CSC 413 Project Documentation

Fall 2018

Jinghan Cao

918818659

01

https://github.com/csc413-01-fa18/csc413-p1-KamyC

Table of Contents

1 Introduction

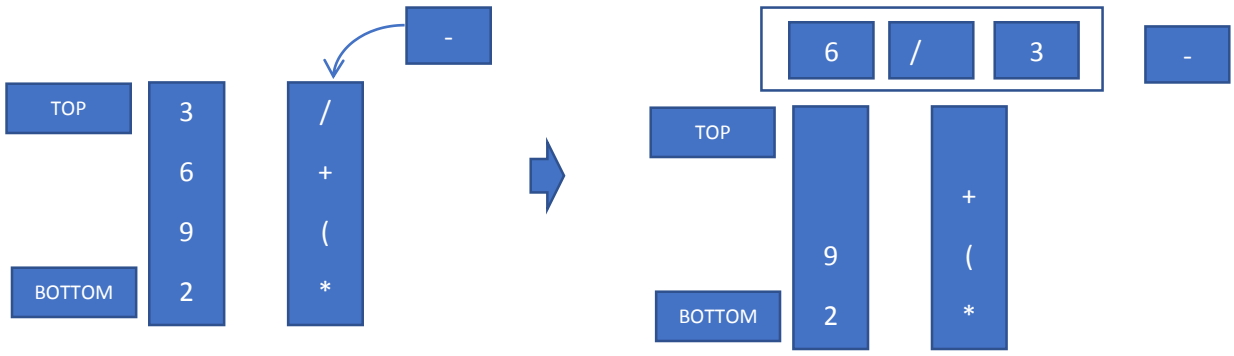
1.1 Project Overview

This project mainly requires to produce a program which is capable to do some mathematical problems. For instance, you type in 1+1 in this program and it will give you the result 2. But this program should do more than the simple calculations. Firstly it should recognize the input numbers and the operators such as addition, subtraction, multiplication, division, exponentiation and parentheses. Then these expressions will be processed in this program, which follows the basic calculation rules. In the end, you will get a right answer for your input. If you type in something that cannot be recognized by the program, you will get some feedback telling you that the input is invalid.

1.2 Technical Overview

From the technical perspective, this program is created by Java. Two packages are created to reduce the coupling of different classes. The first is an operator package which contains an abstract operator class and other subclasses for different operators. In this abstract class, a hashmap object is applied to store the String token as the key and other objects from subclasses as the value. Among the subclasses, each one could return their priority and their method to deal with certain calculation such as addition, subtraction, division etc.

Evaluator class is the major part to process the input expressions. In this class, there are two stacks among which one is for operator and the other is for operands and one eval function to call the certain operators. The logic behind the algorithm is to separate the input expression and categorize them into operator and operand stacks. The rule is that if the input operator's priority is high than the operator stack's top element which is of course also an operator, then this operator is push. If the input operator's priority is less or equal to this stack's top one, then we should use the top operator to deal with top two operands popped from the operand stack. The result will be pushed back to the operand stack and then keep comparing the priority of the input operator and the top operator from operator stack until the top one's priority is higher. But among this process, there should be another consideration regarding the left and right parenthesis. The philosophy should be that the left parenthesis has the lowest priority and the right one's higher than the left one or just give no priority if we add some "if" conditions for the ")". While scanning the expression, make sure all calculations are done within the parenthesis.



Java event listeners are applied to respond to the GUI part. Every time user inputs certain valid token, the button will listen to this event and respond to it by displaying the token in the calculator screen. Particularly, the button C means to delete one displayed token and button CE means to delete all the displayed tokens in the text field.

