CSC 413 Project Documentation

Summer 2023

Ruxue Jin

923092817

Class.Section:01

GitHub Repository Link:

csc413-SFSU-Souza/csc413-p1-RuxueJ: csc413-p1-RuxueJ created by GitHub Classroom

Table of Contents

1 Introduction

1.1 Project Overview

This is a digital calculator. You can click buttons to input your math expression and click "=" to get a result. The buttons are 10 digits (0~9), 7 simple operators (+-*/^ ()) and retrieve button "E", "CE", and one "=" button. Button "C" clears the entire expression. Button "CE" clears the last entry number. For example, if you want to click "10+20", but you mistyped "10+10" instead, you can click "CE", and it will remove the last number you click, display "10+", and you can continue to click your operand. If expression includes invalid token, the "invalid token" will be displayed. And you can click "C" to continue.

1.2 Technical Overview

When calculate the math expression, we divide each entry to two classes, either is an operator (+-*/()) or an operand(0~9). For the operator class, we create an abstract class "Operator", since all operators share similar "priority" property and "execute" method. Also, the abstract class "Operator" include a static initializer – HashMap to store all the operators in memory, with key String "+" and value an object of AddOperator(). This can create one AddOperator object, which using it multiple times.

To demonstrate a GUI for calculator, we create a "EvaluatorUI" which extents JFram, and impletes ActionListener. The math expression input and output show at the top of the calculator. We have 20 buttons in the center of the calculator layout; each can listen for mouse input. The "C" clears the entire expression and start over. The "CE" clear the last number input and enable to continue to input numbers.

1.3 Summary of Work Completed

I implemented Operand class to convert user input to Operand object. I create a private int variable in the class. I finished the check method by try catch the result of Integer.parseInt(token).

I created seven subclasses of Operator class to handle each operator. They are AddOperator, SubtractOperator, MultiplyOperator, DivideOperator, PowerOperator, LefrParenthsisOperator, RightParenthsisOperator. I set up priority and execute method to make them work fluently.

I finished evaluateExpression method in evaluator class to accomplish the calculator algorithm. I break the entire math expression into several tokens with delimiters, check whether each token is an operand or operator. If it is an operand, I push it into operand stack. If it is an operator and the priority is higher than the top operator, I push it into operator stack. If it is an operator, and the priority is lower or equals to the top operator in the stack, I process two top operands in the stack and execute with the top operator in the operator stack. If it is a left parenthesis, I pushed it into operator stack. If it is a right parenthsis, I process the calculator until it encounters the left parenthesis in the stack.

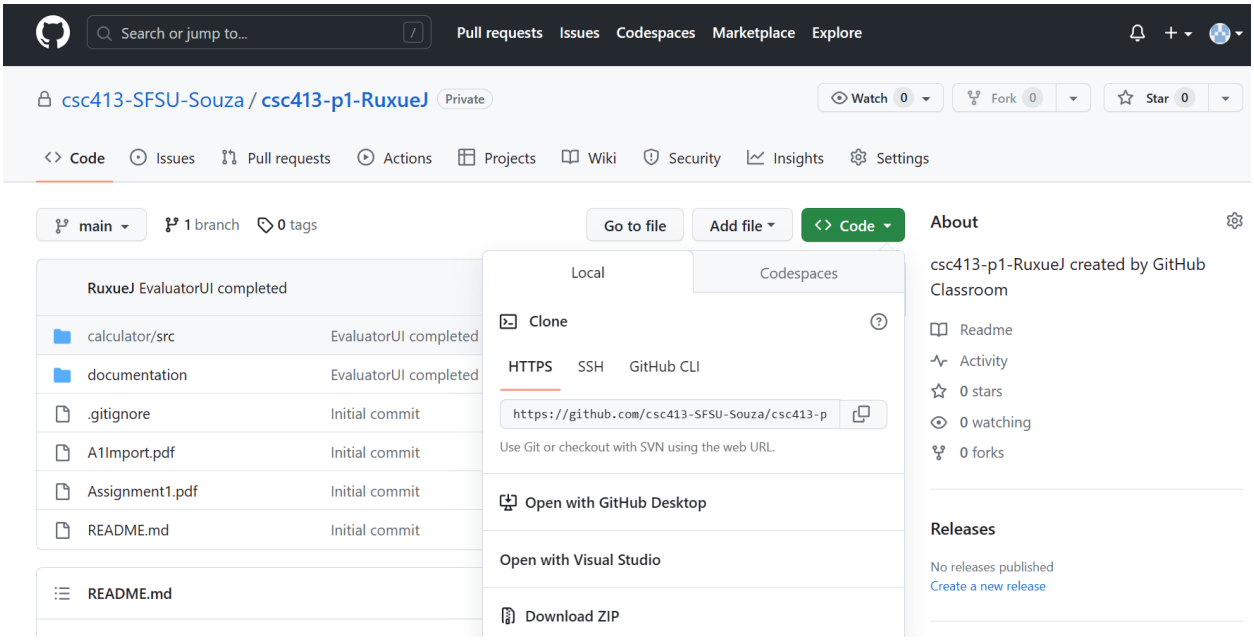
I finished actionPerformed method in EvalutorUI to handle each "click" on the button in the calculator.

