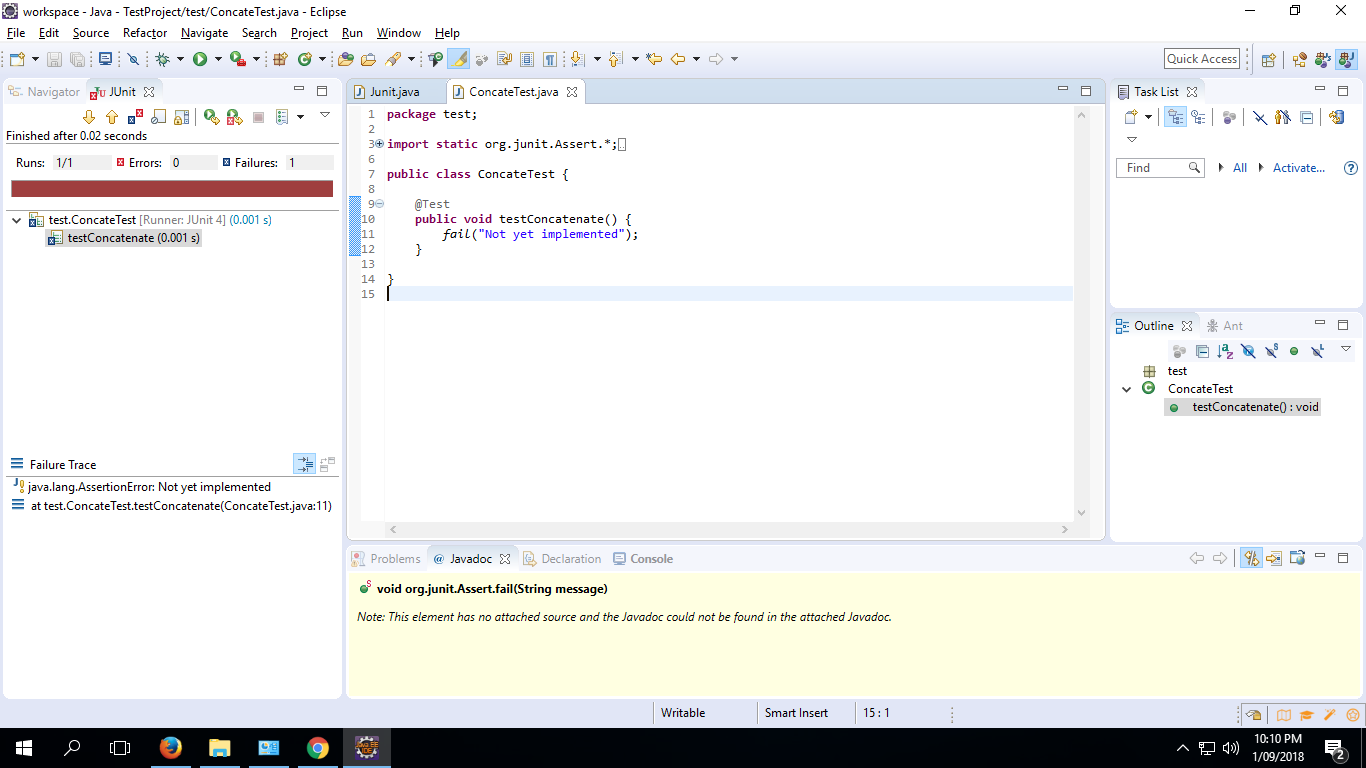
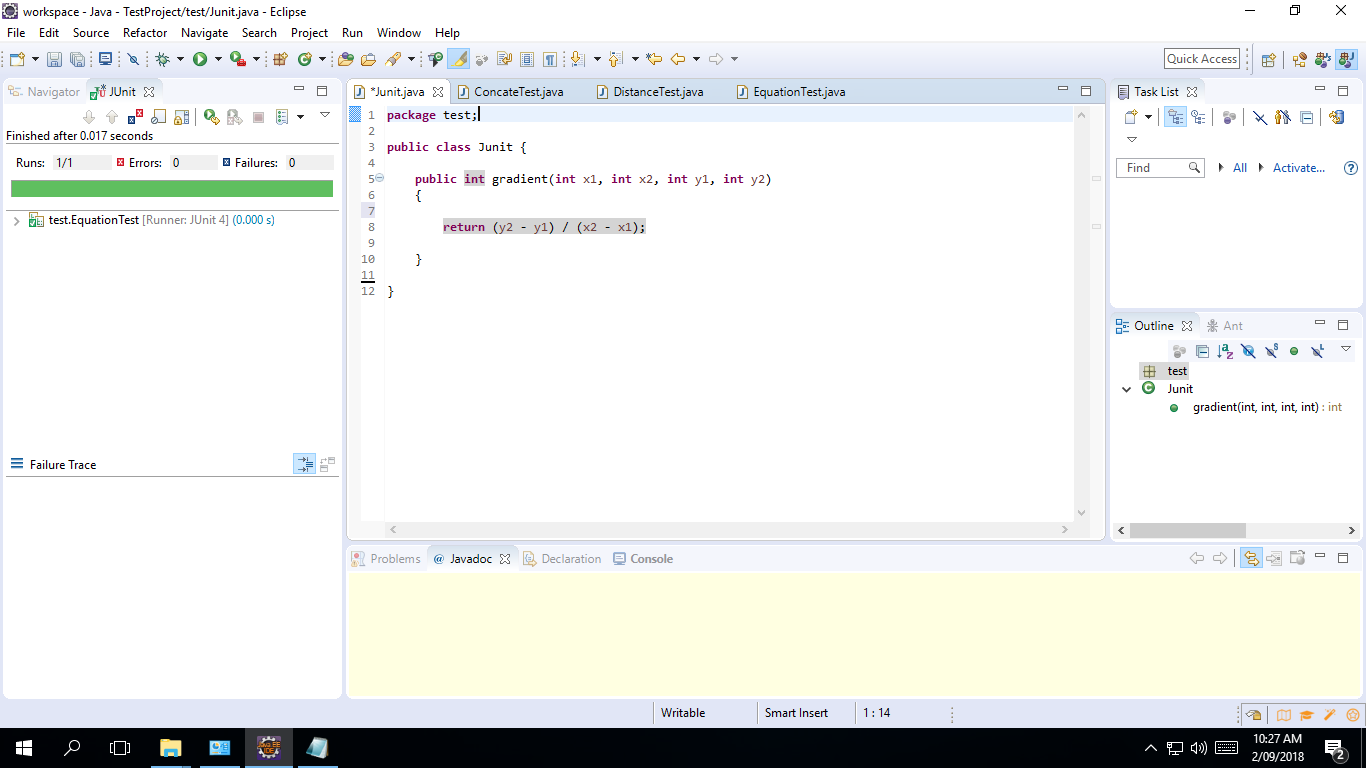
**Unit Test and Refactoring Questions**