1.3 Summary of Work Completed

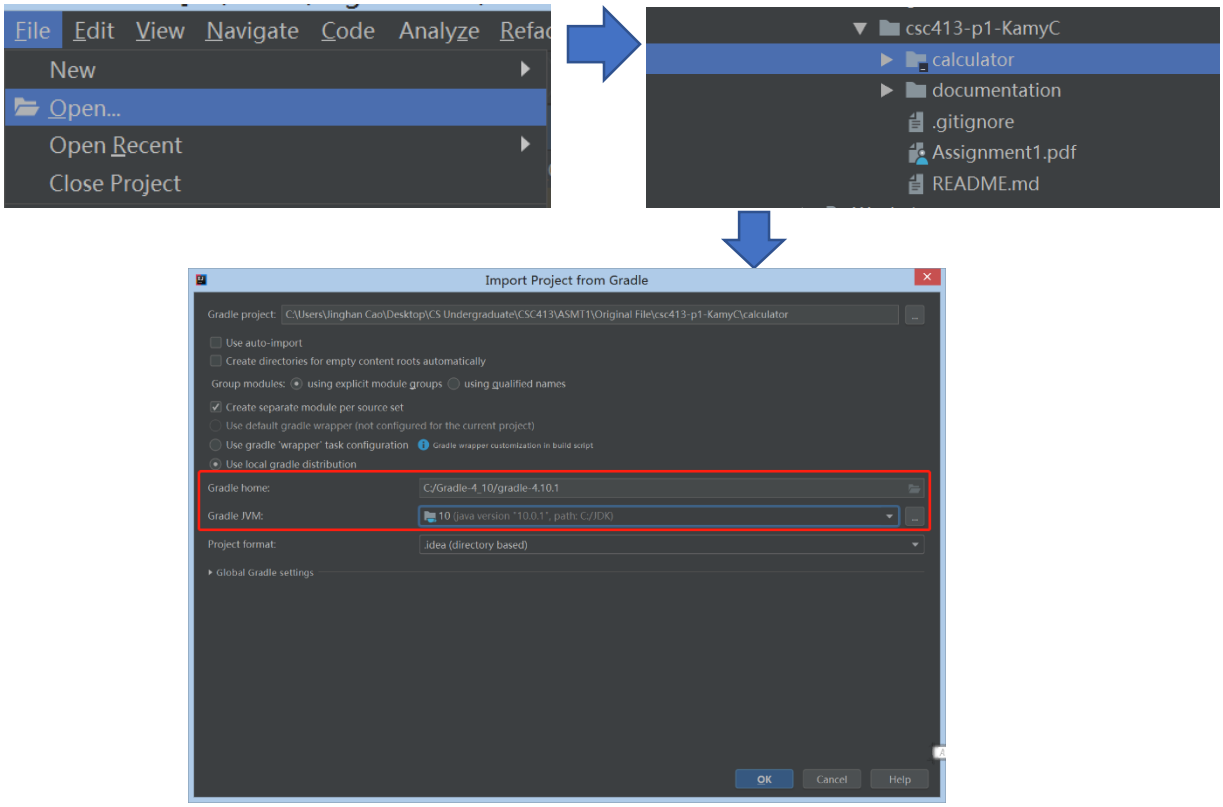
I have completed the required class including evaluator, operator, operand and GUI. There is an inter-connection between each class. The Operator class needs the Operand class to return priority and to execute the certain calculation. And the evaluator depends on the operator to process the input tokens. Therefore I understand that creating a program is like to establish a building with multiple rooms. Each room undertakes certain function and we should carefully combine them together to make sure this building could accommodate people's needs.

2 Development Environment

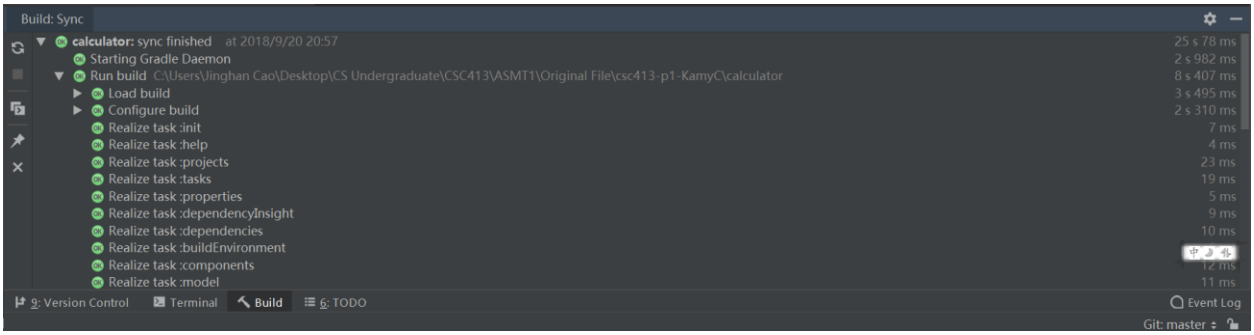
The development environment is JDK 10.0.1 and Gradle 4.10.1 to build the project and then run all the tests. The IDE is intelliJ IDEA community edition 2018.

3 How to Build/Import your Project

I firstly use gitbash to clone the whole repository from the github. The command is "git clone filelink". Then open IntelliJ IDEA and go to the File->Open. Go to the local repository and select calculator as the root file.