2 Development Environment

Java version: 17.0.6

IDE Used: IntelliJ IDEA 2022.3.2(Ultimate Edition)

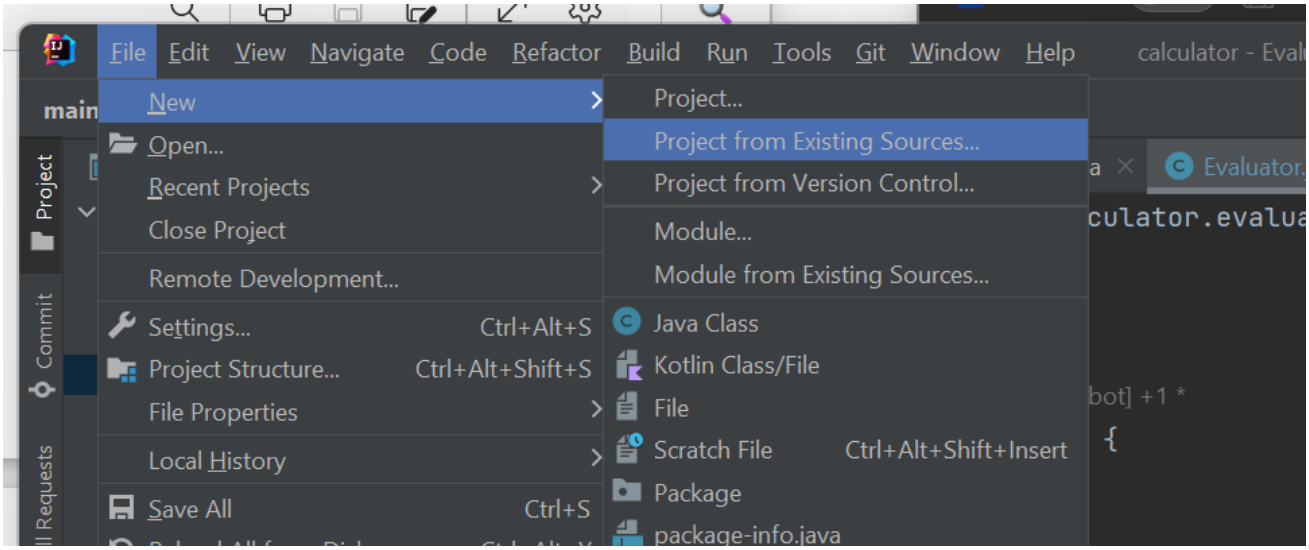
3 How to Build/Import your Project



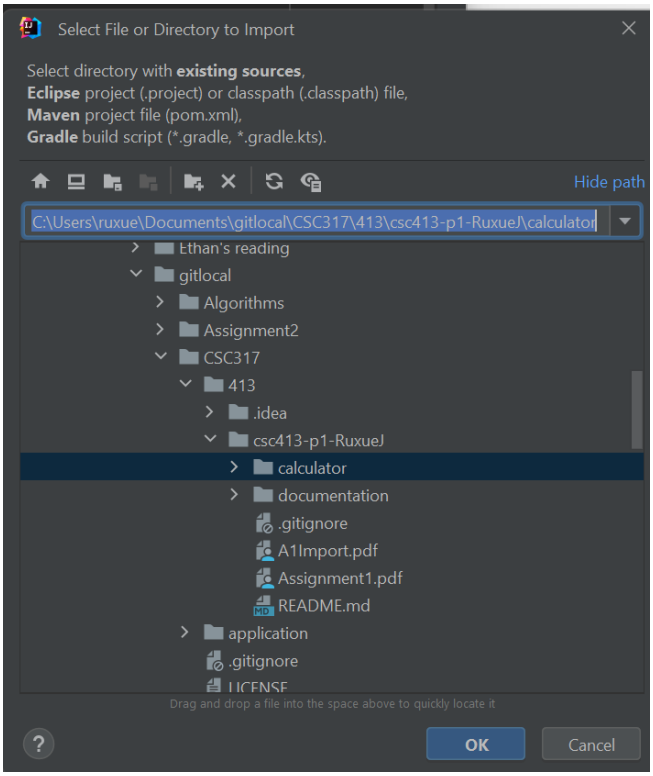
Click the green “Code” button on my repo’s home page. Then copy HTTPS.

In your terminal, cd to the folder you want to store the project.

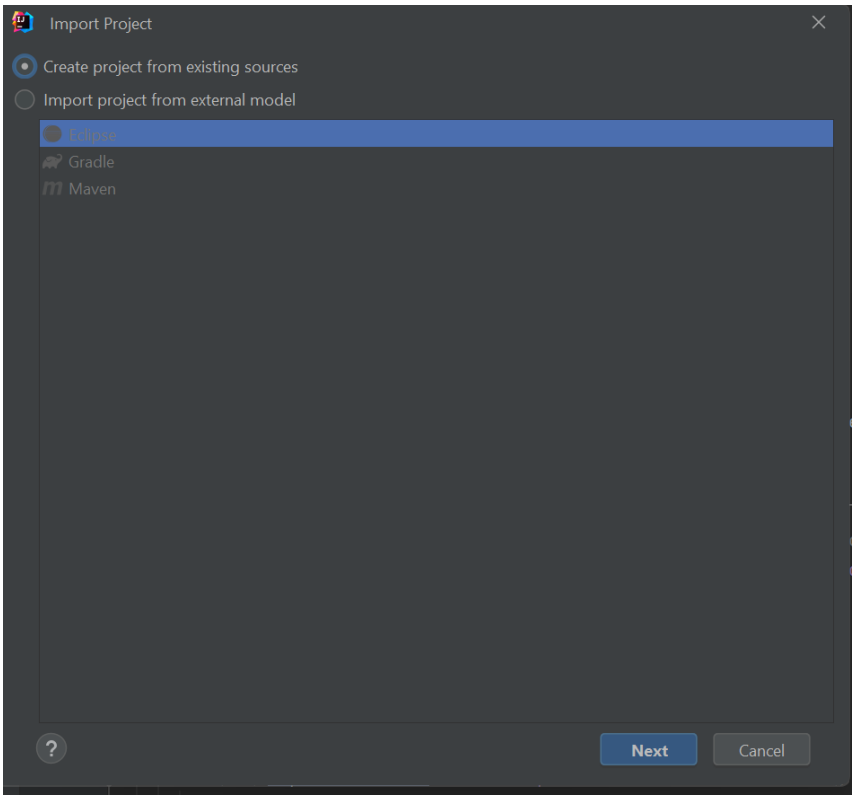
then type: git clone repo_url_you_copied.



