CSC 413 Project 1 Calculator Documentation

Calvin Tam
917902523

CSC413.03
Fall 2019

https://github.com/csc413-03-fall2019/csc413-p1-ctam4

0 Table of Contents

1 Introduction

1.1 Project Overview

This project is to practice object-oriented programming (OOP) implementation on two programs - an object that evaluates mathematical expressions and a graphical user interface (GUI) around the object. In short, this is a project than convert end-user input of mathematical expressions, such as `(1+2)*3^4/5`, into result inside the user interface (UI).

1.2 Technical Overview

The project has been based on the template available by the professor. There are several incomplete classes are provided: 1) `Operand`; 2) `Operator`; 3) `Evaluator`; 4) `EvaluatorDriver`; and 5) `EvaluatorGUI`. On top of these classes, JUnit jars have been provided for importing into the project. These are the basic structure of the project.

From the UI, the user inputs mathematical expressions; `EvaluatorUI` handles user inputs and react accordingly. If the end-user requests result, the program calculates it using `Evaluator`, `Operand`, `Operator` and its subclasses; then program outputs the result on the UI.

1.3 Summary of Work Completed

I have completed `Operand` class, `Operator` class and its subclasses: 1) `AddOperator`; 2) `SubtractOperator`; 3) `MultiplyOperator`; 4) `DivideOperator`; 5) `PowerOperator`; 6) `OpeningParenthesisOperator`; and 7) `ClosingParenthesisOperator`. They are the essentials of the project. `Evaluator` requires them to be functioning.

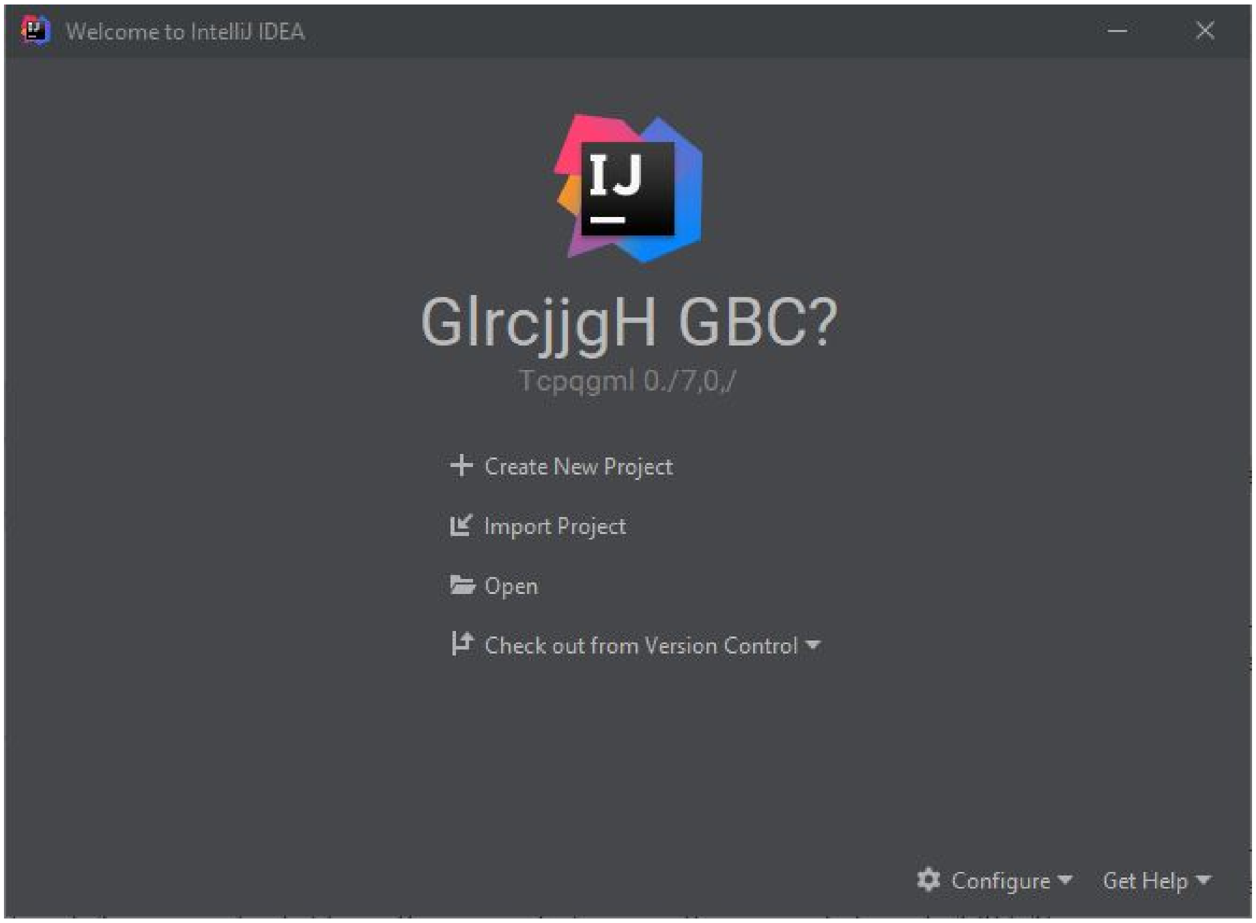
I have implemented the UI actions after buttons were clicked/fired. Based on different user inputs, the reactions are different; hence, there are multiple switch cases to handle such situations.

2 Development Environment

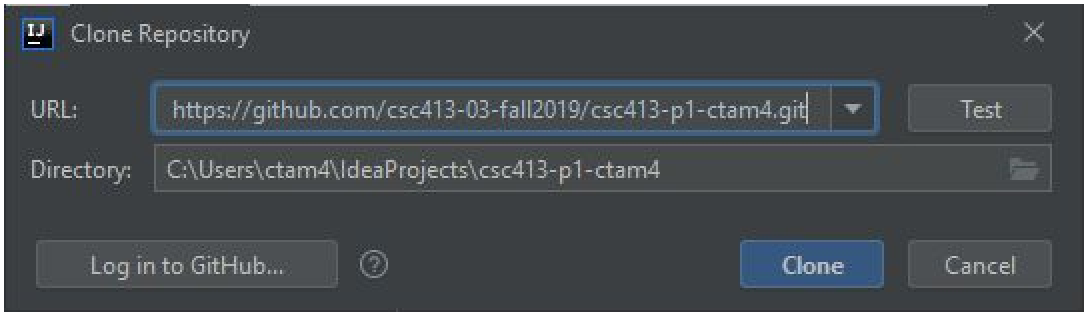
This program has successfully compiled on OpenJDK 11.0.4 in command line and inside IDE IntelliJ IDEA 2019.2.1.

3 How to Build/Import Your Project

Step 1: Open IntelliJ IDEA and select "Check out from Version Control".



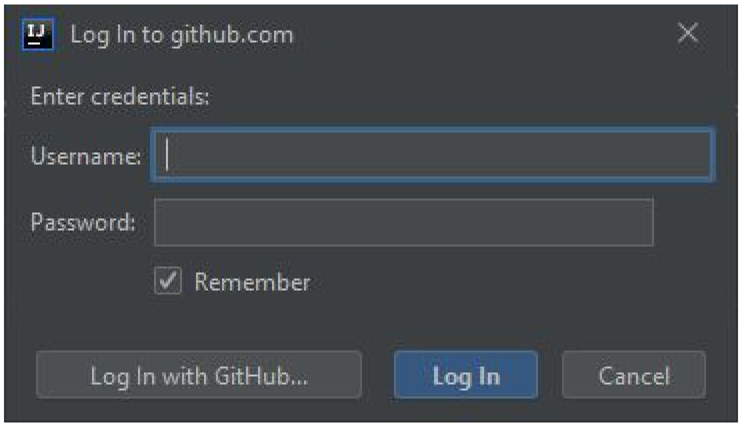
Step 2: Enter "https://github.com/csc413-03-fall2019/csc413-p1-ctam4.git" for URL, select directory path at your preference, and select "Clone".



