CSC 413 Project Documentation

Fall 2024

Po-Han Chen

923446482

Class.Section:02

GitHub Repository Link

https://github.com/csc413-SFSU-FA2024/calculator-eric915c.git

Table of Contents

1 Introduction

In this project, I learned how to implement some basic logic to make a calculator. This file includes its purpose, the technical details of the implementation, and a summary of the work completed.

1.1 Project Overview

This project is a basic calculator application. The app enables users to enter arithmetic expressions and calculate them. The calculator can perform addition, subtraction, multiplication, division, and exponentiation. Briefly, this project comprises both the backend logic and a graphical user interface (GUI) for interacting with the application.

1.2 Technical Overview

The calculator has been developed in Java, with the project organized into multiple classes that handle operands, operators, and the evaluation logic. The graphical user interface (GUI) is built using Java Swing, and unit tests have been incorporated to ensure the accuracy of the operations.

1.3 Summary of Work Completed

I. Firstly, Implemented the Operand class to represent numbers.

II. Created an abstract Operator class and its subclasses (AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator).

III. Created the Evaluator class to analyze and assess arithmetic expressions.

IV. Finished the ‘actionPerformed’ in EvaluatorUI class to provide the logic about next steps of ‘C’,’CE’,and’=’

V. Captured the entire project, detailing assumptions, implementation specifics, and reflections.

2 Development Environment

- Operating System: Windows 10
- IDE: IntelliJ IDEA
- java version "22.0.2"
- git version 2.45.1.windows.1

