Assignment 03: Simple Calculator

CS 140 with Dr. Sam Schwartz

Due: Sunday, September 29 at 11:59pm via Canvas Upload

1 Purpose

The purpose of this assignment is to:

1. Reinforce conditional logic and Boolean expressions in Java.
2. Apply if/else structures to implement decision-making in code.
3. Validate user input and handle error cases (e.g., division by zero).
4. Produce professional, readable console output.

2 Tasks – High Level Overview

In this assignment, you will:

- Build a two-mode calculator program: Calc Mode (arithmetic) and Compare Mode (relational comparison).
- Use if/else chains and Boolean expressions to decide behavior.
- Validate inputs (numbers, operators, comparators).
- Format and present results clearly to a user.

3 Tasks – Detailed Requirements

3.1 Task 1: Setup

1. Create a new Java project called `Assignment03` in Eclipse.
2. Create a new class called `Calculator`.
3. At the beginning of the file create a multi-line comment containing:
    - Your name
    - A short description of what the program does.

3.2 Task 2: Mode Selection & Input Normalization

1. Prompt the user to choose a mode: either `calc` or `compare`.
2. Normalize the input (e.g., trim whitespace; allow case-insensitive entries).
3. If the mode is not recognized, print an error and terminate gracefully.

3.3 Task 3: Calc Mode (Arithmetic with Conditionals)

1. If the user chose `calc`, prompt for:
   - Two numbers (use `double`).
   - An operator: `+`, `-`, `*`, or `/`. (Also accept the phrases `plus`, `minus`, `times`, `divide`.)
2. Use `Scanner` and appropriate checks (e.g., `hasNextDouble()`) to validate numeric input before reading.
3. Use an if/else-if/else chain to select the operation. Reject unrecognized operators with a clear message.
4. Guard against division by zero with a Boolean condition and print an error if it occurs.
5. Add a Boolean-controlled behavior:
   - **Rounding mode**: If enabled (via a `boolean` or Y/N prompt), present the result rounded to 3 decimal places; otherwise present the exact `double`.

3.4 Task 4: Compare Mode (Relational Operators)

1. If the user chose `compare`, prompt for:
   - Two numbers (use `double`).
   - A comparator: one of `<`, `<=`, `>`, `>=`, `==`, `!=`.
2. Validate the comparator with an if/else chain. If invalid, print an error and terminate gracefully.
3. Evaluate the comparison with a Boolean expression and print the `true`/`false` result.

3.5 Task 5: Output Formatting & UX

1. Print the full expression and result in a clean, readable format. Examples:
   - `7.0 * 4.0 = 28.0`
   - `3.5 <= 3.6 is true`
2. Use either string concatenation or `printf` to control decimal places.

3.6 Task 6: Decomposition & Reflection

1. Decompose your solution into **multiple methods** (at least three) with clear responsibilities. Examples:
   - `parseOperatorOrWord(String s)`, `validComparator(String s)`, `formatResult(...)`.
2. In a short comment block (3–5 lines), reflect on one validation you implemented (e.g., division-by-zero or invalid comparator) and describe a test case that would have failed without it.

3.7 What to Upload

You should upload three files to Canvas:

1. `Calculator.java`
2. A screenshot of `Calculator.java` running in Calc Mode
3. A screenshot of `Calculator.java` running in Compare Mode

4 Grading Criteria

Scale

Elements of nearly all assignments in this class will be broken down into a 0-1-2 scale. “2” means, “Nailed it!” “1” means, “Umm, kinda got it, but not really.” “0” means, “Uh-oh. Didn’t get it.” The points are averaged out and multiplied by 100. The ceiling becomes the final score. (If you get a 92.01%, we “round up” to 93%.)

Rubric

Validity

The author submitted a `Calculator.java` file to Canvas	2	1	0.
… which included a multi-line header comment with name and description	2	1	0.
… which prompted for a mode (`calc` or `compare`)	2	1	0.
… which handled stray whitespace and case normalization where appropriate	2	1	0.
… and which rejected an unrecognized mode gracefully by printing an error message	2	1	0.
In Calc Mode, it prompted for two numbers and an arithmetic operator	2	1	0.
… validated numeric input using appropriate checks (e.g., `hasNextDouble()`)	2	1	0.
… implemented + and plus with an if/else-if/else chain	2	1	0.
… implemented - and minus with an if/else-if/else chain	2	1	0.
… implemented * and times with an if/else-if/else chain	2	1	0.
… implemented / and divide with an if/else-if/else chain	2	1	0.
… handled division by zero with a Boolean guard	2	1	0.
… rejected unsupported operators (including misspelled words) gracefully	2	1	0.
… and demonstrated at least one compound Boolean expression (e.g., `||` or `&&`)	2	1	0.
In Compare Mode, it prompted for two numbers and a valid comparator operator	2	1	0.
… validated numeric input using appropriate checks (e.g., `hasNextDouble()`)	2	1	0.
… evaluated the comparison and printed a `true`/`false` result	2	1	0.
… decomposed the program into multiple methods (at least three)	2	1	0.
… and rejected unsupported operators gracefully by printing an error message	2	1	0.

Screenshot

The author included a screenshot showing Calc Mode run	2	1	0.
… and included a screenshot showing Compare Mode run	2	1	0.

Readability

The author included a brief reflection comment describing a validation and test case	2	1	0.
… and contained a sufficient number of comments explaining and documenting the code, each of which was written in an appropriate registrar of professional English	2	1	0.

Fluency

… and the deliverable was executed in such a way that an experienced practitioner would not find the deliverable “weird-in-a-bad-way” or unduly jarring	2	1	0.

Subtotal (of 48) .....

Total ($\lceil 100 \cdot \frac{\text{subtotal}}{48} \rceil$) .....