JOHN SABOUR

917832180

CSC413, SUMMER 2019

https://github.com/csc413-02-summer2019/csc413-p1-jpsabour

Project Overview:

The expression evaluator evaluates an infix expression. It utilizes operators in order to perform operations between values given by the user. The program utilizes the +,-,*,/, and (,) operators to evaluate expressions.

Technical Overview:

The expression evaluator was intended to be a demonstration of inheritance and encapsulation. This is demonstrated through private access and inherited classes in the operators.

The project also includes a stack, necessary to make sure the correct operator is being used. All of these are encapsulated to ensure that there is no spill over. The stack can be used to determine if there is an operator or parenthesis being pushed onto the stack, or if there is a different result for the infix expression.

The Evaluator class is responsible for determining the status of the stack and determining if there needs to be a pop or if the stack is empty already.

Summary of Work Completed:

The project is able to be built and run seamlessly. The GUI and keys all work as intended. Negative numbers do not cause any issues.

The project is able to correctly evaluate expressions involving all the operators, however, has issues evaluating complex expressions as shown in 8 of the test cases. I presume it may be an issue with how the parenthesis function.

Development Environment:

a.Java 1.8 was used to develop this project.
b.Intellij Idea was used to develop this project.

How to import/build the project in Intellij:

To import the project in Intellij, go to the GitHub link where the repository is located. Either copy the link to import, or download the zip file containing the project. When in Intellij, use the import functionality to open the project in the IDE. At this point, you can use the build drop down menu to build the project.

How to run the project:

Once you have imported and built the project in Intellij, you can simply use the dropdown menu under “run” and use the run functionality built into Intellij.

Assumptions Made when designing and implementing my project:

Assumptions were made involving priority levels of parenthesis, multiplication and division operators, addition and subtraction operators, exponents, and that the values of the operands are all going to be given as integers.

Implementation Discussion:

The expression evaluator greatly utilized the idea of encapsulation. The attributes of classes and some methods were properly encapsulated with private access.

The idea of Inheritance was also implemented. All the operator classes inherit the parent “Operator” class. This allows them to have a parent-child relationship with the operator class.

Project Reflection:

When designing this project, I took the very slow and simple process of writing out each and every one of the op classes first, then looking to complete the program as a whole. It was a lot easier to have completed and working operator classes waiting to be used, rather than attempting to do them last.

Project Conclusion and Results:

This project gave me a deep dive into the inheritance and encapsulation principles in a way that I hadn’t yet encountered. By splitting the project into parts and organizing into roles, I was able to develop in a way that would allow me to see the results as I code. This was extremely important when utilizing the operator classes and constantly running the GUI in order to see if they were working properly! I feel that after this project I will have a much better understanding of these root OOP principles!