CSC 413 Project Documentation (Calculator)

Fall 2019

Angelo Reyes

917566993

CSC413.02

Repo Link: ***https://github.com/csc413-02-fall2019/csc413-p1-angiereyes99***

Table of Contents

1 Introduction

1.1 Project Overview

The program in this repository is a calculator GUI with simple calculator operations such as add, subtract, multiply, divide, and power operators for integers.

1.2 Technical Overview

Using a variety of abstract classes, subclasses, constructors, as well as Action Event methods, all worked to together in implementing the correct functions for creating a calculator GUI that executes the proper function based on the math operator. In addition, multiple unit tests for the operators were also implemented to test the function of the operators.

1.3 Summary of Work Completed

The key highlights of work done in the program was creating the proper subclasses for the operators so that they are given with the correct "priority" and execution. "Priority" meaning that they have the correct order when calculating multiple operators at once. In other words, I made sure they follow the PEMDAS rule. I created constructors that can be used in other classes to get operand and operator values. I contributed to the "Evaluator" class that used peek(), pop(), and push() methods to check for priority of operands and operators in their stack. I also made the correct actions for the buttons in the GUI in the "actionPerformed" method in the EvaluatorUI class.

2 Development Environment

The IDE I Preferred to use in this project was IntelliJ with a java version of 1.8.0_161.

3 How to Build/Import your Project

To import this project (Assuming you have git installed in your device), clone the SSH or HTTPS key in the GitHub link above in the title. Open your terminal or GitBash and type "git clone -SSH or HTTPS keys you copied -. Once you have successfully cloned the repo, you should be able to see it in your computer.

Afterwards, go on the IDE of your choice and choose "Import Project" (This is shown in the opening screen on IntelliJ). Make sure to click the calculator folder as the root and leave the default settings and click next until you see "Finish". You should be able to see the calculator folder along with its existing files in your IDE now.

4 How to Run your Project

There are multiple ways to run this particular project. If you want to run the actual calculator GUI, then you simply find the file that is named "EvaluatorUI" inside the "evaluators" folder inside the calculator project folder. Right click the file and hit "Run EvaluatorUI". In the next few seconds, it should redirect you to a small calculator where you can interact with the keys. Note that when you finish a calculation, you need to hit the "C" button first to clear previous calculations so that you can start a new one. Anything divided by 0 will also not be calculated.

To run the test scripts, go to the test folder inside the calculator project folder and follow the same exact steps from the GUI and it should run the given tests inside the folder.

