**Step 1: Open Eclipse IDE**

Make sure you have Eclipse IDE for Java Developers installed.

If not installed yet, download from:  
https://www.eclipse.org/downloads/

**Step 2: Create a New Java Project**

1. Go to **File > New > Java Project**
2. Give your project a name (e.g., SwingExample)
3. Click **Finish**

**Step 3: Create a New Java Class**

1. Right-click on the **src** folder in your project
2. Select **New > Class**
3. Enter a name for your class (e.g., MySwingApp)
4. Tick the box for public static void main(String[] args)
5. Click **Finish**

**Step 4: Write Your Swing Code**

import javax.swing.\*;

public class MySwingApp {

public static void main(String[] args) {

JFrame frame = new JFrame("My First Swing App");

JButton button = new JButton("Click Me");

button.setBounds(100, 100, 120, 40);

// Action Listener

button.addActionListener(e -> JOptionPane.showMessageDialog(frame, "Button Clicked!"));

frame.add(button);

frame.setSize(300, 300);

frame.setLayout(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

**Step 5: Run the Program**

1. Right-click on the file (MySwingApp.java)
2. Click **Run As > Java Application**

A new window will appear with a button.

**What is Event Handling in Java?**

**Event Handling** in Java is a **mechanism that controls the events** and decides what should happen when an event occurs.

**Real-Life Example:**

Like pressing a button on an elevator → elevator goes up/down.  
Similarly, in Java GUI:

* **Button click** → perform an action
* **Key press** → take input
* **Mouse movement** → change cursor or highlight

**Why Use Event Handling?**

* To **respond to user actions** (like button click, key press, mouse click)
* Makes GUI **interactive**
* Used in **Swing, AWT, JavaFX** applications

### Common Event Listener Interfaces in Java

| **Interface** | **Purpose** |
| --- | --- |
| ActionListener | Handles button clicks |
| MouseListener | Handles mouse actions |
| KeyListener | Handles keyboard events |
| ItemListener | Handles checkbox/dropdown |
| WindowListener | Handles window actions (close) |

### Without Event Handling:

Your GUI components (buttons, text fields, etc.) would be **static and unresponsive**—just like a picture.

With event handling, they become **active and intelligent**, ready to interact with the user.

### Real-Life Analogy:

Think of:

* A **TV remote** – Button press (event) triggers channel change (response).
* A **doorbell** – Button press (event) triggers sound (response).

## **1. What is JFrame?**

* JFrame is a **window** on your screen — the main container where other components (like buttons, text fields) are placed.
* It’s similar to a **blank canvas** or **form** in Windows applications.

### Why Use JFrame?

| **Feature** | **Description** |
| --- | --- |
| Main window | Acts as the main **application window** |
| Container | Holds all GUI components: buttons, labels, panels |
| Methods to control UI | Set size, layout, visibility, close operation, etc. |

## **2. What is JButton?**

* JButton is a **push button** component used to **trigger actions**.
* When clicked, it **generates an event** (like opening a new form, saving data, etc.).

### Why Use JButton?

| **Feature** | **Description** |
| --- | --- |
| Interactive | Lets users click to perform actions |
| Event support | Works with ActionListener to handle clicks |
| Customizable | Text, color, size, icons can be set easily |

### Example: Button Click Event using JFrame and JButton

import javax.swing.\*;

import java.awt.event.\*;

public class EventExample {

public static void main(String[] args) {

// Create a frame (window)

JFrame frame = new JFrame("Event Handling Example");

// Create a button

JButton button = new JButton("Click Me");

// Set button position and size

button.setBounds(100, 100, 120, 40);

// Add event handler to button

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// This code runs when the button is clicked

JOption Pane.showMessageDialog(frame, "🎉 Button was clicked!");

}

});

// Add button to frame

frame.add(button);

// Frame settings

frame.setSize(350, 250);

frame.setLayout(null); // No layout manager

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true); // Show frame

}

}

### Explanation:

| **Code Line** | **Purpose** |
| --- | --- |
| JFrame | Creates the application window |
| JButton | Creates a clickable button |
| setBounds(x, y, w, h) | Sets the position and size of the button |
| addActionListener(...) | Attaches an event handler to the button |
| JOptionPane.showMessageDialog | Shows a pop-up message box |

### ****Example 1: Multiple Buttons with Different Actions****

import javax.swing.\*;

import java.awt.event.\*;

public class MultiButtonExample {

public static void main(String[] args) {

JFrame frame = new JFrame("Multiple Buttons");

JButton helloBtn = new JButton("Say Hello");

JButton byeBtn = new JButton("Say Bye");

helloBtn.setBounds(50, 80, 120, 30);

byeBtn.setBounds(180, 80, 120, 30);

// Hello button event

helloBtn.addActionListener(e ->

JOptionPane.showMessageDialog(frame, "Hello, User!")

