

Java Swing tutorial 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

There are many differences between java awt and swing that are given below.

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.

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.

Java Swing Examples

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

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

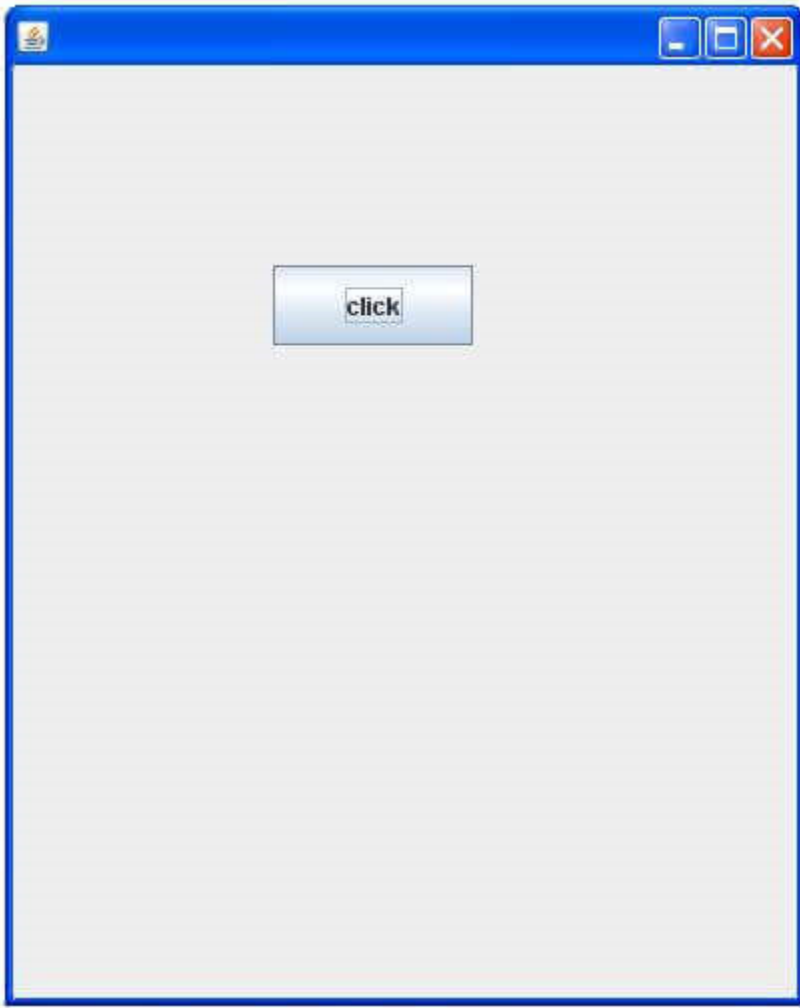
File: FirstSwingExample.java

```
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
    }
}
```



Example of Swing by Association inside constructor

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

File: Simple.java

```
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();  
}  
}
```

The `setBounds(int xaxis, int yaxis, int width, int height)` is used in the above example that sets the position of the button.

Simple example of Swing by inheritance

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

File: Simple2.java

```
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();
}}

```

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. It inherits AbstractButton class.

JButton class declaration

Let's see the declaration for javax.swing.JButton class.

1. **public class** JButton **extends** AbstractButton **implements** Accessible

Commonly used Constructors:

Constructor	Description
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.
JButton(Icon i)	It creates a button with the specified icon object.

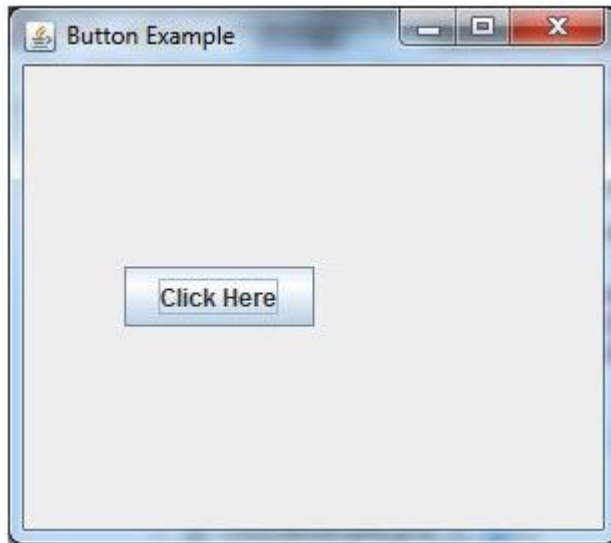
Commonly used Methods of AbstractButton class:

Methods	Description
void setText(String s)	It is used to set specified text on button
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

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);
}
}
```

Output:

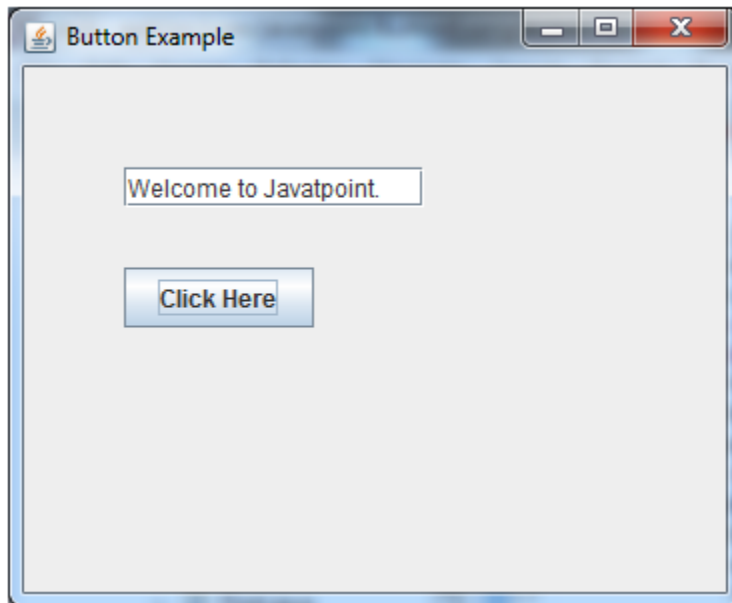


Java JButton Example with ActionListener

