Assignment 1 – Expression Evaluator
CSC 413 Project Documentation
Summer 2020

Name - Rajdeep Singh
ID: 920258882

CSC 413-02 Summer 2020
GitHub Repository Link:

https://github.com/csc413-02-SU2020/csc413-p1-Rajdeep0303.git

Table of Contents

1 Introduction

The main goal of the project is to program the calculator to evaluate the given expression in the Java language. In this project, I build a simple calculator that can perform simple addition, subtraction, multiplication, and division.

1.1 Project Overview

It is a simple calculator that can achieve basic mathematical expressions and execute easy algorithm work. The expression evaluates an infix expression which uses operators +,-,*,/,(,) to evaluate the expressions. This project performs like a regular calculator we use every day with simple arithmetic.

1.2 Technical Overview

This project includes a stack to create the proper operator that is being enrolled. All of them are encapsulated to confirm that there's no issues left. The stack can be used to determine if an operator or parenthesis being pushed onto the stack, or if there's a distinct result for the infix expression.

The Evaluator class is responsible for checking the status of the stack and checking if we need pop statements or if the stack is empty already. Evaluator class takes a single String parameter to represent a mathematical expression.

It is intended to be a demonstration of inheritance and encapsulation, which is demonstrated through private access and inherited classes in the operators. String parameters in the Evaluator class represent a mathematical expression.

1.3 Summary of Work Completed

In this project, we learned how to use the stack, hash map, implement various methods, abstract operator class, etc. Negative numbers do not cause any issues in the program, and GUI and all keys work correctly.

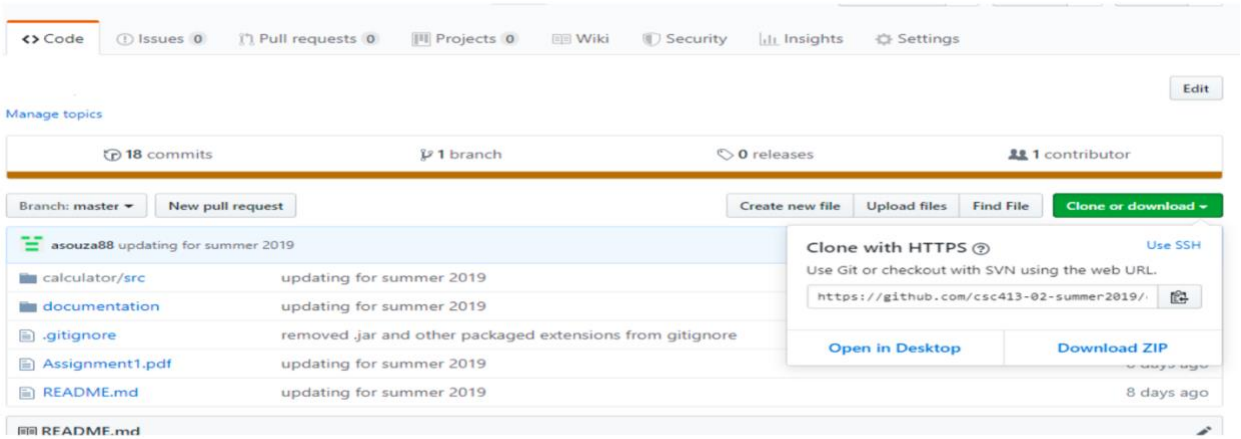
2 Development Environment

For this project, I use IntelliJ IDEA CE to compile it, java version 2020.1.2.

Java version 13.0.2 was used to develop this project in IntelliJ IDEA CE was used to develop this project.

3 How to Build/Import your Project

To be able to build the project in IntelliJ IDEA CE, go to the GitHub link where you can see the repository is located. Copy the link to import, or you can download the zip file containing the whole project.

