calculator in Java using both console and GUI (Swing)

## JFrame

* JFrame is a top-level container.
* It holds all components like buttons, panels, and text fields.

## JButton

* Every button can trigger actions.
* Used addActionListener() to handle clicks.

## JTextField

* Can be used for input or output.
* setEditable(false) makes it read-only.

## JPanel

* JPanel is a generic container.
* Used for organizing layouts.

## GridLayout

* GridLayout(rows, columns) places components evenly in a grid(Calculator keypad).

## ActionListener

* ActionListener is an interface.
* You must override actionPerformed().
* It allowed me to check which button was pressed and perform the correct calculation.

## ActionEvent

* getActionCommand() returns the label of the clicked button

**📌 Java Swing Calculator - Mini Project**

A simple yet powerful **GUI Calculator App** created using **Java Swing**. This project supports basic arithmetic operations, advanced functions, and a clean user interface.

**🧠 Features**

* ➕ Basic Operations: +, -, \*, /, %
* ✴️ Advanced: ^ (power), √ (square root)
* 🔢 Decimal number support
* 🧹 Clear (C) and ⌫ Backspace support
* 🖥 Built using **Java Swing** components: JFrame, JTextField, JButton, GridLayout
* 🧠 Input validation and exception handling

**🚀 How to Run the Project**

1. Open IntelliJ IDEA (or any Java IDE)
2. Create a new Java Project
3. Create a file named CalculatorApp.java
4. Copy and paste the full code inside the file
5. Run the main() method

**🖥 Sample Output (UI):**

A maximized calculator window with:

* Top display box
* Buttons laid out in a 5x4 grid:  
  7 8 9 /  
  4 5 6 \*  
  1 2 3 -  
  0 . = +  
  C ⌫ ^ √

**💡 Technologies Used**

* Java
* Java Swing (JFrame, JButton, JTextField, JPanel, GridLayout, etc.)

**🧾 Resume Project Description**

**Java Swing GUI Calculator**

Built a GUI-based calculator using Java Swing that supports both basic and advanced arithmetic operations. Implemented responsive UI using JFrame, JButton, JTextField, and GridLayout. Integrated features like square root, power, decimal handling, and backspace. Used ActionListener for real-time user interaction and robust error handling for smooth UX.

**📁 Project Folder Structure**

CalculatorProject/

├── CalculatorApp.java

**📦 Build & Package (Optional)**

To create a runnable .jar file:

1. Compile the program:

javac CalculatorApp.java

1. Create JAR:

jar cfe CalculatorApp.jar CalculatorApp CalculatorApp.class

1. Run it anywhere:

java -jar CalculatorApp.jar

**🧑‍💻 Interview Preparation Report**

**🔹 What problem does this project solve?**

It provides a lightweight desktop calculator for quick arithmetic and scientific operations with a user-friendly interface, mimicking real-world calculator usage.

**🔹 Why did you build this?**

To strengthen my Java programming fundamentals, especially in GUI development, event handling, and user interaction logic.

**🔹 What did you learn?**

* Creating user interfaces with **Java Swing**
* Implementing **event-driven programming** using ActionListener
* Managing **input parsing**, **operator precedence**, and **UI updates**
* Handling **edge cases** like divide-by-zero and invalid input

**🔹 What challenges did you face?**

* Managing operation flow while maintaining input clarity
* Updating UI dynamically based on user actions
* Ensuring all input and exceptions are handled gracefully

**🔹 What would you improve or add next?**

* Add a secondary display to show operation history
* Introduce a toggle for light/dark mode themes
* Save calculation logs to a file for later use
* Make UI responsive to various screen sizes

**🔹 Key Java Concepts Demonstrated**

* **Object-Oriented Programming** (extends JFrame, modular design)
* **Swing Components** (JTextField, JPanel, JButton, layout managers)
* **Event Handling** (implements ActionListener, actionPerformed)
* **Math and Parsing** (Double.parseDouble(), Math.pow(), Math.sqrt())

**🔹 Sample Interview Answer:**

"I built a Java Swing Calculator as a mini project to demonstrate my understanding of GUI programming and event handling in Java. It includes both standard and advanced features like power, square root, decimal handling, and a backspace key. I used ActionListener to capture button clicks and updated the display in real time based on input. I also implemented proper error handling for user input and edge cases. This project gave me hands-on experience with Java Swing and helped me build a strong foundation for future GUI-based applications."

**📌 Final Note:**

This project is a great starting point for learning Java GUI development. It can be extended with more features like theme switching, history, scientific functions, and file I/O logging for further enhancement.