CSC 413 Project Documentation

Summer 2020

Jarvis Yu

920584779

CSC 413-02

https://github.com/csc413-02-SU2020/csc413-p1-yujarvis08

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 project is a calculator with a GUI. Its purpose is to calculate expressions by using operators such as addition, multiplication, division, subtraction, and power and calculating with the same order of operations in math.

## Technical Overview

We take in an expression as a token in a string, before converting it to integers. We cerate 2 stacks, one for the operand and one for the operator. The tokens will be stored into the stacks. The token is stored into the operator stack if the hashmap contains the item, and the token is stored into the operand stack if the check returns an integer from parseInt. When the expression is calculated, the operators and operands are popped from the top of the stack.

## Summary of Work Completed

I created classes for the operators: AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, LParenthesisOperator, and RParenthesisOperator. Overridden the functions priority and execute for the operators. I written logic for when the expression contains parentheses in the evaluator class and filled the operand constructors.

# Development Environment

Java SE Development Kit 12.0.2

IntelliJ IDEA 2020.1.1 Ultimate Edition

# How to Build/Import your Project

Make sure Java is installed

In IntelliJ, click import project

Choose the calculator folder

Keep clicking next

Click finish

Build the project

# How to Run your Project

To run the project in IntelliJ, navigate to EvaluatorUI

Right click EvaluatorUI

Click Run

# Assumption Made

HashMap for operators must be private no matter what

Parentheses are also operators

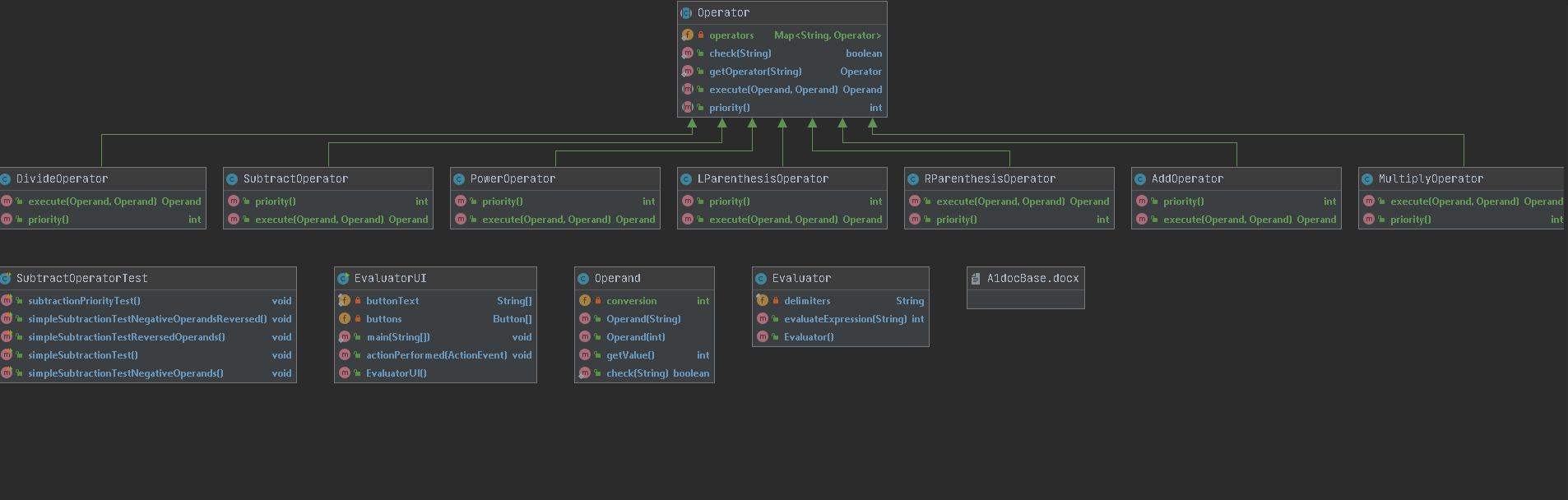
Project is complete when all Junit tests pass

Operators and operands are pushed and popped

# Implementation Discussion

The operator hashmap is private to protect it from classes outside. A hashmap was created to make the project not hard coded. Encapsulation and polymorphism are needed to protect data. The priorities of the operators is determined by how high of a number the priority method returns. The higher the number, the higher the priority.

## Class Diagram



# Project Reflection

For the first assignment, I thought this would be relatively easy. As explained in the beginning, it was an incomplete version of a calculator. I thought that I would just have to write a few lines of code and it would be done, but I was wrong. The implementation of the calculator logic was very difficult. I was stuck for a few days before getting all the Junit tests passing. I kind of gave up when I could not get the logic correct as the result was returning null for the medium and difficult expressions. I thought I was going to turn in a project that only had 9 tests passing, but I somehow was able to correct it.

# Project Conclusion/Results

Overall, this project was difficult. It was kind of a test to see if I remember java polymorphism and encapsulation from the lower division classes I taken before. It is the same kind of material, where you had different classes and had to implement a way to make the driver work. The project was difficult, but surely there will be more difficult assignments to come.