CSC 413 Project Documentation

Spring 2019

CSC0413-01

Naweeda Qeyam

920462358

https://github.com/csc413-01-SU2020/csc413-p1-Naweeda.git

Table of Contents

1 Introduction

The purpose of the project is to practice object-oriented design to create two programs, an object that evaluates mathematical expression and a GUI around the artifact from the object that evaluates arithmetic expressions.

1.1 Project Overview

This project is a good knowledge of the Object-Oriented Programming principles. Implementation of Encapsulation that hides details in a class and provides method. Next is the implementation of Polymorphism which explains the concept of same name, different behavior, and based on type. And the implementation of Inheritance which captures the common attributes and behaviors in a base class and extend it for different types.

1.2 Technical Overview

The mathematical expressions are composed of integer operands and operators from the set +, -, *, /, ^, (, and). Each of these operators have their own priorities. In the usual arithmetic expressions, the operator is written between the operands. Such operators are called binary operators. Such expressions are called infix expressions. And we are implementing Stack data structure to keep track of operator priority. We use operand stack to keep values and operator stack to keep operators.

We implement an important process for the stack operations which is pop operand stack once, pop operator stack once, pop operand stack again, compute first and second value of the operator and lastly push the value obtained in operand stack.

The algorithm part is implementing a while loop until the end of the expression. If the character is an operand, push it onto the operand stack. If the character is an operator, and the operator stack is empty then push it onto the operator stack. If the character is an operator and the operator stack is not empty, and the character's precedence is greater than the precedence of the stack top of operator stack, then push the character onto the operator stack. We are going to keep doing this until there are no more input characters, and the stack becomes empty. The values left in the operand stack is the result of the expression.

1.3 Summary of Work Completed

I basically started the project working on the Operand class which is self-contain class. I stored the operand tokens and implemented the functions which was converting string to the int, and the getter function for returning a value. Then I implemented the operator class, which is a parent class, and it contains the child classes, and I inherited the methods form the parent class inside the child classes. After I implemented the Evaluator class by the features of the stack and some logics when the stack is empty and when it is not empty. Lastly, I started working on the EvaluatorUI, which is triggering anytime a button is pressed.

2 Development Environment

- Version of Java: Java 13
- IDEA: IntelliJ

3 How to Build/Import your Project

a- From the Git Bash clone the repository from GitHub using the “git clone” along with the GitHub repository link.

b- Open IntelliJ and choose Import Project.

c- Keep clicking next, and lastly click finish.

4 How to Run your Project

- In order to run the ‘Evaluator Driver’, under the Run menu select the run Evaluator Driver.
- In order to run the ‘EvaluatorTest’ file go to the test folder, then select the EvaluatorTest file and each of the test classes contains the green arrow if you right click on them, you can see the Run option.
- In order to run the ‘EvalatorUI’ file go to the evaluator folder. Then right click on the EvaluatorUI and select run.

5 Assumption Made

The first assumption I made was the permitted operand and the operators. I assumed that the permitted Operands are like A, B, C, and the permitted Operators are like mathematical expressions. And all the values of the operands must be integers. And how a better logic would be applied to the right and left parenthesis. And I assumed that each of the operators have their own priorities and how to implement them in a method. Also, for the GUI, I assumed that there should be a better and shorter way to trigger the buttons.