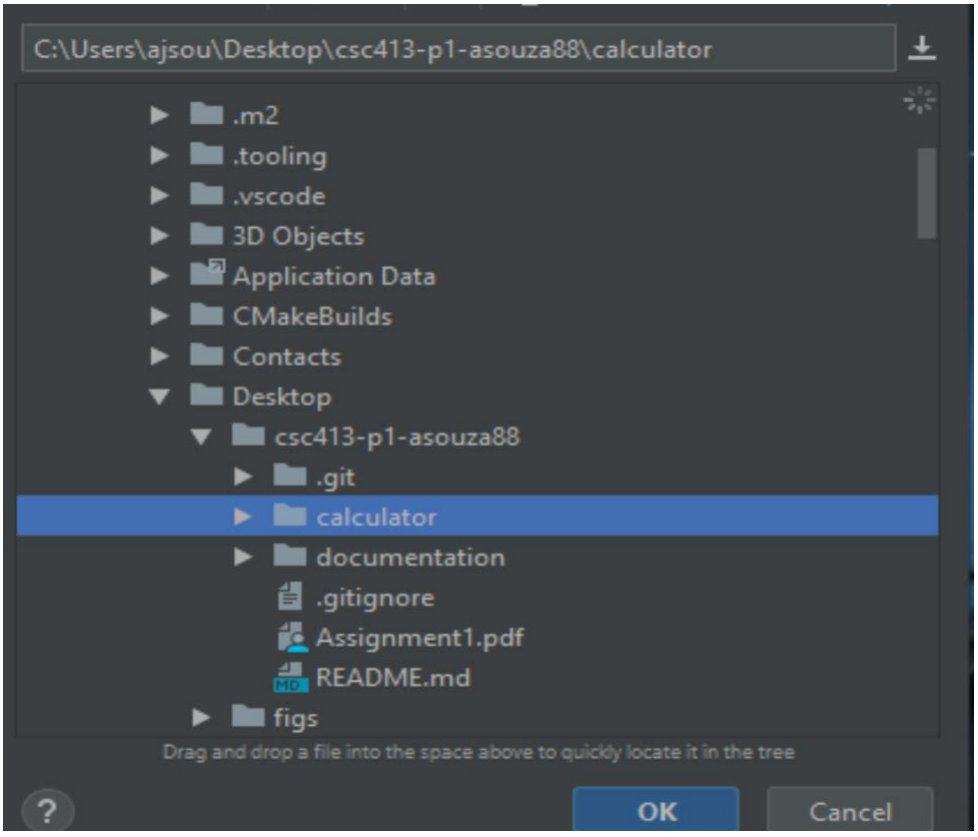


Click on “Clone or Download” button. Then select the HTTPS or SSH if you have an SSH keys set up.

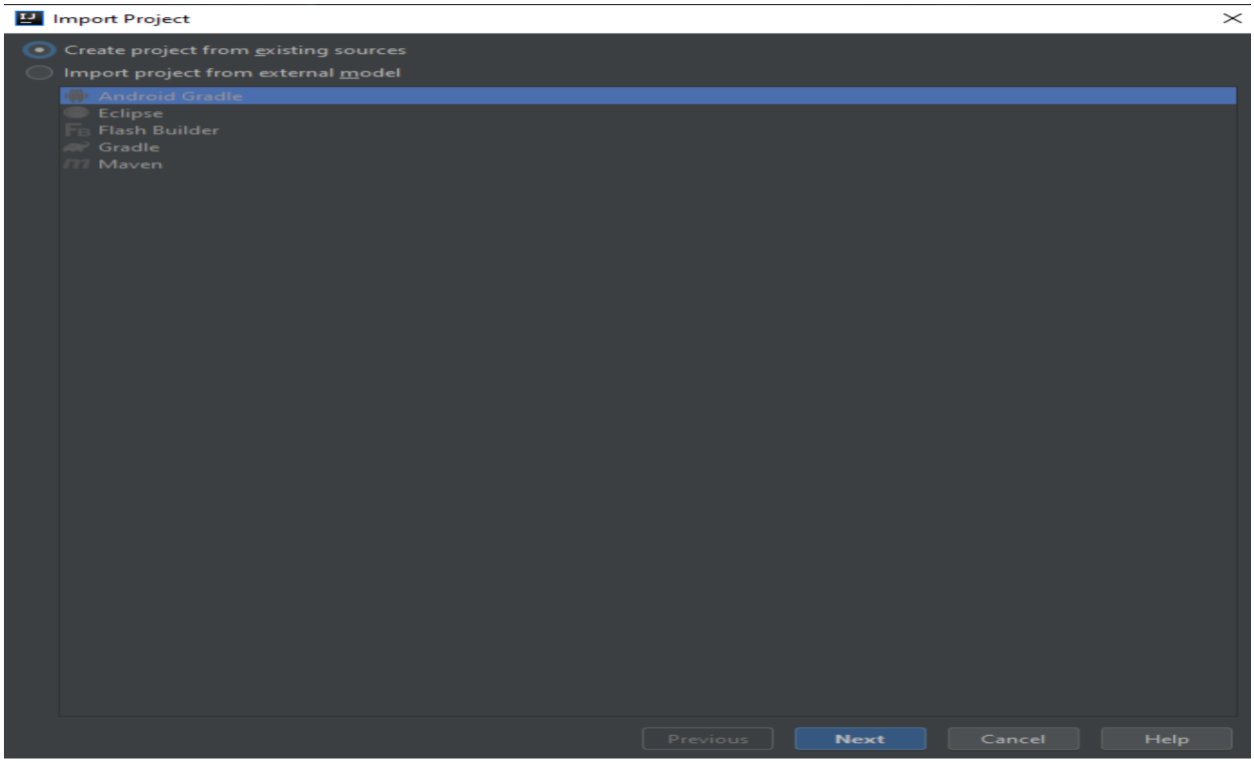
Then, copy the link in the terminal. For example, git clone repo_link_here

Once the repo is cloned, follow the following steps to import your project into IntelliJ. Steps are very easy, just follow the direction.

