***CSC 413 Project Documentation***

***Summer 2023***

***Wing Lee***

***920558688***

***CSC-0413-01***

[***https://github.com/csc413-SFSU-Souza/csc413-p1-ayanamesu***](https://github.com/csc413-SFSU-Souza/csc413-p1-ayanamesu)

* Introduction
* Project Overview

On this assignment, the goal is to build a program that is an Expression Evaluator, that is able to take in an expression with any operators and give a logical result. The second part of the Project is to finish building the calculator GUI so we can put in expressions on the Calculator and get a result from the calculator.

* Technical Overview

An in depth explaination of this project would be that we are given a half complete program of an Evaluator Class which takes in operands and operators the user inputs. Inside the Evaluator Class we have a constructor which initializes two stacks operatorStack and operandStack these two stacks are used to store operands and operators during the evaluation. We are also given the evaluateExpression method, this method takes an expression as input and returns the evaluated result as an integer. Next we are also given an abstract Operator Class that serves as a main class for different types of operators and expressions.   
Inside the Operator Class is a hashmap that maps between the operator tokens and instances. After the Hashmap we need to set our priorities and checks and getOperator with their respective return values. Next is the Operand Class which has a method that retrieves the value of an operand and checks if a given token is a valid operand by parsing it as an integer. Then we need to work on the Evaluator GUI, where we use the given interface and, inputting the actionPerformed method to make it so that the calculator works in a similar way as the evaluator Driver, along with the added on 20 button display when the program is "run".

* Summary of Work Completed

The Summary of the work Completed is as follows I first started with the Operand Class and From there I got the Operator class working by finishing the requirements of the hashmap which are operators such as --> " +/\*-^()". All of these Operators has its own Classes which defines their uses. After finishing with the Operator Classes I then moved on to the Evaluator Class, this is where all the problems started happening when trying to run "All the expressions failed but 3, but even then 2 out of the 3 aren't even right results" I then wrote a private construtor called process and called it each time when needed to perform an evaluation of an operator with its operands . Then I wrote an if, else if statement to check for the left parenthesis.

* Development Environment

The Java version used is Java 15 SDK   
The IDE used is IntelliJ IDEA 2023 1.2 Ultimate Edition

* How to Build/Import your Project

I first imported this assignment from GitHub repository and copied the HTTP link and then went to my Terminal and used the command git clone <https://github.com/csc413-SFSU-Souza/csc413-p1-ayanamesu.git>. Then I began building the project by opening IntelliJ IDEA and selecting the Import Project, Then select the calculator folder as the source root of the project > Create Project from existing sources > Enter in a project name then keep hitting next > Make sure the 2 items in each tab are selected one for test case and one for the code assignment. Once finished you should see the C symbol next to the files.

* How to Run your Project

After Importing the project to run the project you would click the green button that looks like ">" on the top right of IntelliJ IDEA next to the "Edit Run/Debug dialog". You should be able to run the program but the results would not be as expected the programmer "me" would have to change that. To edit the configurations you would select edit configurations then select either the evaluator Driver or the Evaluator UI in this case if you want the print out all the accepted and expected results you would have to input "AUTO" on the arguements section which is under the main class "edu.csc413.calculator.evaluator.EvaluatorDriver"

* Assumption Made

Some assumptions to be made when I implemented my project is that the calculator does not take in floats but integers, Another assumption is that the calculator does not take in negative numbers in its expression meaning you cannot enter in "-6+2" when entering this in and pressing enter or "=" sign you would recieve an error or null.

* Implementation Discussion

a)   
Implementation for this project, first we would start with the Operand class and work our way to the Evaluator Class by finishing the Operator Class along with its hashmap and its fellow Operator Classes such as these symbols --> " +/\*-^()". Then we are finally allowed into the Evaluator Class which alot of things started to go wrong. First we needed to change the delimeters by adding in "()" into it since the Accepted and Expected Results require a parenthesis check, then I wrote a if else statement to check If the token is either a left parenthesis or right parenthesis, it then handles them accordingly. If it's a left parenthesis then it pushes a LeftParenthesisOperator onto the operatorStack. If it's a right parenthesis, it calls the processOperatorStack() method to perform operations on the operator stack and then discards the left parenthesis from the operatorStack. Then if the token is a valid operator other than the left and right Parenthesis then it gets the operator object by using the Operator.getOperator() method. Then I changed one of the while loops with its contents copy and pasted into a new method called Process() which when needed to I would just use the line "process()". Then I used similar logic on the implementation of the Evaluator UI basically making an if statement stating that if the calculator button "C" is pressed then it clears all entry by printing (" ") which basically deletes all entry. Then I continued off that with a statement declearing if the button "CE" is pressed it would clear last entry meaning if you have "77+" when "CE" is pressed you would have "77".

b)   
For UML diagram Please look at Documentation Folder because the Image is way too big for this docx. The png is called "diagram Calculator.png"

* Class Diagram

For UML diagram Please look at Documentation Folder because the Image is way too big for this docx. The png is called "diagram Calculator.png"

* Project Reflection

In my opinion the project isn't seen as too difficult at first glance but the hardest part for me was where to start. I find that really difficult when coding and programming but luckily this time I was told where to start and it was the Operand Class then into the Operator class and so on... I also spent majority of my time the first day trying to piece everything together like looking through multiple test case and see what is the return values of the operators and so on and also reviewed java since I hadent taken Data structures and Algorithms in 2 years time. Another thing to reflect on is that I have a hard time on this documentation because its hard for me to explain and express thing properly like the way I want it. Like inside my head I have this all planned out but when it comes to explaining it I tend to go all over the place.

* Project Conclusion/Results

In Conclusion this project was a good exercise and experience in reviewing java programming as a whole using topics taught from CS 220 and applying it to this project. Even though everyone has taken CS 220 there are multiple ways and steps to get the same results on the Project meaning all passing results and getting the calculator UI to write and print.