Java Swings

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.

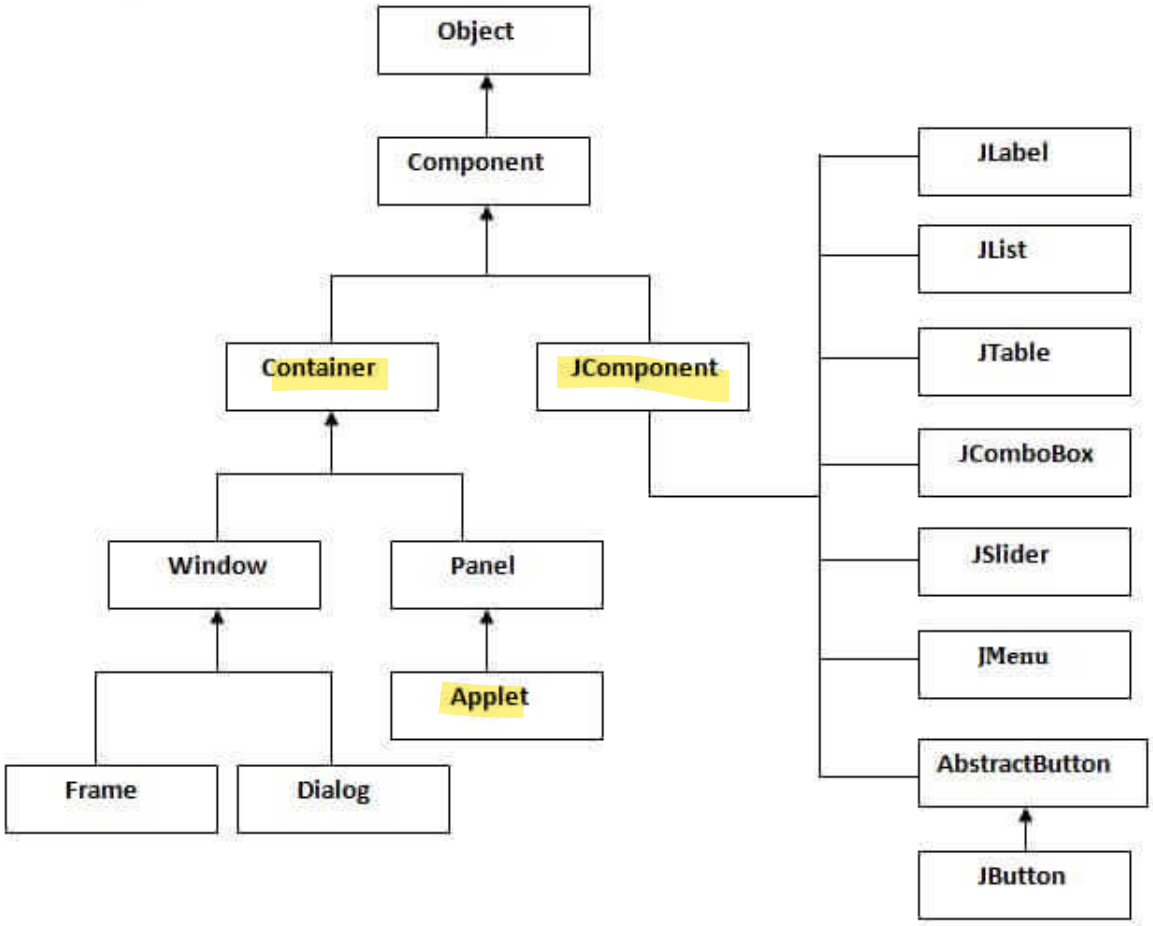
Java AWT	**Java Swing**
AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
AWT components are **heavyweight**.	Swing components are **lightweight**.
AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
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.

Hierarchy of Java Swing classes

The hierarchy of java swing API is given below.