3 How to Build/Import your Project

I. Clone the project repository from Github

II. Open the project in IntelliJ IDEA

III. Ensure that the Java SDK is correctly configured.

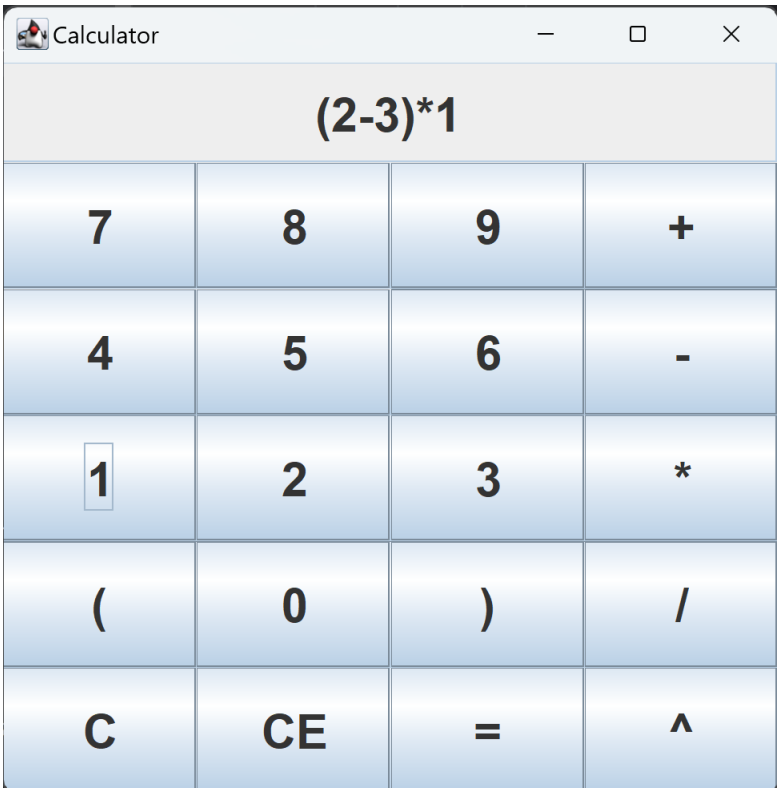
IV. Using these commands

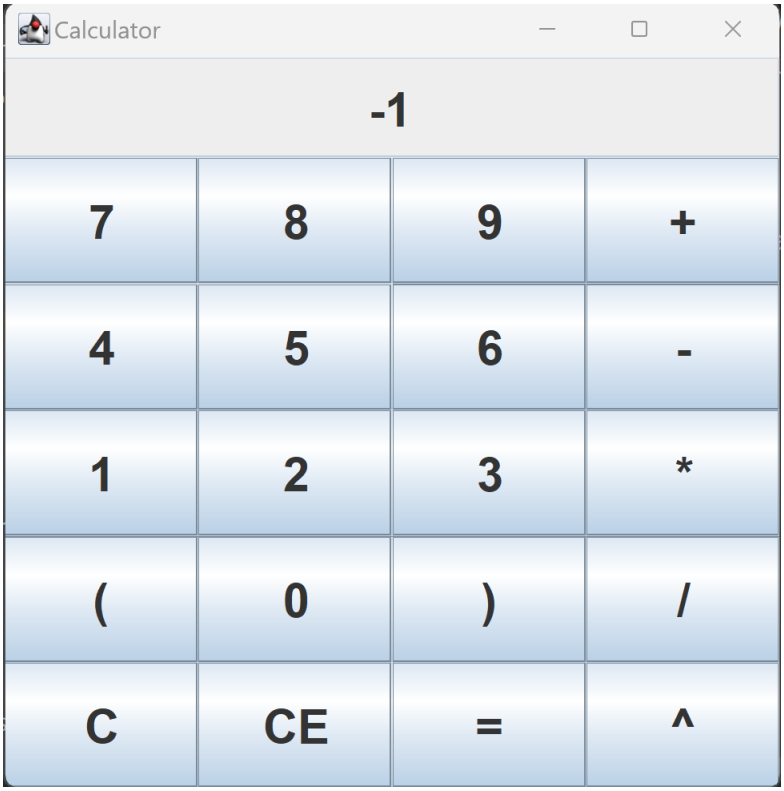
```
javac -d target .\calculator\evaluator\Operand.java

javac -d target .\calculator\operators\*.java
```

