

# SE 317: Lab 7 - The Scientific Calculator

TEAM:

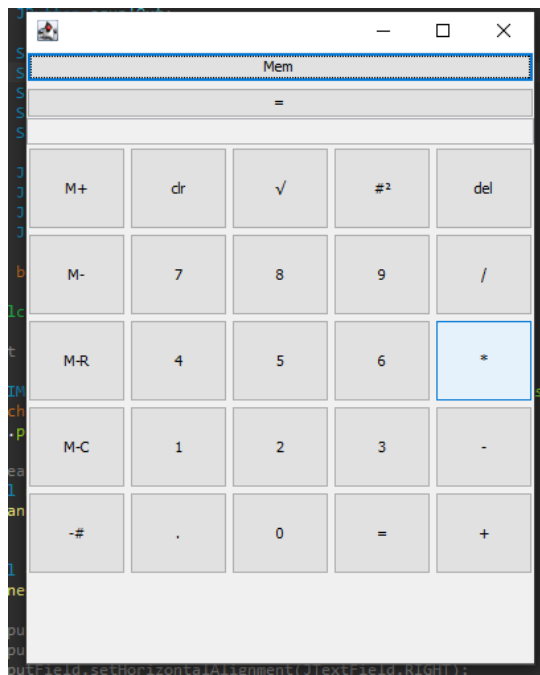
Lindsey Wessel - lkessel

Camden Fergen - cfergen

In this assignment, you will build and test a simple GUI calculator using the MVC pattern or any of its simplified versions. The Observer class in Java is a good example.


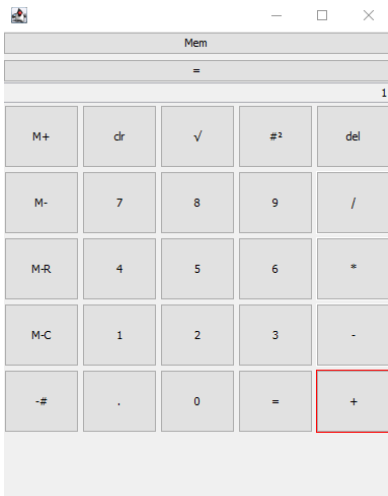

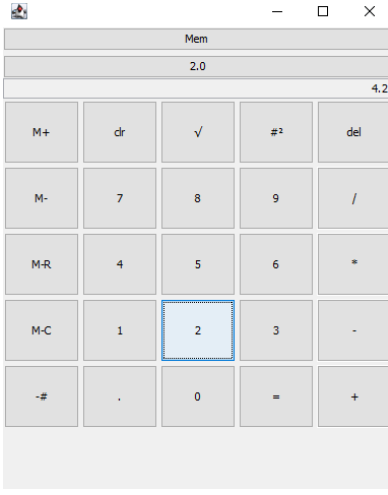
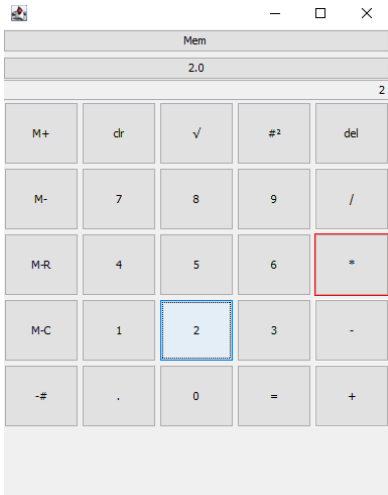

