CSC 413 Project Documentation

Spring 2023

Naing Htet

921634165

CSC413.01

GitHub Repository Link

Table of Contents

1 Introduction

1.1 Project Overview

The program is basically a calculator program. It is a program which allows the basic function of a real calculator aka it allows the user of the application perform simple addition, subtraction, multiplication, and division. The calculator has additional features like the ability to calculate powers and handle parentheses to perform more complex calculations in the right order. The calculator is created using multiple different parts which together combine to a basic calculator.

The program is a combination of 5 main java files, they are the Evaluator.java, the EvaluatorDriver.java, the EvaluatorUI.java, the Operand.java and the Operator.java. There are also other minor files such as the files that give priority to addition, and other aforementioned operations.

In this program, Evaluator.java is the brain of the calculator by breaking down the given math problem into smaller, manageable pieces and perform the operation like addition or subtraction. Whereas EvaluatorDriver.java is basically more backend in a way that it is basically for the person creating the code. EvaluatorDriver is a test runner for our calculator and through that file, the programmers would know if his code met certain requirements. On the other hand, EvaluatorUI.java is the visual aspect of the program since it is a just a interface with a layout of buttons which together work to allow users to enter basic math problems that they want to solve. Finally the Operand.java and the Operator.java is the foundation behind the program by helping the program recognize and deal with numbers and mathematical operations provided by the user.

1.2 Technical Overview

The program comprises of an implementation of a mathematical expression evaluator in Java which uses the Stack data structure and HashMap. The Evaluator.java class is the main central component which utilizes two Stack objects, one for operands called operandStack and one for operator called operatorStack, and implementing a method called evaluateExpression that tokenizes input expressions applies the data of the HashMap to handle operator precedence and parentheses. Operand and Operator objects are represented with their respective classes, Operand.java and Operator.java. The Operator class is abstract, extended by specific operator classes, and uses a static HashMap to map strings to Operator instances. The EvaluatorUI.java class provides a GUI, built with Swing and other, for user input. Whereas the EvaluatorDriver.java functions as the main driver program, offering a command line interface that allows manual input of expressions or automated testing via a pre-populated HashMap of test expressions read to be used for testing purposes. Finally, the exception handling is applied for invalid tokens and expression evaluation issues.

1.3 Summary of Work Completed

I have completed the calculator program. I have implemented the handling of parentheses so that it could be used to have complex mathematical expressions by allowing correct order of operations following mathematical conventions. The implementation of handling for parentheses is very important since without it, the expression evaluator would not be able to handle complex calculations accurately. In addition, I have worked on the Operand class to ensure it function correctly and allow the program to recognize the mathematical operations.

For that to happen, I had to write code to validate and manage numerical tokens(operands) in the mathematical expressions since the Operand class serves as a foundational building block of the expression evaluation and without it, the program would not have done any mathematical expressions correctly. Finally, I constructed a HashMap in the Operator class to handle different mathematical operations. The map links string symbols to specific instances of Operator subclasses, allowing the evaluator to recognize and execute these operations based on the priority of the token in the input expressions. Through the use of HashMap, I was able to create a flexible mechanism which support a variety of mathematical operations, removing the hassle of having to take too many memories. The result of all of those additions, I made my calculator program functions by allowing it to work correctly and efficiently.

2 Development Environment

The version of Java that I am using is the latest standard version of java which is Java 19. Did not use Java 20. Finally, I used the latest version of IntelliJ IDEA Ultimate which is 2023.1.2.

3 How to Build/Import your Project

1. Clone the Repository. Open the terminal/command prompt and navigate to the directory where you want the project to be located. Then use the command git clone https://github.com/csc413-SFSU-Souza/csc413-p1-Codingjackking.git to clone the repository.
2. Install Java Development Kit (JDK): Ensure that you have JDK installed on your system. If not, you can download it from Oracle's official website.
3. Install a IDE: If you don't have one, install a Integrated Development Environment (IDE) like IntelliJ, NetBeans, Eclipse or other.
4. Open the Project in the IDE:
    a. For IntelliJ IDEA, choose "Open", then navigate to the cloned repository and select the project folder.
    b. For Eclipse, choose "Import" -> "Existing Maven Projects", then navigate to the cloned repository and select the project folder.
5. Check Project SDK: Make sure the project's SDK is correctly set to your installed JDK.
6. Build the Project: In IntelliJ, you can use the Build -> Build Project menu option. In Eclipse, you can use Project -> Build All.

4 How to Run your Project

Look for a Driver or any classes with the main class, which in this project is called EvaluatorDriver.java. Right click on it then look for a green play button on the menu and click on it then the program should be up and running.