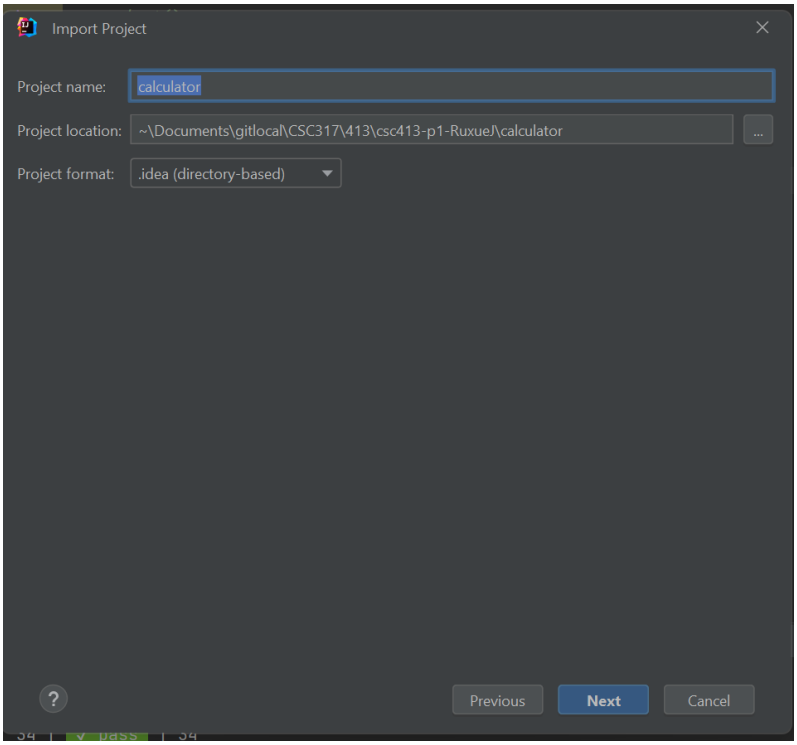
Open IntelliJ, click File -> New-> Project from Existing Sources…



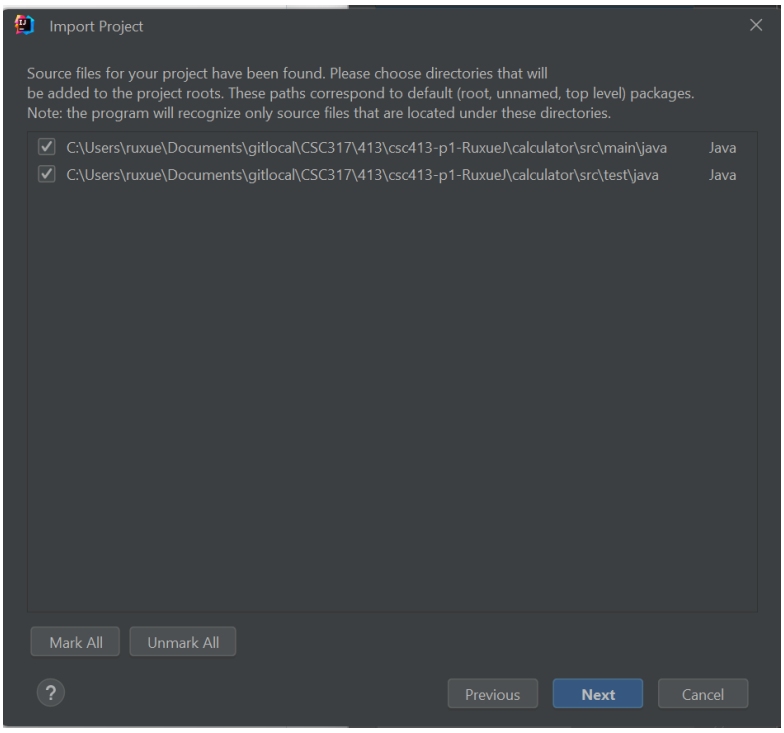
Select the folder your store the project, click “calculator” package, and click OK.



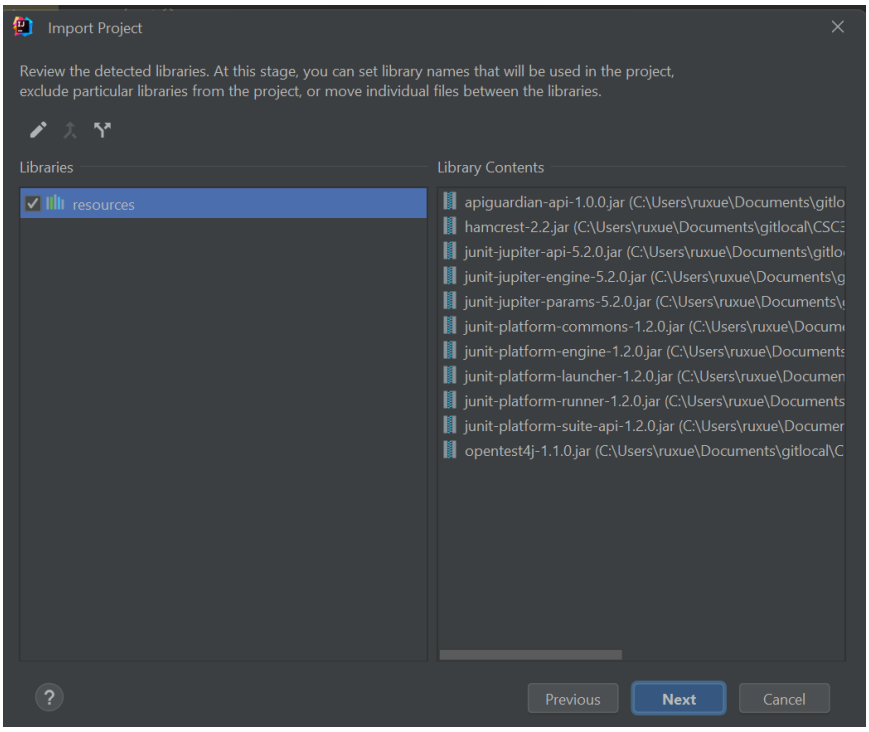
Keep the “Create project from existing resources” radio button selected.



