Author: Hashim Jacobs
Package: Evaluator
Github: github.com/hashjaco

Introduction:

Description:

The Evaluator Program and its GUI displays a calculator that evaluates an expression input by the user. The user clicks the buttons corresponding to the numbers and operators they desire. Upon clicking the equality operator, the solution is displayed in the text field. The program and it's UI were implemented in Java and compiled within NetBeans.

To compile and run as a jar file in terminal:

Write: java -jar EvaluatorUI.jar.

Scope:

1) Program utility classes Operand and Operator (COMPLETE). Also, program extensions of Operator class (COMPLETE).
2) Complete Evaluator class and test successfully (COMPLETE).
*The Evaluator implements a single public method, int eval(String token), that takes a single String parameter that represents an infix mathematical expression, parses and evaluates the expression, and returns the integer result.
3) Complete the programming for the Evaluator UI by utilizing Evaluator class (COMPLETE).

Assumptions:

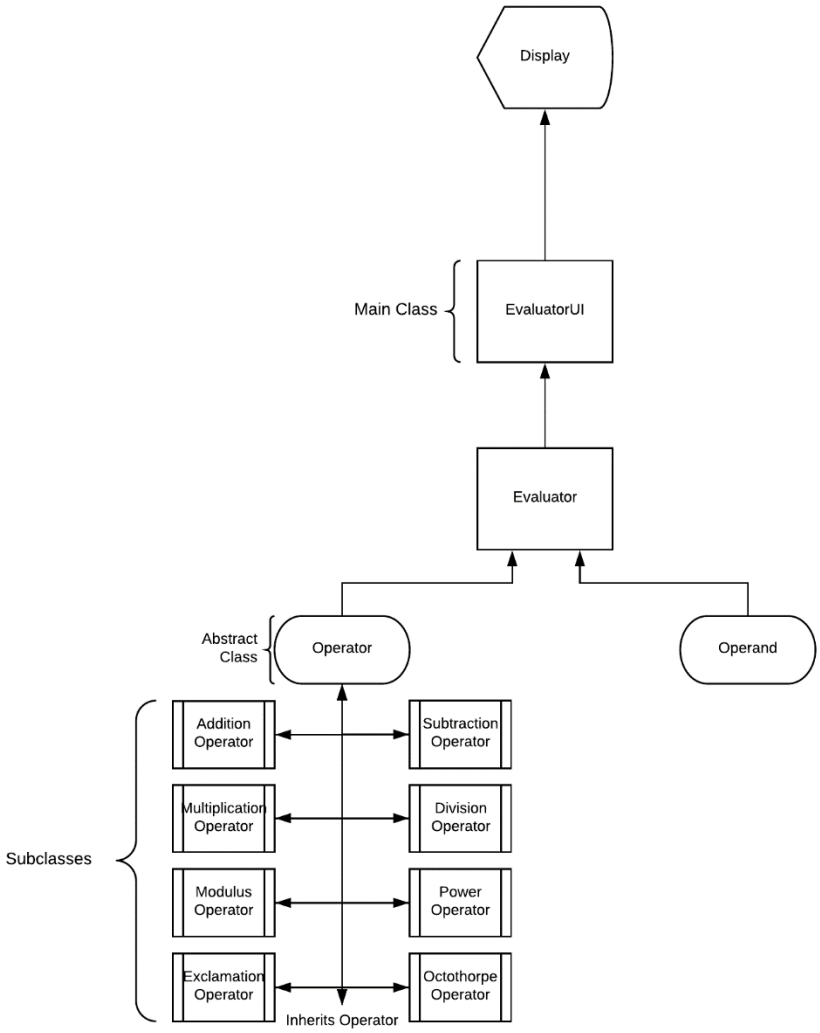
We could assume that all variables/operands were input as integers and also that parenthesis were balanced.

Implementation:

To utilize all of the different operators efficiently, it was necessary to implement subclasses of the operator class, which inherit the superclass, and create an instance of a Hashmap containing all of the subclasses within the abstract Operator class. Being able to call an operator subclass by using its key as an argument to perform the desired operation made life a bit easier. The only issue I had was getting the parentheses to work because for a while, my program was not reading them as valid operators even though I had created subclasses for them in order to push them onto the Operator stack. Because of that issue, the program would fail as it returned the error for the invalid token.

All of the operator subclasses contain a different implementation of the abstract methods, priority() and execute(), to perform their respective operation on operands.

I was a bit torn while deciding how to implement my eval method to take care of parentheses. There was the obvious if/else push/pop until...but I figured there was a way to do it recursively. However, to be honest, I was unsure of how much more efficient or inefficient it would have been.


Display
Main Class
EvaluatorUI
Evaluator
Abstract Class
Operator
Operand
Addition Operator
Subtraction Operator
Multiplication Operator
Division Operator
Modulus Operator
Power Operator
Exclamation Operator
Octothorpe Operator
Subclasses
Inherits Operator