CSC 413 Project Documentation

Summer 2020

Student name: YANGESH KC

Student ID:920771082

Class: CSC413 Section 02

GitHub Repository Link

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

Table of Contents

Introduction

This program is a implementation of a simple java calculator. Normally, In the expression, an operator is written between the operands. This program performs the arithmetic operation of given values of the operand A, B, C and D and operators (+, -, *, /, and ^) including left parenthesis and right parenthesis. For example: A + ((B * C) – D /A)

1.1 Project Overview

Arithmetic expressions are formed by the combination of operands and operators. Operands are usually numbers denoted by A, B, C, D and operators are usually (+, -, *, /, and ^). Each operations are classified and executed according to their respective priority.

1.2 Technical Overview

In this project, we implement the object-oriented approach in java using arrays, stacks and HashMap.

Algorithm:

1) We get the user Input form the GUI as a block of string which gets parsed and stored as tokens.
2) The tokens are compared and pushed into operator or operand stack depending upon their priority following the LIFO convection (For stack)
3) For the parenthesis, the left is set to the low priority and the right is basically ignored by setting to priority 4.
4) Right parenthesis: Pops the operator from the operator stack, pops the operand twice from the operand stack and operates and pushes the result to the operand stack.

5) An operator call newOperator: While the operator stack is not empty, and the operator stack has similar or greater precedence as newOperator it pops the operator form the operator stack and along with two operands and pushes result onto the operand stack.
6) If the operator is not empty it does the similar process and pushes the result on the operand stack.

1.3 Summary of Work Completed

The project contains several classes as shown in the chart below. All the classes build and run without any errors. The GUI also works as expected without any kind of issue. All the requirements are met.

2 Development Environment

I wrote the program in IntelliJ IDE. However, any java compactible ide should be able to build and run my program.

3 How to Build/Import your Project

For simplicity, I have created a jar folder ("calculator.ja") which will easily run my program without going the traditional way. However, If anyone wants to import my project, they can clone it from github and open in any java compactible ide and simply build and run my calculator.

4 How to Run your Project

STEP 1: Download the Entire Program

STEP 2: Open folder and look for calculator.jar

STEP 3: The calculator GUI runs. Perfom any Mathematical operations. Enjoy!!

5 Assumption Made

- Delimiters are pre-defined and can only operate

  - Addition, Subtraction, Divide, Multiply, Power and expressions with Parenthesis of any size.

  - The programs cannot execute complex mathematical expression like (Trigonometric & Scientific expression)

6 Implementation Discussion

The objective of project was for students to be able to implement three fundamental principles of Object-Oriented Programming (OOP) namely:

1. Encapsulation

• It refers to an OOP concept that binds together the data and functions that manipulate the data, and that keeps both safe from outside interference and misuse. A class can enforce the desired level of restriction to both the data and functions using access modifier such as private, public or protected keywords.

• Implementation: In the expression evaluator class I made sure that the attributes of the classes are properly encapsulated by using a private access modifier. Also, some of the methods such as our HashMap was made private.

2. Inheritance and Composition

• One class may share attributes with other classes either in an “is-a-type-of” relationships known as inheritance or “has-a” relationship known as composition. In such a scenario, we can create a parent/superclass, and the child class can inherit from the parent class.