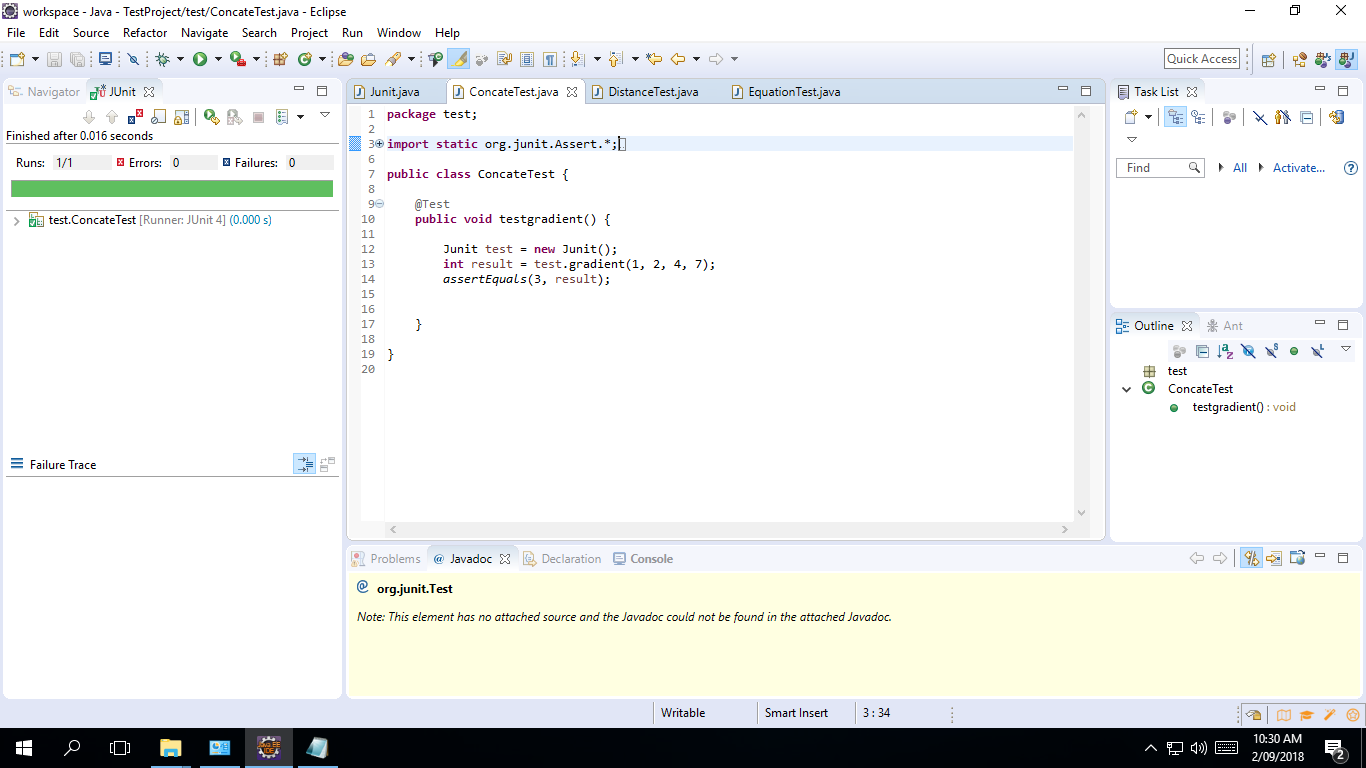
1. Write a program using TDD to do the following (15 marks):

a. Step one: This is the test case using for the function of gradient and failed.

