Calculator Project Documentation

(HW 1)

Spring 2019

Stephanie Sechrist

918679078

413.02

https://github.com/csc413-02-spring2019/csc413-p1-stephlsechrist

Table of Contents

1 Introduction

1.1 Project Overview

This project was designed to help us practice object-oriented design. We created an object (Evaluator) that calculates expressions. In addition, we were asked to create a GUI to interact with this object. Some code was filled out for us, and given the class hierarchy, we had to fill in the rest and pass all given tests. The calculator, in addition evaluating according to parentheses, processes the following operations: addition, subtraction, multiplication, division, and power.

Technical Overview

1.2 Summary of Work Completed

The only file that has not been edited is EvaluatorDriver. Classes that were added implement each operator (7 child classes of Operator). These are: AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator, OpenParenOperator, and CloseParenOperator. Each of these subclasses implements the priority() and execute(Operand op1, Operand op2) methods from the abstract methods in Operator.

Code was also added to Evaluator to: add " ", "(", and ")" to the delimiter list; take into account open and close parentheses; and continue executing the expression (emptying the stack and processing operators and operands) even after the tokenizer has reached the end of the expression string.

The Operand class was filled out, implementing two constructors: one from a string token, and one from an int. Also implemented in this class is getValue(), where we get the value of the operand, and check(String token), where we check to see if the token is a valid operand.

The Operator class was also filled out. Here, I create an instance of a HashMap and initialize it with a static block of each operator subclass. I also implemented getOperator() and check(String token), which makes sure the token is a valid operator.

Lastly, in EvaluatorUI, I implemented the actionPerformed(ActionEvent arg0) method with a series of if-else statements to allow the user to use his or her mouse to enter an expression and evaluate it.

2 Development Environment

For this project, I worked solely in IntelliJ IDEA 2018.3.4 (Ultimate Edition) on Windows 10. The version of Java being used is 11.0.2.

3 How to Build/Import your Project

1. Use given git repository link to clone the project into desired folder.
2. Open IntelliJ IDEA and select Import Project in the Welcome window.
3. Find the folder that you cloned the git repository to and open the project folder (should be named csc-p1-stephlsechrist).
4. Highlight the calculator folder to be the root of the source files. Click OK.
5. Select "Create project from existing sources" and click Next.
6. Accept the defaults on the next page by clicking Next (should be naming your project calculator).
7. Accept the defaults on the next page by clicking Next.

8. Accept the defaults on the next page by clicking Next. “resources” should be selected under “Libraries.”
9. Select both “main” and “test” modules, if not already done for you. Click next.
10. Select project SDK. You should be using JDK 11. In Name field, type 11. If not already found for you, find your JDK home path. Click Next.
11. Next page should say “No frameworks detected.” Click Finish.

4 How to Run your Project

To run the project, one may either use EvaluatorDriver.java or EvaluatorUI.java.

In the Project window on the left, open folders until “evaluator” folder is found. To use the driver, right click EvaluatorDriver.java and click “Run ‘EvaluatorDriver.main()’”. Enter an expression and hit enter key to evaluate it. Manually stop the process by clicking the stop button to left of dialog box.