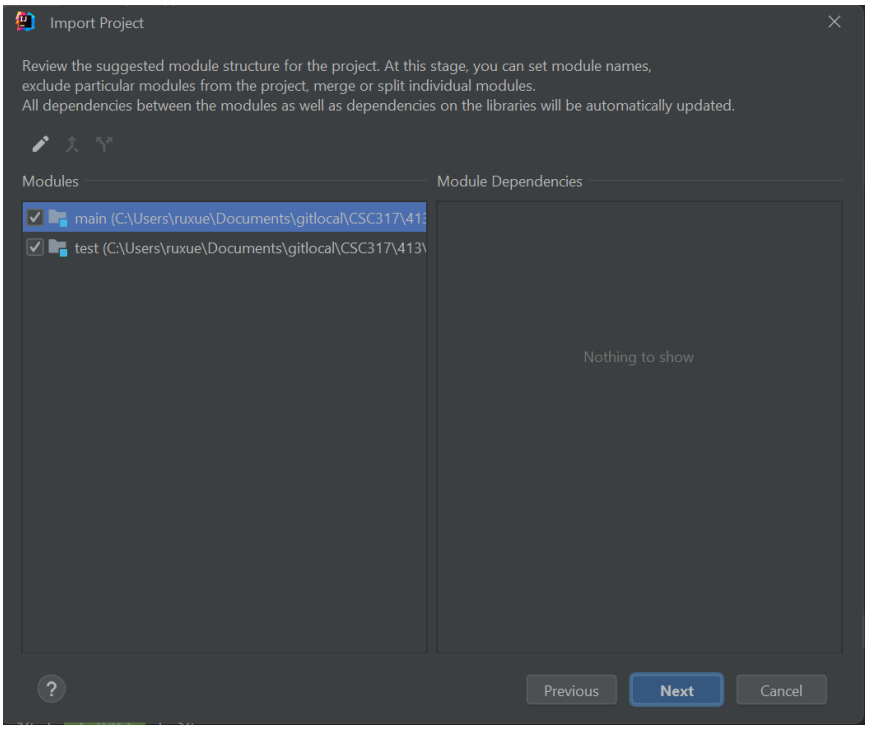
All default fields can be left alone here.



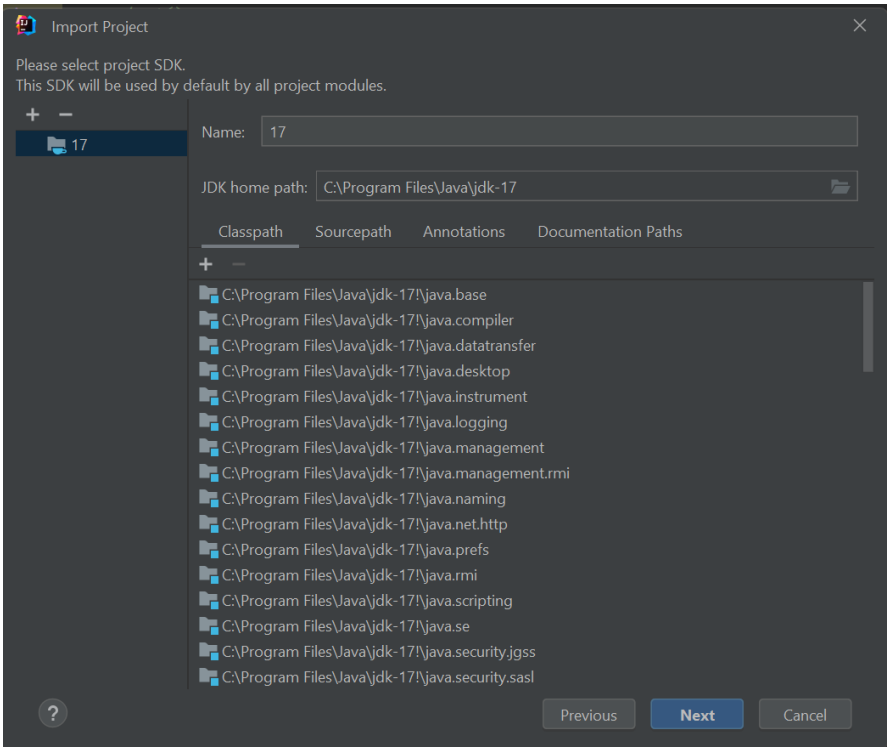
You should see two items shown here. One is the folder for the source code of your assignment. The second is the source folder for the unit tests provided.



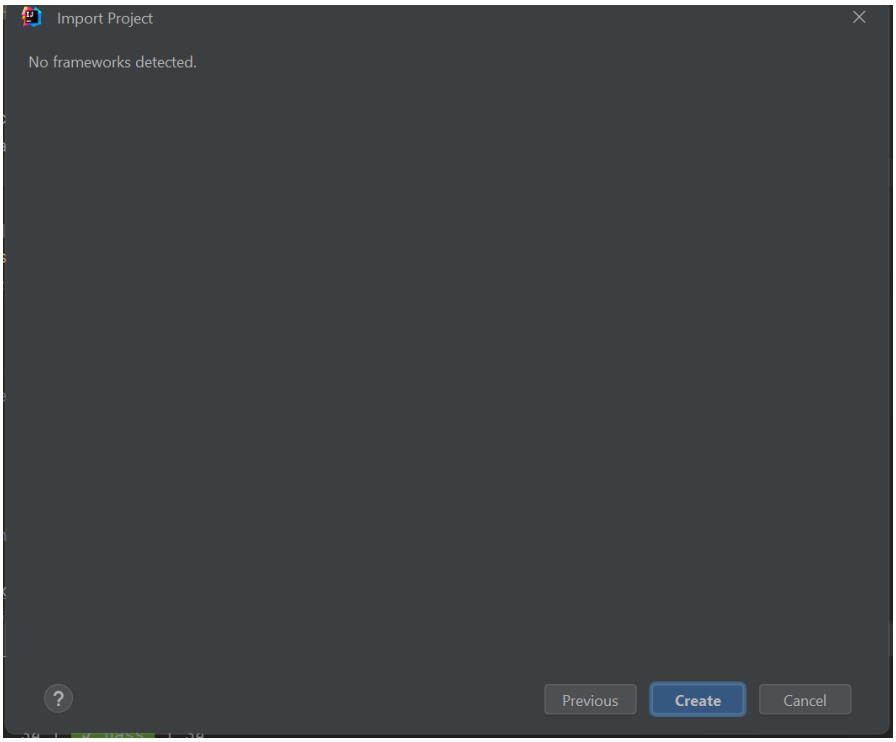
On this pane you should see a list of JARs. These are used for the units tests.