## Features and Functions:

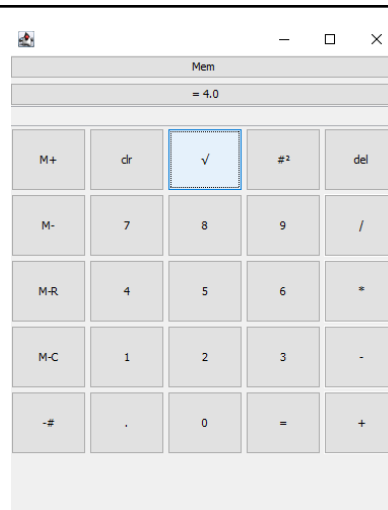
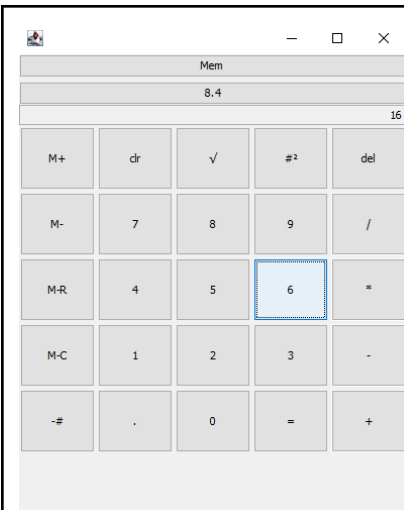
- ☒ 0–9 digit buttons
- ☒ Decimal point
- ☒ 4 basic math operations: add, subtract, multiply, divide
- ☒ 2 advanced operations: square and square root
- ☒ Memory functions:
  - ☒ M+ to add the output of a successfully executed math operation to the value currently stored in memory (negative values allowed)
  - ☒ M- to subtract the number on the calculator screen from the number stored in memory (negative values allowed)
  - ☒ M-Recall to recall the number in the memory and use it as an input operand in the current calculation
  - ☒ M-Clear to delete the memory contents
- ☒ Delete: will delete the last digit or the decimal point entered by the user. Multiple Delete operations will delete multiple digits in reverse order (including the decimal point)



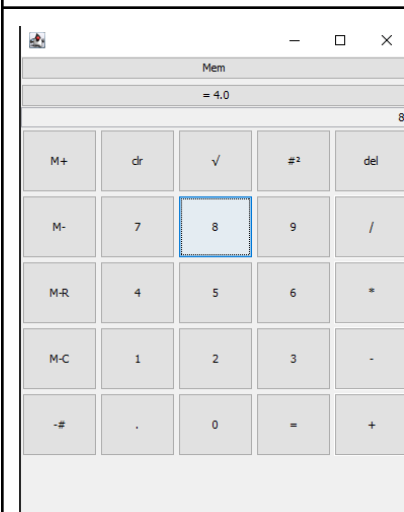
## Notes:

- ✓ Only the operands and the result are displayed on the calculator screen; the operations are NOT displayed.
- ✓ Each operation button on the screen changes its visual appearance to show it is pressed and currently active.
- ✓ When pressed, each operation button on the screen changes its visual appearance to show it is pressed and is currently active (they get bold or change color to show being active).
- ✓ Clear will reset the calculator to a blank start, including clearing the memory.

