CSC 413 Project Documentation

Fall 2018

Mohamed Farah

918378258

413.03

https://github.com/csc413-03-fall2019/csc413-p1-FarahMo24

Table of Contents

1 Introduction

[redacted] Project Overview

In this first project of the semester we designed a basic calculator. This basic calculator can calculate any valid expression that is fed in. The program works by receiving a mathematical expression, these expressions are then broken down into smaller pieces. Through the help of order of operations, the smaller pieces are evaluated in an orderly manner. After evaluating the expression, the valid answer is promptly displayed on the text field of the calculator.

The appearance of the calculator is very simple and intuitive. The calculator is presented as a normal calculator where the numbers between one and nine are displayed. All the basic operations are displayed to the right at the bottom, making it easier to find the correct operation while your operands are easily displayed.

[redacted] Technical Overview

In this project, the idea is built around infix expressions. The algorithm is built around to handle infix expressions and thus evaluate them in accordance to which value is an operand and which value is an operator.

To evaluate the expression, there is a method that takes the entire expression as a string. This string is broken down using the tokenizer, one by one we check whether or not it's an operand or an operator. In the case of an operand, the operand object is instantiated and called upon to not only check if the current token is an operand but to also return the value of the token as an integer. In the case of an operator, the object is instantiated, and called upon to check if the current token is an operator and then it is identified as to which operator the token really is.

Each token is pushed to the respected operator or operand stack, the method scans through the entire expression as to make sure nothing is missed. Once the scan is completed and every token is pushed to their respected stack, the program will process and determine which result is the correct answer.

[redacted]Summary of Work Completed

In this projected I started out by working on the class Operand, which is a utility class that the class Evaluator uses. By working on the utility class, it gave me the foundation I needed to understand what is going on in this project. Operand is a simple class setup but the only method that seemed difficult was the check method, this method checked whether or not the string token passed in was a number or not.

After the initial utility class, I moved over to the Operator package and worked on the other utility classes. In this case I started out with the abstract class Operator, in this class I made a hashmap. The key for the hash map was a string operator and the value was a concrete class that extends the abstract Operator class. Each operator has its own concrete class that extends the abstract class, each operator does its own operation and then returns an object. Each operator also has a unique priority number that classifies its order of operation. The abstract Operator class, like the operand class, has a check method to make sure that the token passed in is an operator.

Lastly, the class Evaluator was implemented slowly using both the understanding of operand and operator class. The eval method was partially implemented, the holes that needed to be fix was adding two operators, the open parenthesis and the closed parenthesis to the delimiter. Both of these operators were implemented and tested to make sure it worked with the operand and operators.

2 Development Environment

IntelliJ IDEA 2019.1.3 (Community Edition)

Build #IC-191.7479.19, built on May 27, 2019

JRE: 1.8.0_202-release-1483-b58 x86_64

JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o

macOS 10.14.6

3 How to Build/Import your Project

Step 1: Clone Project to a known folder on your desktop

Step 2: Create a new project on your preferred IDE

Step 3: Import only the “calculator” folder from your “Step 1” folder.

4 How to Run your Project

After importing this project from the previous steps, locate the “evaluator” folder.

After locating the “evaluator” folder, proceed to the “EvaluatorUI” class.

Depending on your IDE, locate your “Run” button and Run the “EvaluatorUI” class

A GUI calculator should pop open, enter your expression and click the “=” button to evaluate the expression.

5 Assumption Made

The first assumption is the user will not need any decimal answers; therefore, this calculator only returns whole numbers.

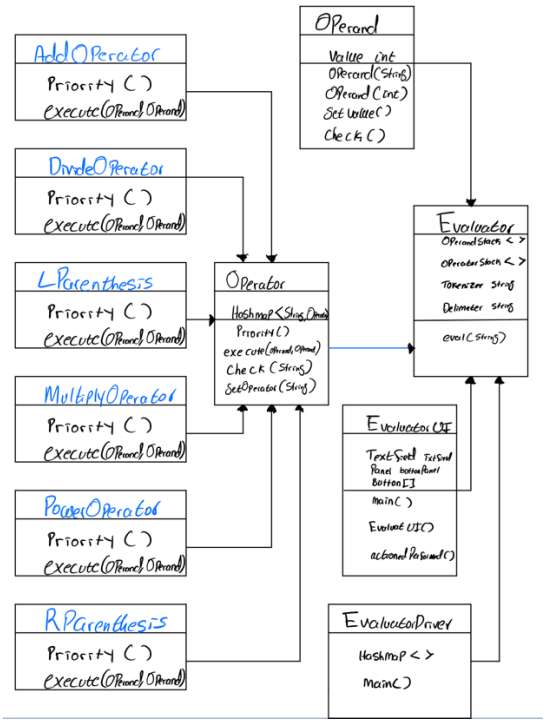
The second assumption is the user will enter valid expressions that make sense, any invalid expression entered will return an invalid answer.

The user will only need the basic operations to use this calculator, no other operators are included besides the basic arithmetic operators.

Lastly, the calculator will return negative answers depending on your operation, but the negative operator is not included in the functionality of this calculator.

6 Implementation Discussion

Class Diagram



The main classes above are, the Operator class, and the Evaluator class. Both of these classes are highly necessary for the program to function correctly. The operand class is also another important class, that not only checks if the operand is a number but also returns the value of the operand, which is crucial for

the final result. The EvaluatorUI is a user interface to setup the calculator design and print out the result clearly.

I've included both Right side parenthesis and left side, both of which have the same priority in the number system. The Abstract Operator class uses a HashMap which really consists of the operators as key values and the class of the operators as the values themselves.

Now with the Evaluator class, it uses both Operand and Operator class, to pass in tokens and to make sure each token is valid. After which the evaluator class uses stacks to execute the different operations. If certain operator is on the stack then an the second operand is popped off from the stack followed by the operator and finally the first operand. After which it is executed by calling the Operator and printed using the value of Operand.

7 Project Reflection

In this first project for this semester, I've refreshed a lot in the process of figuring out how to implement a class, an abstract class and a HashMap. The biggest difficulty in this project was figuring out how to construct an algorithm with the given incomplete algorithm. I googled lots of ways on how to figure out these methods and ideas on how to evaluate the numbers in the most consistent manner. Struggling through that process has given me the foresight to make sure I start early on these upcoming projects and to ask more questions when I get stuck on certain areas

8 Project Conclusion/Results

Overall this project has been a great introduction to the class, I will say that I have lots of things to learn and in the coming weeks I will take the time to practice problems on my own, such that it will help me

solve problems in future projects. I have in fact been using this opportunity for this project to review certain principles and thus was the key in helping me complete this first project.

One of the implementations that I have yet to complete is on the Evaluator UI, where I have not discerned the difference between CE and C. I will work on it until the due timestamp and see whether or not I can implement a better idea on how to discern both within my if statements.