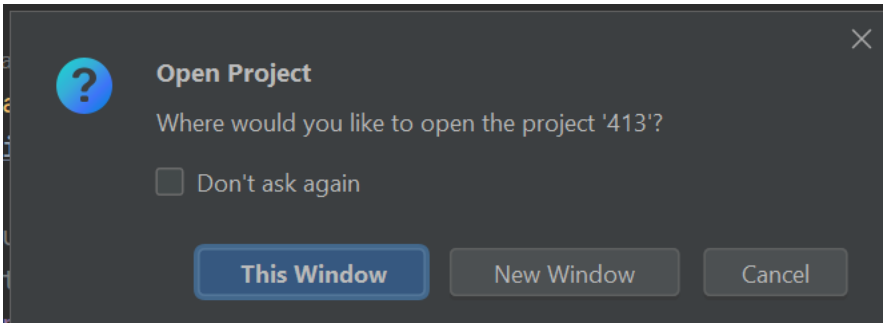
Like the image a few pages back, you should see 2 items here. We a module entry for our source code and a module entry for our unit tests.



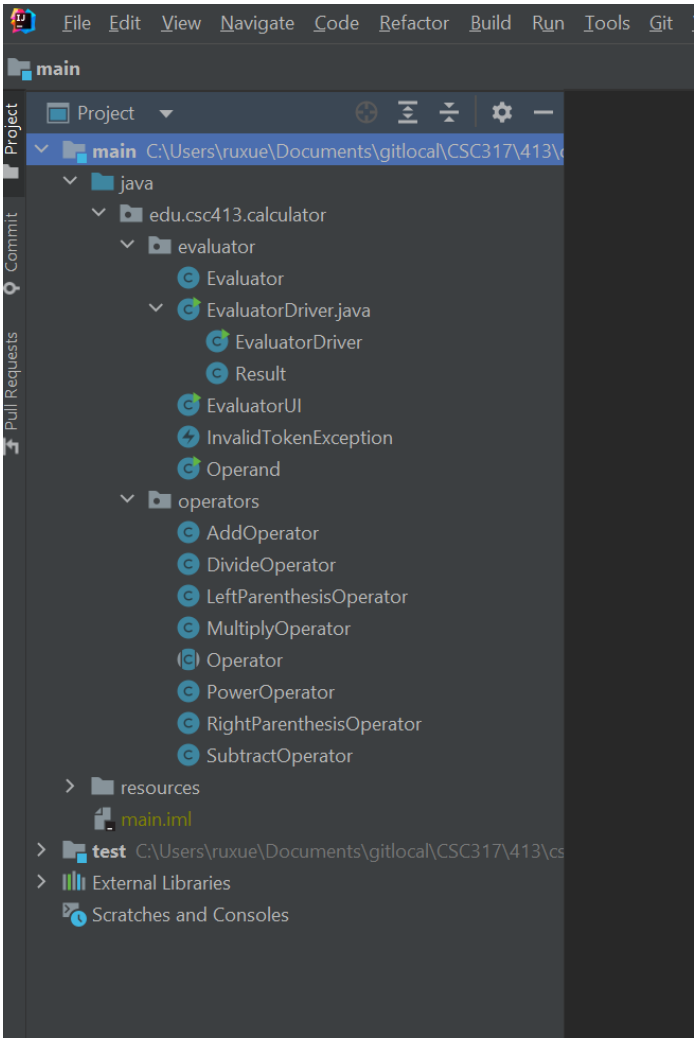
On this image you should see a selected JDK. A Java JDK needs to be installed before moving on from this screen. If one is not shown and you do have a JDK installed, click the + symbol near the top and find where the JDK folder is installed and select it. This will add a new JDK to inteliiJ.



This window should empty since we do not have any frameworks in our project.



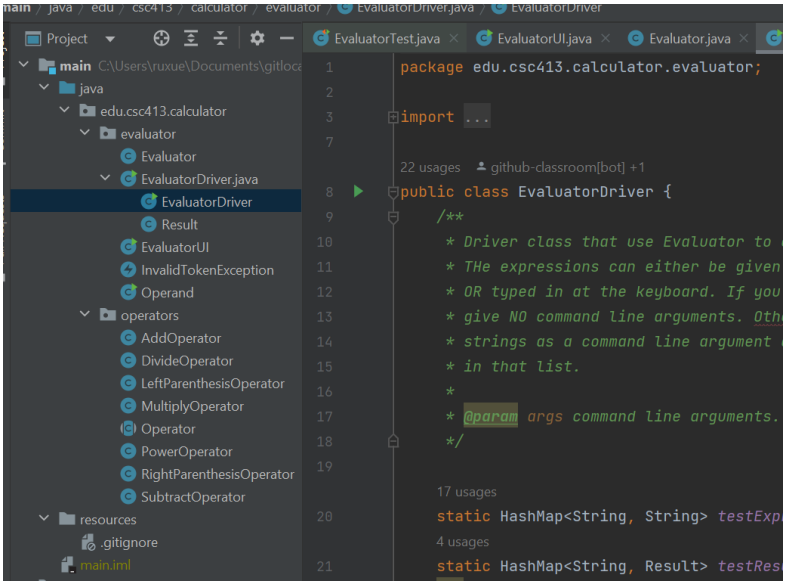
Click New Window.