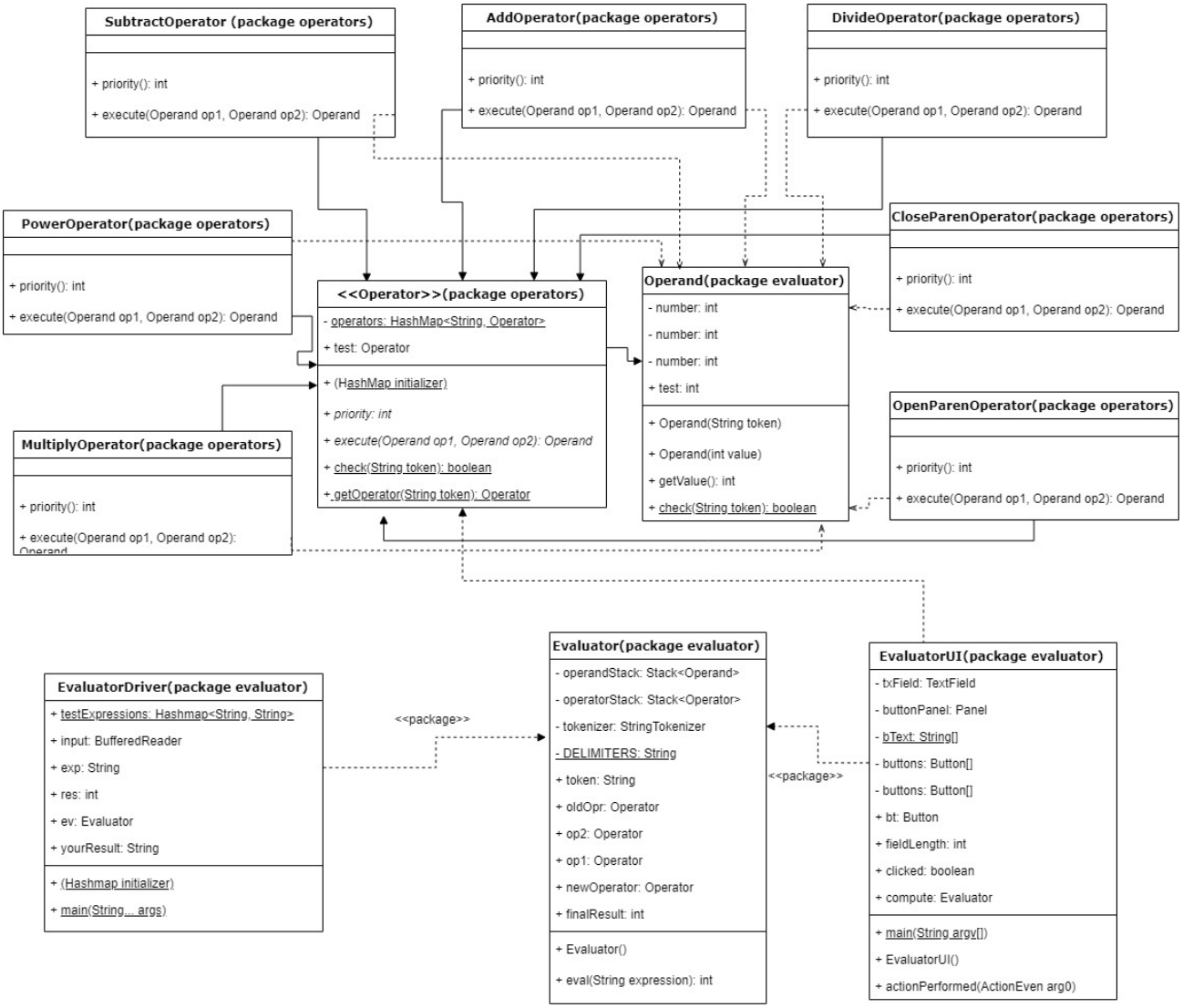
To us GUI, right click on EvaluatorUI.java in Projects window and run it. Use your mouse to click on buttons to type an expression. Exit the window to close the program.

5 Assumption Made

In this project, we assume that the user is going to use the GUI to interact with the program, so we do not have to worry about invalid operators, such as], [, %, !, x, etc. Our calculator is a very basic calculator that only uses +, - , /, *, (, and). I also assumed that I could edit any file provided in order to achieve the expected results. An overview of edits and additions made are in section 1.3.

6 Implementation Discussion

6.1 Class Diagram



Disclaimer about UML: I have not created a UML diagram for a couple years and definitely not with these many files. It is possible I have too many details, or not enough!

7 Project Reflection

This was a great first project for CSC413 in my opinion. In past CS classes, we always had to work in teams. This time, I got to implement the program the way I wanted to 100% of the time. I still had to get some help from my peers, such as in the GUI, but not as much as I thought I would need to. Honestly, I had a lot of review to do, because there was a lot of moving parts. It took me a while to get started (I was a little afraid I would not be able to do it), but once I sat down and wrote down the basic structure and made sure I understood the algorithm to evaluate expressions, it was pretty fun. Also, passing all those tests in the end was a great feeling!

8 Project Conclusion/Results

I am happy with the results of the project. There are definitely places I wish I had more time to find a better algorithm to improve efficiency. For example, another student on slack mentioned using one class for both open and close parentheses; I would like to implement that or a recursive solution if I had an extra day. Furthermore, I would fix the StringIndexOutOfBoundsException in the GUI for when CE is used to clear the field completely. Additionally, I would like to add to the UI, allowing the user to use the keyboard for input. I am also curious to see how easy it would be to add other operations, such as square root and factorial. The HashMap was presumably used to be able to add operations with relative ease, so I think it would be feasible. This project is something that can be upgraded continuously, adding more and more features.