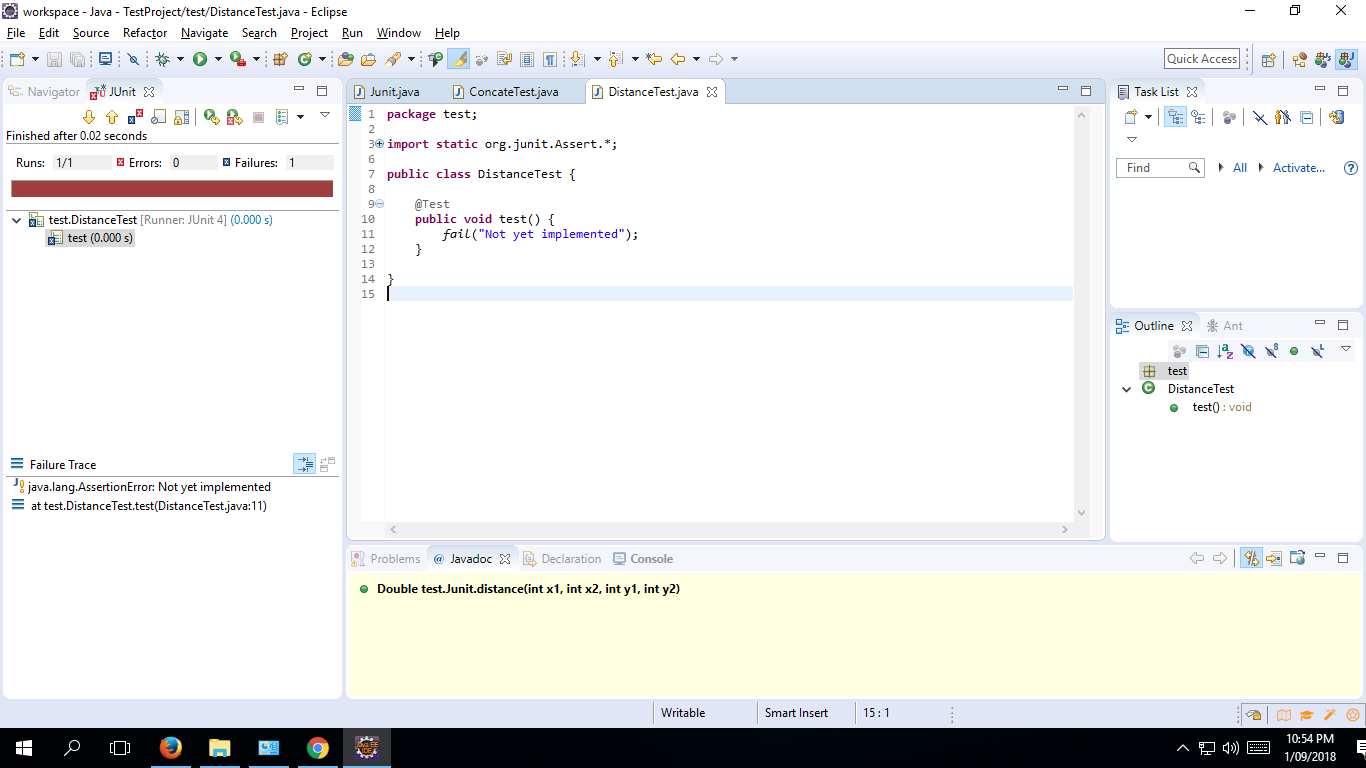


Step two: This is the gradient test case which is successful when input right characters and results.

