Java Swing Calculator Application Documentation

1. Project Overview

The Java Swing Calculator is a desktop application built using Java and the Swing library. It offers a modern, user-friendly interface for performing basic arithmetic and scientific operations. The calculator features a clean layout with a seven-digit display, meaning only up to seven digits are visible at any one time.

2. Key Features

The calculator implements the following functionalities:

- **Addition (+):** Adds two numbers.
- **Subtraction (-):** Subtracts one number from another.
- **Multiplication (×):** Multiplies two numbers.
- **Division (÷):** Divides one number by another. Notably, when a division by zero occurs, the calculator shows "Infinity" as per Java's handling of this operation.
- **Equals (=):** Evaluates the expression using the stored operator and operands.
- **Percentage (%):** Calculates a percentage value based on the current input.
- **Plus/Minus Toggle (+/-):** Changes the sign of the current number.
- **All Clear (AC):** Resets the display and clears any stored operations.
- **Decimal Point (.):** Supports floating-point number input.
- **Square Root (√):** Computes the square root of the current number.
- **Seven-Digit Display:** Ensures that only up to seven digits are visible, maintaining clarity in the display.

3. Installation and Usage Instructions

Installation

1. **Clone or Download the Repository:**

   GitHub: Clone the repository using:
   - *git clone <repository_url>*
   - or download the ZIP file and extract it.
2. **Setup Development Environment:**

   - Ensure you have Java Development Kit (JDK 8 or higher) installed.
   - Use an IDE such as IntelliJ IDEA, Eclipse, or NetBeans, or compile from the command line.

3. **Build and Run the Project:**
    - Open the project in your chosen IDE.
    - Compile and run the `CalculatorApp` class located in the `com.mycompany.calculatorapp` package.

Command Line Alternative:
javac -d bin src/com/mycompany/calculatorapp/.java*
java -cp bin com.mycompany.calculatorapp.CalculatorApp

Usage

Upon launching the application, the calculator window displays two main areas:

- **Display Area:** Shows the current input and results. The display is designed to show up to seven digits.
- **Button Panel:** Contains buttons for digits, operators, and functions, with color-coded differentiation:
    - **Operators and Square Root:** Highlighted with a distinct color for easy recognition.
    - **Special Functions (AC, +/-):** Styled in light gray with dark text.
    - **Numeric Buttons and Decimal Point:** Displayed with a dark theme for consistency.

Steps to Use:

- **Entering Numbers:** Click the digit buttons. If the display initially shows “0”, it will be replaced by the new digit; otherwise, digits are appended.
- **Inputting Decimals:** Use the “.” button to add a decimal point (only one is allowed per number).
- **Performing Operations:**
    - Enter the first number.
    - Click an operator (e.g., +, -, ×, ÷, %, √). For binary operations (addition, subtraction, multiplication, division, percentage), the first operand is stored and the display resets.
    - Enter the second number.
    - Click “=” to display the result.
- **Special Functions:**
    - **Square Root (√):** Directly calculates and displays the square root of the current number.
    - **Plus/Minus (+/-):** Toggles the sign of the current input.
    - **All Clear (AC):** Clears all stored values and resets the display.
- **Division by Zero:** When a division by zero is attempted, the calculator handles it by displaying “Infinity,” leveraging Java's inherent behavior for such operations.

4. Code Walkthrough

Main Components and Layout

- **Frame and Panels:**
    - A main window (`JFrame`) with a fixed size and a centered position on the screen.
    - A display area managed by a `JLabel` within a `JPanel` using a BorderLayout.
    - A button panel arranged in a 5x4 grid using `GridLayout` to organize the calculator buttons.
- **Color Customization**

  Custom colors differentiate various button types:

  Color customLightGray = new Color(212, 212, 210);
  Color customDarkGray = new Color(80, 80, 80);
  Color customBlack = new Color(28, 28, 28);
  Color customOrange = new Color(255, 149, 0);

Button Initialization and Event Handling

- **Button Creation:**
    - The `buttonValues` array holds labels for all buttons, while `rightSymbols` and `topSymbols` classify operator buttons and special function buttons, respectively.
    - A loop creates each button, applies styling, and adds it to the button panel.
- **Event Listeners:**
    - **Operator Buttons:** When an operator is pressed, the current value is stored and the display resets. The selected operator is updated accordingly.
    - **Equals Button:** Computes the result using the stored operator and operands. For division, if the second operand is zero, the result becomes Infinity.
    - **Square Root Button:** Computes the square root of the current input.
    - **Numeric and Decimal Input:** Manages digit concatenation and ensures that only a single decimal point is added.
    - **Special Functions:**
        - The **AC** button resets all values.
        - The **+/-** button toggles the sign of the displayed number.
- **Helper Methods:**
    - `clearAll()`: Resets variables (`A`, `operator`, `B`) to default values.
    - `removeZeroDecimal(double numDisplay)`: Formats the number for display, removing unnecessary decimal places for whole numbers.

Application Entry Point

The `CalculatorApp` class contains the `main` method that instantiates the `Calculator` and launches the application:

```
public class CalculatorApp {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
    }
}
```

5. Potential Future Enhancements

While the current implementation covers basic arithmetic and a few scientific functions, the following enhancements could further improve usability and functionality:

1. **Clear Button for Single Input/Digit:**
    - **Enhancement:** Add a backspace or clear-entry button that removes only the last entered digit.
    - **Benefit:** Allows users to correct mistakes without clearing the entire entry, improving the overall user experience.
2. **Trigonometric Functions (sin, cos, tan):**
    - **Enhancement:** Incorporate trigonometric functions to allow calculations of sine, cosine, and tangent.
    - **Benefit:** Expands the calculator's functionality for users who need basic scientific computations.
3. **Logarithmic Functions (log and ln):**
    - **Enhancement:** Add buttons for common logarithm (log base 10) and natural logarithm (ln).
    - **Benefit:** Enables users to perform logarithmic calculations, which are essential for various scientific and engineering applications.
4. **Power Function:**
    - **Enhancement:** Integrate an exponentiation function to compute powers (e.g., x^y).
    - **Benefit:** Provides users the ability to handle more complex mathematical operations easily.
5. **Pi (π) Constant:**
    - **Enhancement:** Include a button to insert the value of π.
    - **Benefit:** Allows for quick access to this common mathematical constant, aiding calculations in trigonometry and geometry.

6. Conclusion

The Java Swing Calculator is a compact yet powerful desktop application that demonstrates effective use of Java Swing for building intuitive user interfaces and handling user input. With a well-structured codebase, clearly defined functionalities (including handling division by zero by displaying Infinity), and scope for further enhancements, this project is an excellent showcase of both foundational and advanced Java programming.