URL: https://github.com/csc413-03-fall2019/csc413-p1-ctam4.git

Directory: C:\Users\ctam4\IdeaProjects\csc413-p1-ctam4

Step 3: Enter your GitHub username and password and select “Log In”.

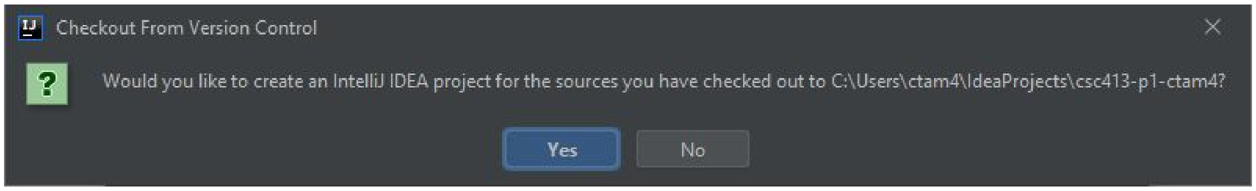

Log In to github.com

Enter credentials:

Username:

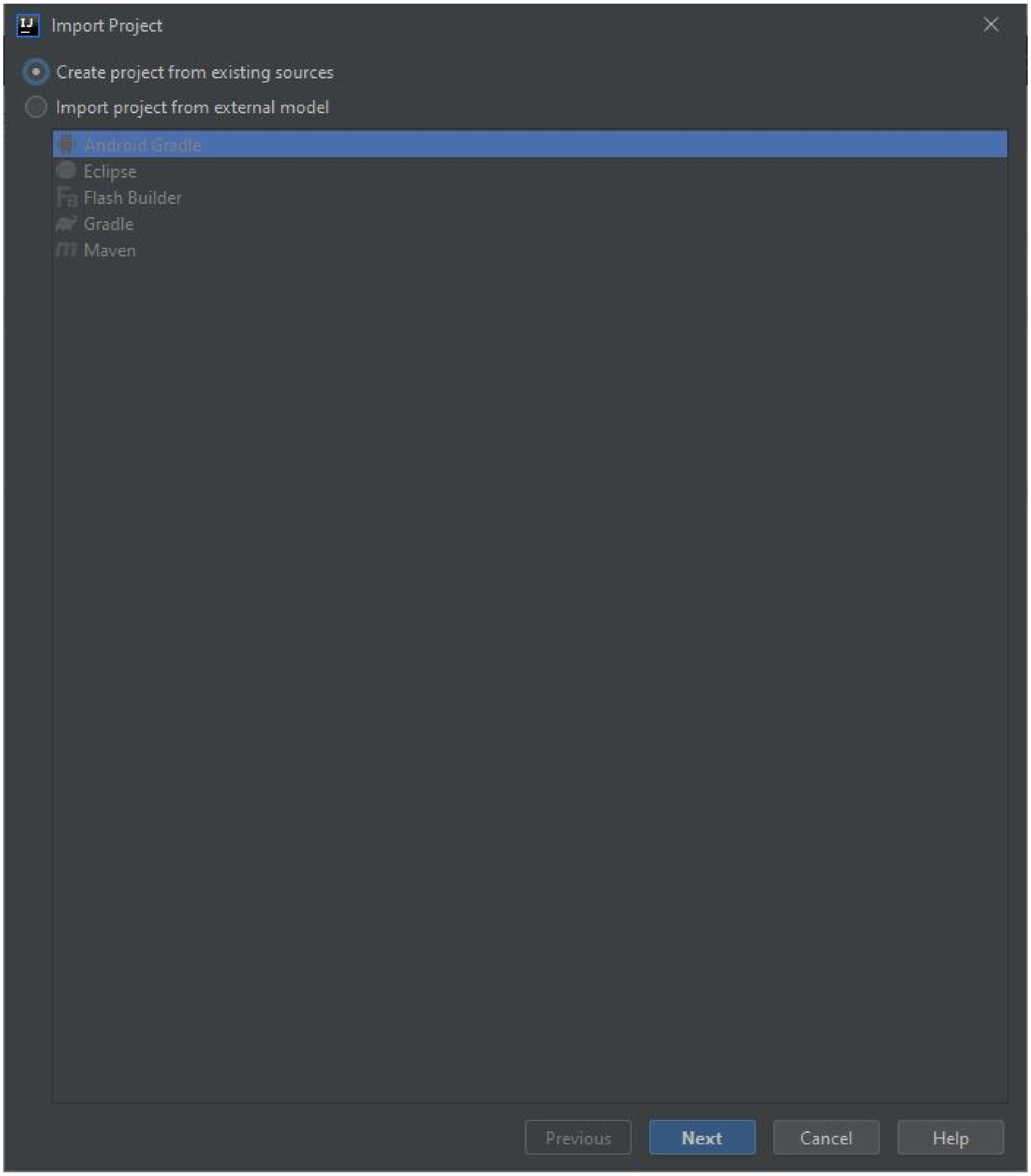
Password:

Step 4: Select “Yes”.