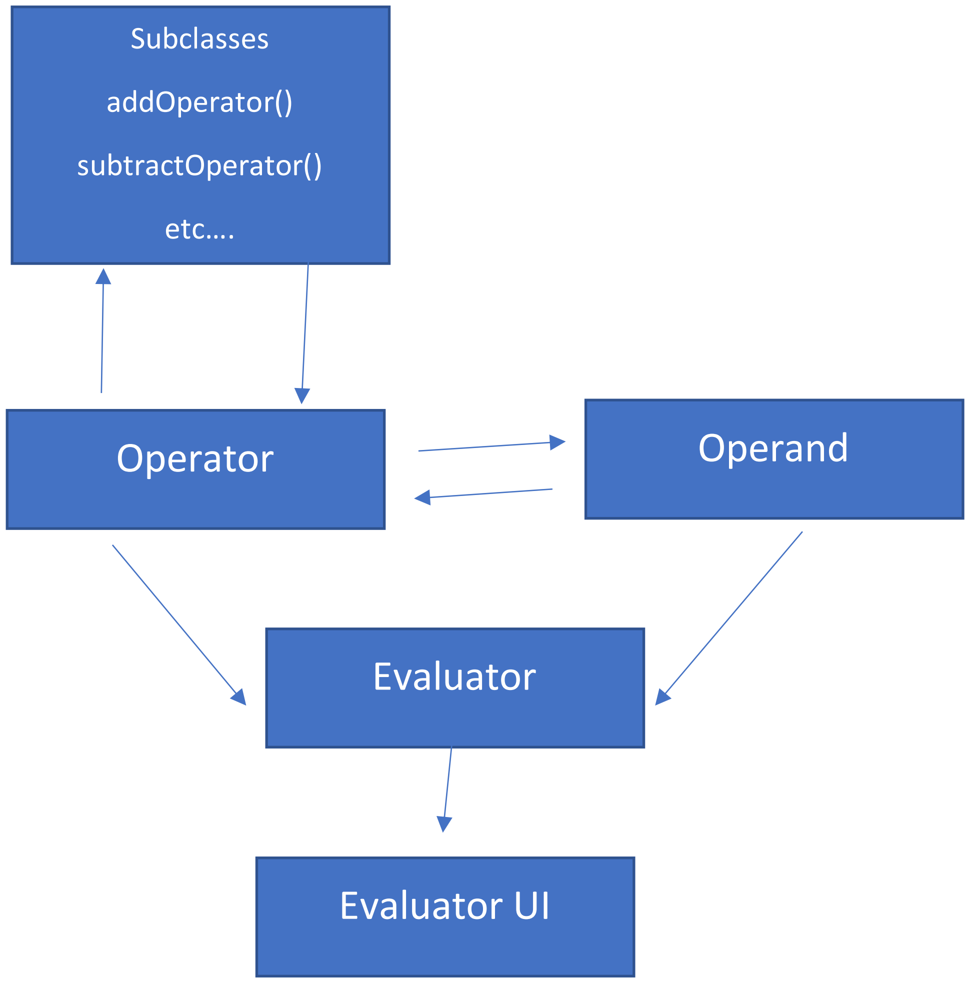
5 Assumption Made

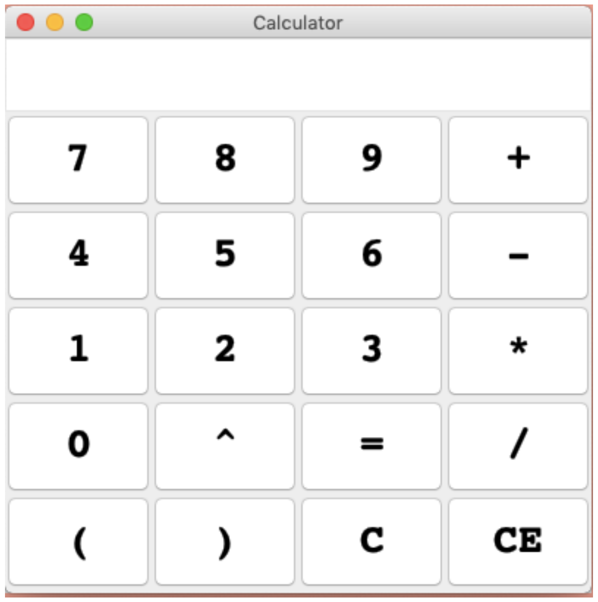
I assumed that the only major part of the skeleton code given was the actual GUI class and that the others needed to be completely changed or added. But the given code for the other classes played a crucial role in my understanding of what the code does and tries to do for the rest of the program.

6 Implementation Discussion

The main data structures utilized in this project were Stacks and a HashMap. The HashMap was implemented to hold the operator symbols and their function, while the stacks were for organizing the priority of the operators and operands based on calculations done in the calculator. Much of the process of said priority was mainly implemented in the specific operator classes and the Evaluator class. The Evaluator class was implemented using numerous for and while loops to push and pop operands and operators in and out of the stacks based on the correct priority so that calculations will be correct. There are numerous docstrings and comments throughout the code that try to explain my thought process in specific code as well.

6.1 Class Diagram





7 Project Reflection

The project overall was a very good experience in refreshing java concepts that I haven't touched in quite some time. My process of coding the program mainly revolved around the "divide and conquer" technique. I was able to be self-sufficient in the work I did and utilize problem solving techniques to get the work done. I viewed numerous resources as well as the tutoring sessions to help me progress through the project. I made a couple mistakes along the way; specifically, the Evaluator class and Evaluator GUI. I also would try to clean up come of my code in the "check" method as the only approach I took was iterating through the strings in the HashMap and check to see if the token is equal to the keys.

An important piece of the project to note was when I created the "execute" method for the divide, power, and subtract operator classes. As you may see in the code, the arguments the method takes were op1 and op2. So, when doing the correct operation for the methods in its specific class (i.e. for divide it should be op1/op2 and subtract should be op1 - op2), the results would be incorrect. An example would be, if I were to do an operation such as 10 - 5, it would give me -5. Meaning that it was actually doing it backwards. This also applied when diving such as, 5/10 would give me 2. Seeing this, I ran the test scripts for the operators and they all passed. But when running the Evaluator test, only 3/8 tests passed. I decided to just switch the op placements (i.e. op2/op1, op2 - op1) and it surprisingly passed all tests for the Evaluator test class as well as correctly compute in the GUI. But for the particular operator tests, they would fail. I was not able to debug this issue, so I just finished off the project with the working GUI and the passed Evaluator test.

8 Project Conclusion/Results

Even though the bug mentioned in the reflection caused the Subtract, Divide, and Power Operator tests to fail, the calculator GUI is fully functional and calculates correct operations in the correct order that follow the PEMDAS rule. In addition to a working GUI, the Evaluator, Operator, and Operand Test classes all passed. The code provided tried to be readable and understandable for someone who views it. In certain methods, I still provided comments to clarify any thoughts about the code. I also provided citations of code that was influenced by outside sources.