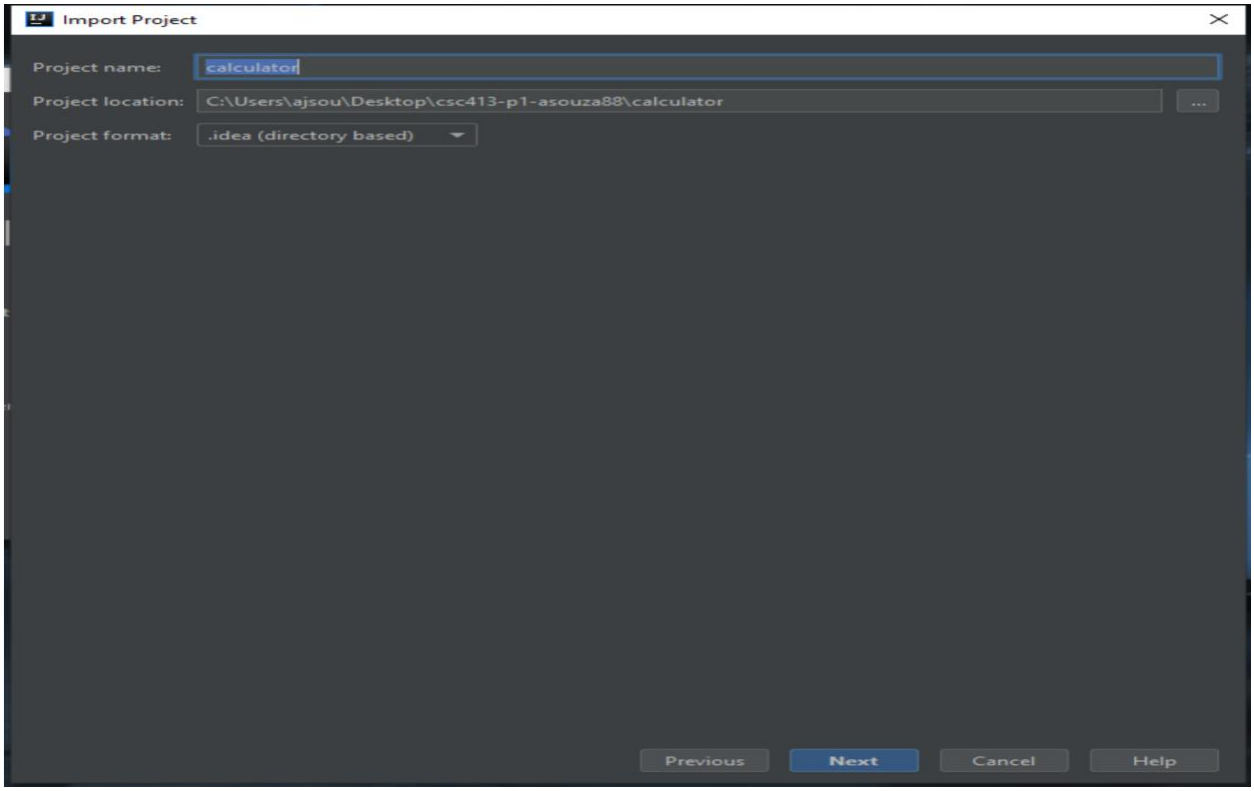
Click on “Import Project”