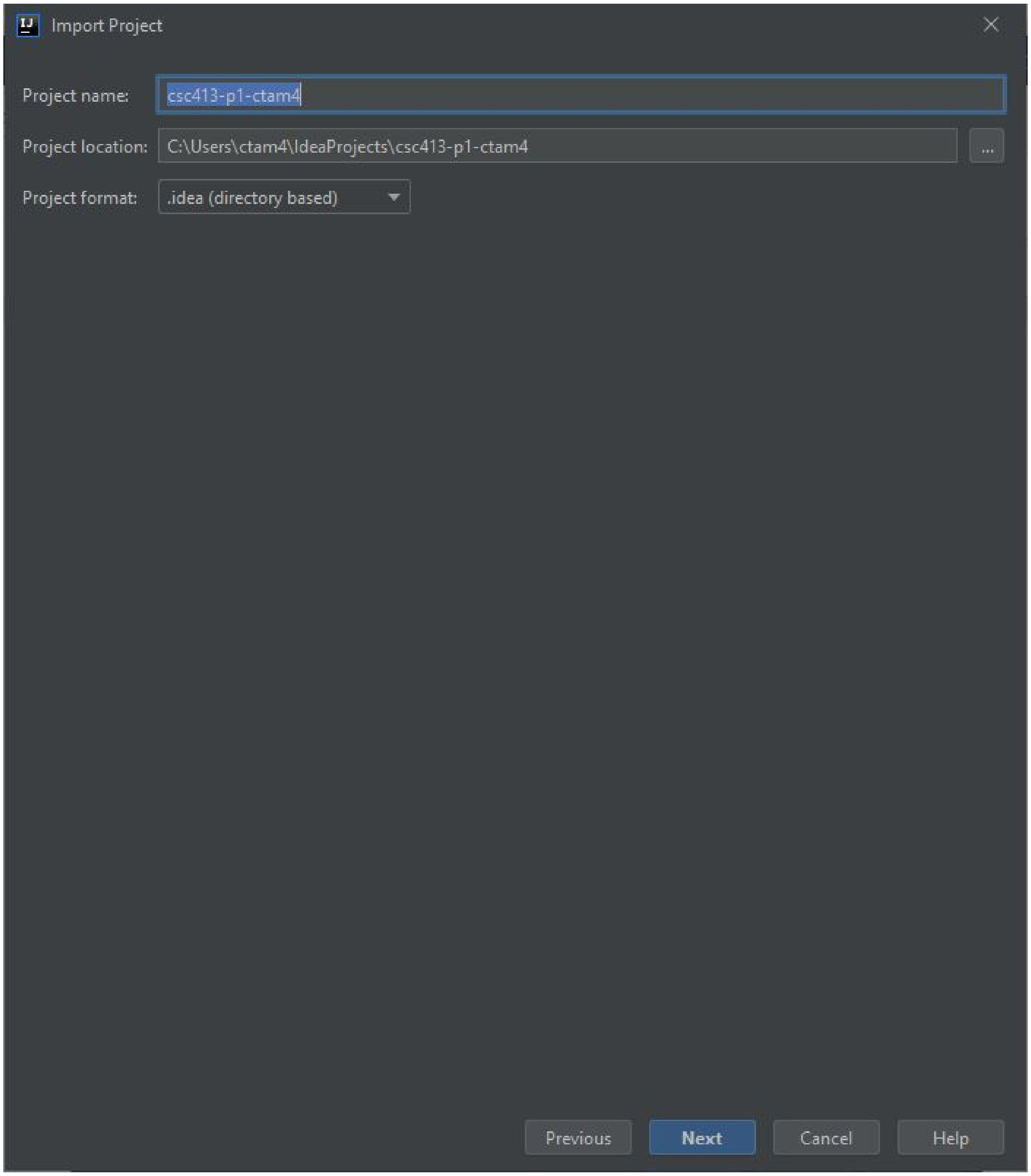

Checkout From Version Control

Would you like to create an IntelliJ IDEA project for the sources you have checked out to C:\Users\ctam4\IdeaProjects\csc413-p1-ctam4?

Step 5: Select “Next”.



Step 6: Select “Next”.

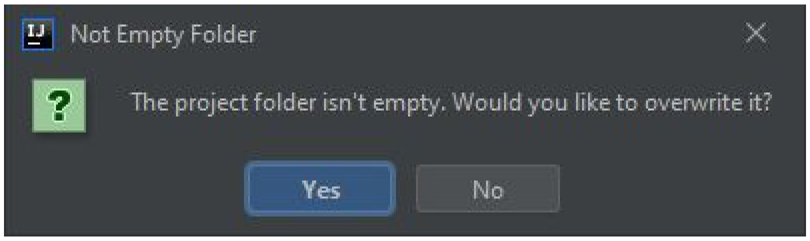

Import Project

Project name: csc413-p1-ctam4

Project location: C:\Users\ctam4\IdeaProjects\csc413-p1-ctam4

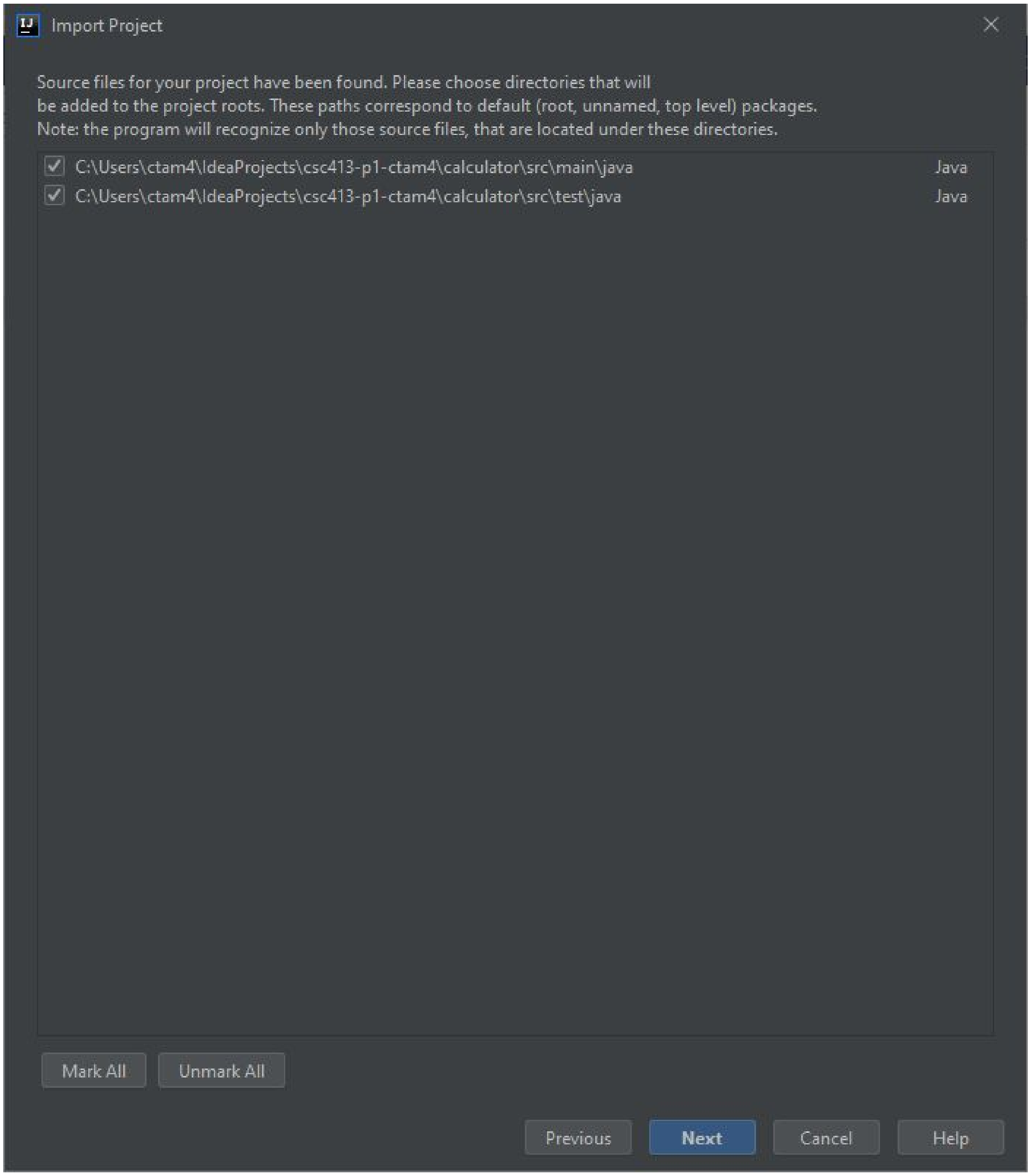
Project format: .idea (directory based)

Step 7: Select “Yes”.


Not Empty Folder

The project folder isn't empty. Would you like to overwrite it?

