CSC 413 Project Documentation

Spring 2022

Beatrice Aragones

920614978

CSC413.02

https://github.com/csc413-SFSU-Souza/csc413-p1-bearagones

https://github.com/csc413-SFSU-Souza/csc413-p1-bearagones.git

Table of Contents

1 Introduction

1.1 Project Overview

The purpose of the project was to finish an almost-complete calculator program that evaluates mathematical expressions. It allows the user to use a user interface that resembles a simple calculator that one would normally see, such as all 10 digits of 0 to 9, an equals sign to evaluate the expression, as well as function buttons that allows the user to use addition, subtraction, multiplication, division, exponents, and parentheses. Lastly, the user is allowed to either clear the last integer or function using the “C” button, or the entire mathematical expression that they have typed using the “CE” button.

1.2 Technical Overview

The purpose of the project was to practice object-oriented programming as one of our first assignments in software development. We finished a calculator program that implemented the use of various data structures, such as Stacks and HashMap, as well as the three pillars of Object-Oriented Programming (OOP). The HashMap was used to create a key-value map, where the key was the operator and the value was the creation of a new object in the child class to reference the priority of the operator (which determines the order in which the operators are executed in comparison to each other) and the way the operator should execute on two operands together, which was in the parent class. Two stacks, an operator stack, and an operand stack, were used to demonstrate the concept of “Last in, first out” (LIFO) to determine the order in which different operators are executed. The Operator class is an abstract class that contains abstract methods. This is because it serves as a base class for the different child classes in the program.

1.3 Summary of Work Completed

Overall, I believe that I was able to complete the entire program. All the 61 tests, including the 16 tests in the Evaluator class pass with a green checkmark. Whenever I type in a mathematical expression, I get the answer that I was expecting. Additionally, I get expected errors if there are extra parentheses or repeating operators in a row. The GUI also seems to work as intended and displays the functions correctly.

2 Development Environment

This program was implemented in IntelliJ IDEA 2021.2.1 (Ultimate Edition) on Java 14.0.2

3 How to Build/Import your Project

To build and import my project from GitHub to your IDE such as IntelliJ, you first need to clone the repository. First, click the green "Clone" button on the right side of the page and copy the link. In the terminal, then type "git clone [insert repo URL link here]". The repo will be placed in a folder that you will have to locate when importing the project into your IDE. After opening your IDE, select "Import Project" and then locate the folder where the repo was cloned. Here, be sure to click on the folder labelled as "calculator" as the source root of the project. Next, make sure that "Create project from existing sources" is selected before continuing. In the page where it has the project name, location, and format, it can be left alone, and you can move on. For the page after that shows two folders (a source code folder and a unit test folder), have both items selected. Then, you'll see a list of resources that are used for the unit tests. Select next after the two modules (source code and unit tests) are selected. Lastly, select the JDK version (I used 14). Now, the importing process is finished, and the project is imported into your IDE.

4 How to Run your Project

When running my project, there are two options that you can choose.

If you want to run the program with the user interface that resembles an online calculator, then locate the file under src → main → java → edu.csc413.calculator → evaluator → EvaluatorUI and right click it. You should be presented with the option to "Run 'EvaluatorUI.main()". Once clicked, you will see a calculator on your screen. Then, you can use your mouse and create a mathematical expression that you want to evaluate. Once you're done using the program, you can click the red X button to exit out of it.