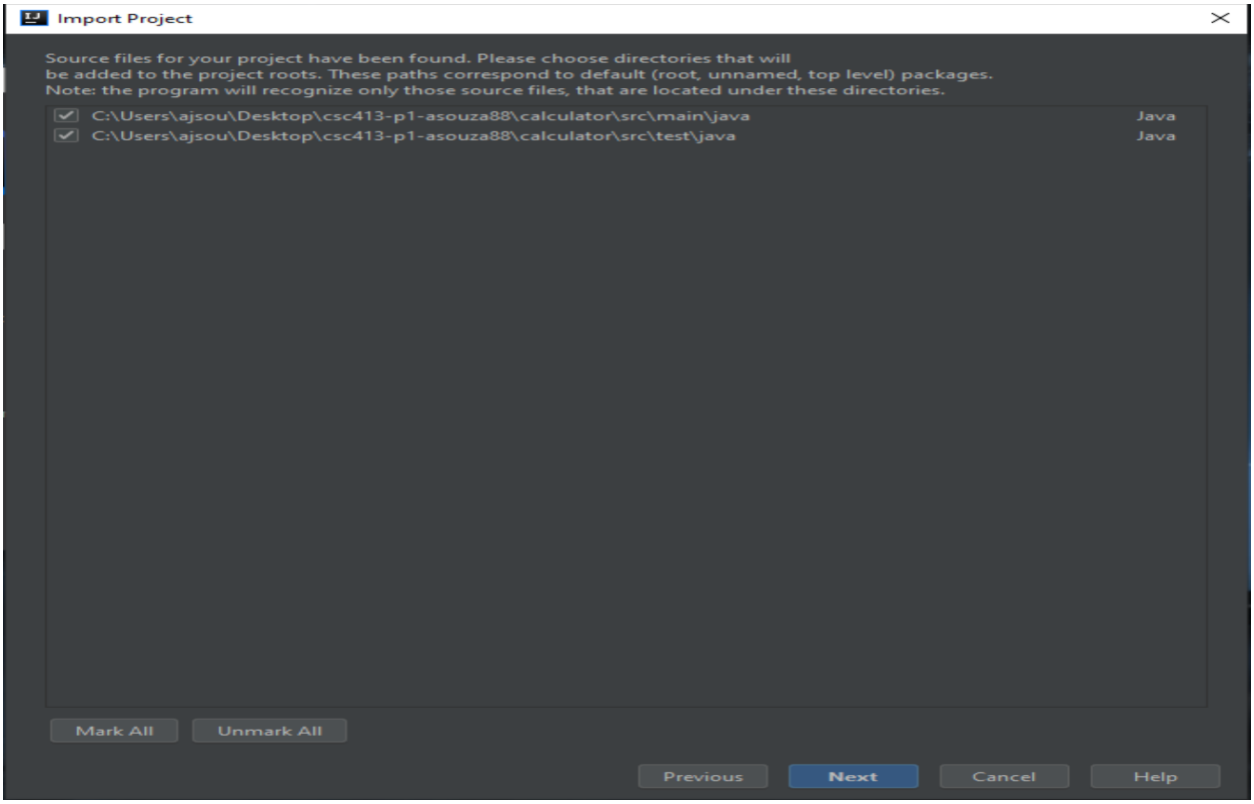
Select the calculator folder



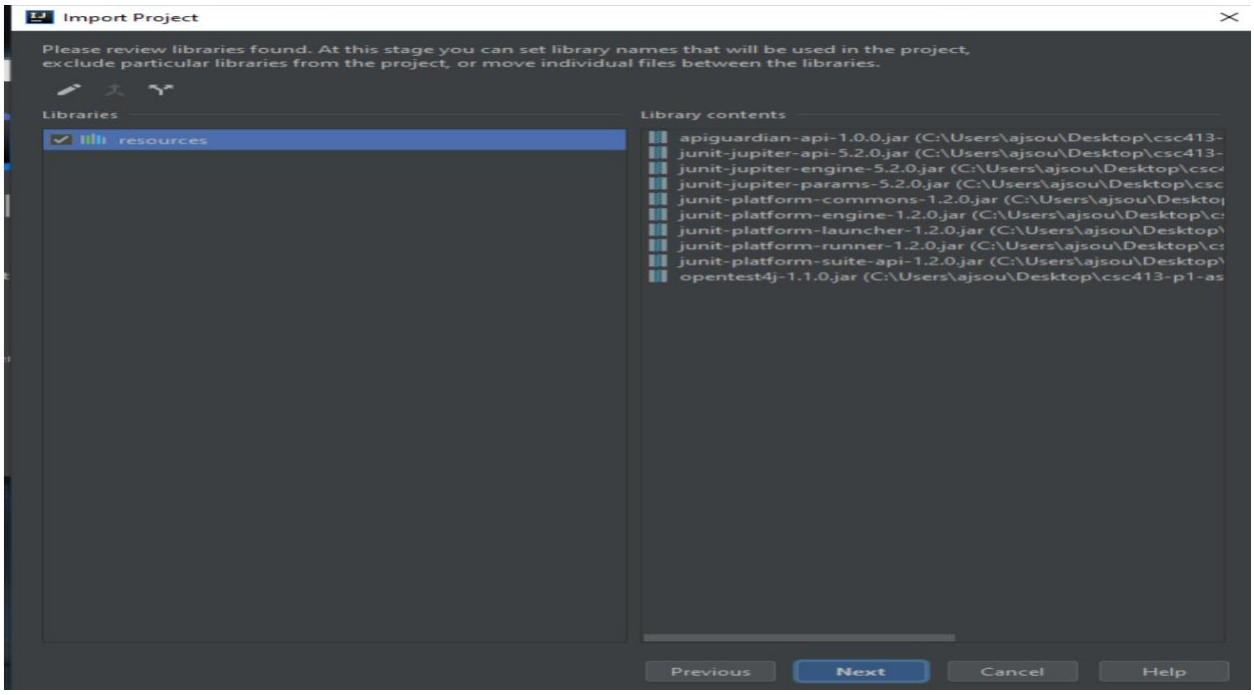
Click on “Create project from existing resources” and then click “Next” button.



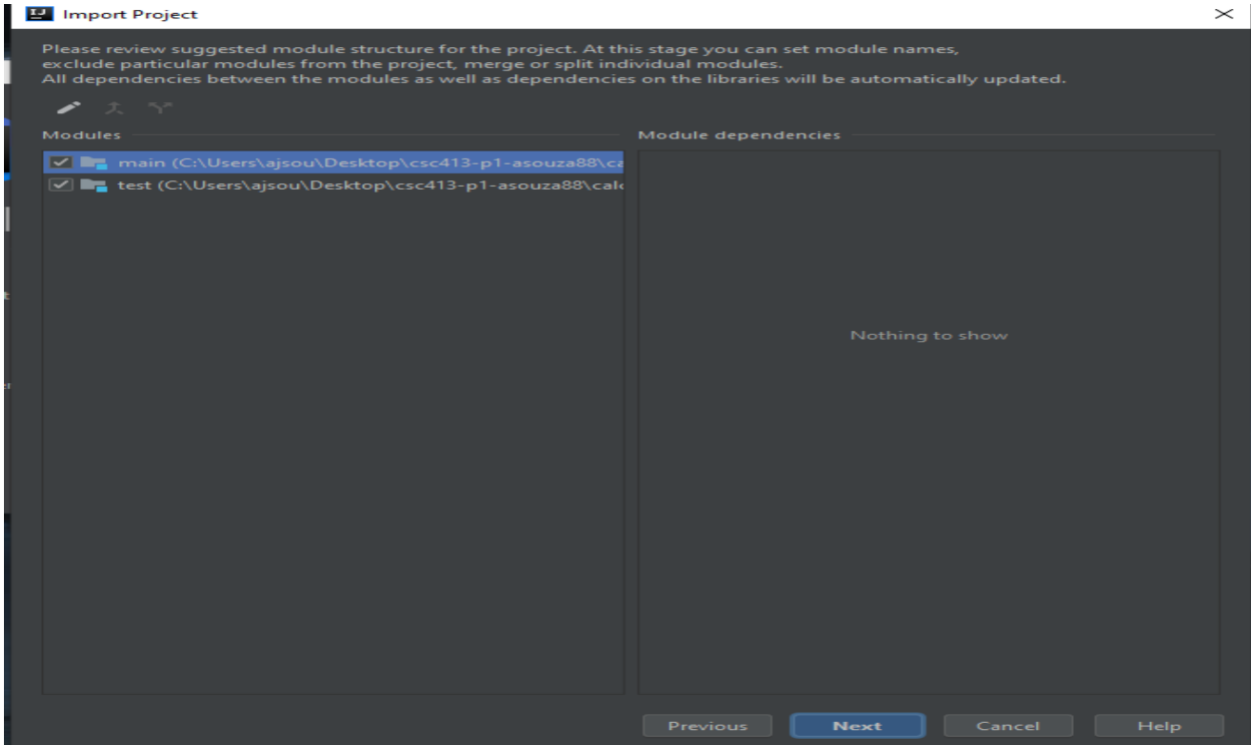
Leave the all default fields alone.