1 + 1 = 2		
		
4.2 * 2 = 8.4		
		
16 (sqrt) = 4		

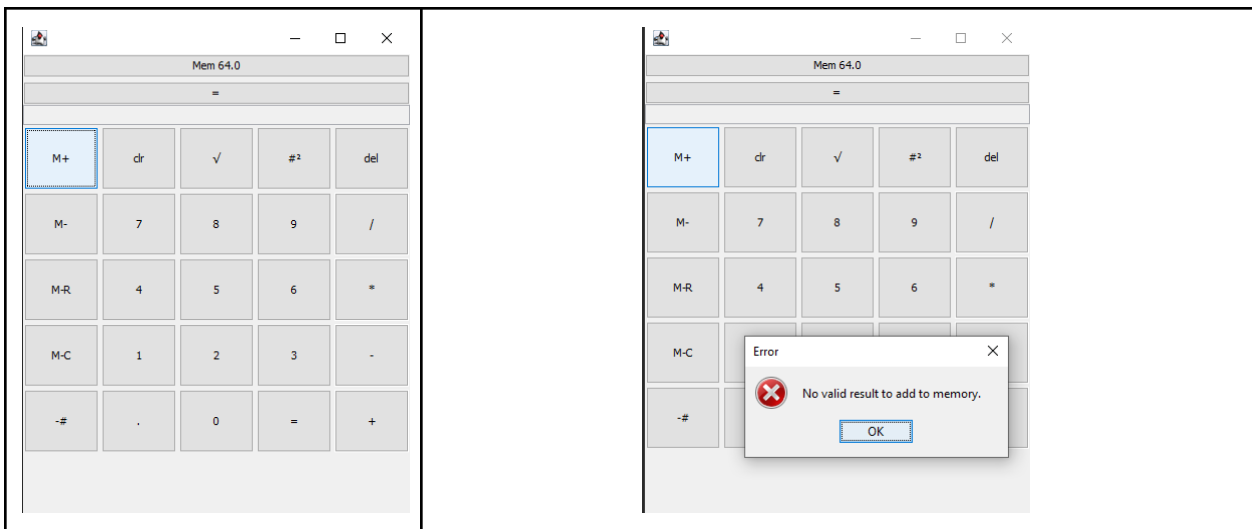


8 (sqr) = 64

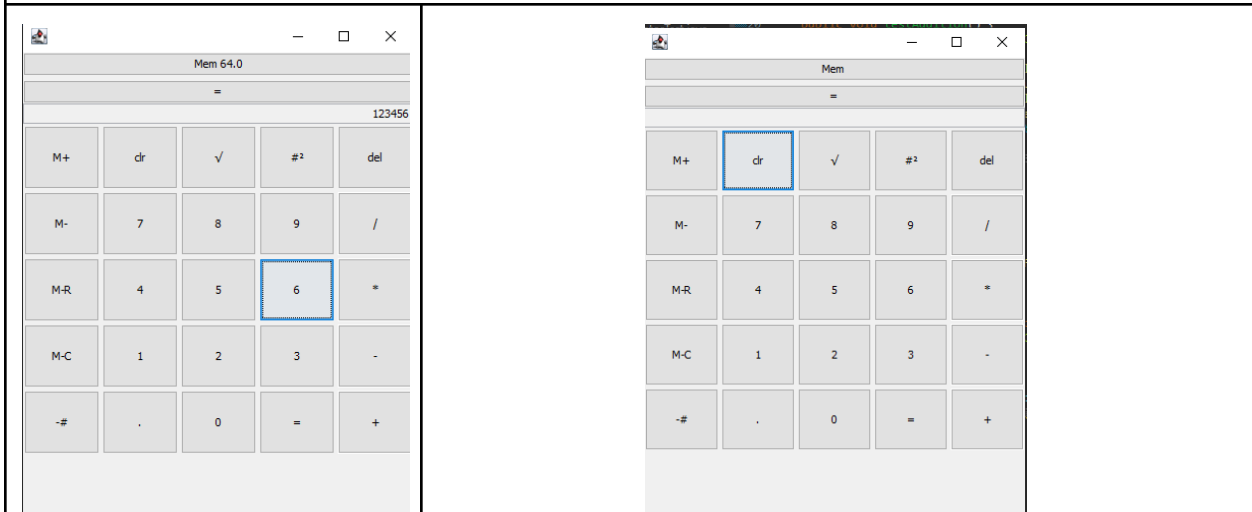


ADD 64 to mem

ADD NO RES to mem (error)