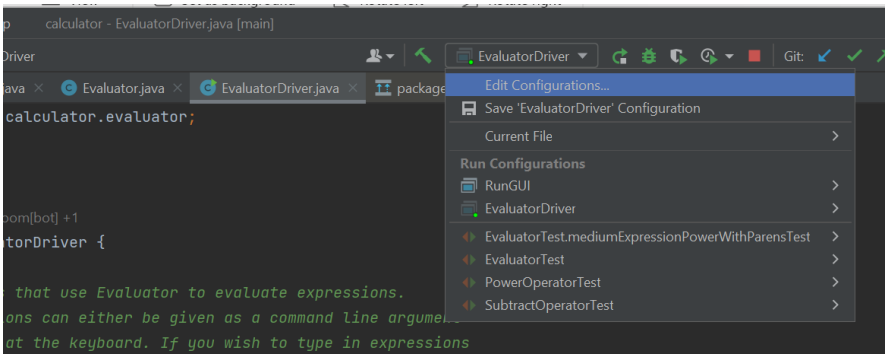
If the import process is done correctly, you should see the following in the Project browser. How to Run your Project.

4 How to Run your Project

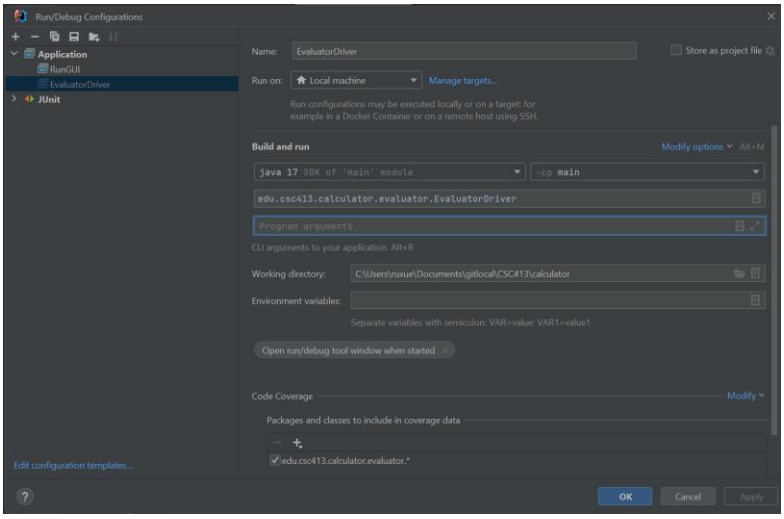
To run the program from keyboard input:



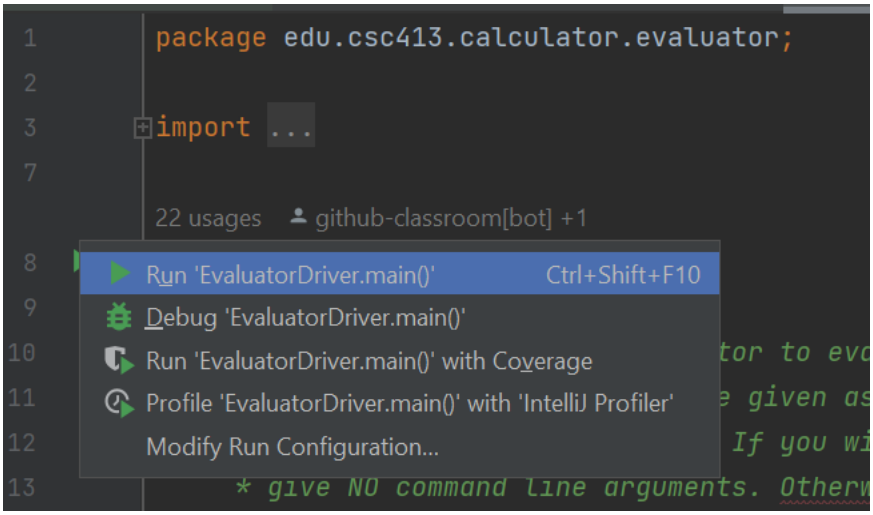
Find EvaluatorDriver class in evaluator package.



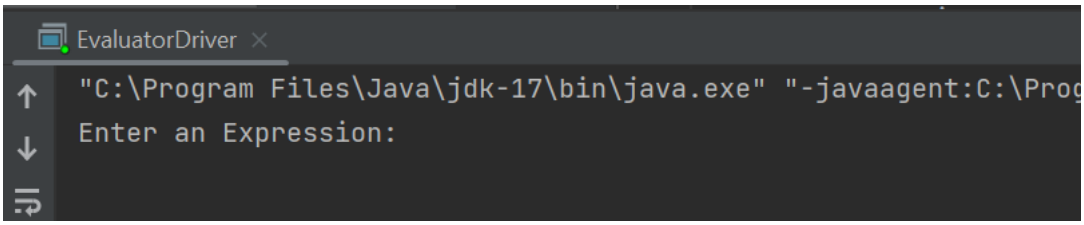
Left click the Edit Configurations…



Set the parameters as above, click OK.

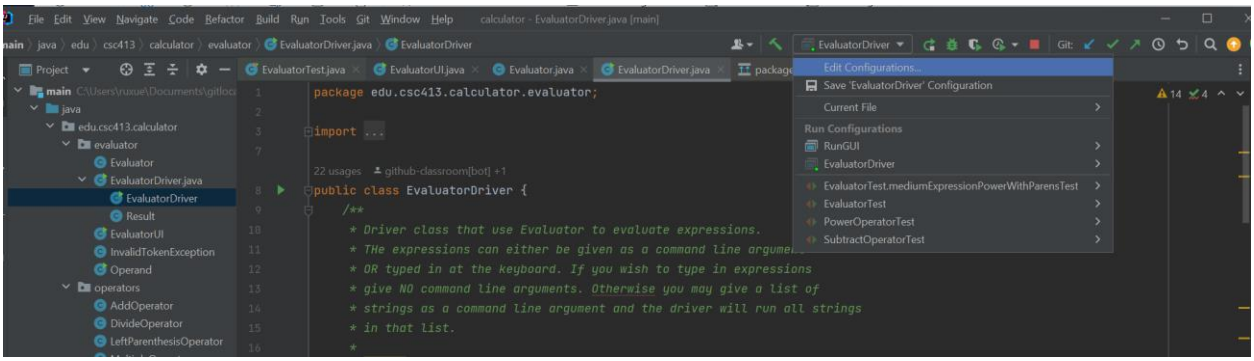


Click green triangle “Run “EvaluatorDriver.main()””