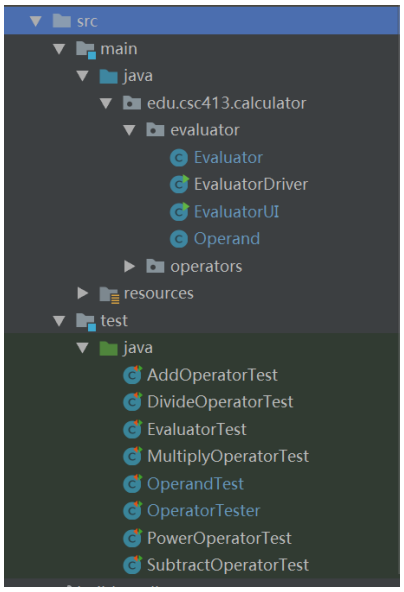


Remember to set the Gradle home directory to the local Gradle filefolder which make sure that you have already download one in your own pc. The Gradle JVM will be the JDK version. In this case I choose 10.0.1 version for the building environment. After clicking OK, there is a syncing process and you just need to wait until it's been done

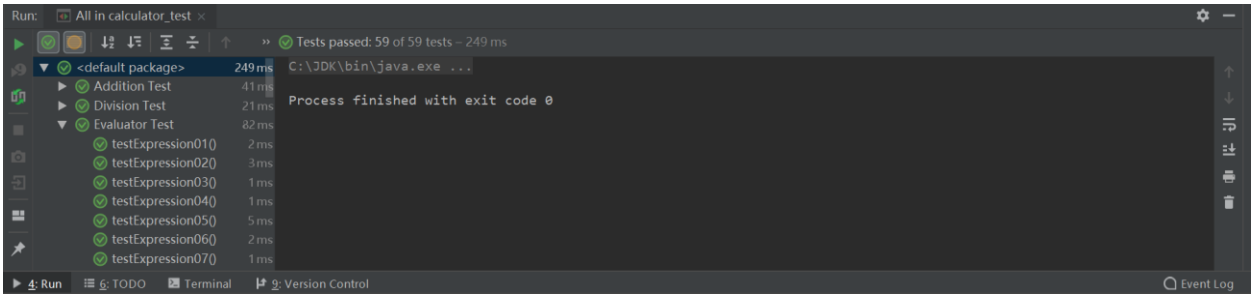


Now you can start to add codes or fix the bugs in this project.

4 How to Run your Project

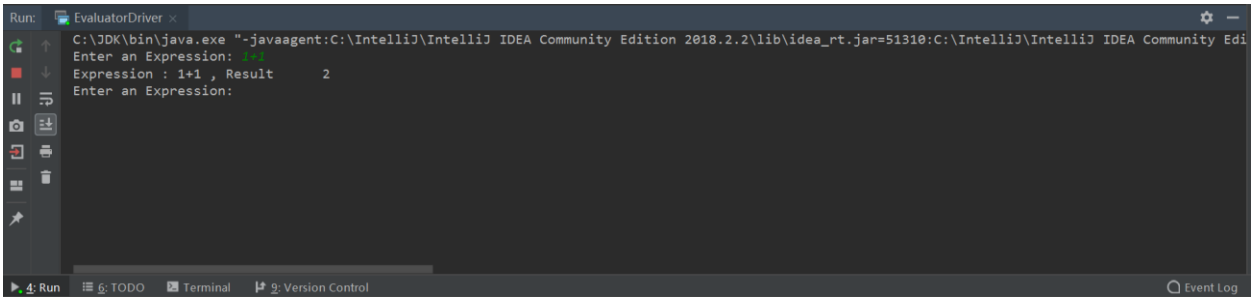


If you hope to test your code in a lazy way, you can just right click the java folder and click “Run All Tests”. If your code is bug-free and your result will be like this:

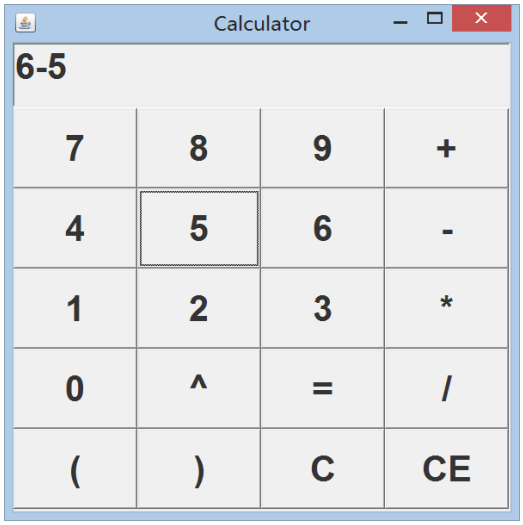


If you get some errors, it means that your program fails in some way and you can just go to the certain test and find out which part makes your program fail.