5 Assumption Made

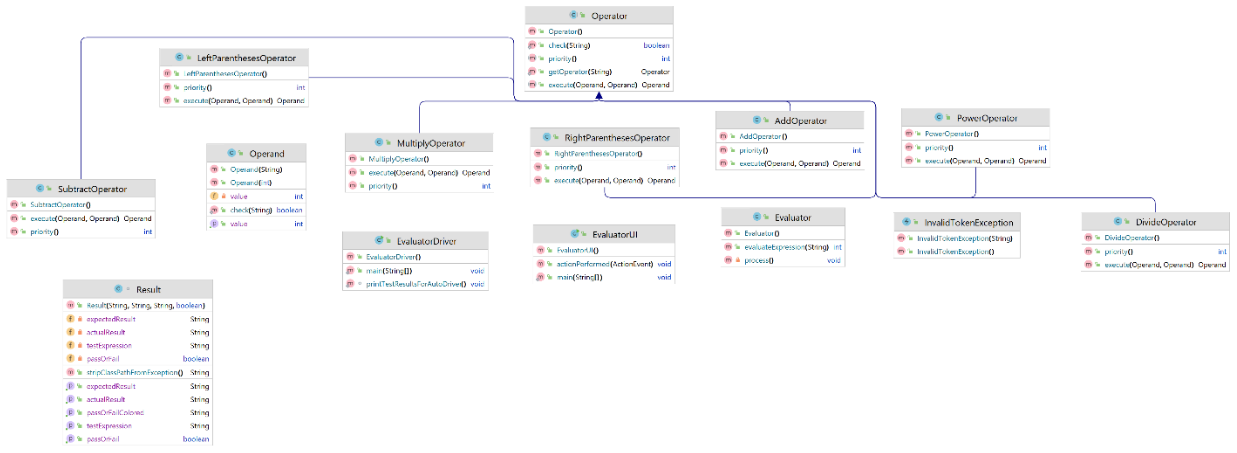
1. You should assume that the program will not accept negative number to be inputted into the expression. Might break the program by inputting negative number.
2. You should assume that you are only allowed to use binary operators.
3. You should assume that the program will not accept floats and will only work with integer values.

4. You should assume that only use of parentheses is allowed, no usage of bracket, and curly bracket.
5. You should assume that any error resulting from proper operations in the expression, would result in Invalid Token Exception to be caught and thrown.

6 Implementation Discussion

1. Changed the evaluateExpression in Evaluator.java to have parentheses handling so that it allows for more complex mathematical expressions, honoring the standard order of operations like PEDMAS.
2. Created two subclasses of operator called left and right parentheses operator and added them into the HashMap because it allows the program to recognize and correctly process parentheses as operators and not as some type of potential error. This maintains the standard order of operations and allow for the program to handle complex mathematical expressions that required usage of parentheses as a operator.

6.1 Class Diagram



7 Project Reflection

Reflecting on the project, I am in awe of how complex a simple application such as a calculator is and how a simple application like a calculator requires many other codes like apiguardian-api other than the main required code used to program the calculator. By programming a calculator which I considered as simple, taught me so much more about the different knowledge you need. Through just the adding of new codes to the already almost code have taught me many things from the concepts of mathematical operations to the handling complexities like the parentheses. It also allowed me to realize how the order of thing were done in a program, and how you must use data structure like Stack for doing things like order of operations. I for sure have learnt the value of a well-structured program which is not only are they flexible, but they also are created with the foresight of future edge cases and test cases. Despite facing challenges with operator precedence especially the parentheses precedence, I learnt the value of navigation through hurdles with continuous debugging and code refining, fostering my problem-solving skills. I also learnt the value of testing, which help ensure that my program have completed the required cases. Overall, I appreciate what implementing the calculator program have taught me and I am excited to apply what I have learned from this program to other future program and future jobs.

8 Project Conclusion/Results

In conclusion, the calculator project has been a success, in term of the teaching me many valuable lessons such as the importance of testing your code for errors. It also helps me learn the value of what it means to create a effective yet flexible application with usage of object-oriented programming principles, data structures, exception handling, and testing methodologies. In addition, the calculator was able to handles all the test cases that were there and allowed for testing outside those test cases by including potential edge cases.

The final result of this project is a fully functioning calculator which can do addition, subtraction, multiplication, division, and powers. The calculator can also follow the order of operation which is one of the most fundamental yet basic rule of math. The calculator also has exception handling capabilities by providing error messages for invalid tokens or expressions. Furthermore, the comprehensive suite of tests, including the set of pre-defined test expressions, ensures that the calculator can handle those type of expressions and probably more.

Overall, the calculator is a great start to future applications, and also shows potential future improvement such as the addition of new operations like the square root or the exponent, it also includes potential in improvement on the user interface.