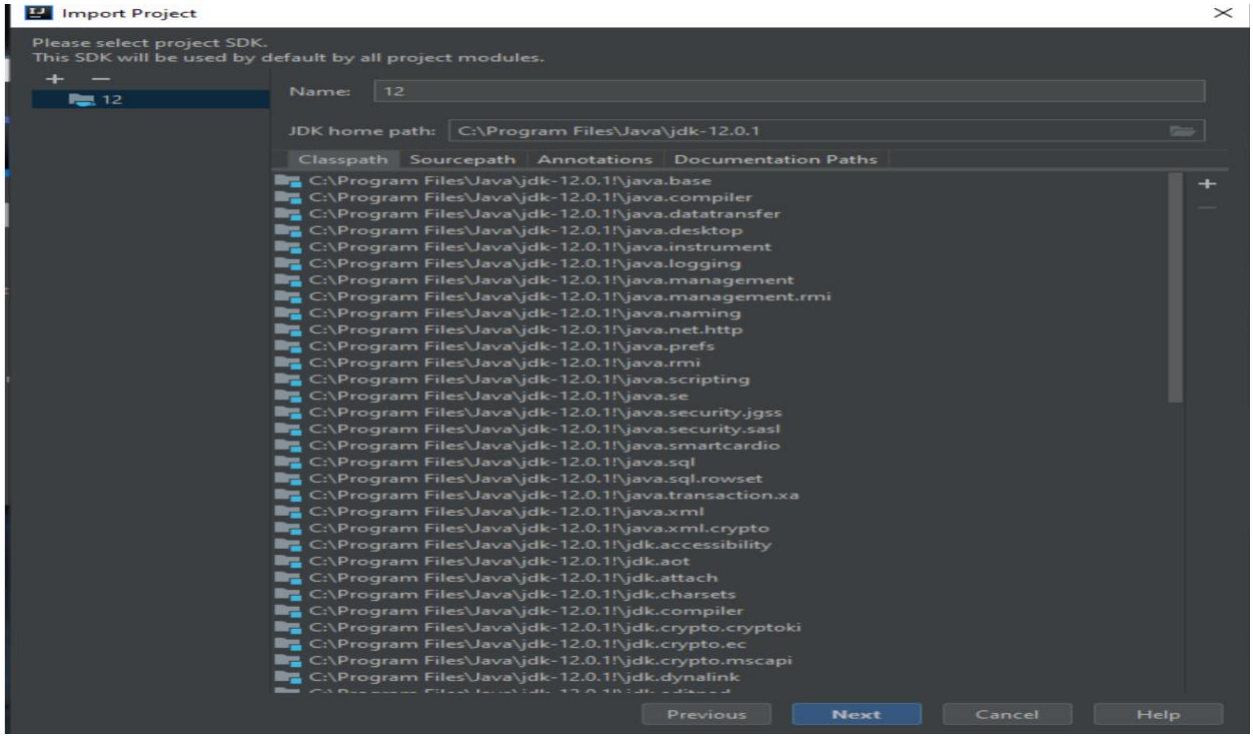
Mark both items. One is the source code folder and second folder contain source of unit tests. Then, click on “Next” button.



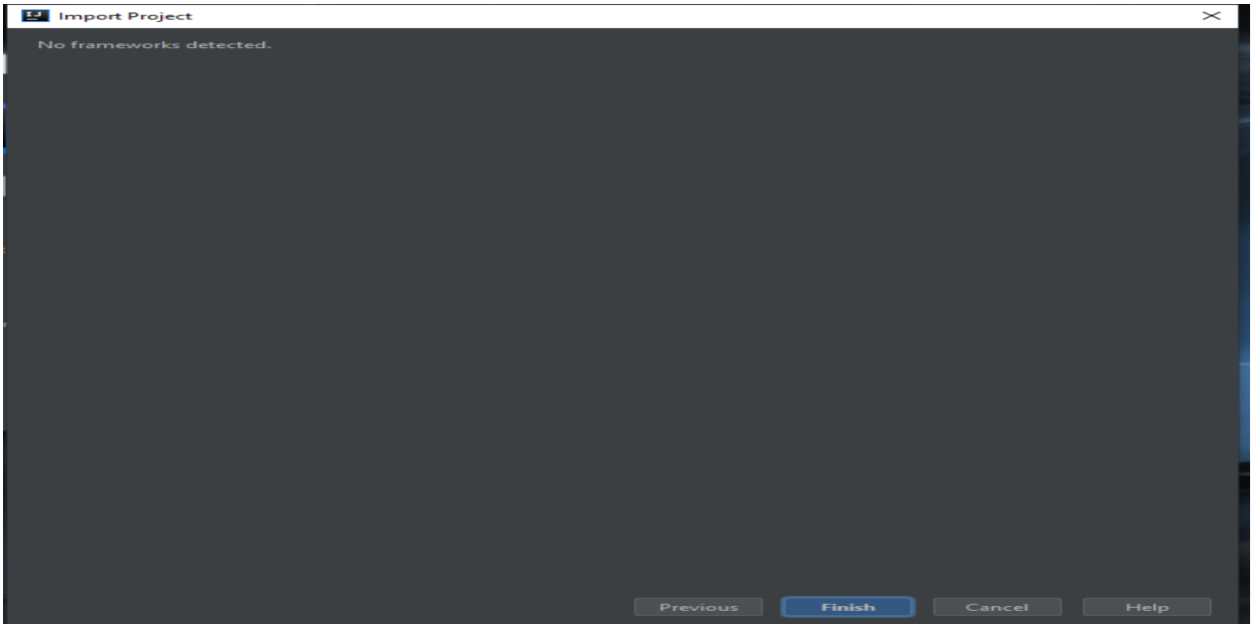
Here, you will see a list of JARs. Click “Next” and all default fields can be left alone here.



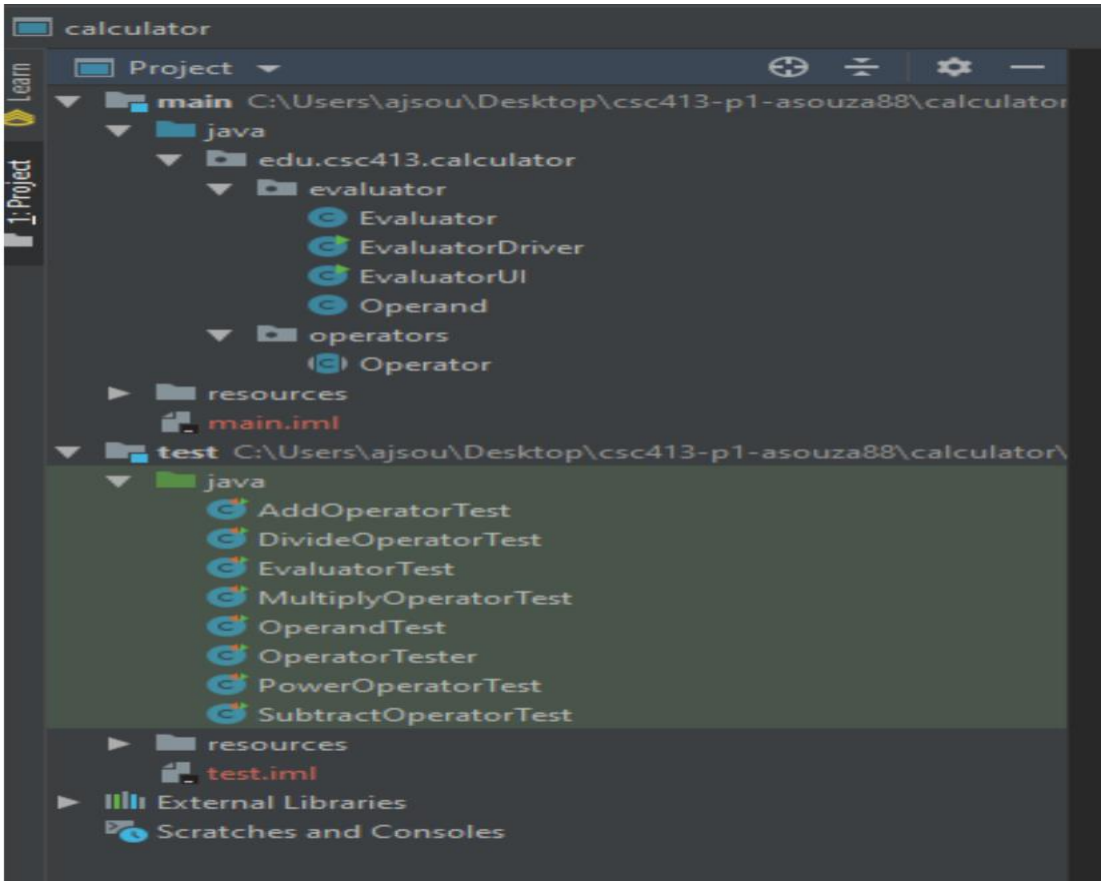
Click “Next” and all default fields can be left alone here.



If you see a screen like this then click “Next” otherwise java JDK need to be installed before going forward. If you could not see this screen then click on + symbol on the top left. After JDK add click “Next”.



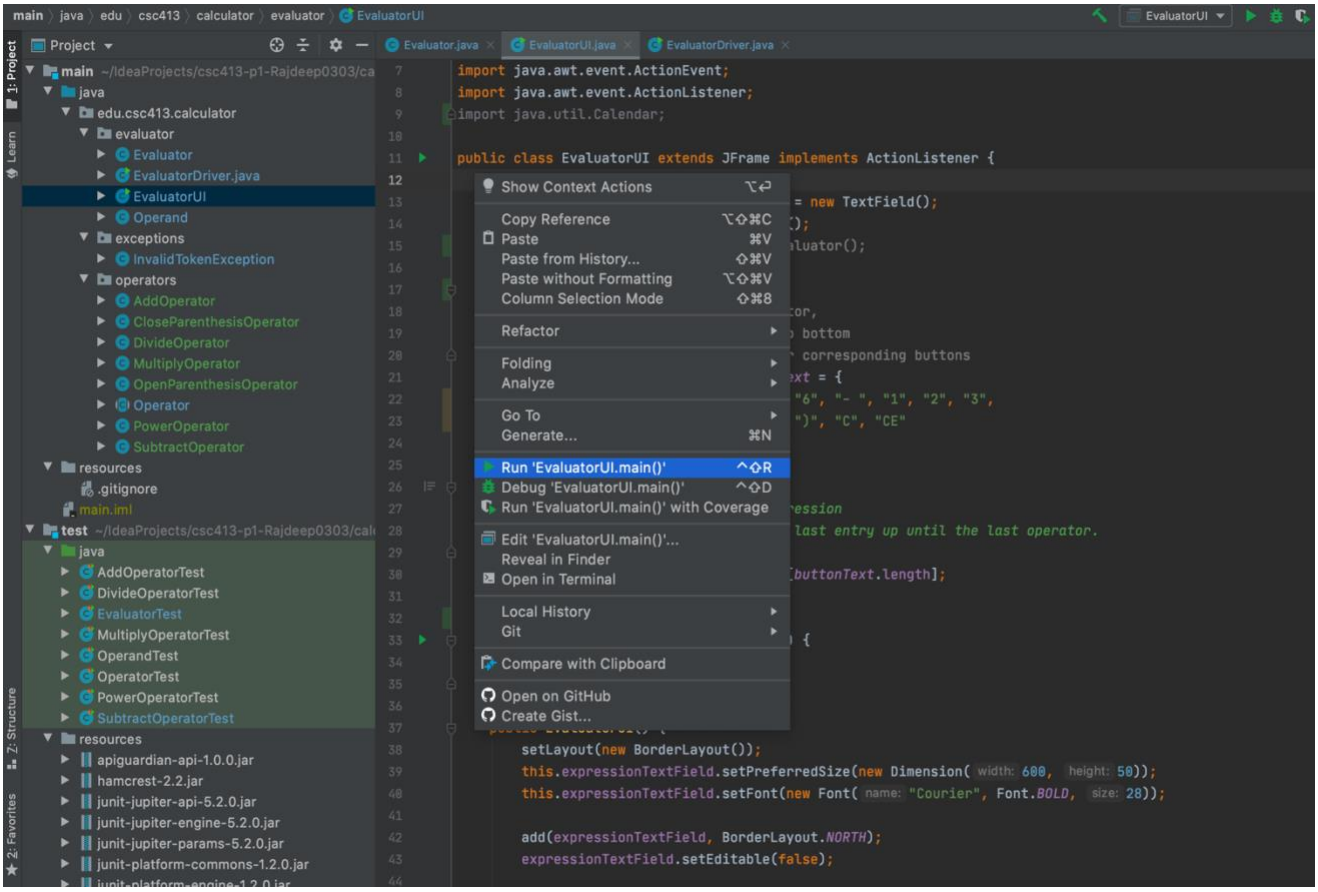
Click “Next” because this window should empty because we do not have any framework.



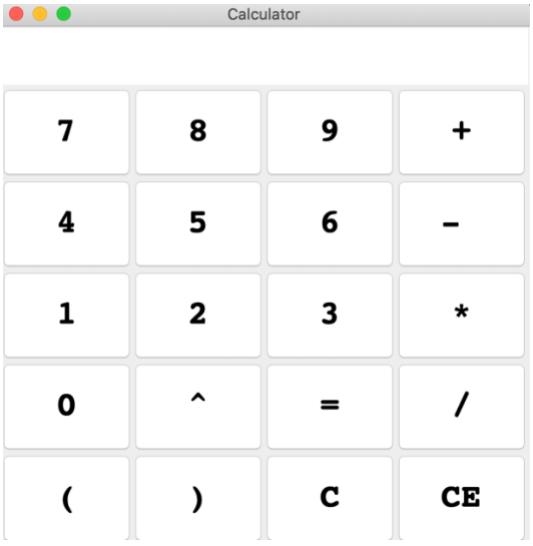
When your import finished, you will see the following classes, test files, resources, etc. If you do not see then, your import process is not done correctly.

4 How to Run your Project

After importing and building the project, you can use the “Run” button to see the calculator.

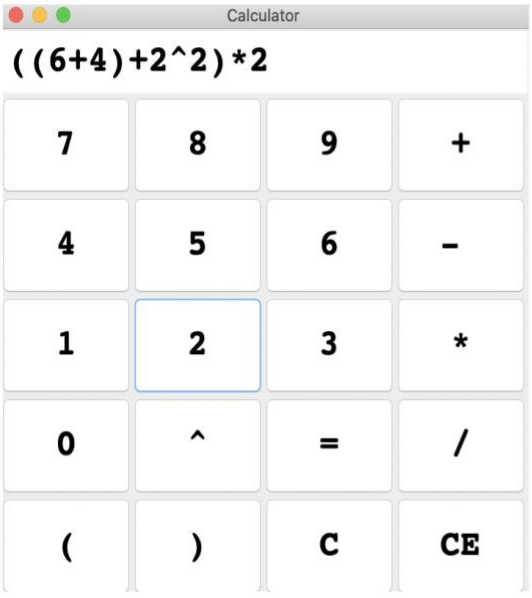


Right click the EvaluatorUI and choose Run ‘Evaluator.main()’ to run this project. Then this calculator will come up.