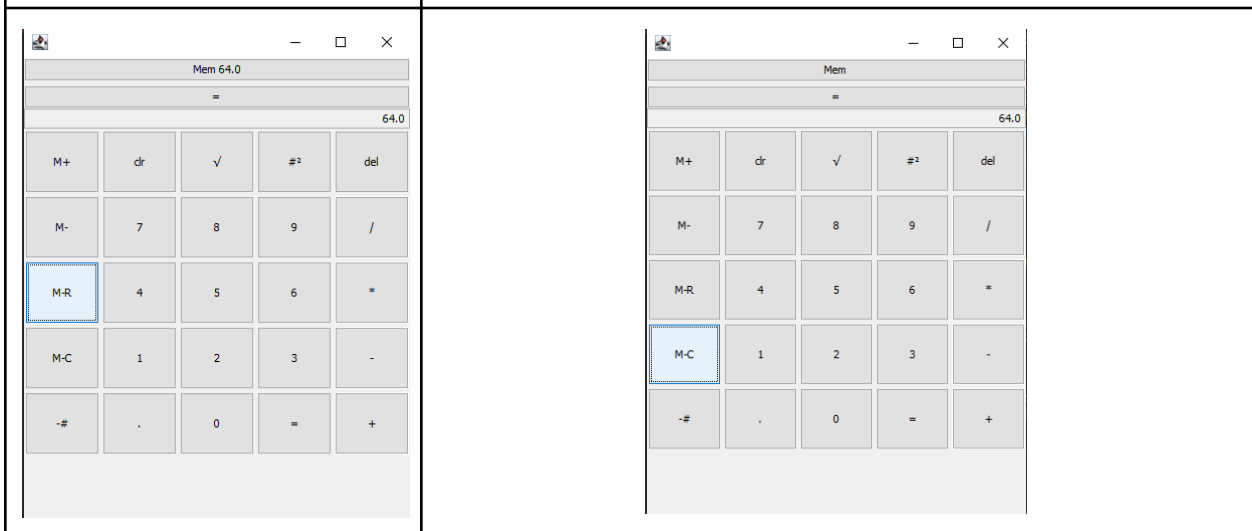


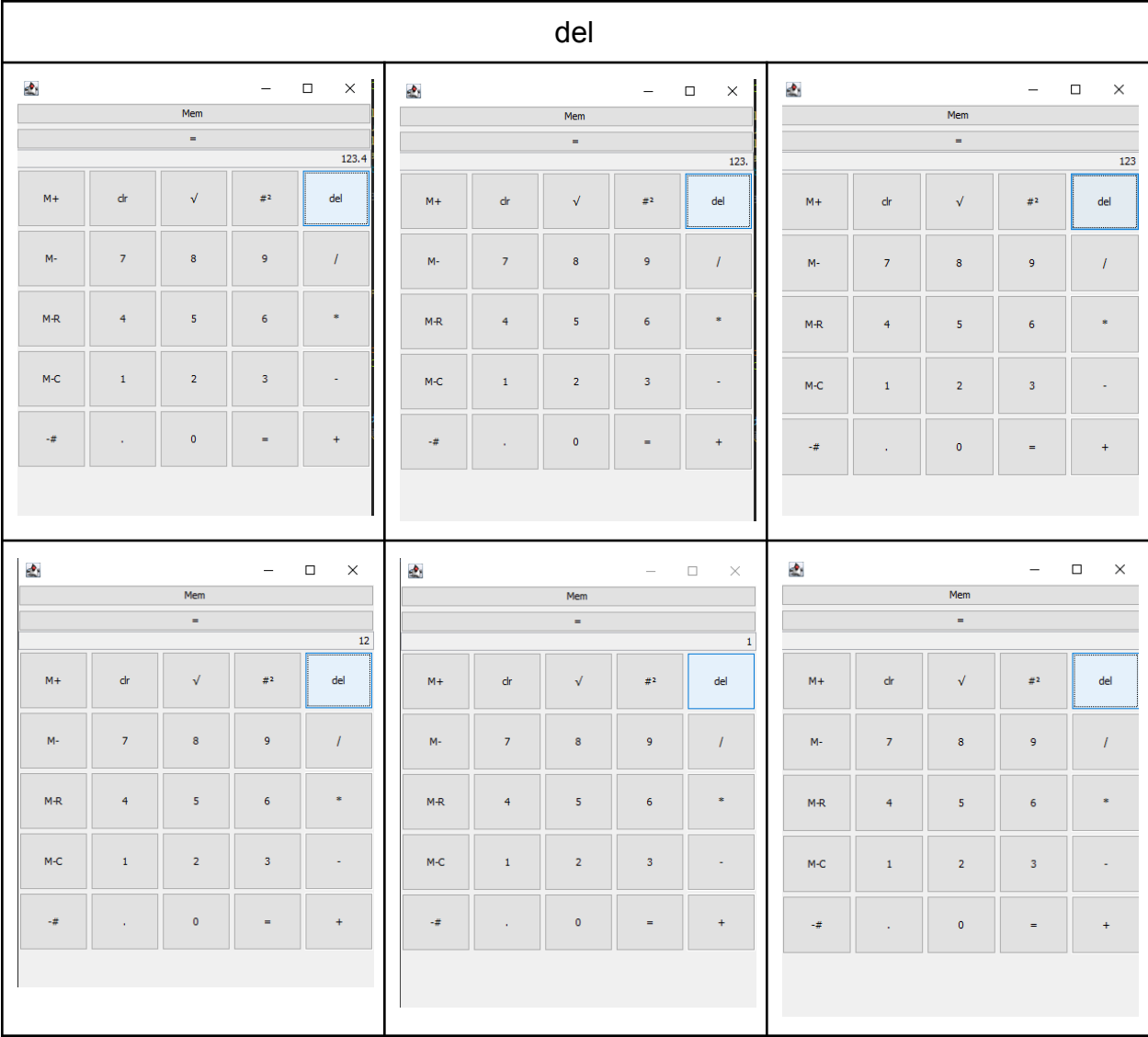
CLR



M-R

M-C





- Guidelines:**
- ☒ Use MVC or the Java Observer class for your assignment.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.math.*;
```

```
for (String label : buttonLabels) {
    if(label.equals("+")) {
        add = new JButton("+");
        add.addActionListener(new ButtonClickListener());
        buttonPanel.add(add);
        continue;
    } else if(label.equals("-")) {
```

- ☒ Use exception handling.
- ☒ Add appropriate code to prevent illegal operations at the model level.
- ☒ Interaction I/O will be done using interactive GUI.
- ☒ Add appropriate code to validate user input and prevent illegal entries at the UI level.

```
// Set the look and feel to the system default
try {
    UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
} catch (Exception e) {
    e.printStackTrace();
}
```

```
case "+, -":
    if((displayField.getText().length() == 0) {
        JOptionPane.showMessageDialog(null, "Please enter a number before making it negative", "Error", JOptionPane.ERROR_MESSAGE);
        break;
    }
    displayField.setText(String.valueOf(0 - Double.parseDouble(displayField.getText())));
    break;
```

```
messageDialog(null, "No valid result to add to memory.", "Error", JOptionPane.ERROR_MESSAGE);
```

- ☒ Add a Read-Me file of how to deploy and use your calculator.