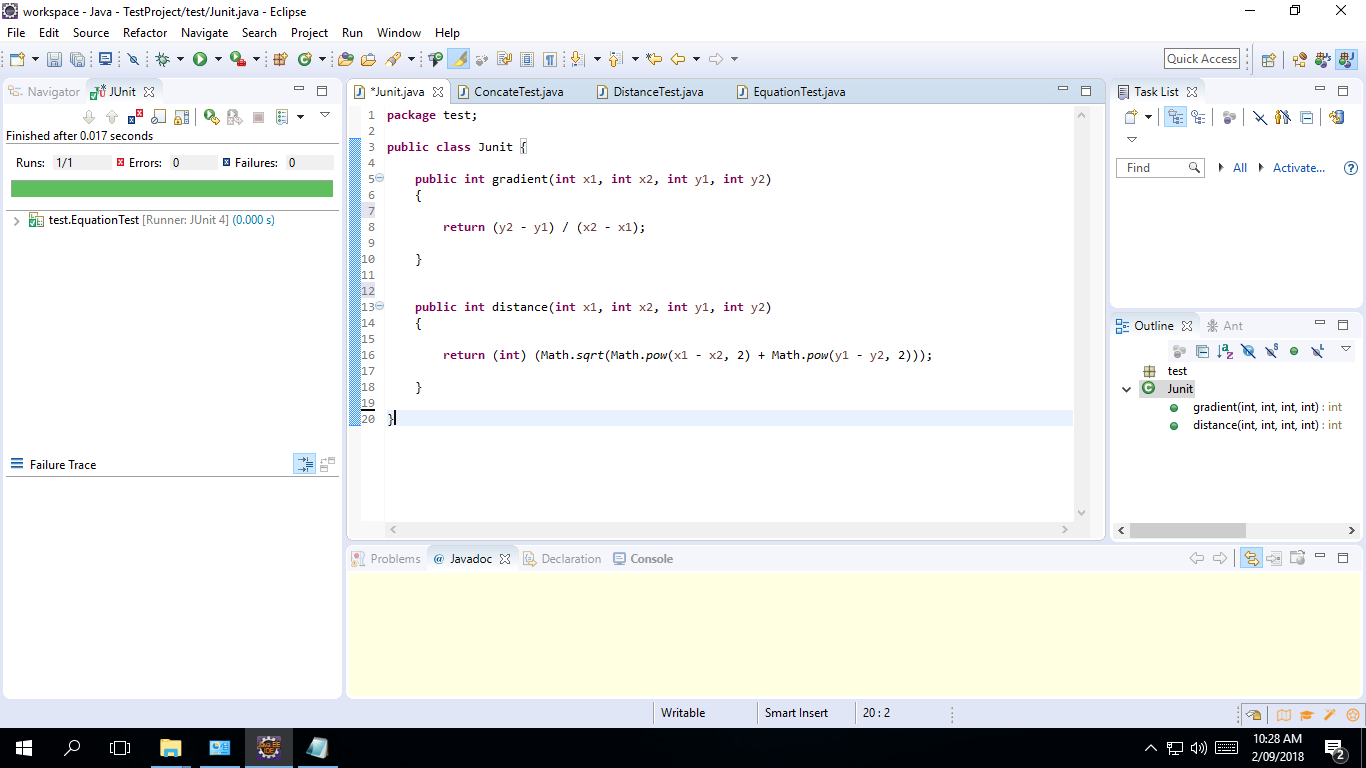


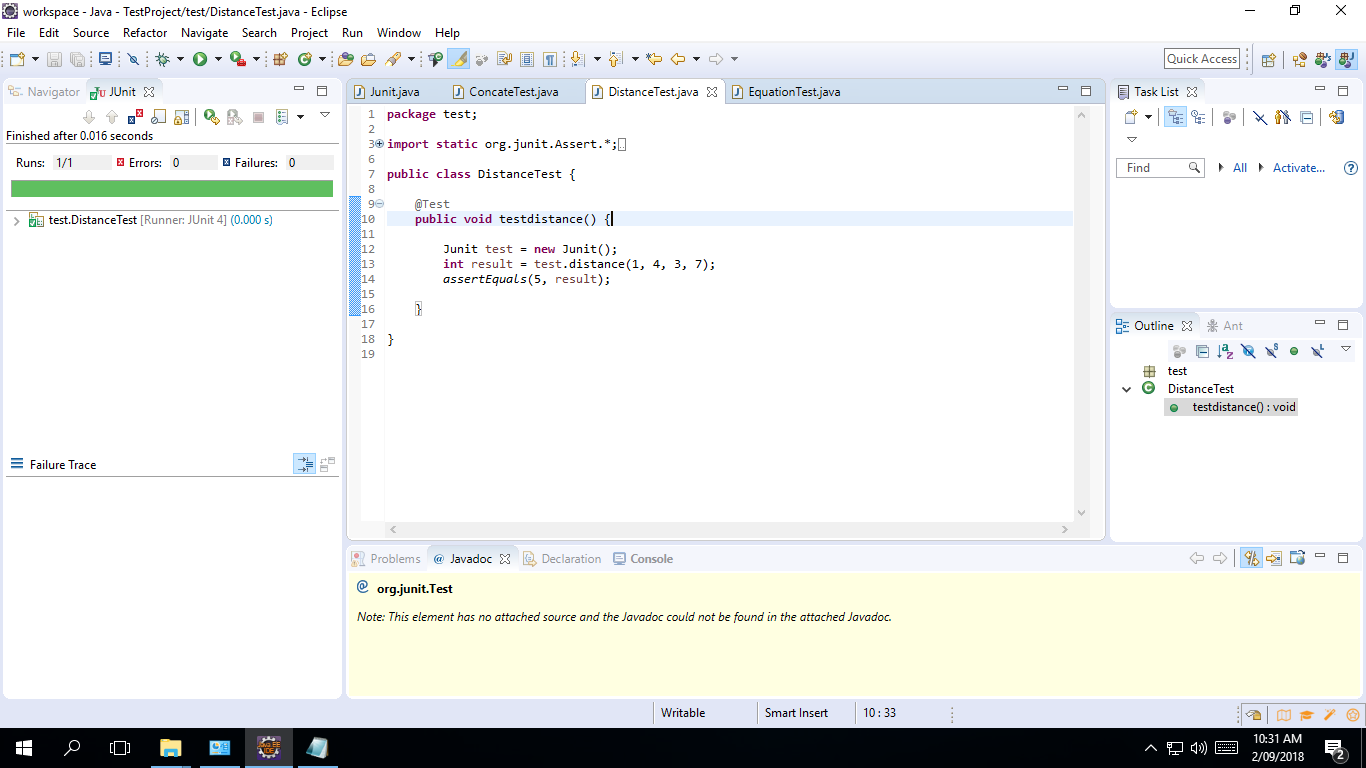


Step Three: This is the second function test case about distance and it is failed.

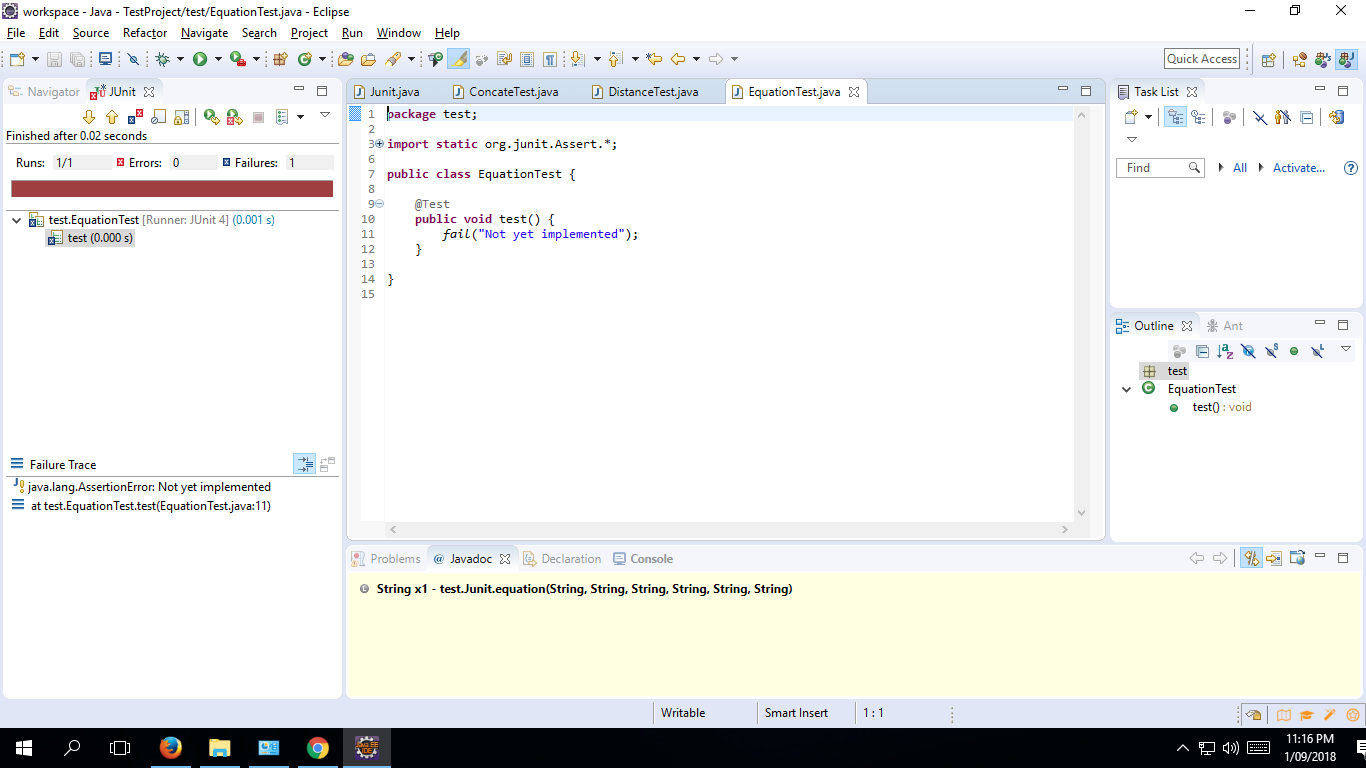


Step Four: This is the second function test case about distance and it is successful.

