# SE 317, Lab 7

Name: Aina Qistina Binti Azman

Net ID: 457 464 051

## CalculatorApp Program

**Screenshots of a Simple Mutiplication Program Execution:**

|  |  |
| --- | --- |
| **Phase/Action** | **Screenshot** |
| Initial View |  |
| Clicking on the first operand (7) |  |
| Clicking on the operation chosen (\*) |  |
| Clicking on the second operand (2) |  |
| Clicking on calculate result (=) and obtain the result |  |

## Unit Testing

1. The Model will be tested at the code level by simulated function calls using java test code, with screenshots of the output (like all previous assignments)

A screenshot of a computer

Description automatically generated

### Unit Testing – Data Analysis & Interpretation

|  |  |  |
| --- | --- | --- |
| **Test Criterion** | **Test Requirement Set** | **Test Set** |
| Addition | Ensure that the values are added mathematically correct. | * testAdditionPositiveNumbers() * testAdditionFirstNegativeNumber() * testAdditionNegativeNumbers() |
| Subtraction | Ensures that the values are subtracted mathematically correct. | * testSubtractionPositiveNumbers() * testSubtractionNegativeNumbers() * testSubtractionFirtsNegativeNumber() |
| Multiplication | Ensures that the values are multiplied mathematically correct. | * testMultiplicationPositiveDecimal() * testMultiplicationTwoNegativeNumbers() |
| Division | Ensures that the values are divided mathematically correct. | * testDivision() * testDivisionByZero() |
| Square | Ensures that the values are squared mathematically correct. | * testSquarePositiveNumber() * testSquareNegativeNumber() |
| Square Root | Ensures that the values are being square root mathematically correct. | * testSquareRoot() * testSquareRootNegative() |
| Memory | Ensure that the memory functions are working as per requirements. | * testMemoryAdd() * testMemoryAddFail() * testMemoryMinus() * testMemoryRecall() * testMemoryClear() |
| Delete & Clear | Ensure that the Delete and Clear Function are functioning as intended. | * testDelete() * testClear() |
| Out-of-Range | Ensure that the calculator application receives value in allowed range (-100000000 to 100000000) | * testOutOfBoundPositive() * testOutOfBoundNegative() |

### Unit Testing – Summary

* Total Number of Test Criteria: 9
* Total Number of Test Cases: 23
* Test Set Size: 23
* All Pass

## UI Testing

1. The GUI will be tested MANUALLY by just providing the test cases of given features to be run on the interactive GUI with the expected results. You will need to run each UI test as a sequence of GUI inputs (manually) and record the final output of each test as pass or fail, next to the expected one.

### UI Testing – Data Analysis & Interpretation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case** | **Input Sequence** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| Addition of Two Positive Numbers | 67 + 42 = | 109 |  | Pass |
| Addition of Two Negative Numbers | -7 + -42 = | -49 |  | Pass |
| Addition of a Negative & Positive Number | -7 + 42 = | 35 |  | Pass |
| Subtraction of Two Negative Numbers | -7 - -90 = | 83 |  | Pass |
| Subtraction of Two Positive Numbers | 7 – 90 = | -83 |  | Pass |
| Subtraction of a Positive & Negative Number | 67 - -54 = | 121 |  | Pass |
| Multiplication of a Positive Decimal with a Negative Number | 67.2 \* -4 = | -268.8 |  | Pass |
| Multiplication of Two Negative Numbers | -98 \* -3 = | 294 |  | Pass |
| Division of a Positive Number with a Negative Number | 100 / -2 = | -50 |  | Pass |
| Division of Two Negative Numbers | -99 / -13 = | 7.615384615 |  | Pass |
| Division by Zero | 99 / 0 = | Error |  | Pass |
| Square of a Positive Decimal Number | 6.5 ^2 | 42.25 |  | Pass |
| Square of a Negative Number | -8 ^2 | 64 |  | Pass |
| Square Root of a Positive Number | 100 sqrt | 10 |  | Pass |
| Square Root of a Negative Number | -100 sqrt | Error |  | Pass |
| M+ without a Successful Execution | 100 M+ | Error |  | Pass |
| M+ & M-Recall After a Successful Execution | 10 ^2 M+ 2 + 5 = M-Recall | Memory should be 100 |  | Pass |
| M- After a Successful M+ & M-Recall | 10 ^2 M+ 2 + 5 = M-Recall 2 M- | Memory should be 98 after called 2 and M- consecutively. |  | Pass |
| M-Clear | 10 ^2 M+ 2 + 5 = M-Recall M-Clear M-Recall | Memory should reset to 0 |  | Pass |
| Out of Range Positive | 10000000000 | Out of Range |  | Pass |
| Out of Range Negative | -89546987 | Out of Range |  | Pass |
| Delete | 123.456 Delete Delete Delete Delete | 123 |  | Pass |

### UI Testing – Summary

* Total Number of Test Criteria: 9
* Total Number of Test Cases: 23
* Test Set Size: 23
* All Pass

### Integration Testing

* Since the UI and model functions are interconnected, the UI tests implicitly cover integration testing.

## Conclusion

The testing plan for the calculator application included comprehensive unit tests and manual UI tests. Both tests were executed successfully, with all actual results matching the expected results. This indicates that the calculator application performs as expected, both at the code level and through the user interface. The use of existing exceptions for error handling ensures sturdy performance in edge cases such as division by zero and out-of-bounds value.