Steve Rentschler
Student ID # 916188083
CSC 413 Fall 2018
Link: https://github.com/csc413-01-fa18/csc413-p1-turbosmr

2. Introduction:

a) Project Overview

For Project 1, we are to complete an Expression Evaluator Calculator with a GUI (graphical user interface) in Java. There is some code that is already in place and we are to complete the code to get the Expression Evaluator Calculator working correctly with a GUI. For the Expression Evaluator Calculator to work correctly, we need to be able to evaluate mathematical expressions -- entered into the calculator GUI -- with the proper order of importance. For example, we will need to be able to evaluate the expression: (4 + 4) * 8 = 64. We will need to incorporate classes and packages to work together for the code to work properly.

b) Technical Overview

The Expression Evaluator Calculator is made up of two packages -- the Evaluator package and the Operator package. The Evaluator package has the Evaluator, Evaluator IU, and Operand classes. The Operator package has the Operator, AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator, LeftParenthesis, and RightParenthesis. The Evaluator class imports the Operator package to implement methods stored in AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator, LeftParenthesis, and RightOperator, stored in the Hashmap in the Operator class. The Evaluator class also uses stacks to push and pop operands and operators in a stack to calculate numbers by calling the functions in the Operator classes for the calculations. The Evaluator IU implements a GUI to display a calculator using the algorithm in the Evaluator class to display and calculate numbers entered into the GUI.

c) Summary of work completed

The worked that I completed in this project was completing the algorithm needed for the Evaluator class to implement the stacks correctly. Making a Hashmap of all of the Operators used for the calculations in the Operator class. I created instances of the Operators in the Hashmap where I extended the Abstract class -- Operators -- to implement the given methods; public abstract int priority() and public abstract Operand execute(Operand op1, Operand op2). I also filled out the code for the method public static Operator getOperator(String token) and public static boolean check(String token) in the same class. For the Operator classes -- AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator -- I wrote code for the methods public abstract int priority() and public abstract Operand execute(Operand op1, Operand op2). In the Operand class, I completed code for the 2 constructors public Operand(String token) and public Operand(int value) and the methods public int getValue() and public static boolean().

3. Development Environment:

a) Version of Java Used

The Java version used was 1.8.0_152.

b) IDE Used

The IDE used was IntelliJ.

4. How to build or Import your project

From my GitHub page, download the project folder by pressing on the green “clone or download” button and select download zip. After unzipping the project folder on your computer, open up IntelliJ and click on the file tab. In the file tab go to “new” then to “project from existing sources” and click on it. Then select the project folder and select open. Select the gradle option in the next window, then in the next window select “using explicit module groups”, “create separate module per source set” and “use gradle wrapper task configuration” then press “finish”.