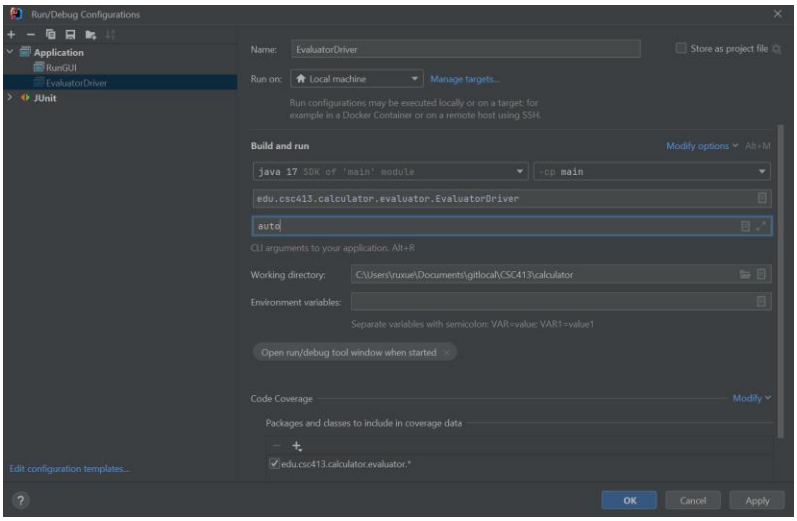


In the console, type the expression.

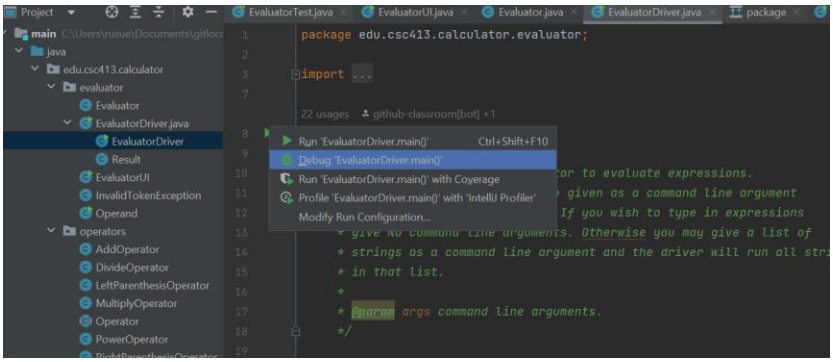
To run the program from the tests in the EvaluatorDriver class:



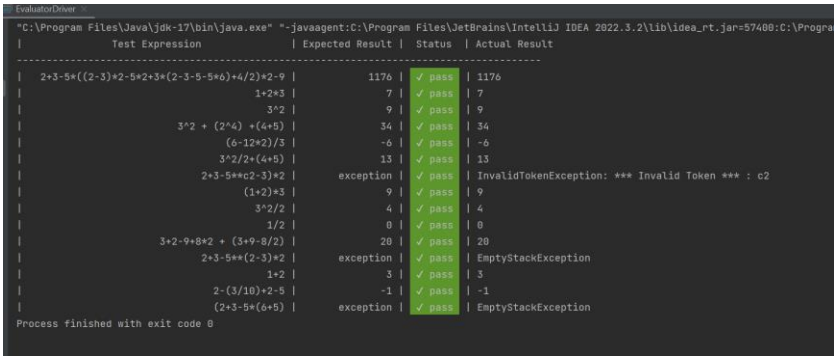
In the EvaluatorDriver class, click “Edit Configurations…”.



Set the Build and Run “auto”, click OK.

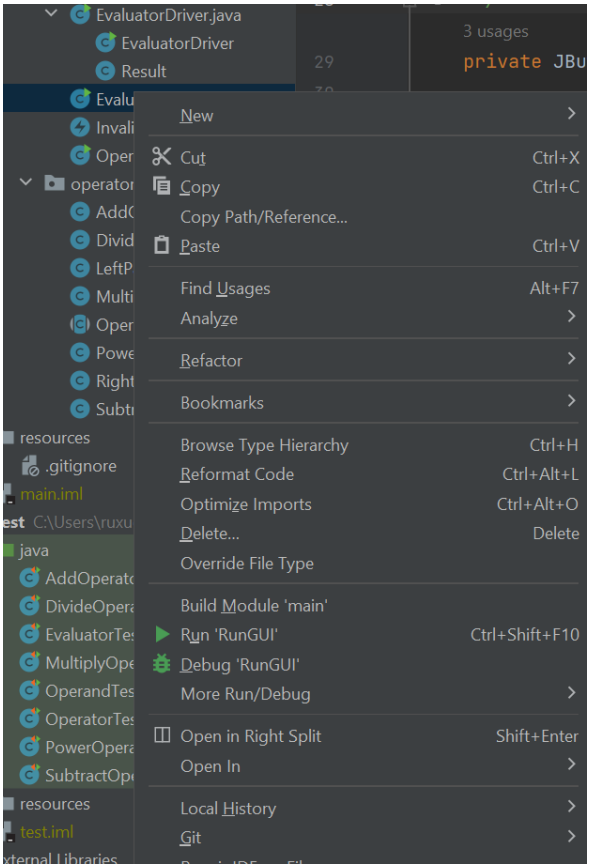


Click the green triangle and click “Run “EvaluatorDriver.main()””

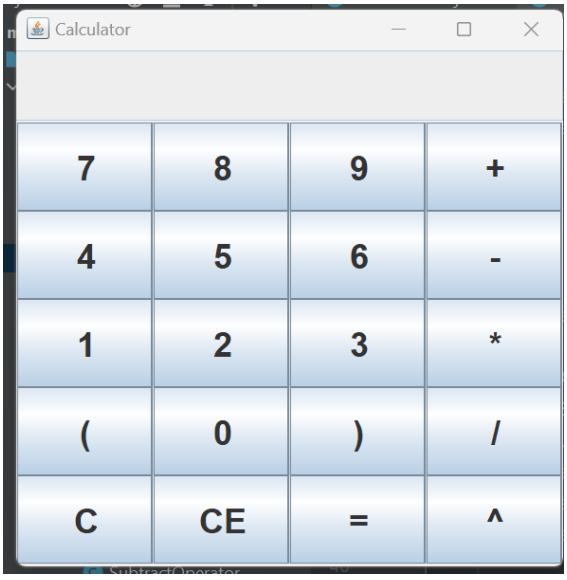


Then the result will come out.

To run the GUI program:



Right click EvalutorUI class. Left click green triangle “Run the ‘RunRUI’”



Start click the expression.

5 Assumption Made

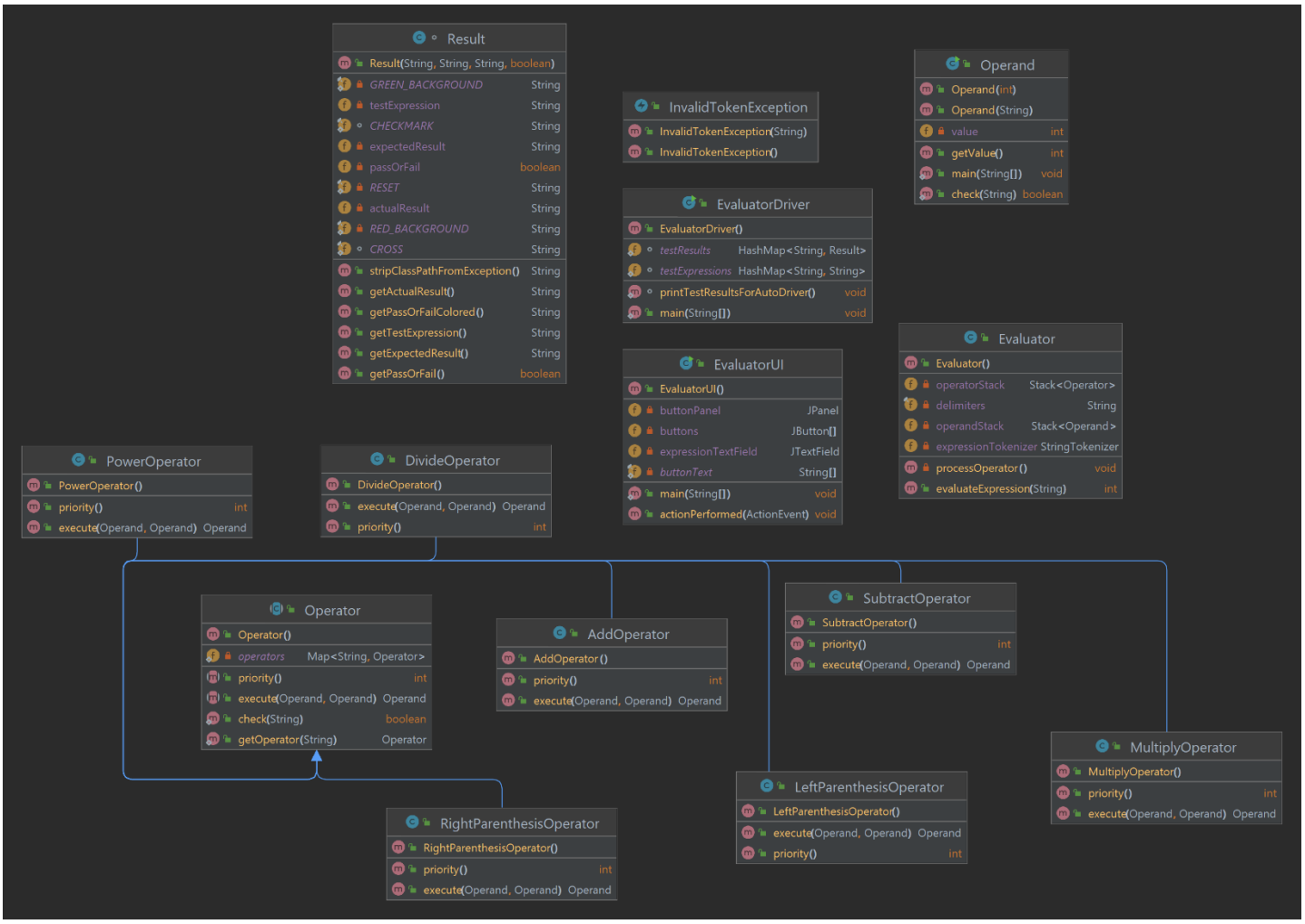
The operands cannot be negative numbers nor floats.

The operators accept only +, -, *, /, (,)

6 Implementation Discussion

Class Operator is an abstract class, Class AddOperator, SubstractOperator, MultiplyOperator, DivideOperator, PowerOperator, LeftParenthesisOperator, RightParenthsis are concrete class extends Operator. Because they all have properties priority and execute method.

6.1 Class Diagram



7 Project Reflection

I think the most important part of this assignment is getting to know the structure of project and the relationship between different classes. In the beginning, I felt it is hard to start because I did not know where to start. Then I started with “Operand” class, pass the “OperandTest”; worked on “Operator”class, pass the “OperatorTest”, then I can see the big picture of this program. Then I feel more confident.

The “evaluator”class costs me the longest time. One reason is that I used to delete some working codes and attempt to start over, instead of commenting them. I spend too much time start over. Another reason is that I spend too much time thinking the efficient code instead of working code. Next time I will make the code work first, then I can optimize them.

Overall, this program is not that bad. And I think I will make it done more quickly next time.

8 Project Conclusion/Results

The program works well!