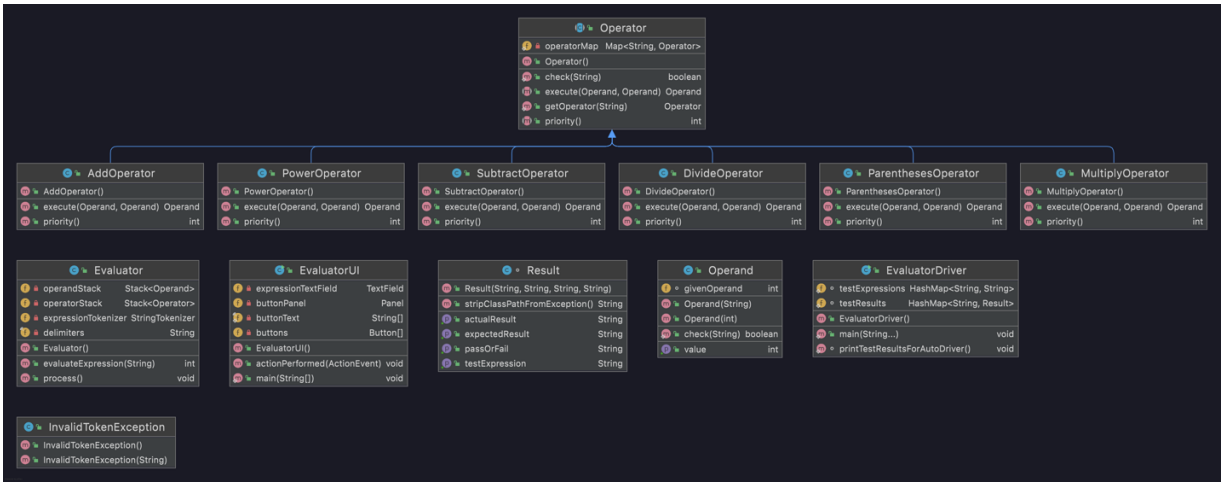
If you want to run the program without the user interface, then locate the file under src → main → java → edu.csc413.calculator → evaluator → EvaluatorDriver.java and right click it. Then, you will be presented with a prompt to enter an expression. Here, you can type out any mathematical expression that you wish to evaluate and press the "Enter" key once finished. After you are finished running the program, locate the red square icon to stop processing the program.

5 Assumption Made

When running the program, assume that only integer values will be accepted when trying to evaluate an expression. This means that no decimals can be used, and when evaluating division, it will be rounded to the nearest whole number. Next, you are not allowed to use negative values in the program. However, it is possible to get negative values after evaluation. Another assumption to be made is that only parentheses, or round brackets will be accepted. The program will not recognize square brackets or curly brackets. Lastly, the calculator can only perform simple math. It cannot calculate things like factorials (!) and inequalities (<, >). If these are included in the expression, you will get an invalid token error and the program will stop. If a space is entered, you will get a null expression and the program will end as well.

6 Implementation Discussion

6.1 Class Diagram



6.2 Design Choice

When working with the actionPerformed() method, I chose to use nested for loops and if-statements because I felt that it were easy for me to traverse all of the buttons in the array. The if statements would help determine if the buttons that were pressed were special cases, where “C”, “CE”, and “=” had special functions that were different from the other buttons. For my process() method in the Evaluator class, I chose a while loop to evaluate the stack until it was empty because I thought that it fit the situation best. I also chose to use if-statements to determine special cases in the stack, such as the situation where there is an empty stack and when parentheses are used.

7 Project Reflection

When I felt a little bit overwhelmed when looking at the various components of the program that I had to complete by the initial deadline. I started watching the video on how to import and start my program on one of the first days the assignment was posted on iLearn because I didn’t want to fall behind. I personally found it helpful to have all the hints that were given in the various videos. Prior to watching the starter code video, I really struggled with knowing where to start exactly, so having the hints and tips really helped.

One of the biggest struggles I had with finishing this program was implementing the two parentheses and understanding how they work in stacks. Because this portion took the longest for me to understand, I ended up finishing the GUI program first to use my time wisely. What helped me the most in implementing the parentheses was to physically draw out on paper an example of a simple mathematical expression that involved parentheses and try to visualize how it would look like in stacks. After I got the logic down, I still struggled for a while to get all the tests in the Evaluator class to work. Personally, I had never done debugging before, but one of my fellow classmates suggested doing so and gave me a tip to “step into” while debugging the try-catch statements in the test classes. This really helped me understand where my issues in the program were, and it even helped me catch several EmptyStackExceptions that I didn’t even know I had. After doing the necessary changes, I was able to get all the tests to pass, thus completing my calculator program.

8 Project Conclusion/Results

Overall, I would say that this first assignment helped me refresh many of the OOP principles that I previously learned in my past semesters. This assignment also gave me the opportunity to learn how to use Git and GitHub in the programming process. Other than refreshing my memory, I also ended up learning some very useful concepts, such as debugging to help find hidden errors that may be causing my program to not work as intended. I believe that this was a very good first assignment for this class and has started to prepare me for the next one.