5. How to run your project

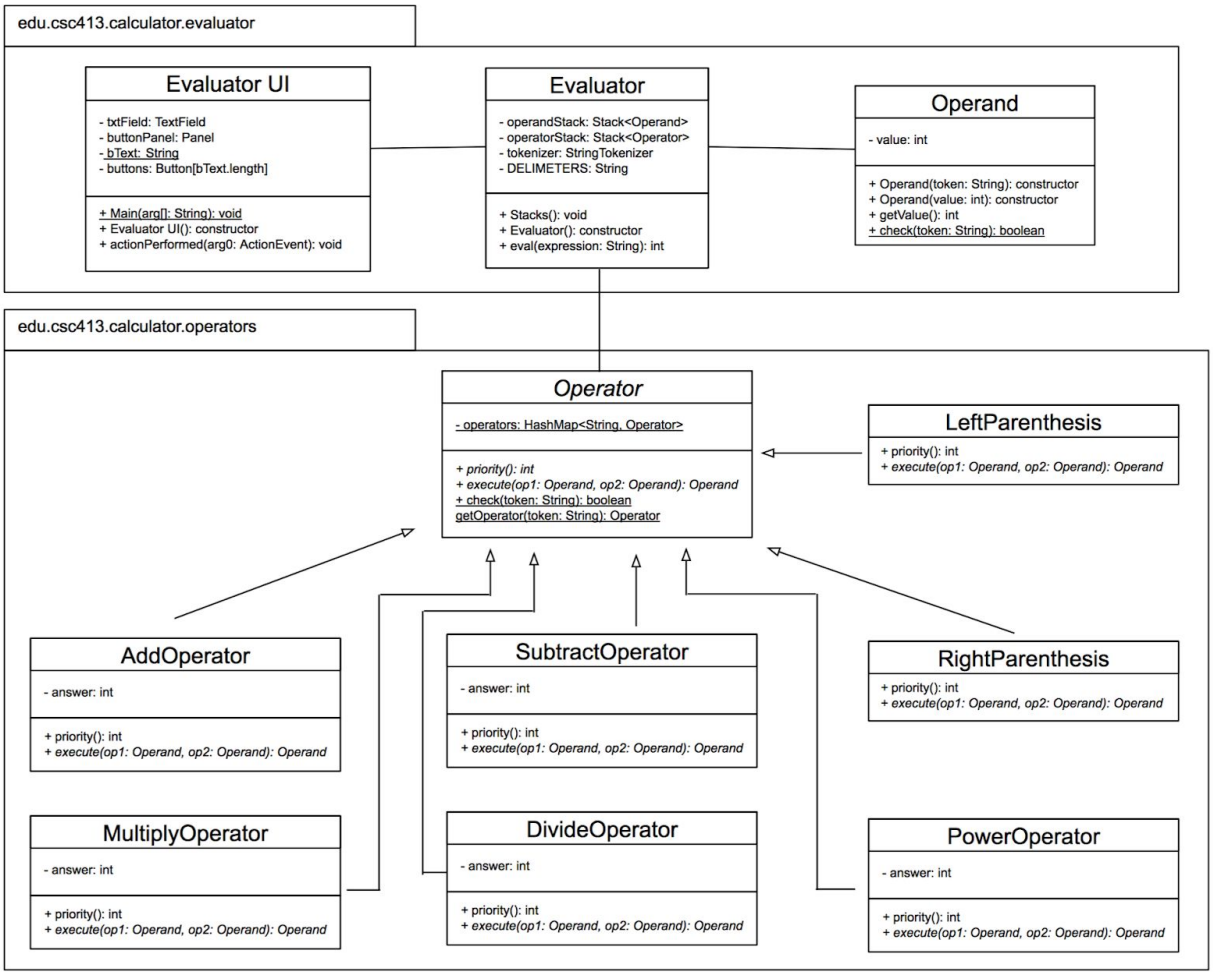
After the project has been imported into IntelliJ, click on the run tab at the top of IntelliJ, then click the “Run EvaluatorUI”.

6. Assumptions Made when designing and implementing this project:

My assumptions when first started designing this project was that it was going to be complicated to fully grasp what was needed to complete the assignment. I didn’t know how Hashmaps worked and I haven’t used Stacks in a while. I knew it would be a challenge to get everything running correctly without a good knowledge of Stacks and Hashmaps. I felt pretty good about my knowledge of how Java works with classes, packages and abstract classes. So I thought I would be able to get everything to work together eventually.

7. Implementation Discussion

Here is a UML diagram to show how the packages and classes work together.



8. Project reflection

After completing this project, I found that my understanding of stacks greatly hindered me from completing it in a timely manner. I needed help to fully understand how the stacks worked to complete the algorithm in the Evaluator class. I started out coding this project in Netbeans because I couldn't get it to run in IntelliJ to begin with. That helped me to get it started, but I should have tried to get it loaded into IntelliJ sooner so that I would have had more time to figure

everything out. Looking back, I'm glad I started coding in Netbeans to get me started. If I would have waited till I got IntelliJ to run correctly, I wouldn't have completed the algorithm for the Evaluator class. For the majority of the coding I did, it went pretty quick, there were just a few classes that took a long time to get to work properly.

9. Project Conclusion and Results

The project was fully completed and should run correctly without any errors.