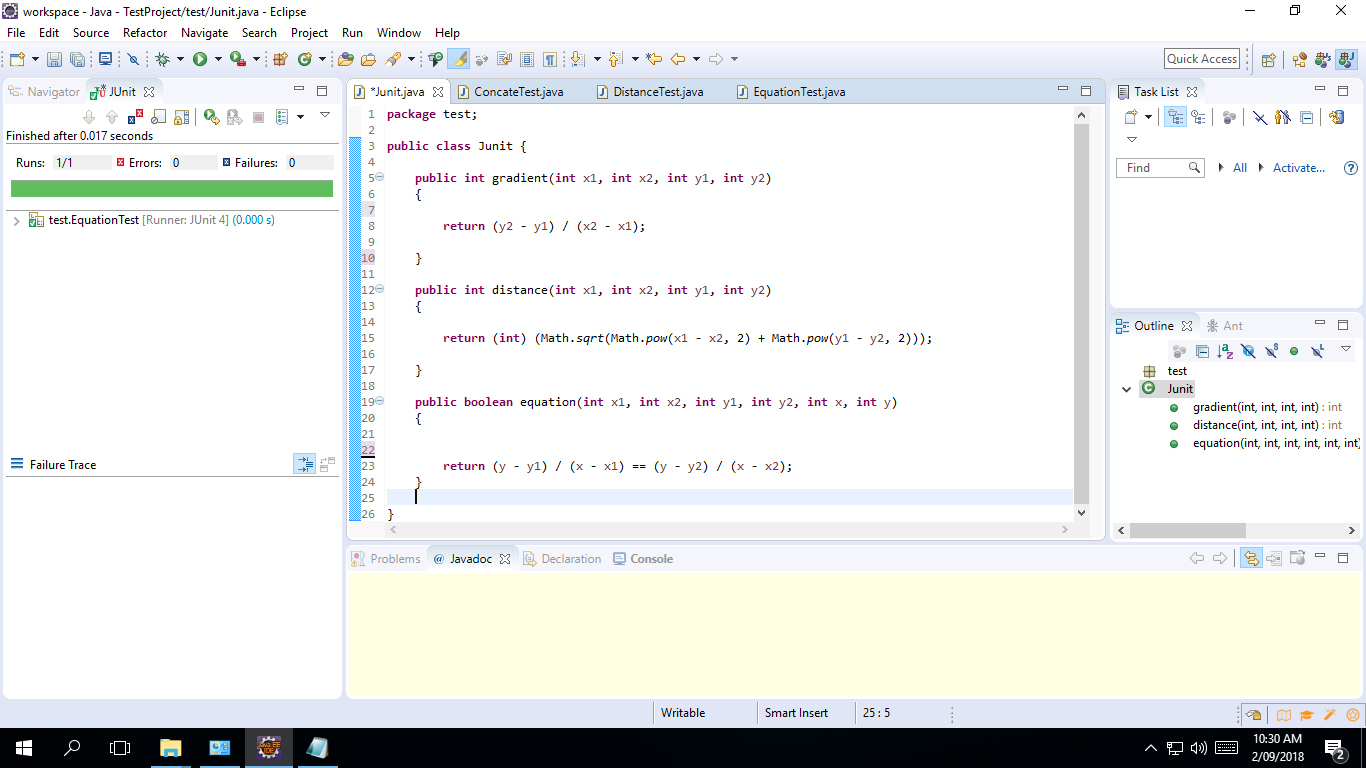


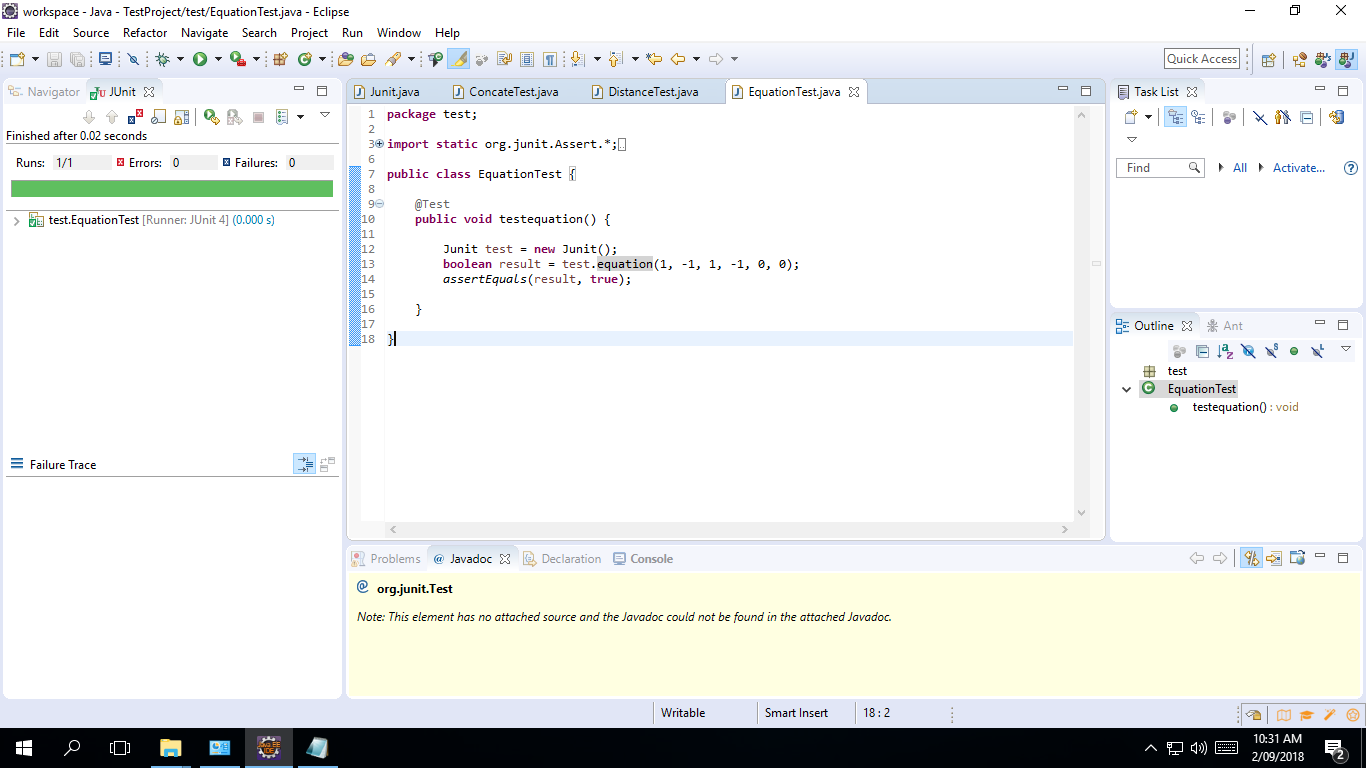


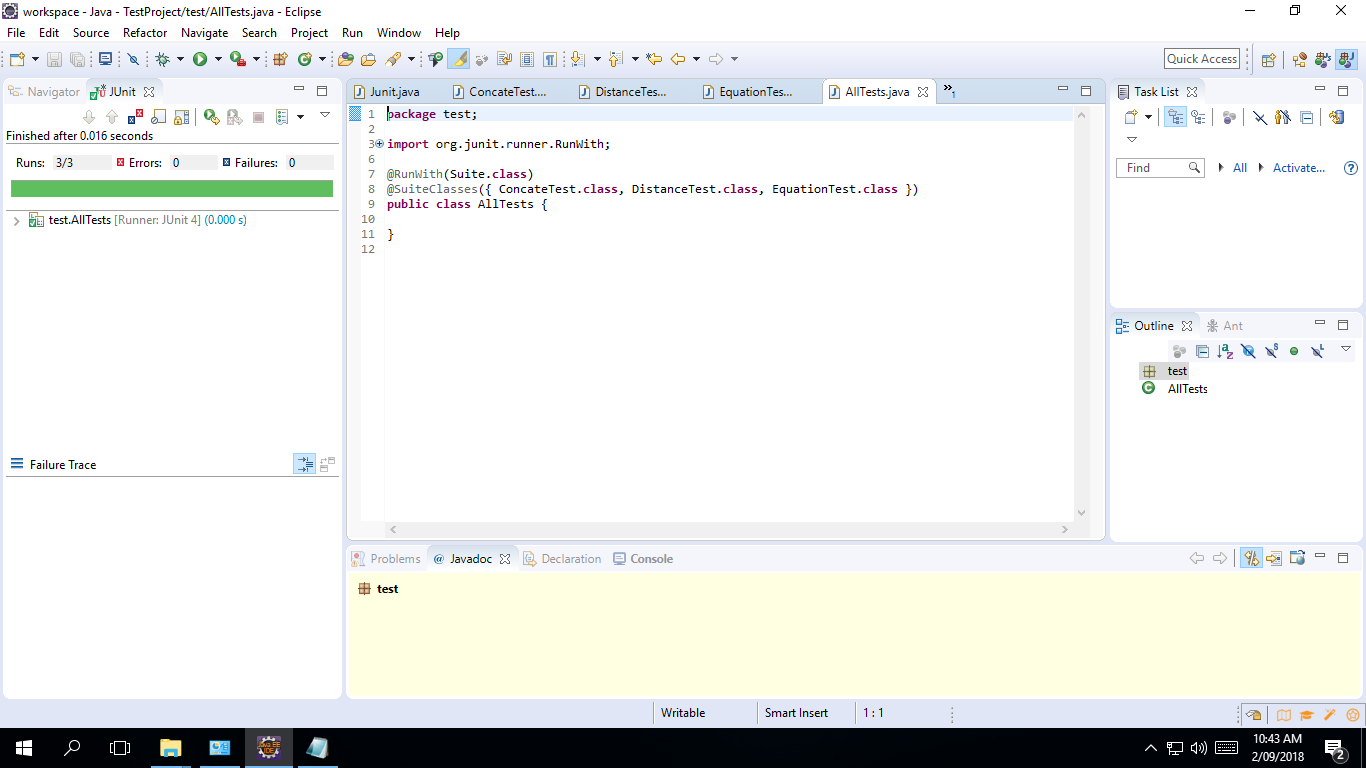
Step Five: This is the Third function test case about equation and it is failed.



Step Six: This is the Third function test case about equation and it goes successfully.