The second way to run the program is to right click the EvaluatorDriver and run the main method in this class. In the consoler you can just type in your own expression and test your project:



The third way to run this project is to use the same way to EvaluatorUI class. If you successfully run this class, you will have a user interface program and just type into the expression and you will get the calculation result after clicking "=".

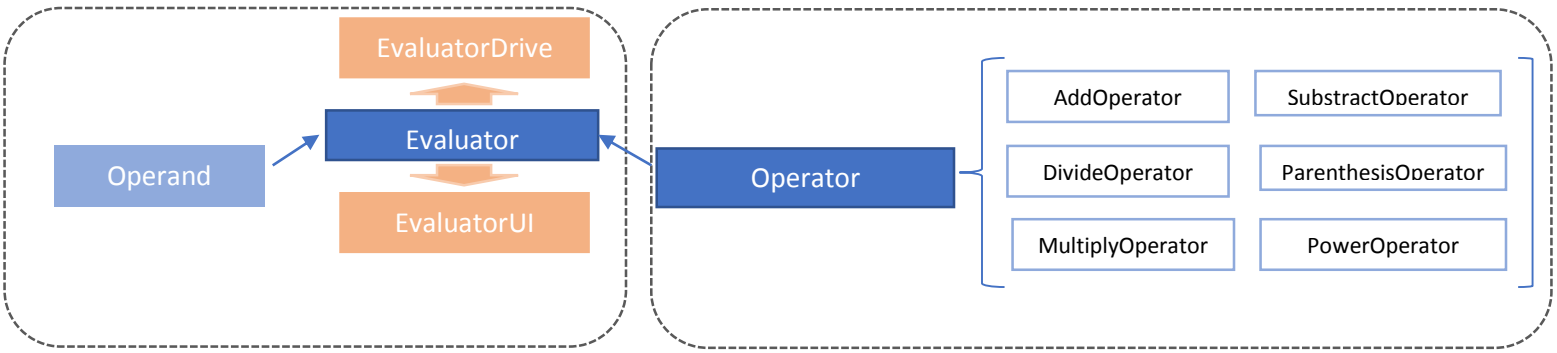


5 Assumption Made

I made a few assumptions before going into the details. I assume that I need to complete all the operator subclasses, because in the evaluator class there is a bug reminder about the initialization of operator class. Since the class is abstract, I need to add a method (getOperator) in operator class. When any operator object is called in evaluator class, the method can be used to get certain operator object. I also assume that the element that will be pushed into stack is the operator or operand instances. They are not String or other basic data type, they are operator and operand objects. In terms of the calculation part, I think there will be some tests regarding the parenthesis. One pair of parenthesis in another pair of parenthesis test may be used to test the evaluator. Moreover, there should be a reasonable priority hierarchy given to the operator tokens to fit into my algorithms

6 Implementation Discussion

6.1 Class Diagram



As the contents in the technical review, there are two packages including evaluator and operator. In operator package, each operator has its priority and a method to deal with two operands. The evaluator class will depend on operator and operand class. In evaluator class, two stacks are ready to accept operator and operand objects and these will fit into the algorithms processing the input tokens. Moreover, the EvaluatorDrive and EvaluatorUI are created so that users could type into any valid expression for tests and also for the user interface.

7 Project Reflection

For this project, I should not spend too much time on the algorithms regarding the detail technical problems. Even though this is the key to passing all the tests, I still need to think in a bigger picture. The time for this part should be balanced, otherwise there will be little time for other parts. Furthermore, the relationship between different classes made me confused at the glance. The reason is that I did not have any experience for a Java project. It took me a few hours to understand how these classes should work together to make a program run. The other reason is that I did not know much about Java language. The lack of experience in a development project resulted in my little understanding about its useful APIs. But I have learned a lot from this project. I now get the point regarding how each class should be created and the useful purpose of package. I also realize that Gradle is quite an efficient tool that we can build our own test files and make it run all the tests automatically.

8 Project Conclusion/Results

This program passes all the given tests and the GUI part is capable to receive the input expression and return a correct result. Users can also type into the expression on their own to test this program. But there is one shortage about the GUI part. Another screen could be created so that the user could know past input. In this case the “C” button can be used to clear the current input and “CE” will be used to clear all the past input. In overall conclusion, this is a good practice for my algorithm skills and it also reminds me to think in a bigger perspective. The technical details are important because they are the indispensable part for a whole project, but an overall control of the project is also vital especially when I lead a team to finish a big project.