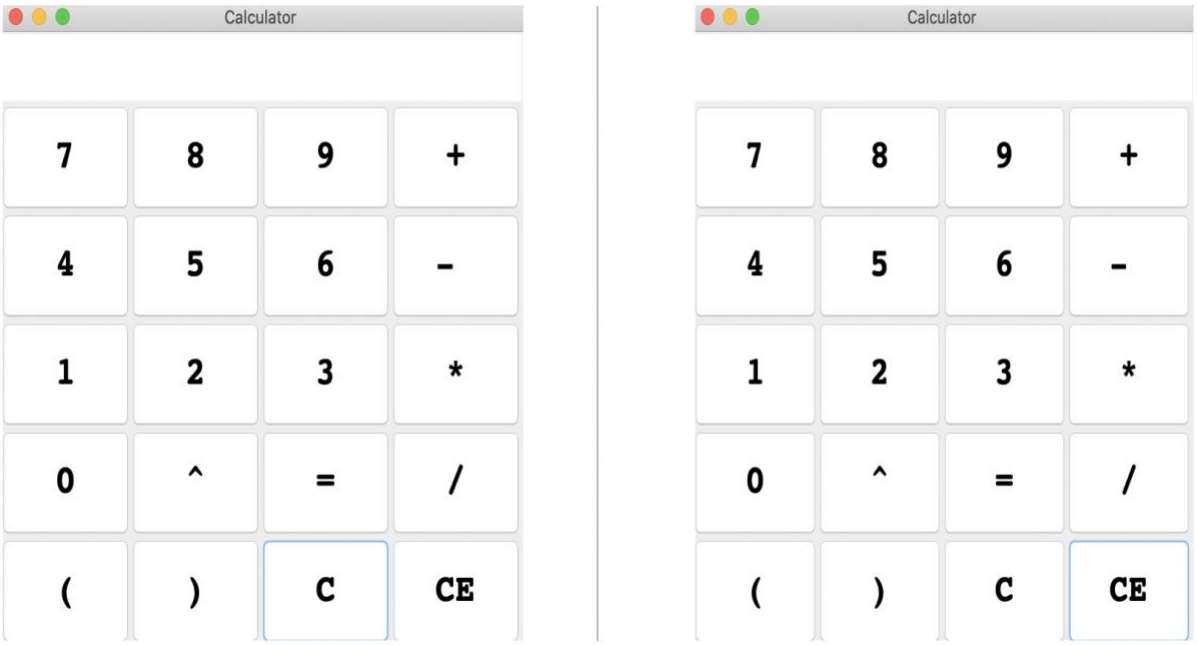


Then we can do some simple calculations. For example, 7+3+5 = 15

((6+4)+2^2)*2 = 28



Click button C to clear the inputs. Also, click button CE to clear the last input.



5 Assumption Made

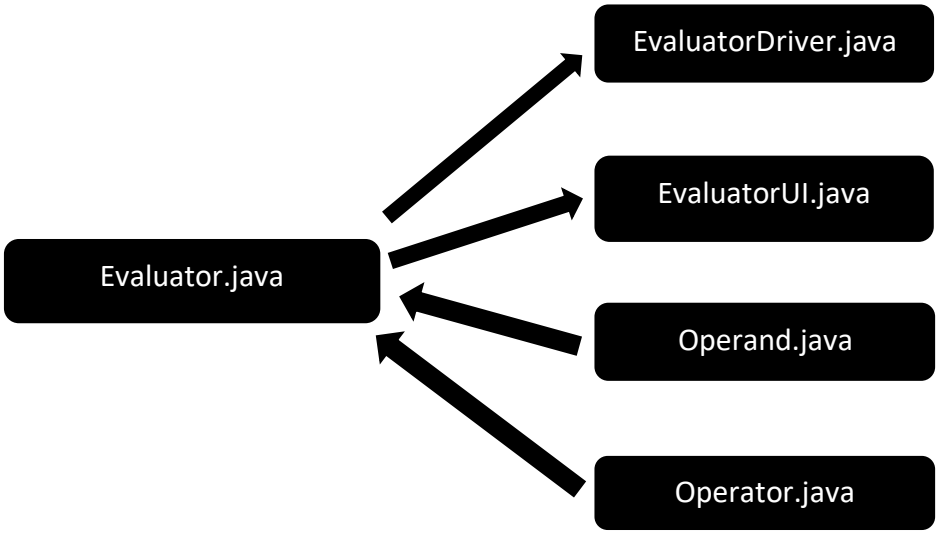
Assumptions were made involving priority of division, addition, subtraction, exponents, parenthesis, and multiplication. I assume that all the input should be integers. Also, I begin with simple algorithms and then slowly test moves to complex algorithms.

6 Implementation Discussion

There are five files in this project:- Evaluator.java, EvaluatorDriver.java, EvaluatorUI.java, Operand.java, and Operator.java. I work on the four files except for the EvaluatorDriver because EvaluatorDriver is a test file for the project.

6.1 Class Diagram

This diagram show the relationship between the five files.



7 Project Reflection

Working on this project since the beginning, I saw there were many Java-related concepts that I almost forgot since I haven't worked with java concepts. The only way I could learn is by studying online and asking CS friends for help and suggestions about coding. I am doing my best to improve my java skills as much as possible. I am getting problems with inheritance and encapsulation, but I am trying to cover them. When I was working on the project, I found some errors, for example, reading the parentheses, unbalance Expression, and reversed input. I debug the whole program to find the issues, I start with operator classes then move to EvaluatorUI, and in last, I debug the Evaluator class.

8 Project Conclusion/Results

Overall, this project gives me an excellent knowledge of inheritance and encapsulation. The project itself performs very well, and it has basic algorithms with simple mathematical expressions, it can handle the input of negative integers. Also, this project cannot handle the fraction because of the result displays in the whole number. I feel better after finishing this project, and I have a better understanding of the OOP principles.