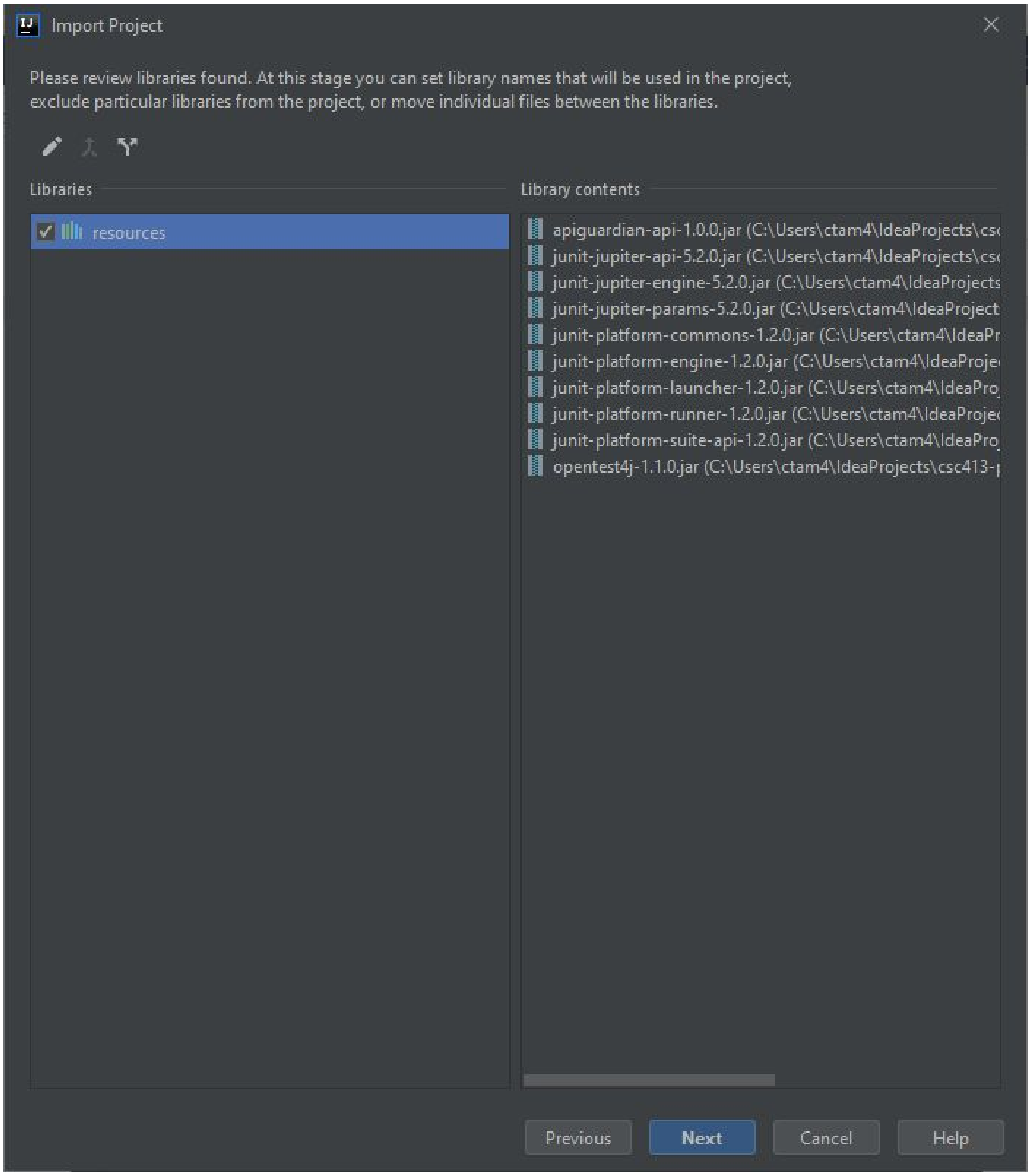
Step 8: Select “Next”.


Import Project

Source files for your project have been found. Please choose directories that will
be added to the project roots. These paths correspond to default (root, unnamed, top level) packages.
Note: the program will recognize only those source files, that are located under these directories.

✓	C:\Users\ctam4\IdeaProjects\csc413-p1-ctam4\calculator\src\main\java	Java
✓	C:\Users\ctam4\IdeaProjects\csc413-p1-ctam4\calculator\src\test\java	Java

Step 9: Select “Next”.


Please review libraries found. At this stage you can set library names that will be used in the project, exclude particular libraries from the project, or move individual files between the libraries.