javac -d target .\calculator\evaluator*.java

4 How to Run your Project

Run EvaluatorUI





Use CMD to navigate to my project. Typing command ‘ls’ to check what’s in the project. And open the winrun.ps1. Typing ‘.\winrun.ps1’ to make project run.

```
PS D:\calculator-eric915c-main> ls


    目錄: D:\calculator-eric915c-main


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        2024/9/7  下午 11:51                .idea
d-----        2024/9/6  下午 06:44                calculator
d-----        2024/9/6  下午 06:44                documentation
d-----        2024/9/6  下午 06:44                lib
d-----        2024/9/6  下午 06:44                out
d-----        2024/9/6  下午 09:53                tests
-a----        2024/9/6  下午 06:44            546 bashrun.sh
-a----        2024/9/6  下午 06:44            527 calculator-eric915c-main.iml
-a----        2024/9/6  下午 06:44           2646 README.md
-a----        2024/9/6  下午 06:44            546 winrun.ps1
```

```
PS D:\calculator-eric915c-main> .\winrun.ps1

安全性警告
只執行您信任的指令碼。來自網際網路的指令碼雖然可能很有用，但是這個指令碼有可能會傷害您的電腦。若信任此指令碼，請使用
Unblock-File Cmdlet 來允許執行指令碼，而不顯示此警告訊息。您要執行 D:\calculator-eric915c-main\winrun.ps1 嗎？
[D] 不要執行(D)  [R] 執行一次(R)  [S] 暫停(S)  [?] 說明 (預設值為 "D"): R

Thanks for using JUnit! Support its development at https://junit.org/sponsoring

.
+-- JUnit Jupiter [OK]
| +-- Multiplication Test [OK]
| | +-- simpleMultiplicationTest() [OK]
| | +-- simpleMultiplicationTestNegativeOperand() [OK]
| | +-- multiplicationPriorityTest() [OK]
| | +-- simpleMultiplicationTestReversedOperands() [OK]
| | '-- simpleMultiplicationTestNegativeOperandReversed() [OK]
| +-- Division Test [OK]
| | +-- simpleDivisionTestNegativeResult() [OK]
| | +-- simpleDivisionTest() [OK]
| | +-- divisionPriorityTest() [OK]
| | +-- simpleDivisionTestReversedOperands() [OK]
| | '-- simpleDivisionTestEvenlyDivisible() [OK]
| +-- OperatorTest [OK]
| | +-- getOperatorReturnTypeTest(String, Class) [OK]
| | | +-- [1] +, class calculator.operators.AddOperator [OK]
| | | +-- [2] -, class calculator.operators.SubtractOperator [OK]
| | | +-- [3] /, class calculator.operators.DivideOperator [OK]
| | | +-- [4] *, class calculator.operators.MultiplyOperator [OK]
| | | '-- [5] ^, class calculator.operators.PowerOperator [OK]
| | '-- operatorCheckTest(String, Boolean) [OK]
| |   +-- [1] +, true [OK]
| |   +-- [2] -, true [OK]
| |   +-- [3] *, true [OK]
| |   +-- [4] ^, true [OK]
| |   +-- [5] /, true [OK]
| |   +-- [6] 156, false [OK]
| |   +-- [7] 156., false [OK]
| |   +-- [8] X, false [OK]
| |   '-- [9] **, false [OK]
| +-- Addition Test [OK]
| | +-- simpleAdditionTestWithNegativeOperand() [OK]
| | +-- simpleAdditionTestReverseOperands() [OK]
| | +-- simpleAdditionTest() [OK]
| | +-- additionPriorityTest() [OK]
```

```
| | +-- additionPriorityTest() [OK]
| | '-- simpleAdditionTestWithNegativeOperandsReversed() [OK]
| +-- Power Test [OK]
| | +-- simplePowerTest() [OK]
| | +-- simplePowerTestNegativeBase() [OK]
| | '-- powerPriorityTest() [OK]
| +-- Evaluator Test [OK]
| | +-- mediumExpressionPowerWithParensTest() [OK]
| | +-- basicExpressionParensOnRightTest() [OK]
| | +-- invalidOperatorExpressionProducesInvalidTokenExceptionTest() [OK]
| | +-- basicExpressionDivisionNumeratorLessThanDenominatorTest() [OK]
| | +-- mediumExpressionParensAsSubExpressionTest() [OK]
| | +-- veryDifficultExpressionMixtureOfOperatorsNestedParensTest() [OK]
| | +-- mediumExpressionParensAndOperatorsTest() [OK]
| | +-- difficultExpressionMixtureOfOperatorsTest() [OK]
| | +-- mediumExpressionPowerTest() [OK]
| | +-- mediumExpressionPowerWithMultipleOperators() [OK]
| | +-- mediumExpressionWithMultipleParensTest() [OK]
| | +-- basicExpressionMultipleOperatorTest() [OK]
| | +-- basicExpressionSingleOperatorTest() [OK]
| | '-- basicExpressionParensOnLeftTest() [OK]
| +-- Operand Test [OK]
| | +-- getValueTest() [OK]
| | +-- operandCheckTest(String, boolean) [OK]
| | | +-- [1] 13, true [OK]
| | | +-- [2] c, false [OK]
| | | +-- [3] *, false [OK]
| | | +-- [4] 430., false [OK]
| | | +-- [5] 430.456, false [OK]
| | | '-- [6] 343324fd, false [OK]
| | '-- getValueTypeTest() [OK]
| '-- Subtraction Test [OK]
|   +-- simpleSubtractionTestNegativeOperands() [OK]
|   +-- simpleSubtractionTestNegativeOperandsReversed() [OK]
|   +-- simpleSubtractionTestReversedOperands() [OK]
|   +-- simpleSubtractionTest() [OK]
|   '-- subtractionPriorityTest() [OK]
+-- JUnit Vintage [OK]
'-- JUnit Platform Suite [OK]
```

```
Test run finished after 190 ms
[         14 containers found      ]
[          0 containers skipped    ]
[         14 containers started    ]
[          0 containers aborted    ]
[         14 containers successful ]
[          0 containers failed     ]
[         59 tests found           ]
[          0 tests skipped         ]
[         59 tests started         ]
[          0 tests aborted         ]
[         59 tests successful      ]
[          0 tests failed          ]

PS D:\calculator-eric915c-main>
```

5 Assumption Made

All operators are binary operators

The calculator was designed assuming that all operators in the arithmetic expressions are binary, meaning each operator takes exactly two operands. For instance, the addition operator '+' is used between two numbers like '3 + 5'. Unary operators such as negation '-' or functions like 'sin' or 'cos' are not supported in this implementation.

6 Implementation Discussion

The calculator application's implementation is broken down into several crucial components, each handling different aspects of the program's functionality.

I. The Operand class is responsible for representing numeric values in the calculator. It is a straightforward class that contains an integer value. This value can be retrieved and utilized by various operators during expression evaluation.

