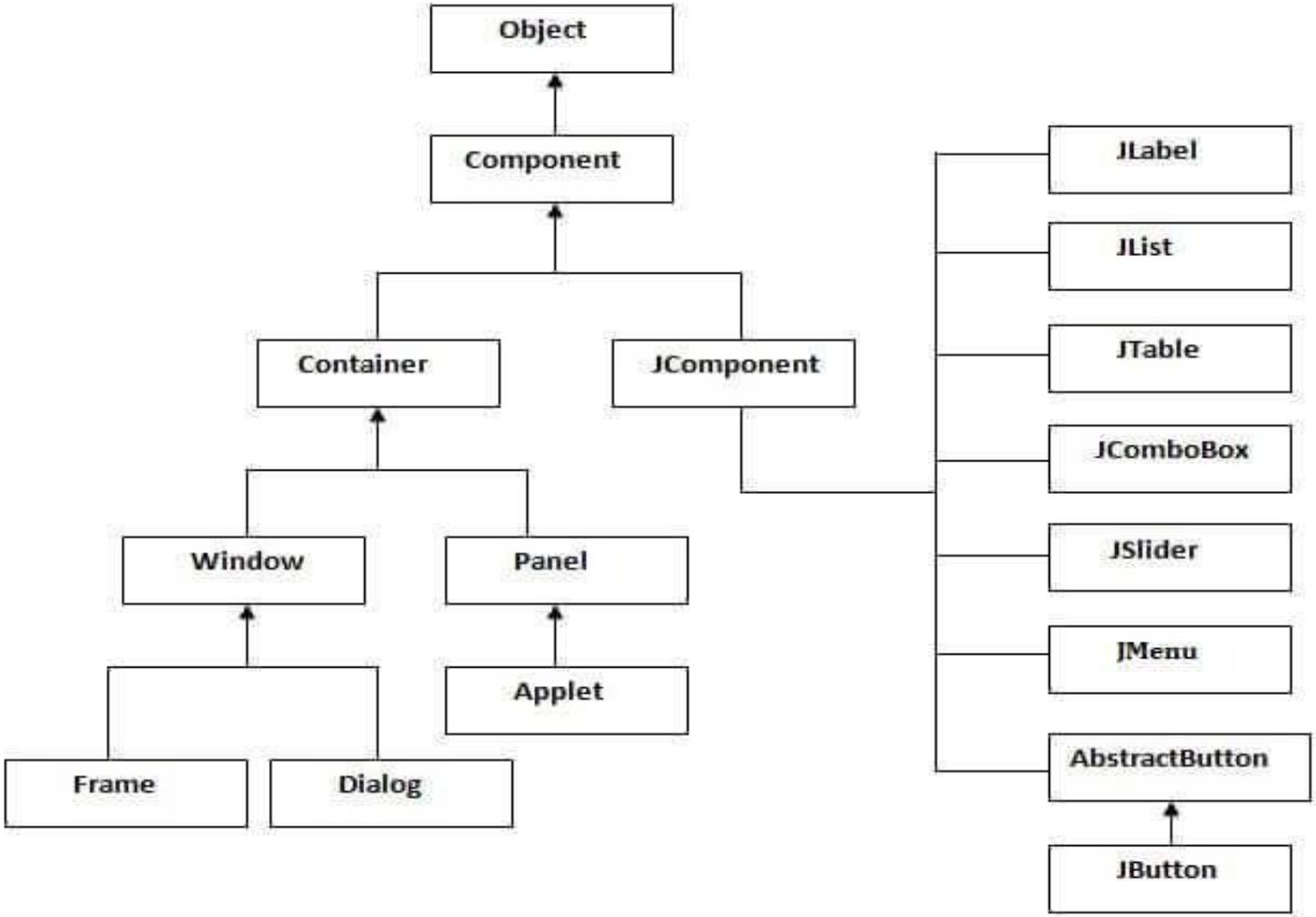
- Unlike AWT, Java Swing provides platform-independent and lightweight components.

- 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**.

Hierarchy of Java Swing classes



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.

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.

Simple Java Swing Example

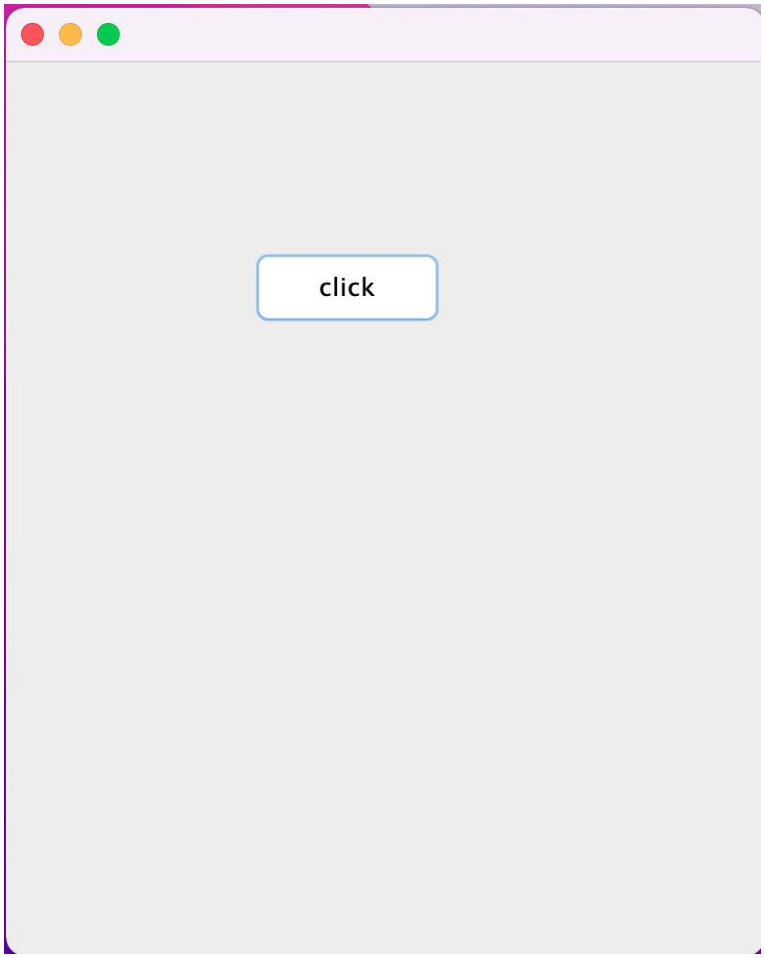
```
import javax.swing.*;
public class FirstSwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();//creating instance of JFrame

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:

Example of Swing by Association inside constructor

```
import javax.swing.*;
public class Simple {
JFrame f;
Simple(){
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 Simple();
}
}
```

Output:



Simple example of Swing by inheritance

```
import javax.swing.*;
public class Simple2 extends JFrame{//inheriting JFrame
JFrame f;
Simple2(){
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 Simple2();
}}
```

Output: