CSC 413 Project Documentation

Summer 2021

Anmol Burmy

921143454

CSC 0413-01

GitHub Repository Link -

https://github.com/csc413-su21/csc413-p1-Burmy

Table of Contents

1 Introduction

1.1 Project Overview

This Java project is a simple calculator, in which a user can enter basic math expressions and get an accurate calculation of that expression. The calculator can perform operations such as addition, subtraction, multiplication, division and power operator for integers.

1.2 Technical Overview

This project is an expression evaluator GUI which calculates basic math expressions given as a string input by the user. These calculations are done by the Expression Evaluator algorithm, in which the string is tokenized and is then stored into two stacks - an operand stack for storing numbers and an operator stack for storing mathematical operators. From these stacks, we are able to pop out operands and operators to complete an accurate calculation of the expression entered by the user using the GUI.

1.3 Summary of Work Completed

- Finished Operand and Operator class, with operator class having subclasses containing different operators so that they can be used in the algorithm to calculate expressions.
- Finished the “evaluteExpression” function in the Evaluator class with the correct algorithm to calculate the expression based on the order of operations including parenthesis.
- Created a method in the Evaluator class that processes the operator stack until empty.
- Created actions for each button in the GUI in the “actionPerformed” method present in the EvaluatorUI class. Made sure that correct expressions and numbers are being shown on the GUI when pressing a specific button.

2 Development Environment

This project was developed using IntelliJ IDEA with Java version 16.0.1.

3 How to Build/Import your Project

To build/import the project, go to my repo and click the green “Clone or Download” button on your repo’s home page. Then select either HTTPS or SSH, If you are not sure what SSH keys are then use the HTTPS method. Copy the link and then open your terminal. This will either be Git Bash on Windows, Terminal on Mac or Linux, or the terminal for your Windows Linux Subsystem if you have it installed. In your terminal type: git clone repo_url_you_copied. Once the repo is cloned, follow the steps below to import your project into IntelliJ IDEA.

1. Select Import Project
2. Select the “calculator” folder as the source root of your project.
3. Keep the “Create project from existing resources” radio button selected.
4. Now keep clicking “NEXT” until the import is finished.
5. Once the import is complete, you can build the project by navigating to Build -> Build Project.

4 How to Run your Project

There are different ways to run the project. If you want to run the calculator GUI, just right click on the “EvaluatorUI” class and click “Run EvaluatorUI” from the dropdown menu. Or you can also select/open the class in the window and press the play button on top of the editor window.

This will launch the GUI Calculator.

5 Assumption Made

One assumption I made was that the EvaluatorUI class was going to be fairly simple to write since the code for the GUI was already given to us. I followed the videos provided by the professor and wrote the code, which worked perfectly, but I believe there was a more efficient and cleaner way to do the action event method. I tried doing it with some if-else loops but was not able to get it working. So I just went with the way the professor showed in the videos.

6 Implementation Discussion

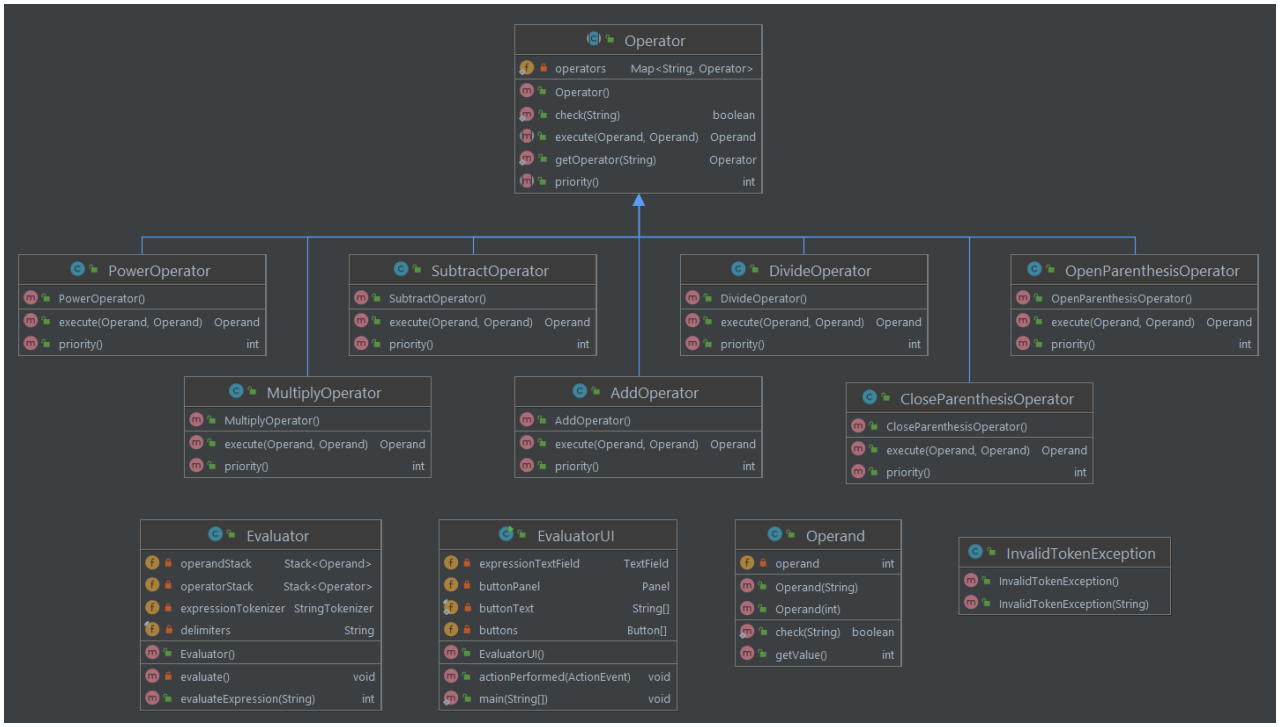
For this project, there is an Operand class that stores the operand tokens of our expression as operand objects. The class contains two constructors, one that takes a

string and one that takes an integer, a getter and a function that checks tokens to make sure the strings are actually numbers.

We also have an Operator class. It is an abstract class where we define our operator abstraction and also define a template for all our subclasses. All the child references are stored in a hash Map. The idea behind this hashmap is that we are going to have a table that maps string tokens to the operator objects.

The Evaluator class contains the main algorithm function of the program. In the Expression Evaluator algorithm, the string is tokenized and is then stored into two stacks - an operand stack for storing numbers and an operator stack for storing mathematical operators. While we have our tokens, if the token is an operand it is going to be pushed in the operand stack. Else if it is not an operand and is a valid operator, we fetch a correct operator from our Hashmap from the operator class and then check for the priority of the operators and do the execution depending on it.

6.1 Class Diagram



7 Project Reflection

I think I was able to get a good overall refreshment of Java OOP. When I first read the instructions for the assignment, I assumed that this project was going to be difficult for me. I couldn't figure out what I was supposed to do or even what files I was supposed to edit. But after reading through Evaluation of infix expressions, the videos provided by the professor and some help from my classmates in discord, made me understand the assignment very well. Also, I feel like if I had more time, I would have played around with the GUI more and maybe change the theme of the GUI calculator or add rounded buttons instead of the default ones.

8 Project Conclusion/Results

After having some difficulty on how to actually start the assignment, I was able to finish and create a fully functioning Calculator. I was also able to pass all the tests that were given in the evaluation driver and in the test folder. I tried to make the code as readable and understandable as possible by adding comments and making the code clean, so people or even I can understand what the code does in the future.