   Attributes:

   int value: The numeric value stored in the Operand.

   Methods:

   'getValue()': Returns the stored value.

II. The Operator class serves as an abstract class, establishing the standard interface for all arithmetic operators.

   Methods:

   int priority(): Returns the precedence level of the operator.

   Operand execute(Operand operandOne, Operand operandTwo): Performs the operation using the two provided operands and returns the result as a new Operand.

III. Many operator subclasses extend the Operator class to perform specialized arithmetic operations.

   AddOperator: Implements addition (+).

   SubtractOperator: Implements subtraction (-).

   MultiplyOperator: Implements multiplication (*).

   DivideOperator: Implements division (/).

   PowerOperator: Implements exponentiation (^).

   Each of these subclasses overrides the execute method to perform the corresponding arithmetic operation. Additionally, they establish their order of importance using the priority method.

IV. The Evaluator class is in charge of parsing and assessing arithmetic expressions. It utilizes a stack-based method and follows the Shunting-yard algorithm to manage operator precedence and parentheses.

Key Components:

Operand Stack: This stack stores numbers as they are encountered, while the Operator Stack stores operators and ensures that operations are executed in the correct order based on precedence.

Algorithm Overview:

The algorithm parses the expression token by token, handles operators by pushing them onto a stack, handles operands by pushing them onto another stack, and finally evaluates the expression to produce the final result.

V. The EvaluatorUI class offers a visual interface for the calculator, utilizing Java Swing to produce a user-friendly environment. This enables users to input expressions and easily view the results.

Components:

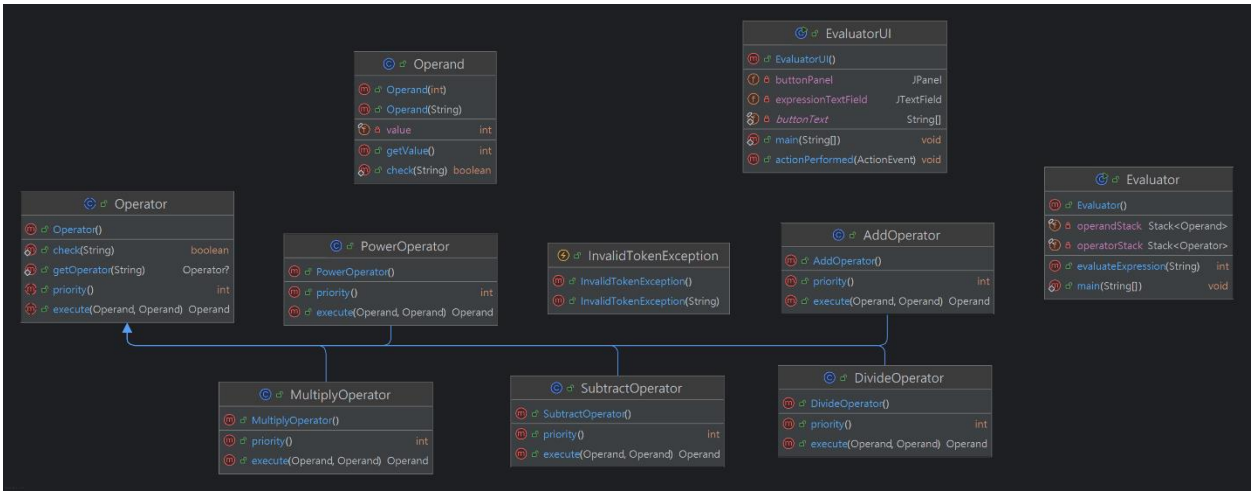
Text Field: Displays the current expression and the result.

Buttons: Represent digits, operators, and control functions (e.g., clear, equals).

Event Handling:

The UI captures user inputs through buttons, updates the expression, and calls the Evaluator class to calculate the result when the equals button is pressed.

6.1 Class Diagram



7 Project Reflection

I gained valuable experience in object-oriented design and Java programming through this project. It emphasized the importance of clear abstraction and the advantages of unit testing for early bug detection. One of the challenges I faced was managing operator precedence in the evaluator, which required careful consideration of the order of operations.

8 Project Conclusion/Results

The project successfully meets the requirements, providing a functional calculator with basic arithmetic operations and with friendly user interface. In conclusion, there are something should be improved, for example, error handling and expanding the range of supported operations.