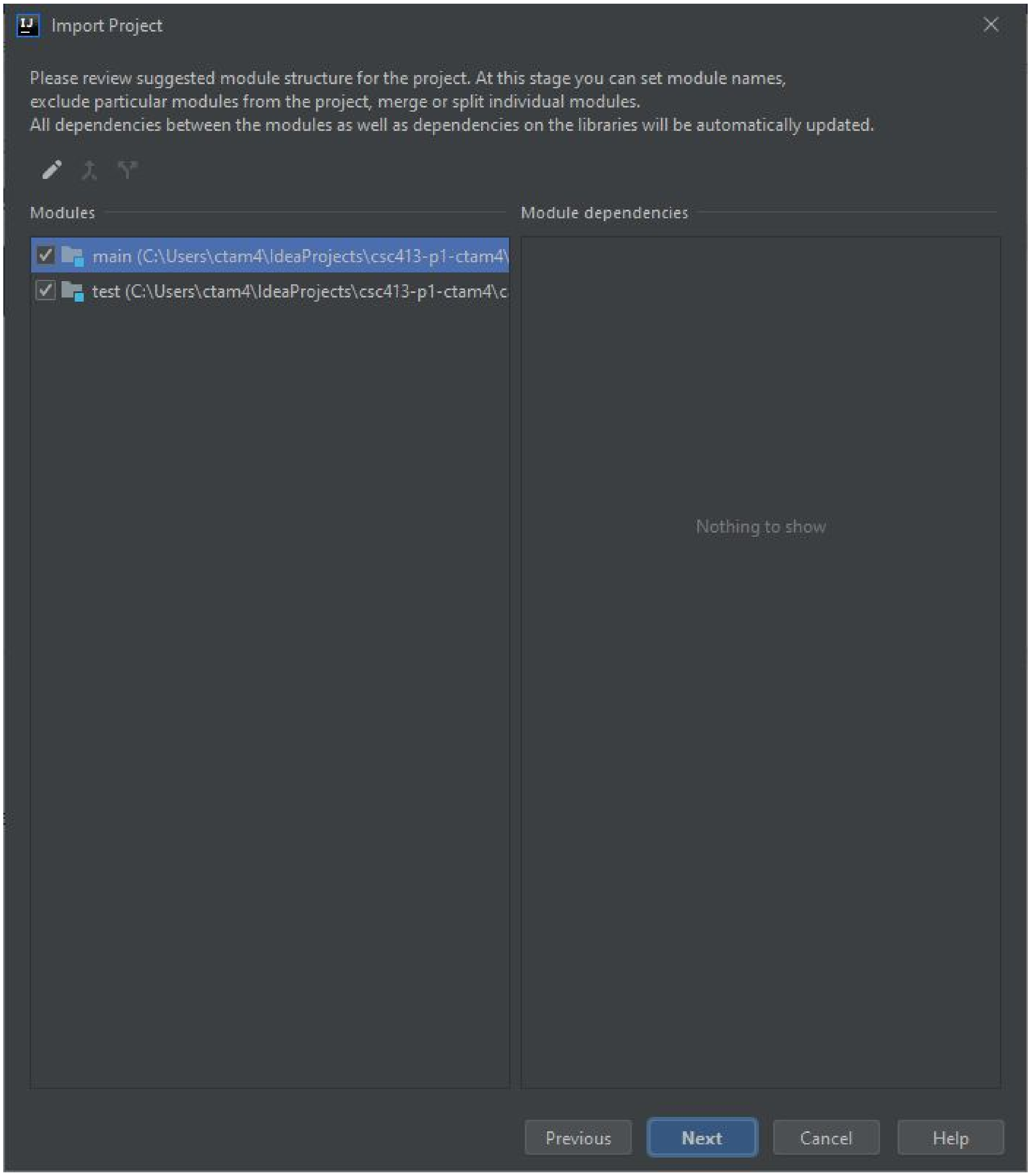
Libraries

✓ resources

Library contents

```
apiguardian-api-1.0.0.jar (C:\Users\ctam4\IdeaProjects\cs
junit-jupiter-api-5.2.0.jar (C:\Users\ctam4\IdeaProjects\cs
junit-jupiter-engine-5.2.0.jar (C:\Users\ctam4\IdeaProjects
junit-jupiter-params-5.2.0.jar (C:\Users\ctam4\IdeaProject
junit-platform-commons-1.2.0.jar (C:\Users\ctam4\IdeaPr
junit-platform-engine-1.2.0.jar (C:\Users\ctam4\IdeaProje
junit-platform-launcher-1.2.0.jar (C:\Users\ctam4\IdeaPro
junit-platform-runner-1.2.0.jar (C:\Users\ctam4\IdeaProje
junit-platform-suite-api-1.2.0.jar (C:\Users\ctam4\IdeaPro
opentest4j-1.1.0.jar (C:\Users\ctam4\IdeaProjects\csc413-
```

Step 10: Select “Next”.


Import Project

Please review suggested module structure for the project. At this stage you can set module names, exclude particular modules from the project, merge or split individual modules.
All dependencies between the modules as well as dependencies on the libraries will be automatically updated.

Modules

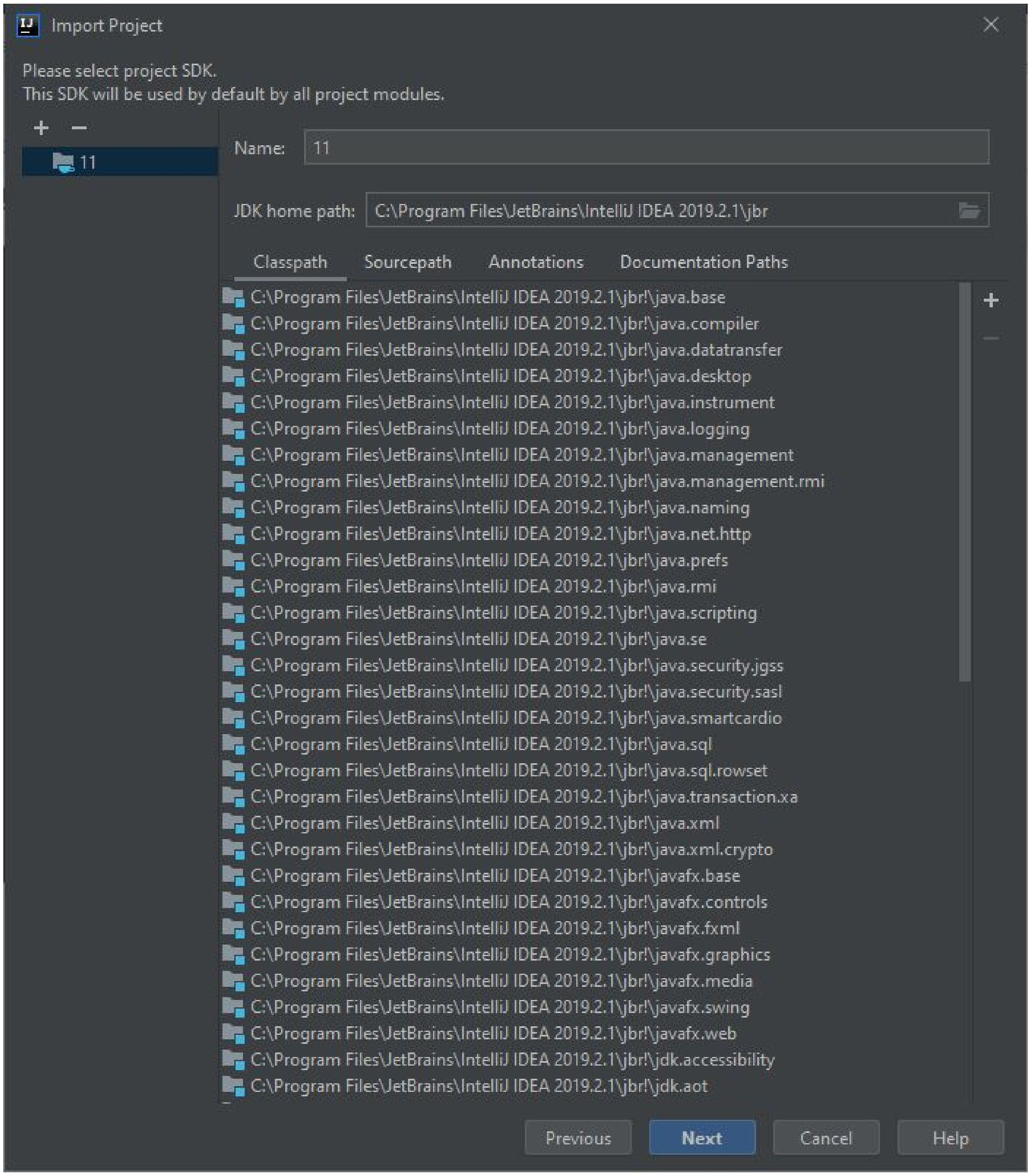
✓ main (C:\Users\ctam4\IdeaProjects\csc413-p1-ctam4\

✓ test (C:\Users\ctam4\IdeaProjects\csc413-p1-ctam4\c

Module dependencies

Nothing to show

Step 11: Select "Next".


Import Project

Please select project SDK.
This SDK will be used by default by all project modules.