);

// Bye button event

byeBtn.addActionListener(e ->

JOptionPane.showMessageDialog(frame, "Goodbye!")

);

frame.add(helloBtn);

frame.add(byeBtn);

frame.setSize(370, 200);

frame.setLayout(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

### ****Example 2: Button + TextField****

import javax.swing.\*;

import java.awt.event.\*;

public class TextFieldExample {

public static void main(String[] args) {

JFrame frame = new JFrame("Input Box");

JTextField textField = new JTextField();

JButton button = new JButton("Show Name");

textField.setBounds(50, 60, 150, 30);

button.setBounds(210, 60, 120, 30);

button.addActionListener(e -> {

String name = textField.getText();

JOptionPane.showMessageDialog(frame, "🧑 Hello, " + name + "!");

});

frame.add(textField);

frame.add(button);

frame.setSize(400, 180);

frame.setLayout(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

### ****Example 3: Counter Button (Increment Value)****

import javax.swing.\*;

import java.awt.event.\*;

public class CounterApp {

public static void main(String[] args) {

JFrame frame = new JFrame("Counter App");

JButton button = new JButton("Increment");

JLabel label = new JLabel("Count: 0");

int[] count = {0}; // Using array to allow change inside lambda

5

label.setBounds(130, 60, 100, 30);

button.setBounds(100, 100, 120, 30);

button.addActionListener(e -> {

count[0]++;

label.setText("Count: " + count[0]);

});

frame.add(label);

frame.add(button);

frame.setSize(300, 200);

frame.setLayout(null);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

## Java Swing Example Programs

### 🔹 1. ****Simple Swing Window****

import javax.swing.\*;

public class SimpleWindow {

public static void main(String[] args) {

JFrame frame = new JFrame("My First Swing Window");

frame.setSize(300, 200); // width, height

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

}

📝 Explanation:  
This creates a basic window using JFrame.

### 🔹 2. ****Label and Button****

import javax.swing.\*;

public class LabelButtonExample {

public static void main(String[] args) {

JFrame frame = new JFrame("Label and Button");

JButton button = new JButton("Click Me");

JLabel label = new JLabel("Welcome!");

button.setBounds(100, 80, 120, 30);

label.setBounds(110, 30, 200, 30);

frame.add(button);

frame.add(label);

frame.setSize(300, 200);

frame.setLayout(null);

frame.setVisible(true);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

📝 Explanation:  
This shows a label and a button on the window.

### 🔹 3. ****Button Click Event (Event Handling)****

import javax.swing.\*;

import java.awt.event.\*;

public class ButtonClickEvent {

public static void main(String[] args) {

JFrame frame = new JFrame("Button Event");

JButton button = new JButton("Click");

JLabel label = new JLabel();

button.setBounds(100, 80, 120, 30);

label.setBounds(100, 130, 200, 30);

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

label.setText("Button Clicked!");

}

});

frame.add(button);

frame.add(label);

frame.setSize(300, 250);

frame.setLayout(null);

frame.setVisible(true);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

📝 Explanation:  
When you click the button, the label text changes.

### 4. ****TextField and Button****

import javax.swing.\*;

import java.awt.event.\*;

public class TextFieldExample {

public static void main(String[] args) {

JFrame frame = new JFrame("TextField Example");

JTextField textField = new JTextField();

JButton button = new JButton("Submit");

JLabel label = new JLabel();

textField.setBounds(50, 50, 200, 30);

button.setBounds(100, 100, 100, 30);

label.setBounds(70, 150, 200, 30);

button.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String text = textField.getText();

label.setText("You entered: " + text);

}

});

frame.add(textField);

frame.add(button);

frame.add(label);

frame.setSize(300, 250);

frame.setLayout(null);

frame.setVisible(true);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

Explanation:  
User types something in the text field → clicks the button → label displays the text.

