Gabriel Alfaro

ID: 913142053

CSC 413.02 Summer 2018

https://github.com/csc413-02-su18/csc413-p1-GabrielAlfaro

2.Introduction

a. Project Review

This project is designed to create a basic calculator that evaluates a mathematical expression. The project takes in a string expression and then evaluates the expression and returns the value of the expression.

b. Technical Review

In this project, we practice using HashMaps, stacks, and abstract classes to perform the functions of a calculator. The Evaluator takes in a string mathematical expression, parses it by token and checks whether the token is an operand or an operator and pushed them onto their respective stacks. Then in the Evaluator class we pop the operands and the one operator from the top of their stacks and execute the operation that matches the operator stack reference. The HashMap inside the Operators class and can be referenced with the .operators extension to access the HashMap from the Evaluator class and matching the token string with the operator. This also allows for us to create another subclass in Operator and Operand to check if the token is an operator or an operand. Then a loop will run until the operator stack is empty and the final value from the mathematical expression is pushed into the Operand stack. The loop will execute the

c. Summary of Work Completed

I have a Evaluator.java file that runs but doesn't produce the correct outputs. I got stuck with the implementation of the second while loop to check for an empty operator stack. Along with not knowing how to process the left and right parenthesis tokens in the Operator subclasses. I did not fully complete the assignment but am still very interested in figuring out what I was missing.

3.Development Environment

a. Version of Java used

JDK 1.8 (Default)

b. IDE used

NetBeans IDE 8.2

4.How to build or import your game in the IDE you used

a. First you clone the java files from the link to the repositories in the title page.
- Unzip the java files to be used later.
- Open up Netbeans IDE at least version 8.2
- Click on File -> new Project and click on Java with existing sources.
- Click Next then browse your computer to locate a version of the cloned java files.
- Once the main source package “calculator” is found select it and click Finish.
- Finally, run the EvaluatorGUI.java file that will create the calculator.

5.How to Run your Project

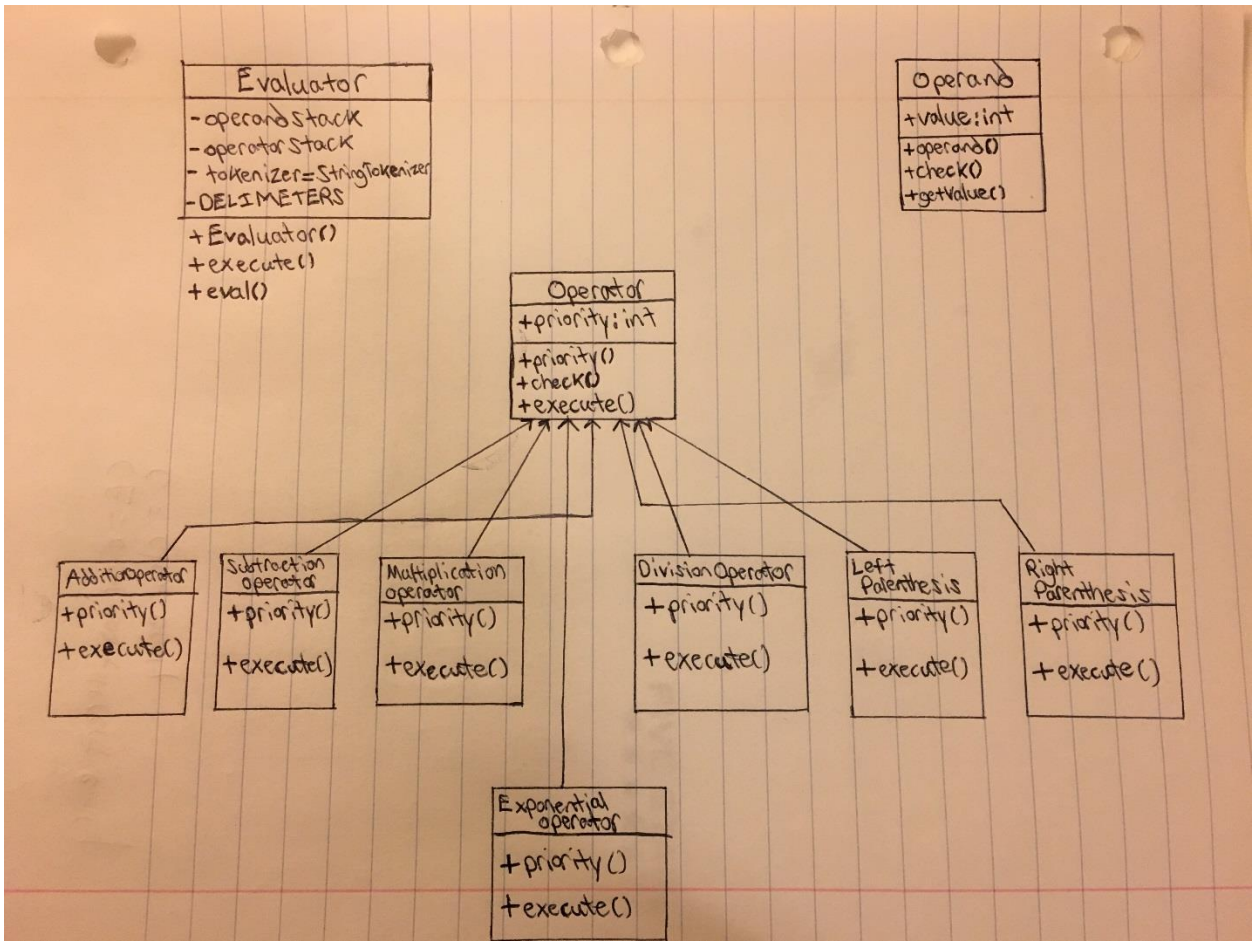
To run the project, go to the EvaluatorUI.java file and then right click and hit run. From there the GUI should run and a calculator will appear ready for user input.

6.Assumptions Made when Designing and Implementing your Project

- User will only input valid expressions from the calculator.
- Assume the user wants an integer value returned.
- Assume all operands are integers.

7.Implementation Documentation

a.



8.Project Reflection

The project was reasonable and very interesting to work on. I was not fully prepared for how fast paced the class was, even though it helped review important topics like inheritance, stacks and hash maps. I don't have any real suggestions regarding ways to improve the project if anything I would want more comments to help explain all the moving parts in the programs.

9.Project Conclusion and Results

The project overall did not work, I get an error for all strings in the output for the EvaluatorTester.java. I had trouble creating the method in Evaluator.java to loop and evaluating the operator stack until it was empty. I was also unable to figure out

how to properly implement the parenthesis in the expression, and how to value their priority. The Evaluator will parse and run through the tokens, pushing them onto the stack and then creating a new Operator object with the correct object reference. When the operators and operands popped they will then be executed by using the operator as the indicator to determine the subclass of Operator and passing the two integers to the correct execute function. The result is then pushed back onto the operand stack. Priority is checked for each operator class and allows for the multiplication and division operators to be executed before the addition and subtraction operators. Overall it wasn't the best evaluator, but it has gotten me more invested into reviewing and eventually figuring out how to finish the code.