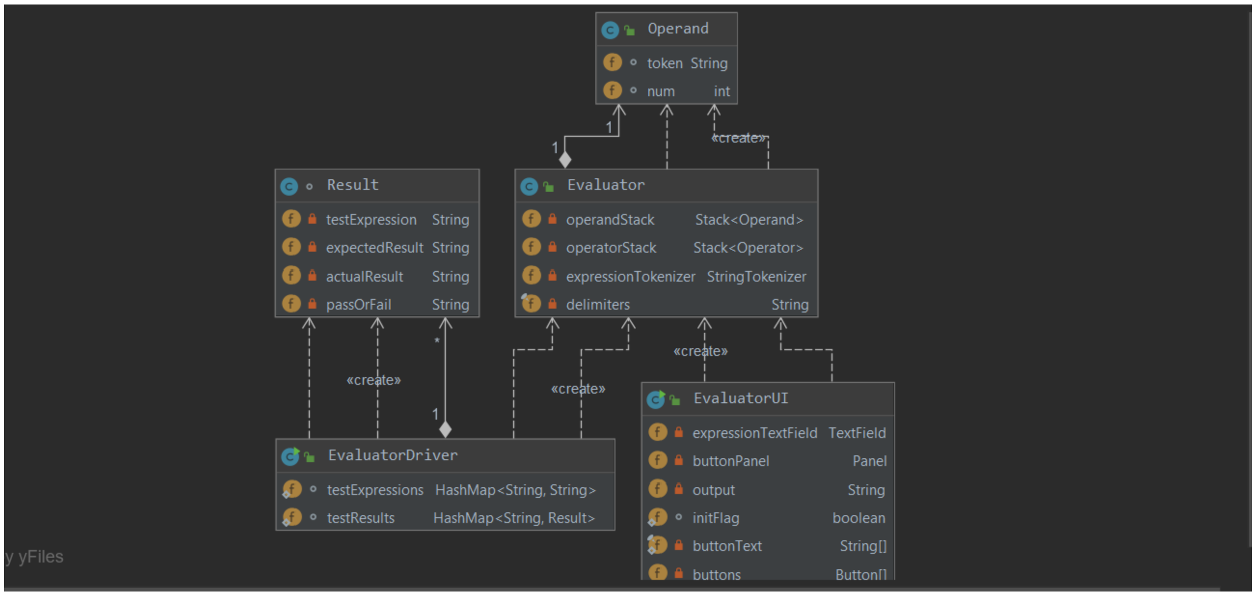
6 Implementation Discussion

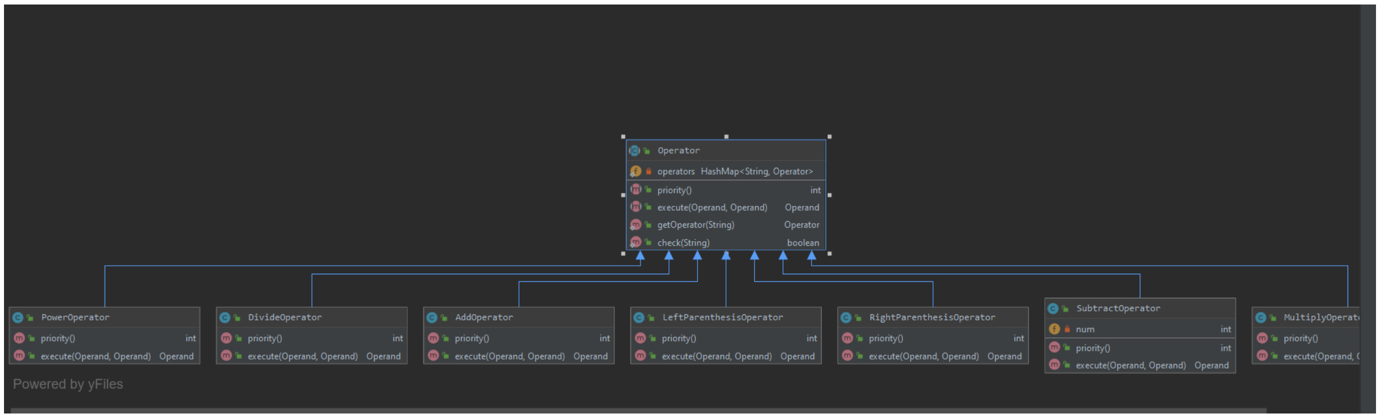
I believe the goal of this assignment is to gain a good knowledge of the Object-Oriented Programming principles. For instance, implementation of Encapsulation that hides details in a class and provides methods. According to the project some attributes of the class are modified public and some of them are private. I have made the Hash Map private so you it would be accessible outside of the class, and it shouldn’t be public, because it would break the encapsulation, and I made it static so we won’t have more than one.

And I want to discuss the implementation of Polymorphism which explains the concept of same name, different behavior, and based on type. For the project, I have implemented Polymorphism in the Operator class. Since the operator class is a parent class and it has child classes like AddOperator, SubtractOperator, MultiplyOperator, and DivideOperator. And they all inherit the methods which are implemented inside the parent class, they behave differently depending on the subclass.

Lastly, I want to discuss the implementation of the Inheritance which captures the common attributes and behaviors in a base class and extend it for different types. For the project I have implemented inheritance for all the operator child class like AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator, LeftParenthsis, and RightParenthesis that inherit from the parent class which is Operator.

6.1 Class Diagram




Operator
operators HashMap<String, Operator>
priority() int
execute(Operand, Operand) Operand
getOperator(String) Operator
check(String) boolean
PowerOperator
priority() int
execute(Operand, Operand) Operand
DivideOperator
priority() int
execute(Operand, Operand) Operand
AddOperator
priority() int
execute(Operand, Operand) Operand
LeftParenthesisOperator
priority() int
execute(Operand, Operand) Operand
RightParenthesisOperator
priority() int
execute(Operand, Operand) Operand
SubtractOperator
num int
priority() int
execute(Operand, Operand) Operand
MultiplyOperat
priority()
execute(Operand, (
Powered by yFiles

7 Project Reflection

Generally, as an intermediate level programmer, this project was super challenging. I have started this project by reading the assignment instructions couples times, and I have watched the recorded lectures on the iLearn. Also, the PDF lectures are super helpful, which helped to understand the concept of Object-Oriented Principles. I spent huge amount of time understanding Hash Map and its features, and I realized that is very amazing for implementing large amount of data. And I learned that implementation of polymorphism is super helpful, like I was able to inherit the methods from the parent class "Operator" to its child classes. It was remarkably interesting for me implementing the Operator as a parent class and the child classes, and not breaking the encapsulation. This project me think that applying logics and problem solving is one of the best techniques for being a skillful programmer.

8 Project Conclusion/Results

The goal of this assignment was to gain a good knowledge of the Object-Oriented Programming principles, like Encapsulation, Polymorphism, and Inheritance. And applying some logic to the Evaluator expression and Evaluator GUI. I have done my best not to break Encapsulation and made sure that my logics are understandable and efficient. The most challenging part was implementing the EvaluatorUI, for the function which is for triggering anytime a button is pressed on the calculator GUI. I struggled to apply a better logic for generating the button, but I did not come up with a shorter or better way. So, implemented the function with some if conditions. And I run all the tests, and all of them were passing. Lastly, I want to include that this project helps me to move to a higher challenge or level. And I am confident that I will be prepared.