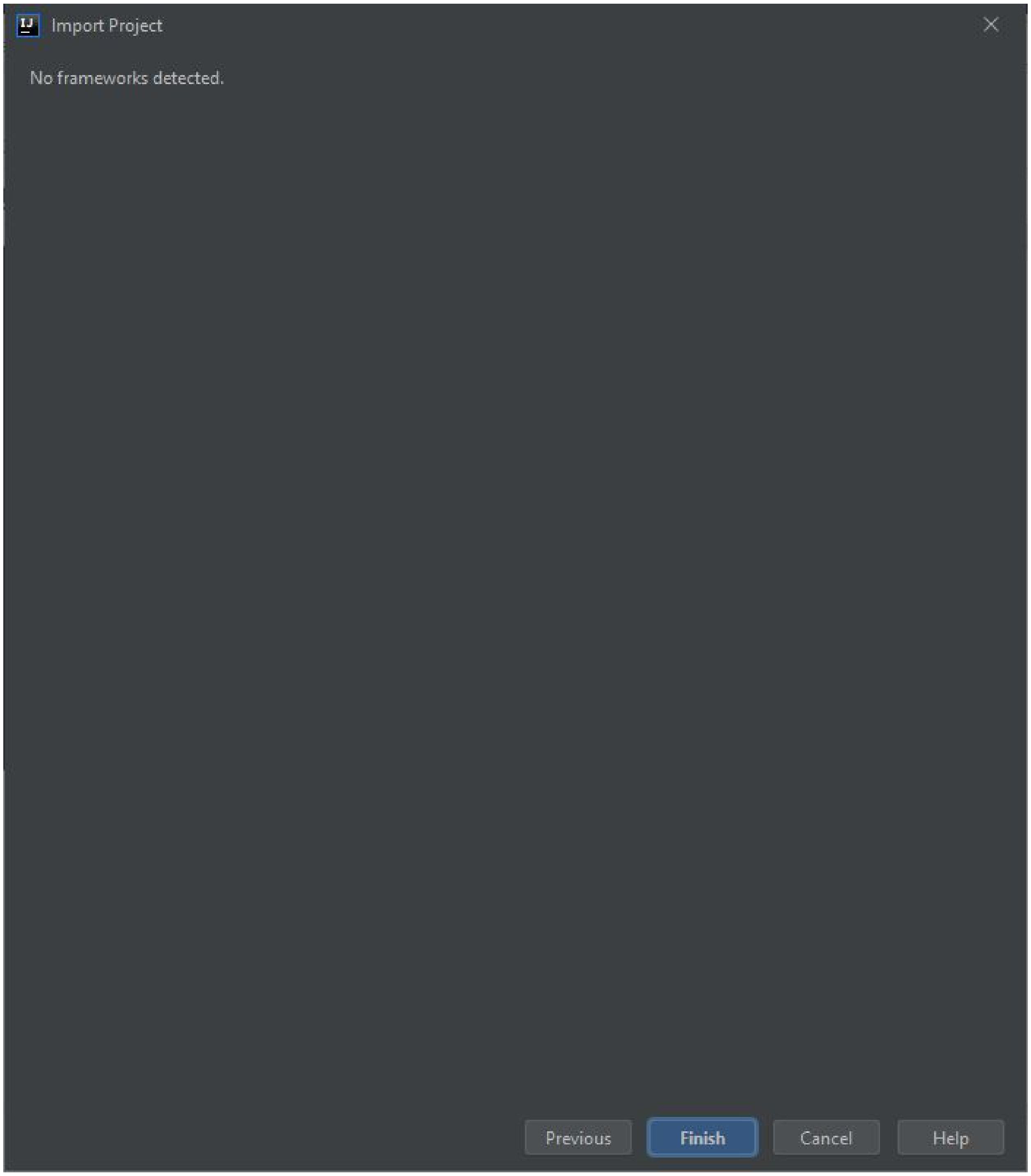
11

Name: 11

JDK home path: C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr

```
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.base
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.compiler
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.datatransfer
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.desktop
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.instrument
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.logging
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.management
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.management.rmi
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.naming
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.net.http
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.prefs
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.rmi
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.scripting
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.se
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.security.jgss
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.security.sasl
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.smartcardio
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.sql
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.sql.rowset
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.transaction.xa
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.xml
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\java.xml.crypto
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\javafx.base
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\javafx.controls
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\javafx.fxml
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\javafx.graphics
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\javafx.media
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\javafx.swing
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\javafx.web
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\jdk.accessibility
C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr!\jdk.aot
```

Step 12: Select “Finish”.

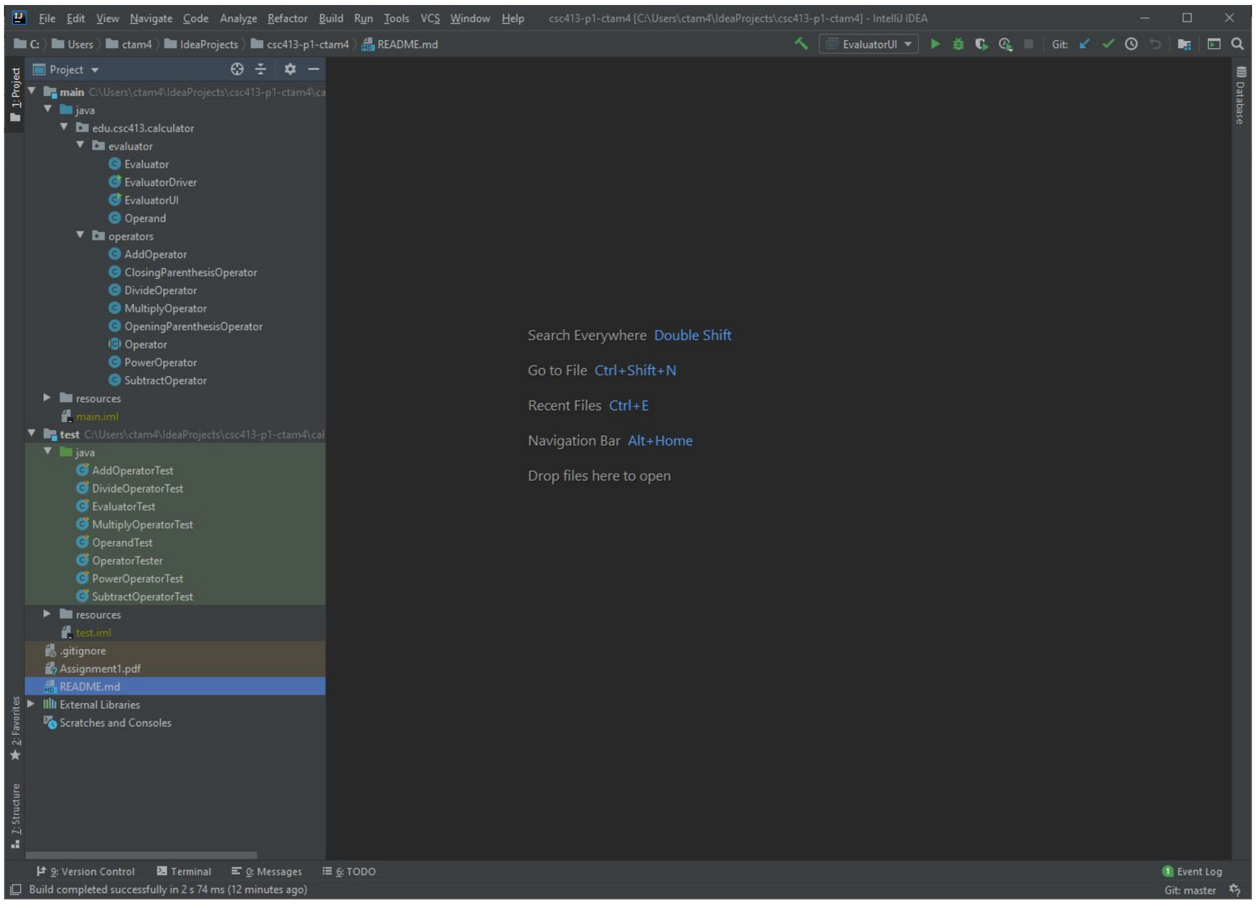


No frameworks detected.

4 How to Run Your Project

Find the java file you want to compile.

For example, if you want to open the GUI, select “EvaluatorUI.java”. Right click and select “Run ‘EvaluatorUI.main()’”. For unit test files, right click on “XXXXXTest” and select “Run ‘XXXXXTest’”.