### 🔹 5. ****Simple Login Form Using Swing****

import javax.swing.\*;

import java.awt.event.\*;

public class LoginForm {

public static void main(String[] args) {

JFrame frame = new JFrame("Login Form");

JLabel userLabel = new JLabel("Username:");

JLabel passLabel = new JLabel("Password:");

JTextField userText = new JTextField();

JPasswordField passText = new JPasswordField();

JButton loginBtn = new JButton("Login");

JLabel result = new JLabel();

userLabel.setBounds(30, 30, 100, 30);

userText.setBounds(120, 30, 150, 30);

passLabel.setBounds(30, 70, 100, 30);

passText.setBounds(120, 70, 150, 30);

loginBtn.setBounds(100, 120, 100, 30);

result.setBounds(100, 170, 200, 30);

loginBtn.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String user = userText.getText();

String pass = new String(passText.getPassword());

if (user.equals("admin") && pass.equals("1234")) {

result.setText("Login Successful");

} else {

result.setText("Invalid credentials");

}

}

});

frame.add(userLabel);

frame.add(userText);

frame.add(passLabel);

frame.add(passText);

frame.add(loginBtn);

frame.add(result);

frame.setSize(350, 300);

frame.setLayout(null);

frame.setVisible(true);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

Explanation:  
Checks if the username is "admin" and password is "1234", and shows result.

## **1) Hello World Swing Window**

import javax.swing.\*;

public class HelloSwing {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new JFrame("Hello Swing");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JLabel label = new JLabel("Hello, Swing!", SwingConstants.CENTER);

frame.add(label);

frame.setSize(300, 200);

frame.setLocationRelativeTo(null); // center

frame.setVisible(true);

});

}

}

**What it shows:**

* JFrame = main window
* JLabel = text display
* SwingUtilities.invokeLater(...) for thread safety

## **2) Button Click Example**

import javax.swing.\*;

import java.awt.event.\*;

public class ButtonClickExample {

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new JFrame("Button Click Example");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JButton button = new JButton("Click Me");

JLabel label = new JLabel("No clicks yet", SwingConstants.CENTER);

button.addActionListener(e -> label.setText("Button Clicked!"));

frame.setLayout(new java.awt.BorderLayout());

frame.add(label, java.awt.BorderLayout.CENTER);

frame.add(button, java.awt.BorderLayout.SOUTH);

frame.setSize(300, 150);

frame.setLocationRelativeTo(null);

frame.setVisible(true);

});

}

}

**What it shows:**

* Handling button events with ActionListener
* Updating UI dynamically

## **3) Simple Login Form**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class LoginForm {

public static void main(String[] args) {

SwingUtilities.invokeLater(LoginForm::createGUI);

}

private static void createGUI() {

JFrame frame = new JFrame("Login Form");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(new GridLayout(3, 2, 10, 10));

JLabel userLabel = new JLabel("Username:");

JTextField userField = new JTextField();

JLabel passLabel = new JLabel("Password:");

JPasswordField passField = new JPasswordField();

JButton loginButton = new JButton("Login");

JLabel message = new JLabel("");

loginButton.addActionListener(e -> {

String user = userField.getText();

String pass = new String(passField.getPassword());

if (user.equals("admin") && pass.equals("123")) {

message.setText("Login Successful!");

} else {

message.setText("Invalid credentials.");

}

});

frame.add(userLabel);

frame.add(userField);

frame.add(passLabel);

frame.add(passField);

frame.add(loginButton);

frame.add(message);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

}

**What it shows:**

* Using JTextField, JPasswordField
* Basic authentication check logic

## **4) Calculator Example**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class SimpleCalculator {

public static void main(String[] args) {

SwingUtilities.invokeLater(SimpleCalculator::createGUI);

}

private static void createGUI() {

JFrame frame = new JFrame("Simple Calculator");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(new GridLayout(3, 2, 10, 10));

JTextField num1 = new JTextField();

JTextField num2 = new JTextField();

JLabel result = new JLabel("Result: ");

JButton addButton = new JButton("Add");

addButton.addActionListener(e -> {

try {

double a = Double.parseDouble(num1.getText());

double b = Double.parseDouble(num2.getText());

result.setText("Result: " + (a + b));

} catch (NumberFormatException ex) {

result.setText("Invalid input");

}

});

frame.add(new JLabel("Number 1:"));

frame.add(num1);

frame.add(new JLabel("Number 2:"));

frame.add(num2);

frame.add(addButton);

frame.add(result);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

}

**What it shows:**

* GridLayout for forms
* Input validation using try-catch

## **5) Table Display Example**

import javax.swing.\*;

import java.awt.\*;

public class TableExample {

public static void main(String[] args) {

SwingUtilities.invokeLater(TableExample::createGUI);

}

private static void createGUI() {

JFrame frame = new JFrame("JTable Example");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

String[] columns = {"ID", "Name", "Country"};

Object[][] data = {

{1, "Akila", "India"},

{2, "John", "USA"},

{3, "Sara", "UK"}

};

JTable table = new JTable(data, columns);

JScrollPane scrollPane = new JScrollPane(table);

frame.add(scrollPane, BorderLayout.CENTER);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

}

**What it shows:**

* Using JTable to display tabular data
* Adding scroll with JScrollPane