CSC 413 Project Documentation

Fall 2022

Ysidro Alfaro De Leon

916365403

CSC413

https://github.com/csc413-SFSU-Souza/csc413-p1-Yalfarodeleon

Table of Contents

1 Introduction

1.1 Project Overview

The evaluator expression project is a calculator that evaluates basic arithmetic expressions, with the appropriate priorities for the expressions " +-*/^()". This calculator project has a user interface that allows for operands to be inputted and evaluated.

1.2 Technical Overview

Create two programs that are object-oriented and implement evaluator expressions public methods to evaluate mathematical expressions and a graphical user interface that will work with the artifact from the created object. Through stacks as a foundation for the data structure, two stacks were implemented, one for the operators and the other for the operands so that each token can be processed according to the priority of the operators. When processing the stacks, the operand stack is popped twice and the operator value is popped once, the operator with the two operands is executed, and the results are pushed onto the operand stack. Once all the tokens are read, the operators are processed until the operator stack is empty.

1.3 Summary of Work Completed

- Evaluator.java
  - in the Evaluator class the parentheses delimiters were added and a HashMap was instantiated for the Operator class that contains instances of the Operators.
  - Modified the loops to make sure to check if the operator stack was empty before we pop anything from the stack and to appropriately handle all the exceptions.
  - A temporary stack for the Operand was created to empty the operand stack so that we can keep evaluating the operator stack until the operator stack is empty.
- EvaluatorDriver.java
  - Minor adjustments to the EvaluatorDriver were implemented to be able to trace and debug the code.
- EvaluatorUI.java
  - The actionPerformed method was modified to get input from the user when pressing the calculator buttons on the GUI. The algorithm is incomplete, was not able to get down the logic for when the user presses the "=" to evaluate and "CE" for when the user wants to clear expression.
- Operand.java
  - In the Operand class, the constructors were modified to appropriately utilize the tokens from the stack with the instances of the values.
  - The Boolean check method was modified using try and catch to check to see if a given token is a valid operand
- Operators folder
  - Operator.java class in the operators folder was modified by instantiating a HashMap called operators. Static initializers were added for each of the subclass operators. The Boolean check method was modified to determine if a given token is a valid operator.
  - In the operators folder, subclasses of the Operator were created for each mathematical operator; AddOperator, SubtractOperator, MultiplyOperator,

DivideOperator, PowerOperator, LeftParenthesesOperator, and RightParenthesesOperator. Each all extends the Operator class, and the appropriate priorities were given as well as the arithmetic needed to execute for each.

2 Development Environment

- Java version: JDK 17
- IDE used: IntelliJ IDEA

3 How to Build/Import your Project

Clone repo to your computer from the repo's home page. On the computer terminal type in:

```
git clone repoLinkCloned
```

and make sure to clone the repo to a folder on your computer that does not require elevated privileges.

Next, the project was created in IntelliJ IDEA using the calculator folder as the source root of the project. Make sure to have the JDK installed when importing the project.

4 How to Run your Project

To run the project, the evaluatorDriver.java class is used as the main for the analysis of the evaluator expressions. To run the user interface calculator, the EvaluatorUI.java