Build completed successfully in 2 s 74 ms (12 minutes ago)

5 Assumption Made

End-user Environment:

1) JRE 11 or above

Developer Environment:

1) JDK 11 or above
2) IntelliJ IDEA 2019 or above, or other IDE alternatives

Program Functionality:

1) The program only computes the following operations:
    a) Addition
    b) Subtraction
    c) Multiplication
    d) Division
    e) Power (Exponent)
2) The program handles parentheses.
3) End-user cannot enter decimal operands.
4) The program does not handle decimal operands.
5) The program does not output decimal results.

6 Implementation Discussion

6.1 Back-end Design Choices

I have implemented more error checkings at `Operator` and `Operand` classes, and `process()` in `Evaluator`. The program checks for disallowed argument, so when the argument is `null`, it prints out error message on the console and throws `IllegalArgumentException`.

I have added `OpeningParenthesisOperator` and `ClosingParenthesisOperator` classes with `0` return on `priority()` and `null` return on `execute()` since they are placeholders for `Evaluator`.

In `Evaluator`, I have added support for handling negative numbers by putting parentheses around the negative numbers. In short, the end-user needs to input at this format: `(-X)`.

6.2 Front-end Design Choices

When the end-user inputs an operand, the program appends the operand into the expression.

When the end-user inputs an operator, the program runs differently based on different conditions:

1) If the last input was an operand, program appends the operator into the expression.
2) If the last input was an operator, the last operator would be replaced by the new one. This does not apply to parentheses.
3) If the last input was an opening parenthesis, and operator entered is a dash, program appends the operator into the expression. This is the special case for `(-X)`.
4) If there was no last input, nothing happens.

When the end-user inputs an opening parenthesis, the program appends the opening parenthesis into the expression only if the last input was an operator or an opening parenthesis, or there was no last input.

When the end-user inputs a closing parenthesis, the program appends the closing parenthesis into the expression only if the number of opening parenthesis minus closing parenthesis is larger than zero.

When the end-user inputs `CE`, the program runs differently based on different conditions:

1) If the last input was an operand, it deletes the operands no matter how long is the operand until it found an operator.
2) If the last input was an operator, it deletes that operator.
3) If there was no last input, nothing happens.

When the end-user inputs `C`, the program clears the text field.

When the end-user inputs =, the program would not compute results if the numbers of parentheses in the expression do not match or the expression ends with any operator.

Due to program complexity, there are few limitations in the UI:

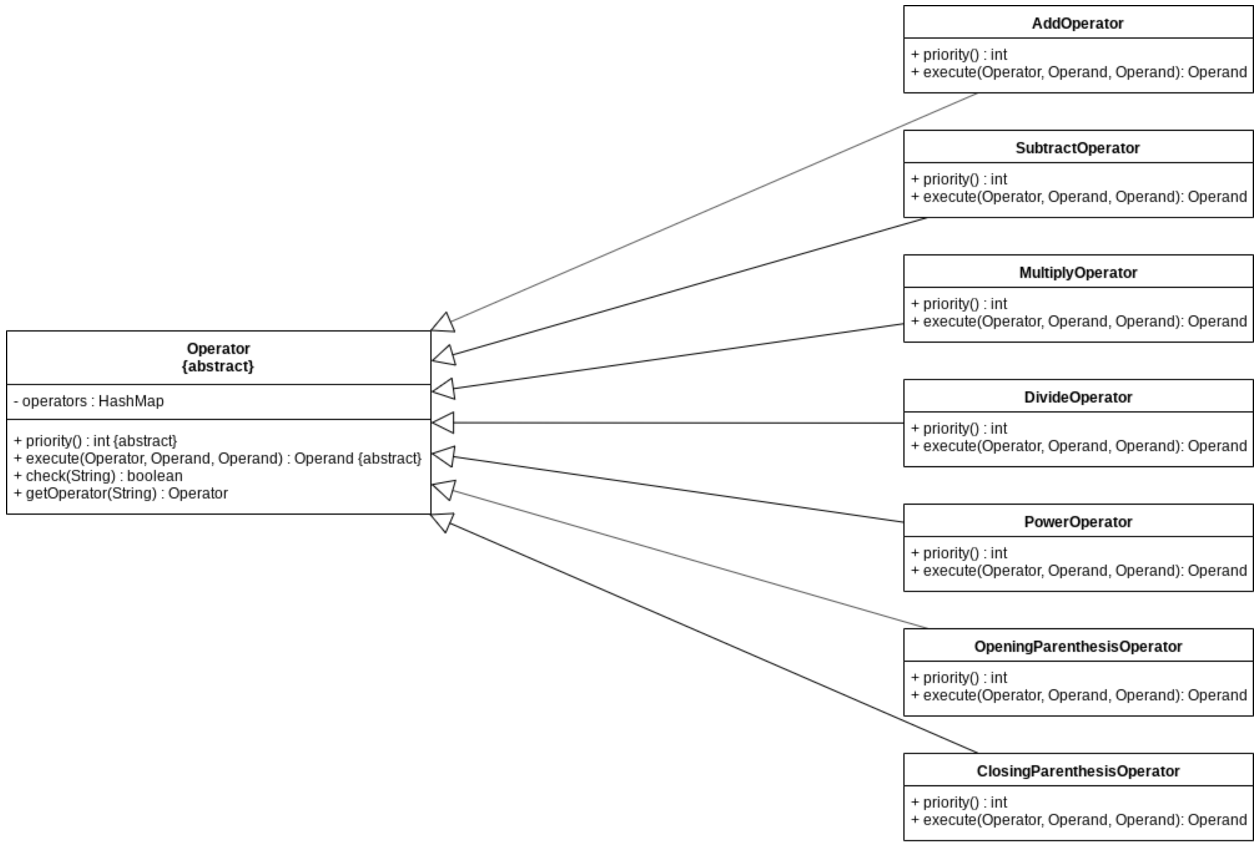
1) If end-user need to enter negative operands, it needs to be have parentheses.
2) If end-user enter 0 after another operand starting with 0, it will accept. No error-checking has implemented. Normally, it would not allow such case.
3) Syntax checking for mathematical expression is limited. User inputs can result in runtime error, and the UI will freeze. End-user needs to rerun the program.

6.3 Class Diagram

`Operator` is an abstract class, and it has one private variable, two public static methods, and two public abstract methods.

The following classes are inherited from `Operator` class: 1) `AddOperator`; 2) `SubtractOperator`; 3) `MultiplyOperator`; 4) `DivideOperator`; 5) `PowerOperator`; 6) `OpeningParenthesisOperator`; and 7) `ClosingParenthesisOperator`.

`OpeningParenthesisOperator` and `ClosingParenthesisOperator` are special cases that use only as placeholders. They have no priority and do not execute any operation.



Operator {abstract}
- operators : HashMap
+ priority() : int {abstract}
+ execute(Operator, Operand, Operand) : Operand {abstract}
+ check(String) : boolean
+ getOperator(String) : Operator
AddOperator
+ priority() : int
+ execute(Operator, Operand, Operand): Operand
SubtractOperator
+ priority() : int
+ execute(Operator, Operand, Operand): Operand
MultiplyOperator
+ priority() : int
+ execute(Operator, Operand, Operand): Operand
DivideOperator
+ priority() : int
+ execute(Operator, Operand, Operand): Operand
PowerOperator
+ priority() : int
+ execute(Operator, Operand, Operand): Operand
OpeningParenthesisOperator
+ priority() : int
+ execute(Operator, Operand, Operand): Operand
ClosingParenthesisOperator
+ priority() : int
+ execute(Operator, Operand, Operand): Operand

7 Project Reflection

The coding part of this project is not difficult. It is complexity of the program design. This project requires to use the OOP approach in Java, but this is not friendly to people who learned OOP in a different language and never practiced in Java.

