CSC 413 Project Documentation

Spring 2020

Duy Nguyen

Student ID: 917446249

CSC 413.03

GitHub Repository Link

<https://github.com/csc413-03-spring2020/csc413-p1-duynguyen2>

Table of Contents

[1 Introduction 3](#_Toc522827688)

[1.1 Project Overview 3](#_Toc522827689)

[1.2 Technical Overview 3](#_Toc522827690)

[1.3 Summary of Work Completed 3](#_Toc522827691)

[2 Development Environment 3](#_Toc522827692)

[3 How to Build/Import your Project 3](#_Toc522827693)

[4 How to Run your Project 3](#_Toc522827694)

[5 Assumption Made 3](#_Toc522827695)

[6 Implementation Discussion 3](#_Toc522827696)

[6.1 Class Diagram 3](#_Toc522827697)

[7 Project Reflection 3](#_Toc522827698)

[8 Project Conclusion/Results 3](#_Toc522827699)

# Introduction

## Project Overview

This is a calculator program that works like a simple calculator would. It can add, subtract, multiply, etc. only whole numbers. It can also use parenthesis to specify certain operations to be done first. It does throw errors when there are too many of one operator in a row, missing parenthesis or not enough numbers. It takes the texts and scans through to get rid of unnecessary spaces and does each operation depending the PEMDAS order (Parenthesis, Exponents, etc.).

## Technical Overview

This is an Expression Evaluator, which takes strings of mathematical expressions and computes them. It looks at each character of the expression as a substring and converts them to either numbers or operators.

## Summary of Work Completed

First, I looked through the given code to figure out what does what, the classes, functions, etc. that need to be added and what they should look like and leaving some notes to help sort it out. I created the proper bodies for the methods in the Operator and Operand classes. I created a class for each operator and gave them priorities starting from -1 to 3, and operators were assigned priorities given by the outline. I stuck to the naming convention suggested to give myself less work in changing the given code. I added a RightParenthesisOperator and LeftParenthesisOperator to give them priority (i.e. -1, 0) which are not used very much at all but are there for conventional purposes. I also rearranged the Evaluator class a little bit and added the parenthesis to the delimiters and changed the initialization of the new operator variable.

# Development Environment

The IDE used to program this project is IntelliJ using JDK 12.

# How to Build/Import your Project

# How to Run your Project

To run the project, simply hit the play button.

# Assumption Made

* There are no negative inputs, only negative outputs
* Will not be tested in a way that tries to intentionally breaks the code
* Inputs and outputs are integers

# Implementation Discussion

## Class Diagram

# Project Reflection

Starting this project took a while since I haven’t coded at all in the past few months. The last time I did do code was in a different language, C++, so some of the general ideas were quick to grasp but getting back into the groove with Java instead took some time and made it a little tough. My knowledge of Java is mainly from CSC 210 and CSC 220, which only briefly covered things like parsing and hash maps, so I had to do some research on what they did, their methods and more. I also had to remind myself of how stack worked and imagining each token should be stacking. One of the road blocks was wrapping my head around the tokens being popped in backwards order of the original expression.

# Project Conclusion/Results

The Expression Evaluator works and properly can compute given expressions as long as the abide by the assumptions.