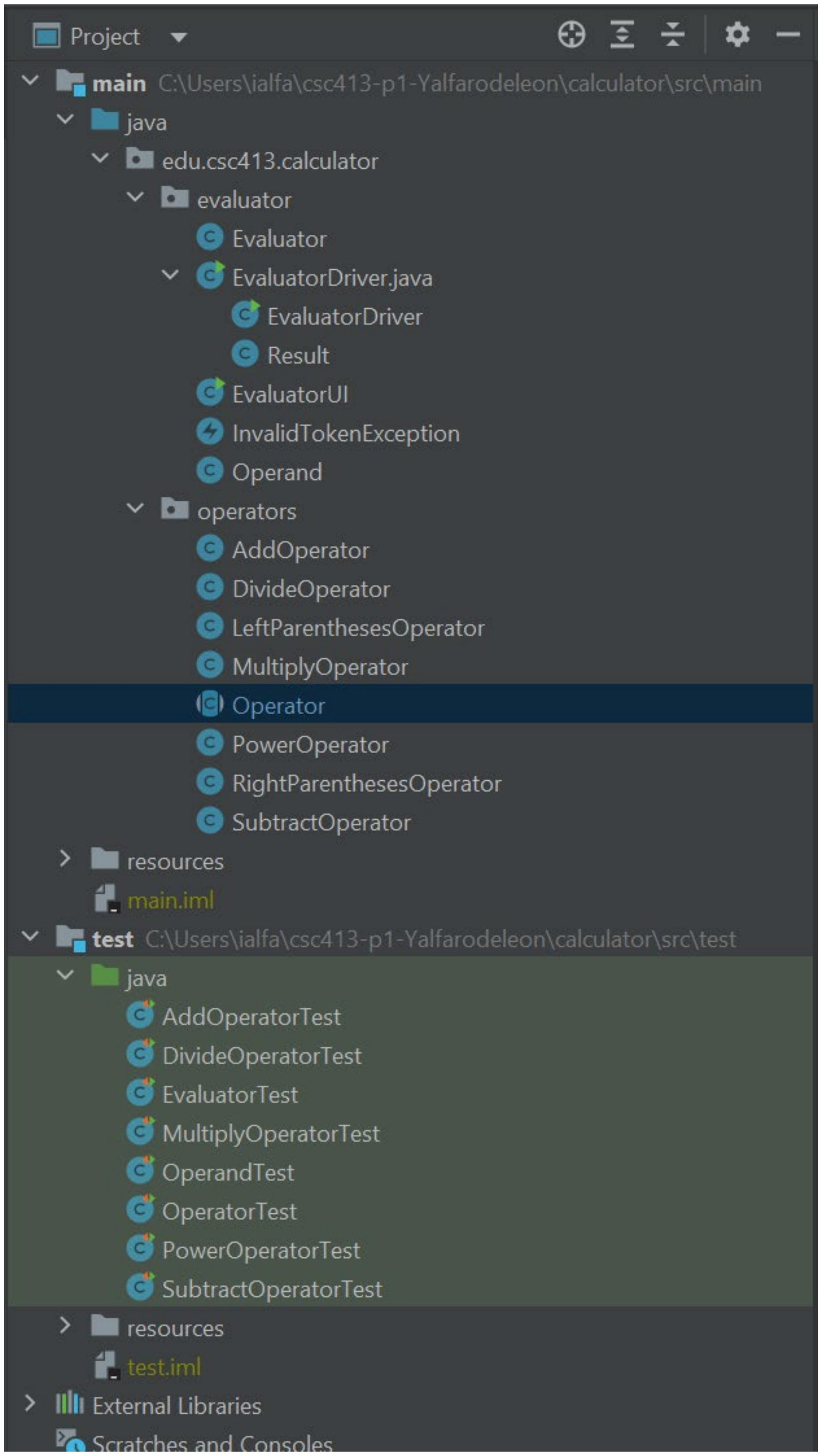
5 Assumption Made

Assumptions made for this project are that this is an object-oriented program working with basic mathematical arithmetic. Assumed that the code was filled with flaws the professor and that basic data structure knowledge was necessary for this project.

6 Implementation Discussion

The evaluator class works with the UI and the driver as the main to implement the operators to proficiently work with the stacks.

6.1 Class Diagram



```
Project
main C:\Users\ialfa\csc413-p1-Yalfarodeleon\calculator\src\main
  java
    edu.csc413.calculator
      evaluator
        Evaluator
        EvaluatorDriver.java
          EvaluatorDriver
          Result
        EvaluatorUI
        InvalidTokenException
        Operand
      operators
        AddOperator
        DivideOperator
        LeftParenthesesOperator
        MultiplyOperator
        Operator
        PowerOperator
        RightParenthesesOperator
        SubtractOperator
  resources
  main.iml
test C:\Users\ialfa\csc413-p1-Yalfarodeleon\calculator\src\test
  java
    AddOperatorTest
    DivideOperatorTest
    EvaluatorTest
    MultiplyOperatorTest
    OperandTest
    OperatorTest
    PowerOperatorTest
    SubtractOperatorTest
  resources
  test.iml
External Libraries
Scratches and Consoles
```

7 Project Reflection

After spending several hours on this project, I realized that my data structure skills need to be worked on. This project relied heavily on data structure fundamentals, and it took me some time for me to get back on track with the appropriate handling of classes, sub/super classes, and stacks. I would get empty stacks and made a lot of errors when working with while, else if, and if loops throughout the project. I struggled a lot but with struggle comes growth in my coding and for that, I am grateful for this assignment that has helped me become a better programmer.

8 Project Conclusion/Results

To conclude this project, it is important to understand that there were many moving parts to this program, many of which were initially given. Small changes can cause a large ripple in the code that with the use of debugging, the appropriate solution can be achieved. The results of my project were that the tests all passed the evaluator expressions the way it was needed. Unfortunately, I was not able to implement the user interface as it should have.