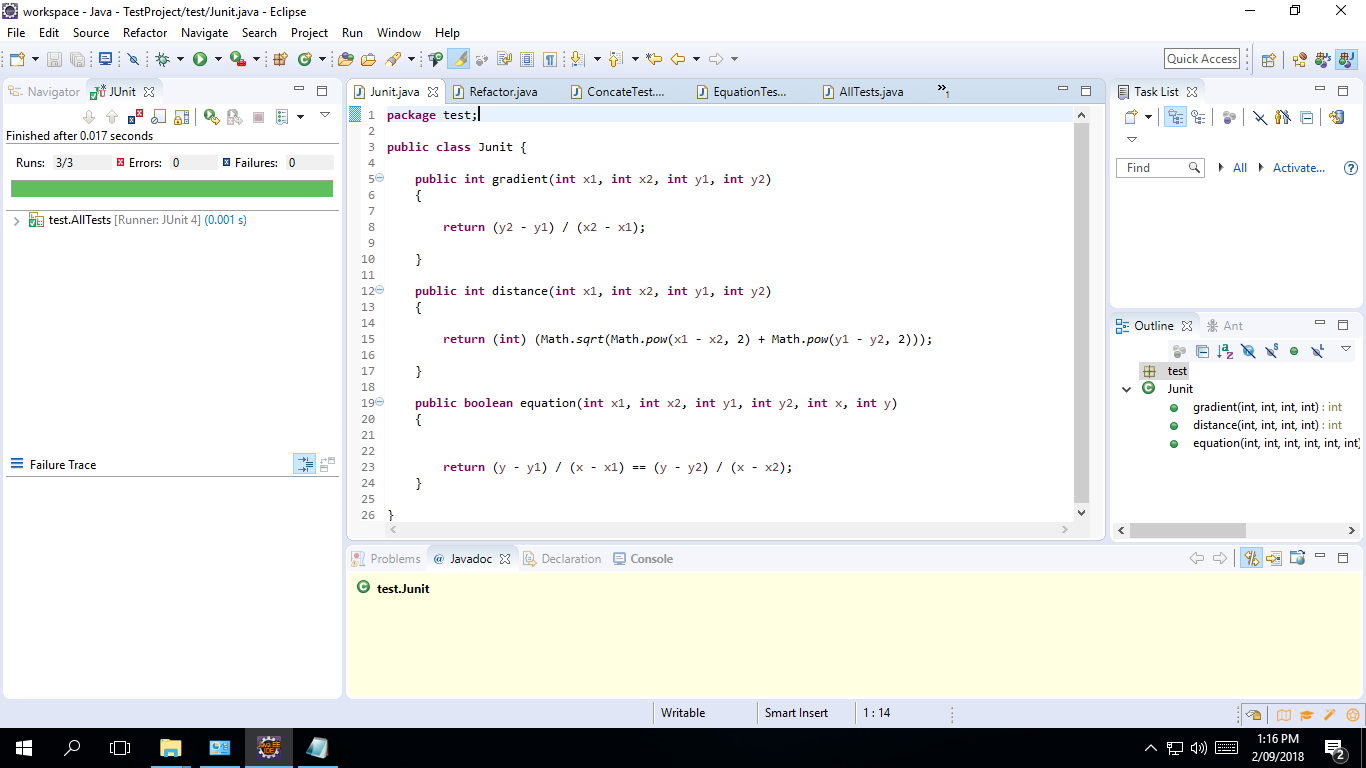




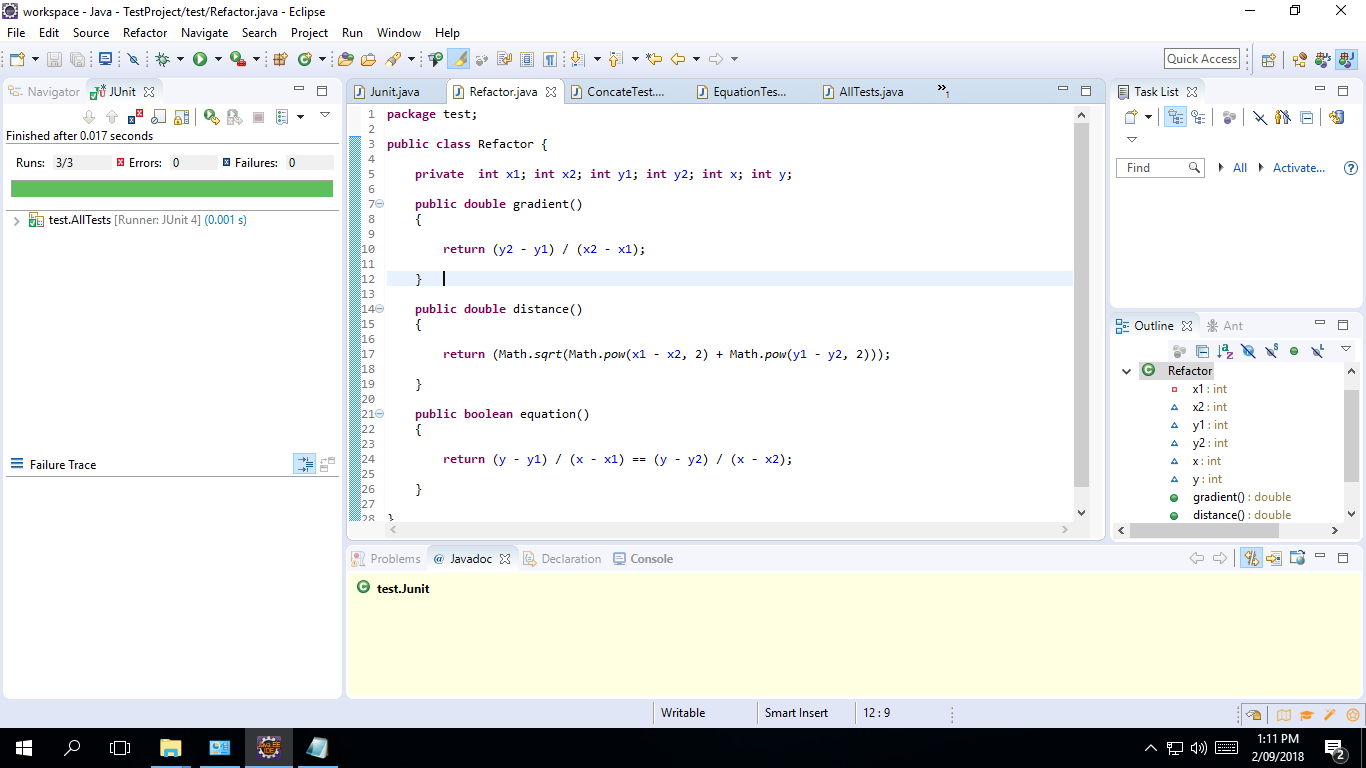


b. Refactor your code

The original program

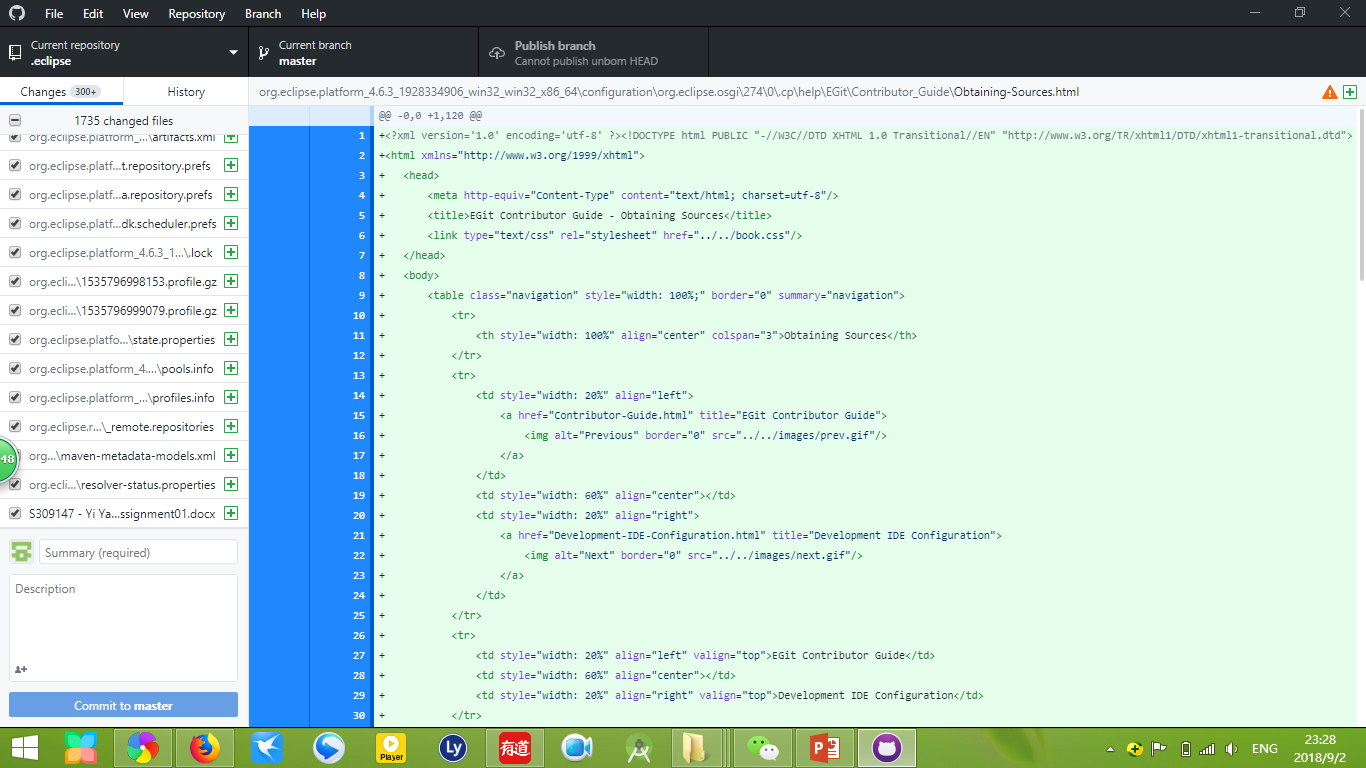


Program by refactor



By refactor, the variable is defined as a macro variable, and only the function type and the return value need to be defined in the sub-function. To do this, the program looks both concise and clear, and it doesn't go wrong when the variable is called and assigned, which is easy to see the importance of refactor.

c. Create a Git directory for your assignment (including word or pdf documents and programming code)



<https://github.com/GroupSurge/.eclipse.git>

d. Capture screenshots and write briefly the steps you have taken to create this program.

The first step: create one class called ‘Junit’ and create one test called ‘ConcateTest’ which is used to test the function about gradient. As the weld code in the test case – ‘ConcateTest’ at the beginning, it goes failed obviously. By modify the correct code in the test case and fulfil the first function in ‘Junit’, it goes successfully. The second step, the third, the fourth and the fifth step, to test the functions about distance and equation, to create the test cases called ‘DistanceTest’ and ‘EquationTest’, they are all failed at the beginning because there don’t have the functions relevant codes. By fulfilled the correct codes in the test cases and functions in ‘Junit’, then they all go toward the succussed. For more details, please refer to the screenshots and answer steps in question a.

2. Code smell can give indications that there is some issue with the codes and can be solved by refactoring. Identify 5 issues and their related code smells. Include solutions to fix up the issues. (5 marks)