```
Calculator.java  Main.java  README.txt
1 Lindsey Wessel (lkwessel) & Camden Fergen (cfergen)
2
3 Import the project into eclipse.
4 Compile and Run the project.
5 Use like a phone or basic math calculator.
6
7 NOTES:
8 This project uses MVC with Java Swing (has a model, view, and controller).
```

## Test Plan:

- ☐ Functionality (the Model functions)
- ☐ Features (The UI features)

- ☐ Boundaries and user space (for both the UI and the Model)

### Execution:

- ☐ The Model will be tested at the code level with function calls using Java test code, with screenshots of the output.
- ☐ The View and Controller (complete GUI) will be tested by providing the test cases to run on the interactive GUI with the expected results.
- ☐ You will need to run each GUI test as a sequence of GUI inputs (manually) and record the final output of each test as a screenshot, next to the expected value.

### Data Collection:

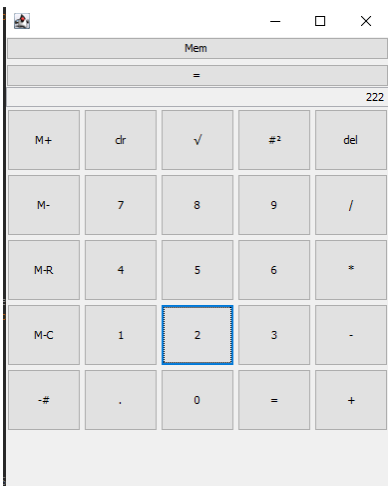

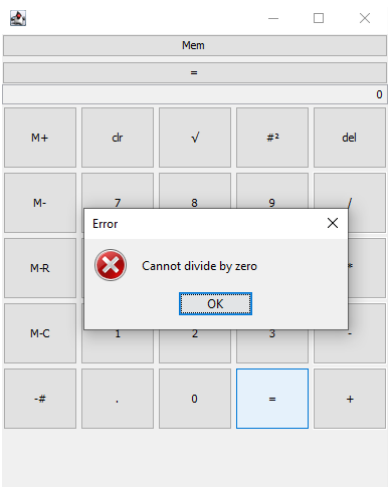
- ☐ Collect your results in the form of a table, including Test method, test criterion, test input values, test expected output, test actual output, success/fail.

### Analysis:

- ☐ Analyze, interpret, identify, and document the results, given any unexpected behavior or wrong results of your assignment.
- ☐ Provide a report, including a conclusion (if tests passed or failed).

### Example Test:

- ☐ Division by zero test case:
  - ☐ Input sequence: 222 / 0 =
  - ☐ Expected result: Error
  - ☐ Actual result: Error

Human Test		
		
Computer Test		

## Test Plan

### 1. Functionality (Model Functions)

**Test Criterion:** The calculator should correctly perform all basic and advanced mathematical operations.

#### Test Requirement (TR) Set:

- **TR1:** The calculator should correctly add two numbers.
- **TR2:** The calculator should correctly subtract two numbers.
- **TR3:** The calculator should correctly multiply two numbers.
- **TR4:** The calculator should correctly divide two numbers.
- **TR5:** The calculator should correctly calculate the square of a number.
- **TR6:** The calculator should correctly calculate the square root of a number.

#### Test Cases:

- **TC1:** Addition:  $5 + 3 = 8$
- **TC2:** Subtraction:  $10 - 4 = 6$
- **TC3:** Multiplication:  $2 * 3 = 6$
- **TC4:** Division:  $10 / 2 = 5$
- **TC5:** Square:  $4^2 = 16$
- **TC6:** Square Root:  $\sqrt{16} = 4$

The above tests can be found in the file “**basicTests.mov**”

### 2. Features (UI Features)

**Test Criterion:** The calculator's UI should correctly display and interact with all buttons and features.

#### Test Requirement (TR) Set:

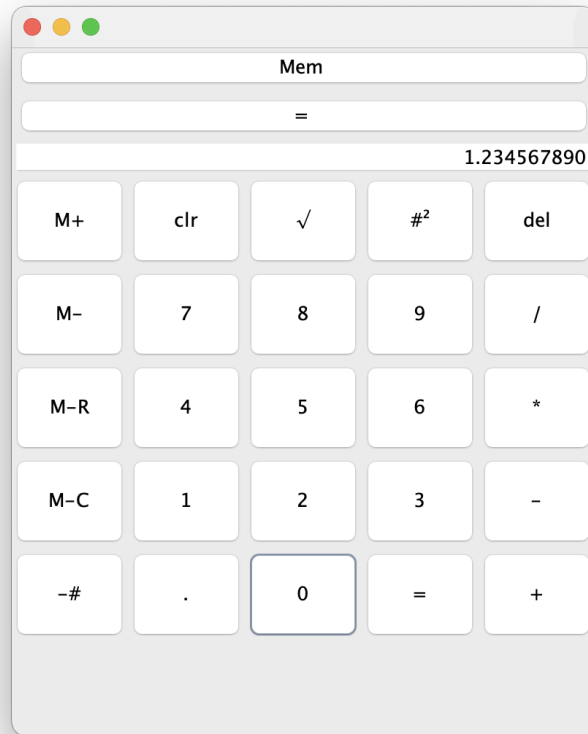
- **TR7:** The calculator should display all number buttons (0-9) and the decimal point.
- **TR8:** The calculator should correctly perform all basic and advanced operations when their corresponding buttons are pressed.
- **TR9:** The calculator should correctly store and recall values using the memory functions (M+, M-, M-Recall, M-Clear).



- **TR10:** The calculator should correctly delete the last digit or decimal point entered by the user.

### Test Cases:

- **TC7:** Display: All number buttons and the decimal point are visible.



- 
- **TC8:** Operation: Performing an addition operation (e.g., 5 + 3) and verifying the result.
- **TC9:** Memory Functions: Storing a value (e.g., 10) and recalling it for a new operation.
- **TC10:** Delete: Entering a number (e.g., 123.45) and deleting the last digit (e.g., 123.4).

The above tests can be found in the file “**UITests.mov**”

### 3. Boundaries and User Space (UI and Model)

**Test Criterion:** The calculator should handle edge cases and user inputs gracefully.

#### Test Requirement (TR) Set:

- **TR11:** The calculator should handle division by zero gracefully.

#### Test Cases:

- **TC11:** Division by Zero: Attempting to divide by zero (e.g.,  $10 / 0$ ) and expecting an error message.
  - Typed in  $10/0$