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
}
}
```

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.

A key element of graphical user interfaces (GUIs) in Java that is used to create interactive buttons is the JButton class. Users can click these labelled buttons to initiate particular operations within the application. Because JButton offers a platform-independent implementation, it can be used in a variety of settings and operating systems. It is descended from the AbstractButton class, which offers shared functionality for all button kinds in the Swing GUI framework and Java's Abstract Window Toolkit (AWT). Developers can improve the usability and interactivity of their Java programmes by adding sensible user interface components to their JButton objects through configuration.

JButton class declaration

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

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

Commonly used Constructors:

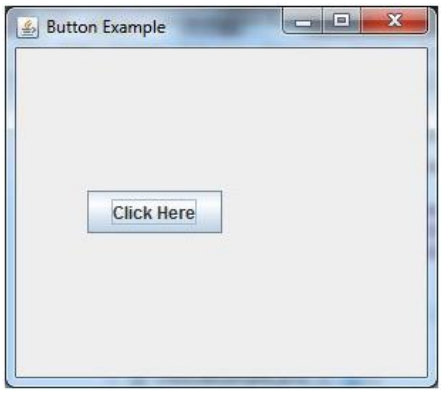
Constructor	**Description**
JButton()	**It creates a button with no text and icon.**

JButton(String s)	**It creates a button with the specified text.**
JButton(Icon i)	**It creates a button with the specified icon object.**
JButton(String s, Icon i)	**Creates a button with both specified text and icon.**

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



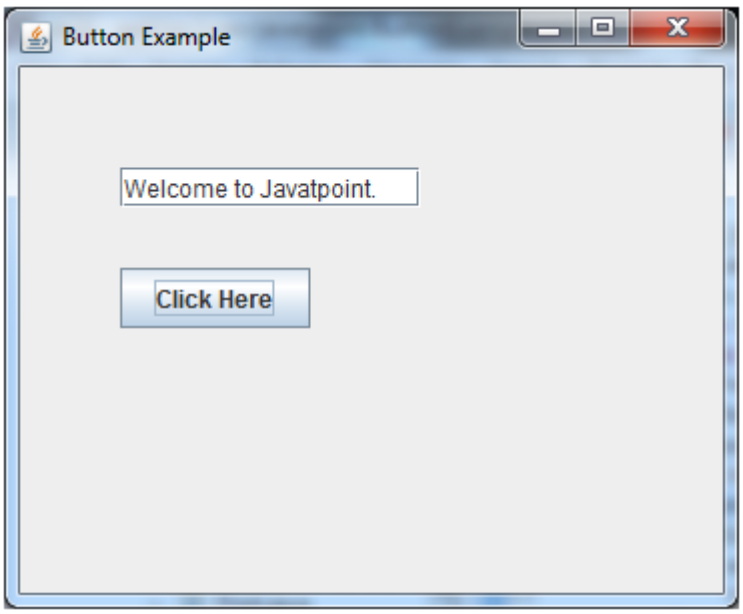
Java JButton Example with ActionListener

Filename: ButtonExample.java

```
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:



Explanation

This extended Java code enhances the previous example by incorporating functionality to the JButton. Upon clicking the JButton labeled "Click Here," the JTextField below it is updated with the text "Welcome to Javatpoint." This functionality is achieved by adding an ActionListener to the button. When the button is clicked, the actionPerformed method of the ActionListener interface is invoked, and the setText method is called on the JTextField to update its content. The JFrame is displayed with both the button and the text field, utilizing absolute positioning through null layout. This example demonstrates how to handle events in Swing applications, enabling interactive user experiences.

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.

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);
}
}
```

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:

Java JTextField

JTextField is a Swing component in Java that allows users to input single-line text. It is essentially a blank space where users can type characters. It is commonly used in GUI (Graphical User Interface) applications to accept textual input from users. The object of a JTextField class is a text component that allows the editing of single-line text. It inherits the JTextComponent class.

JTextField Class Declaration

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

1. **public class** JTextField **extends** JTextComponent **implements** SwingConstants

Commonly used Constructors:

Constructor	**Description**
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

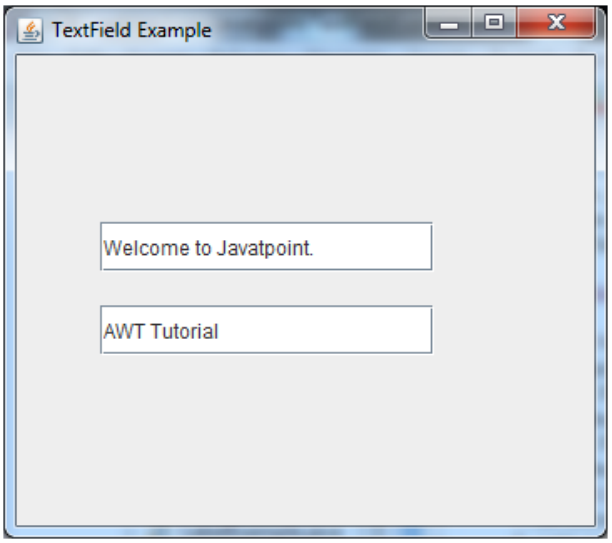
Java JTextField Example

TextFieldExample.java