The code smell may indicate that there are some problems with the codes, which can be solved by refactoring, analysing the code to determine the risks and benefits of the refactoring. If the risk is too high, time is not enough, or refactoring does not bring enough benefits to the project, don't do it. However, some code problems can indeed be solved by code refactoring to reduce costs. Here are the five code smells that can be solved by code refactoring:

Issue 1: Duplicated Code. When programmers can see the same code structure in multiple places, you need to find a way to unify them. You can make the program structure look better by executing the ‘EXTRACT METHOD’ and calling the code from two places.

Issue 2: Long Method. The longer the program, the more difficult it is to understand, and the more time-consuming it is not conducive to code refactoring. Solution: Execute the ‘EXTRACT METHOD’, Decompose the condition or Replace the temporary with a query.

Issue 3: Large Class. When a class tries to comprise too many functions, it often shows lots of instance variables and complex functions in a large class. When other classes or programs need to use some functions, it is not convenient to call, but need to write this again. Partly creates redundancy. Solution: Execute ‘EXTRACT CLASS’ or ‘SUBCLASS’.

Issue 4: Shotgun Surgery. This happens when you make some kind of change, you have to make a lot of small changes to many different classes. Solution: Executing the ‘MOVE METHOD / FIELD’ or ‘INLINE Class’ combines a whole bunch of behaviours.

Issue 5: Switch Statements. They are usually scattered throughout the program. If you add or remove one clause in a switch, you usually must also find and fix the other clauses. Solution: Use the ‘EXTRACT METHOD’ to extract the switch statement and then use the ‘MOVE METHOD’ to put it into a class that requires polymorphism.