CSC 413 Project Documentation

Fall 2018

Keith Rich

915799591

CSC 413.01

https://github.com/csc413-01-fa18/csc413-p1-GuantanamoBae

Table of Contents

1 Introduction

1.1 Project Overview

Implementation and construction of calculator that follows the rules P.E.M.D.A.S. as is Parenthesis, Exponents, Multiplication, Division, Addition, and Subtraction.

1.2 Technical Overview

- Mapping a HashMap with operators with constructors that perform said operation.
- Implementing two separate stacks. One for operators and one for operands.
- Following the algorithm found here: http://csis.pace.edu/~murthy/ProgrammingProblems/16_Evaluation_of_infix_expressions) That was provided in the assignment descriptions.
- Pushing, pooping (heh), and performing the operations in the specified order of the algorithm.
- Lastly returning the value of the expression which is inputted.

1.3 Summary of Work Completed

- Evaluator Class

The evaluator class was the bulk of the work. Created a function that runs outside of the main eval function that performs the operation once a “)” is found.

Towards the middle of the eval function a few checks were implemented. Checks include a check for the function call once “)” is found. Check if the operator stack is empty. If one operator priority is higher then the new operator. Checking to push or pop depending on what operator is added ex “(” and “)”.

Running through one more check on weather or not the operator stack is empty and operator stack priority. Then finally returning the top of the operator stack.

- EvaluatorUI Class

Created a switch statement for the following cases. “=”, “C”, “CE”, and default to other buttons pushed.

“=” evaluate the expression.

“C” and “CE” are implemented the same way and are done by setting the text field to “”.

Default enters the text field and action command and displays it in the calculator view.

- Operand Class

Constructor that takes in a string token and converts it to an integer as well as a integer constructor that returns said integer. A get value functions that returns the values of said token. And lastly a check to find if the token is a int.

- PEMDAS Classes

Constructors that build objects that contain a priority and a operation of the operands that are passed to said constructor. These objects include add, subtract, multiply, divide, and powers.

- Operator Class

This is where the implementation of a HashMap with the corresponding operators.

Because the HashMap is an abstract class we must access it through a public function called "functionCall".

A function that returns the operator from said HashMap.

Lastly a check for if the token is an operator.

- Open & Close Class

Constructor that sets a unique priority and returns a Operator that contains nothing of value.

2 Development Environment

- 1.8 (java version "1.8.0_171")
- IntelliJ IDEA Ultimate

3 How to Build/Import your Project

- If you have GitHub and a computer and fingers.
- Cd to a directory of your choosing
- Git clone the URL of the repository
- In your favorite compiler select "open from exiting sources"
- Find "calculator", select it, specify that you want to wrap the program in Gradle
- Select one of the buildable classes ie. EvaluatorDriver, EvaluatorUI, or any of the tests under "test" folder under src.
- Right click on the chosen class and click "Build" under the class chooses.

4 How to Run your Project

- After building the said class press the green sideways play button located at the upper right side of your IDE

5 Assumption Made

I remember getting about halfway through the implementation of the algorithm from the website that was provided and thinking. "Well this wont work if you have a large string with lots of different priorities." Turns out it works super well. So long as you don't have "))".

When first working within the project I didn't realize that other classes where needed (such as AddOperator, OpenParenthesis ect) and spend a large amount of time working all of these classes in the Operator class. It wasn't till class the following week that I was told we should be keeping all of these functions in separate classes.

Another challenge was because the HashMap was inaccessible due to abstract declaration, having to determine whether to use an interface or a function came into mind. Obviously, function was the best choice but determining that only came after a class discussion as well.

6 Implementation Discussion

All operation classes are extended with Operator class. All of those exist within the Operator package. We also import the evaluator class package as well. As for the Operand class hierarchy being extended by the Evaluator class that contains the stacks that we pop and push from. All of those are manipulated by the Driver or the EvaluatorUI.

6.1 Class Diagram

7 Project Reflection

Denial: After reading the assignment description and struggling though what's is being asked I was thinking to myself. "Eh, can be that hard. I'm going to sleep on this for a few days just to let it sink in. It's the first assignment, it can need that much attention."

Anger: "What the f is Gradle anyway! Why are there options for running the code! I can't even get this to compile without running an error. What do you mean the abstract class can't be initiated, just initiate it!"

Bargaining: "Google isn't helping. Maybe ill talk to Sauza and see if he can help me. I just need to phrase my questions so that he just tells me everything I need to do without looking like a moron or that I don't belong."

Depression: "I can't do this. I'm just going to drop this class and play some Overwatch… Oh but what if I change this bit of code. Dose it push to the stack correctly? No… okay. Time to binge eat."

Acceptance: After banging my head against the keyboard for a while things are actually making sense and I'm able to follow the code. The website provided dose a great job of explaining what needs to be done in what order when it comes to the evaluator class. "Alright, I so I got most of the tests passed but I'm still working on these double open parentheses. What if I pop one if it reads two of those tokens in a row? Huh, that worked. Let me run the test now…"

8 Project Conclusion/Results

All 59 tests pass. The UI is working as intended. Last thing that needs to be done is to make sure this is a PDF and send my code up to the Git Gods.