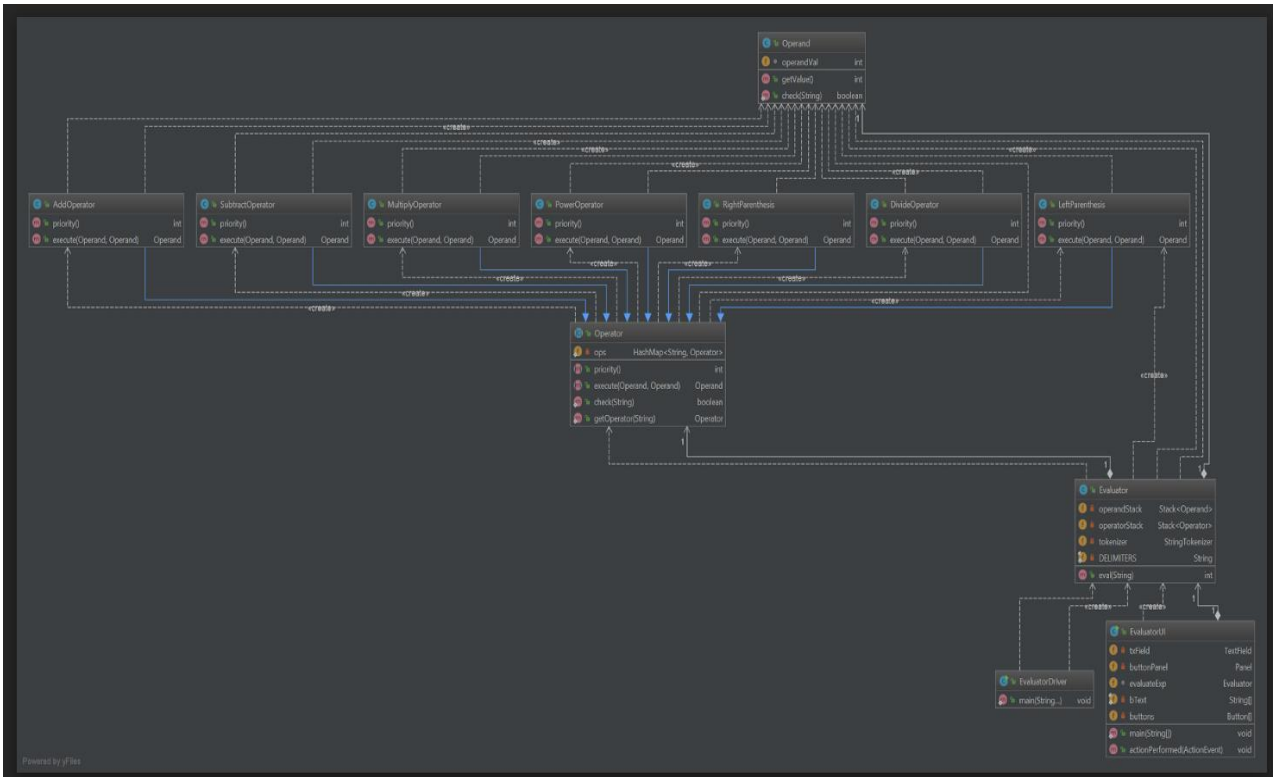
• Implementation: All the operator subclasses/child class such as AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator, LeftParen, and RightParen inherit from the parent Operator class.

3. Polymorphism

• Polymorphism makes OOP programming dynamic in the sense that same operation name may behave differently on different classes. • For example, all animals move. But all movements are not the same. An dog has four legs to move. A bird flaps its wings to fly and also has two legs to walk. And dolphins have fins and tail which they move to help them propel in the water.

• Implementation: Operator class is a parent class and it has priority() and execute() methods as bstract. Each subclass that inherits from the parent class implements those methods, and each behaves differently depending on the subclass. For example: in AddOperator subclass, method execute() adds operand one and operand two with an operator whereas in the SubtractOperator subclass method execute() subtracts operand two from operand one.

6.1 Class Diagram



7 Project Reflection:

This project helps me refresh my java skills. This was my first time building a GUI application which made the entire project realistic and fun experience. Moreover, this project helped me grasp the concept of object-oriented programming in java. The program has a implemented several useful concept-like stacks, hash-maps, arrays; concepts like polymorphism, encapsulation, inheritance which can be implemented on any future projects. It has also developed my problem-solving skills. At the beginning I did not fully understand the importance of clean coding any how simple things like can have a huge impact in our program. During the office hours, I was able to debug my program and solve issues with parenthesis.

8 Project Conclusion/Results

This project helped me dive deeper into implementing the object-oriented principle by creating evaluation calculator and a GUI implementation. Even though, it took me a while to grasp some concepts, the joy of understanding the concept is enormous. My program is fully functional but it still is unable to process the complex mathematical operations like scientific expressions and time efficiency. There is still a big room for improvement which will be my point of focus for the upcoming days.