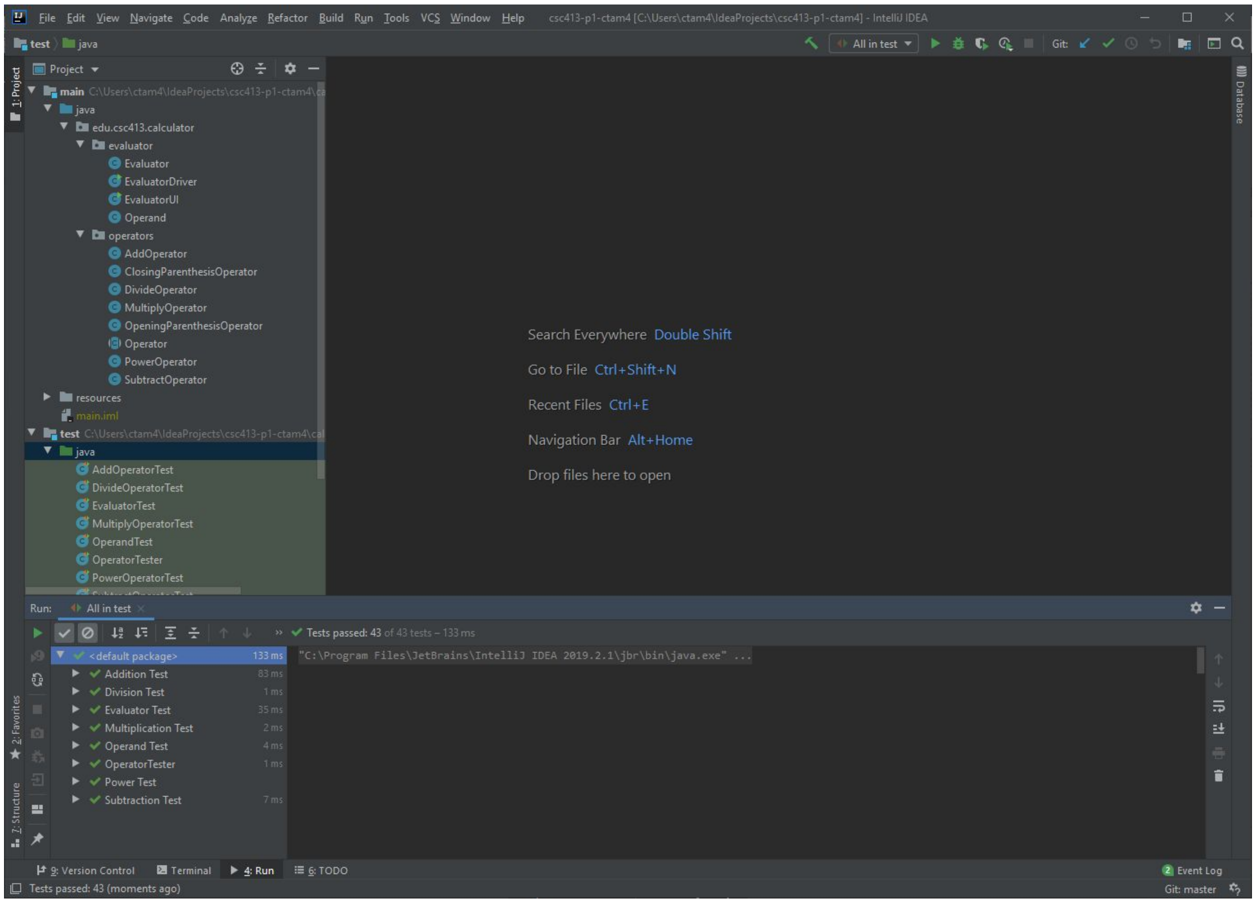
I have been using version control for years; therefore, I am familiar with Git, Subversion (SVN) and other version control systems. Students who are the first time using Git probably feel confusing for some operations, such as amending commits, resetting branch or doing a hard push. They are clueless about any of these. They know nothing about version control.

Requirements listed are descriptive, but it is hard to understand the algorithms without any test cases or activity diagrams. It took me some time to figure out how the algorithm works. Project managers usually just provide the requirements with flexibility on the algorithm implementation (like what is the input and output) or continue development from a previous project, which includes all source code and some documentation. It is rare to be given the descriptive version of the algorithm.

The most difficult part of the project is writing this documentation. This project is slightly too small for a need to write documentation. There are only five main classes and seven simple subclasses. Because of the simple program structure, there is not much stuff to write about.

8 Project Conclusion/Results

All tests are passed.



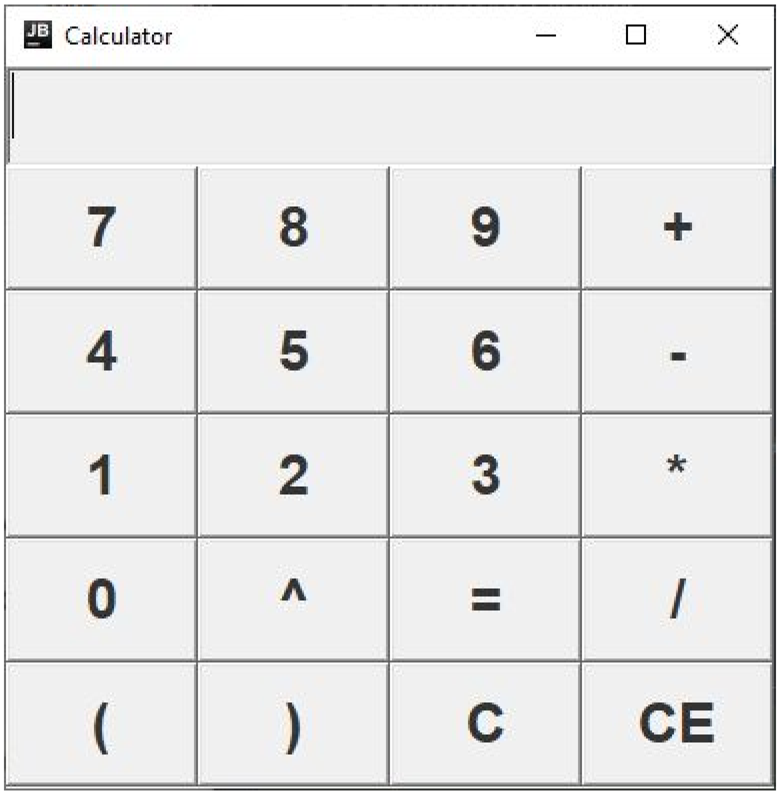
```
Tests passed: 43 of 43 tests – 133 ms

<default package>    133 ms
  Addition Test       83 ms
  Division Test        1 ms
  Evaluator Test      35 ms
  Multiplication Test  2 ms
  Operand Test         4 ms
  OperatorTester       1 ms
  Power Test
  Subtraction Test     7 ms

"C:\Program Files\JetBrains\IntelliJ IDEA 2019.2.1\jbr\bin\java.exe" ...

Tests passed: 43 (moments ago)
```

UI has been completed with some degree of incorrect syntax prevention. The template given does not support handling negative numbers; hence, I made changes for this.


Calculator
7
8
9
+
4
5
6
-
1
2
3
*
0
^
=
/
(
)
C
CE