```
import java.awt.event.*;
import javax.swing.*;

public class ButtonExample {
    public static void main(String[] args) {
        JFrame f=new JFrame("Button Example");
        final JTextField tf=new JTextField();
        tf.setBounds(50,50, 150,20);
        JButton b=new JButton("Click Here");
        b.setBounds(50,100,95,30);
        b.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e){
                tf.setText("Welcome to Javatpoint.");
            }
        });
        f.add(b);f.add(tf);
        f.setSize(400,400);
        f.setLayout(null);
        f.setVisible(true);
    }
}
```

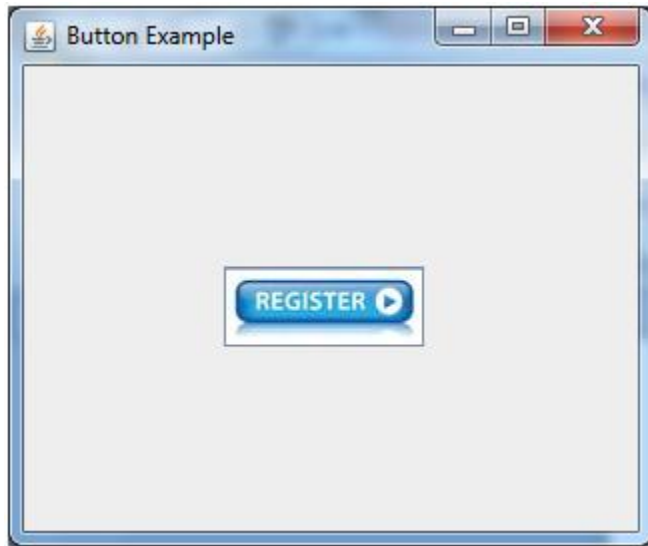

Output:



Example of displaying image on the button:

```
import javax.swing.*;
public class ButtonExample{
    ButtonExample(){
        JFrame f=new JFrame("Button Example");
        JButton b=new JButton(new ImageIcon("D:\\icon.png"));
        b.setBounds(100,100,100, 40);
        f.add(b);
        f.setSize(300,400);
        f.setLayout(null);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public static void main(String[] args) {
        new ButtonExample();
    }
}
```

Output:



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.

JLabel class declaration

Let's see the declaration for javax.swing.JLabel class.

1. **public class** JLabel **extends** JComponent **implements** SwingConstants, Accessible

Commonly used Constructors:

Constructor	Description
JLabel()	Creates a JLabel instance with no image and with an empty string for the title.
JLabel(String s)	Creates a JLabel instance with the specified text.
JLabel(Icon i)	Creates a JLabel instance with the specified image.

JLabel(String s, Icon i, int horizontalAlignment)	Creates a JLabel instance with the specified text, image, and horizontal alignment.
---	---

Commonly used Methods:

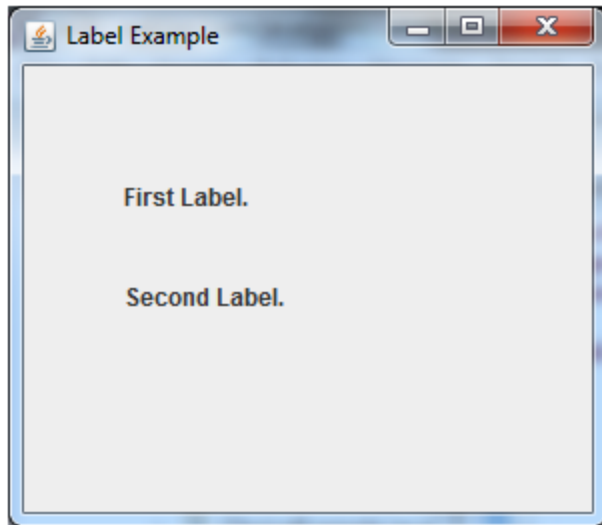
Methods	Description
String getText()	t returns the text string that a label displays.
void setText(String text)	It defines the single line of text this component will display.
void setHorizontalAlignment(int alignment)	It sets the alignment of the label's contents along the X axis.
Icon getIcon()	It returns the graphic image that the label displays.
int getHorizontalAlignment()	It returns the alignment of the label's contents along the X axis.

Java JLabel Example

```
import javax.swing.*;
class LabelExample
{
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);  
}  
}
```

Output:



Java JLabel Example with ActionListener

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
public class LabelExample extends Frame implements ActionListener{  
    JTextField tf; JLabel l; JButton b;  
    LabelExample(){  
        tf=new JTextField();  
        tf.setBounds(50,50, 150,20);  
        l=new JLabel();  
        l.setBounds(50,100, 250,20);  
        b=new JButton("Find IP");  
        b.setBounds(50,150,95,30);  
        b.addActionListener(this);  
        add(b);add(tf);add(l);  
        setSize(400,400);  
        setLayout(null);
```

```
setVisible(true);  
}  
public void actionPerformed(ActionEvent e) {  
    try{  
        String host=tf.getText();  
        String ip=java.net.InetAddress.getByName(host).getHostAddress();  
        l.setText("IP of "+host+" is: "+ip);  
    }catch(Exception ex){System.out.println(ex);}  
}  
public static void main(String[] args) {  
    new LabelExample();  
} }  
Output:
```