```
import javax.swing.*;
// Main class TextFieldExample
class TextFieldExample
{
   public static void main(String args[])
   {
      // Creating a JFrame object with title "TextField Example."
      JFrame f= new JFrame("TextField Example");
      // Creating two JTextField objects
      JTextField t1, t2;
      // Initializing the first JTextField with default text "Welcome to Javatpoint."
      t1 = new JTextField("Welcome to Javatpoint.");
      // Setting the position and size of the first JTextField
      t1.setBounds(50,100, 200,30);
      // Initializing the second JTextField with default text "AWT Tutorial"
      t2 = new JTextField("AWT Tutorial");
      // Setting the position and size of the second JTextField
      t2.setBounds(50,150, 200,30);
      // Adding JTextFields to the JFrame
      f.add(t1);
      f.add(t2);
      // Setting the size of the JFrame
      f.setSize(400,400);
      // Setting layout to null to use absolute positioning
      f.setLayout(null);
      // Making the JFrame visible
      f.setVisible(true);
   }
```

```
}
```

Output:



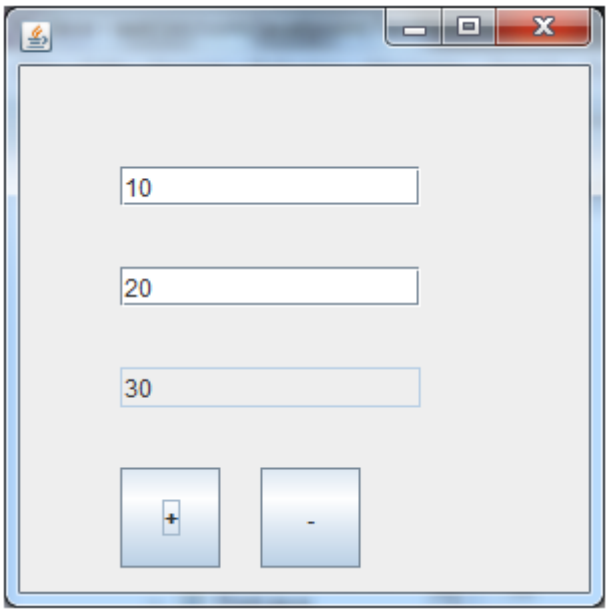
Java JTextField Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class TextFieldExample implements ActionListener {
    // Declaration of JTextField and JButton objects
    JTextField tf1, tf2, tf3;
    JButton b1, b2;
    // Constructor
    TextFieldExample() {
        // Creating a JFrame object
        JFrame f = new JFrame();
        // Creating JTextField objects
        tf1 = new JTextField();
        tf1.setBounds(50, 50, 150, 20);
        tf2 = new JTextField();
        tf2.setBounds(50, 100, 150, 20);
        tf3 = new JTextField();
        tf3.setBounds(50, 150, 150, 20);
```

```
    // Making tf3 non-editable
    tf3.setEditable(false);
    // Creating JButton objects
    b1 = new JButton("+");
    b1.setBounds(50, 200, 50, 50);
    b2 = new JButton("-");
    b2.setBounds(120, 200, 50, 50);
    // Adding ActionListener to buttons
    b1.addActionListener(this);
    b2.addActionListener(this);
    // Adding components to the JFrame
    f.add(tf1);
    f.add(tf2);
    f.add(tf3);
    f.add(b1);
    f.add(b2);
    // Setting JFrame size and layout
    f.setSize(300, 300);
    f.setLayout(null);
    // Making JFrame visible
    f.setVisible(true);
}
  // actionPerformed method from ActionListener interface
public void actionPerformed(ActionEvent e) {
    // Retrieving text from text fields
    String s1 = tf1.getText();
    String s2 = tf2.getText();
    // Converting string inputs to integers
    int a = Integer.parseInt(s1);
    int b = Integer.parseInt(s2);
```

```
        // Variable to hold the result
        int c = 0;
        // Checking which button is clicked
        if (e.getSource() == b1) {
            c = a + b; // Addition
        } else if (e.getSource() == b2) {
            c = a - b; // Subtraction
        }
        // Converting result back to a string
        String result = String.valueOf(c);
        // Setting the result in the third text field
        tf3.setText(result);
    }
    public static void main(String[] args) {
        // Creating an instance of TextFieldExample
        new TextFieldExample();
    }
}
```

Output:



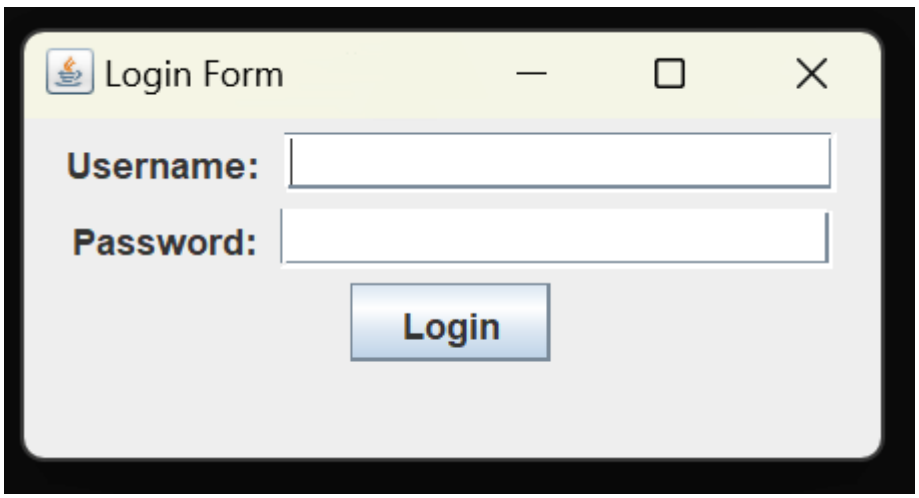
Example: Creating a Simple Login Form

LoginForm.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class LoginForm {
   public static void main(String[] args) {
      // Create the main frame
      JFrame frame = new JFrame("Login Form");
      // Create a panel to hold components
      JPanel panel = new JPanel();
      // Create text fields for username and password
      JTextField usernameField = new JTextField(20);
      JPasswordField passwordField = new JPasswordField(20);
      // Create a button for login
      JButton loginButton = new JButton("Login");
      // Add components to the panel
      panel.add(new JLabel("Username: "));
      panel.add(usernameField);
      panel.add(new JLabel("Password: "));
      panel.add(passwordField);
      panel.add(loginButton);
      // Add action listener to the login button
      loginButton.addActionListener(new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            // Retrieve username and password entered by the user
            String username = usernameField.getText();
            String password = new String(passwordField.getPassword());
            // Validate username and password
```

```
            if (username.equals("admin") && password.equals("password")) {
                // Show a success message if login is successful
                JOptionPane.showMessageDialog(frame, "Login successful!");
            } else {
                // Show an error message if login fails
                JOptionPane.showMessageDialog(frame, "Invalid username or password");
            }
        }
    });
    // Add the panel to the frame
    frame.add(panel);
    // Set frame properties
    frame.setSize(300, 150);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

Output:



Example: Calculating the Area of Rectangle

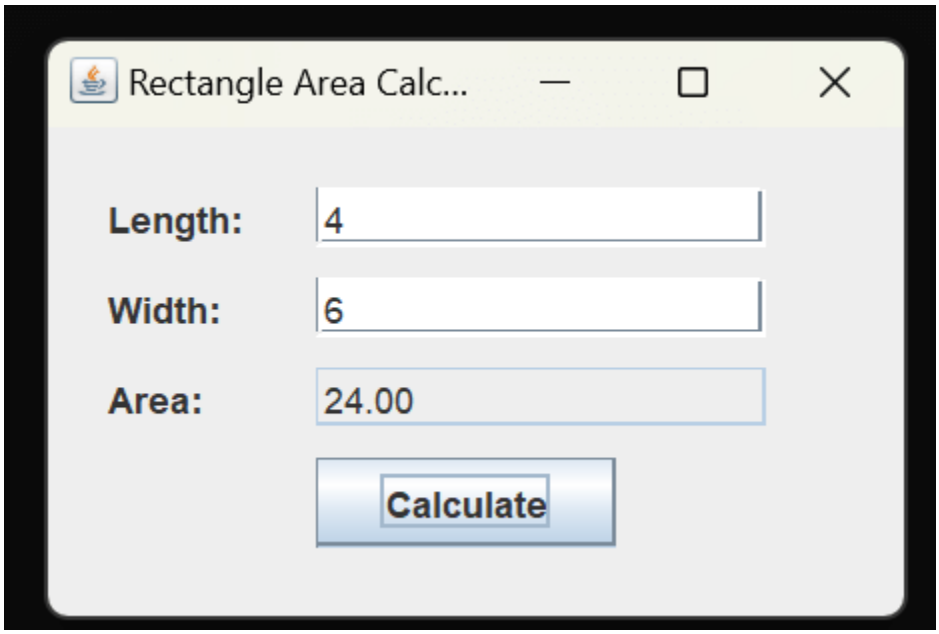
RectangleAreaCalculator.java

```
import javax.swing.*;
import java.awt.event.*;
public class RectangleAreaCalculator implements ActionListener {
   JTextField lengthField, widthField, resultField;
   JButton calculateButton;
   RectangleAreaCalculator() {
      // Create JFrame
      JFrame frame = new JFrame("Rectangle Area Calculator");
      frame.setSize(300, 200);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setLayout(null);
      // Create JLabels
      JLabel lengthLabel = new JLabel("Length:");
      lengthLabel.setBounds(20, 20, 60, 20);
      JLabel widthLabel = new JLabel("Width:");
      widthLabel.setBounds(20, 50, 60, 20);
      JLabel resultLabel = new JLabel("Area:");
      resultLabel.setBounds(20, 80, 60, 20);
      // Create JTextFields
      lengthField = new JTextField();
      lengthField.setBounds(90, 20, 150, 20);
      widthField = new JTextField();
      widthField.setBounds(90, 50, 150, 20);
      resultField = new JTextField();
      resultField.setBounds(90, 80, 150, 20);
      resultField.setEditable(false);
      // Create JButton
      calculateButton = new JButton("Calculate");
```

```
        calculateButton.setBounds(90, 110, 100, 30);
        calculateButton.addActionListener(this);
        // Add components to the frame
        frame.add(lengthLabel);
        frame.add(widthLabel);
        frame.add(resultLabel);
        frame.add(lengthField);
        frame.add(widthField);
        frame.add(resultField);
        frame.add(calculateButton);
        // Make the frame visible
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        // Calculate area when the button is clicked
        if (e.getSource() == calculateButton) {
            try {
                // Get length and width from text fields
                double length = Double.parseDouble(lengthField.getText());
                double width = Double.parseDouble(widthField.getText());
                // Calculate area
                double area = length * width;
                // Display result in resultField
                resultField.setText(String.format("%.2f", area));
            } catch (NumberFormatException ex) {
                // Handle invalid input
                JOptionPane.showMessageDialog(null, "Please enter valid numbers for length
and width.");
            }
        }
    }
```

```
    public static void main(String[] args) {
        // Create an instance of RectangleAreaCalculator
        new RectangleAreaCalculator();
    }
}
```

Output:



Java JPasswordField

The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.

JPasswordField class declaration

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

1. **public class** JPasswordField **extends** JTextField

Commonly used Constructors:

Constructor	**Description**
JPasswordField()	Constructs a new JPasswordField, with a default document, null starting text string, and 0 column width.

JPasswordField(int columns)	Constructs a new empty JPasswordField with the specified number of columns.
JPasswordField(String text)	Constructs a new JPasswordField initialized with the specified text.
JPasswordField(String text, int columns)	Construct a new JPasswordField initialized with the specified text and columns.

Java JPasswordField Example

```
import javax.swing.*;
public class PasswordFieldExample {
   public static void main(String[] args) {
   JFrame f=new JFrame("Password Field Example");
    JPasswordField value = new JPasswordField();
    JLabel l1=new JLabel("Password:");
      l1.setBounds(20,100, 80,30);
       value.setBounds(100,100,100,30);
         f.add(value);  f.add(l1);
         f.setSize(300,300);
         f.setLayout(null);
         f.setVisible(true);
}
}
```

Output:

Java JPasswordField Example with ActionListener

```
import javax.swing.*;
import java.awt.event.*;
public class PasswordFieldExample {
   public static void main(String[] args) {
   JFrame f=new JFrame("Password Field Example");
    final JLabel label = new JLabel();
    label.setBounds(20,150, 200,50);
    final JPasswordField value = new JPasswordField();
    value.setBounds(100,75,100,30);
    JLabel l1=new JLabel("Username:");
      l1.setBounds(20,20, 80,30);
      JLabel l2=new JLabel("Password:");
      l2.setBounds(20,75, 80,30);
      JButton b = new JButton("Login");
      b.setBounds(100,120, 80,30);
      final JTextField text = new JTextField();
      text.setBounds(100,20, 100,30);
```

```
        f.add(value); f.add(l1); f.add(label); f.add(l2); f.add(b); f.add(text);
        f.setSize(300,300);
        f.setLayout(null);
        f.setVisible(true);
        b.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
          String data = "Username " + text.getText();
          data += ", Password: "
          + new String(value.getPassword());
          label.setText(data);
        }
      });
  }
  }
```

Output: