**3.1 Task 01: Terminology Illustration**

Please READ [junit.pdf](https://favour-link.feishu.cn/file/boxcnFyJVx8ZGtO73ataKo7zmMh) and illustrate the following Terminologies about Unit Test:

1. What’s test suite?

* A collection of test cases

1. What’s test case?

* Tests the response of a single method to a particular set if inputs

1. What’s unit test?

* A test of the smallest element of code we can sensibly test, usually a single class

1. What’s test fixture?

* The environment in which a test is run

1. **Annotations**. JUnit 4.0 uses **annotations** rather than special names for setting up, tearing down and testing. Please list the mainly used **annotations** in JUnit 4.0.

* @Before, @After, @BeforeClass, @AfterClass, @Test

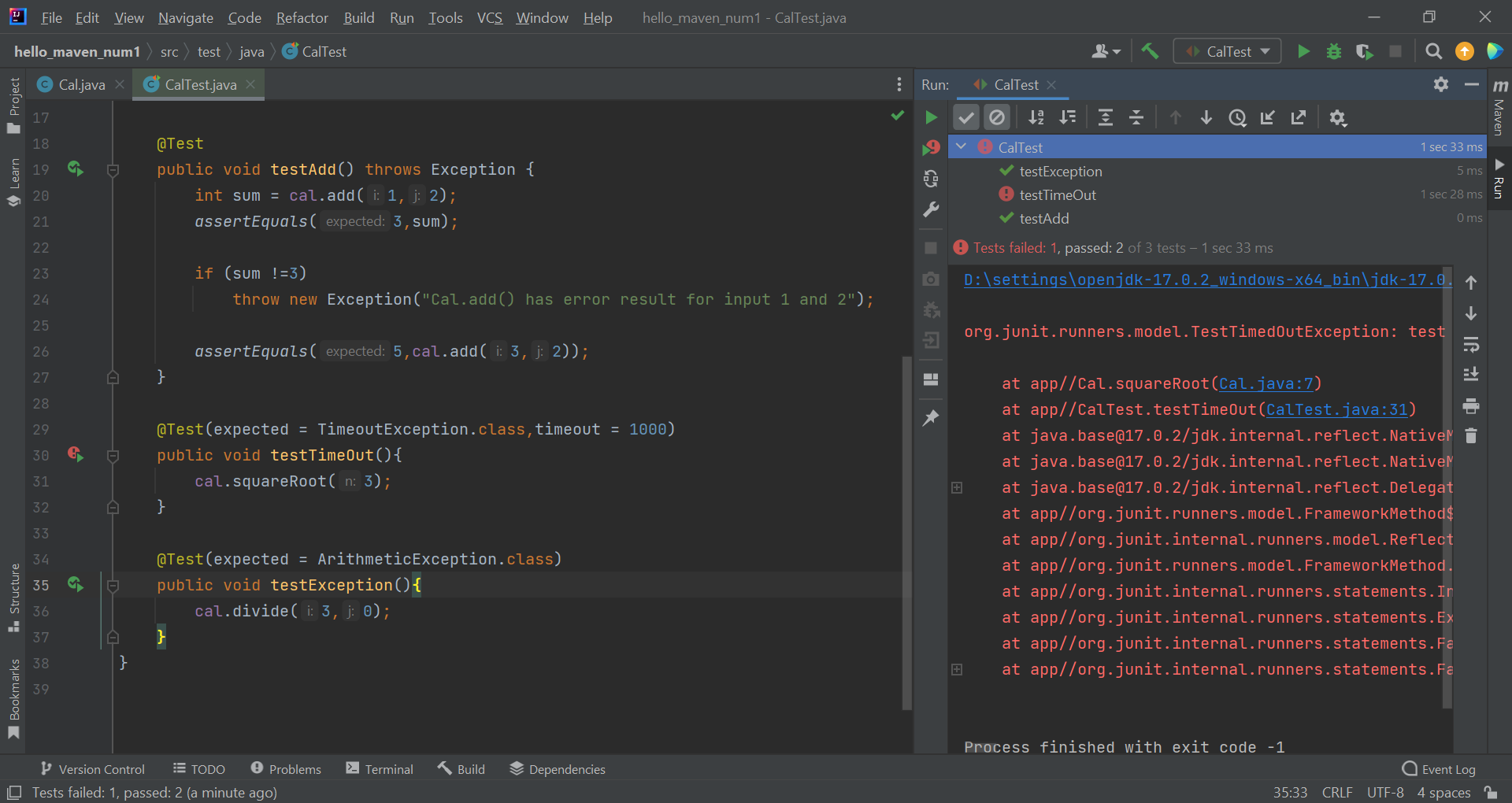
1. **Assert Statement**. There are two forms of the assert statement. Please illustrate what they are.

* Assert *Boolean\_condition ;*
* Assert *Boolean\_condition : error\_message;*

Both forms throw an AssertionFailedError if the *Boolean\_condition is false.* the second form, with an *explicit error\_message,* is seldom necessary

**3.2 Task 02: Config JUnit 4/5 in a Maven Project**

1. Please read [*Create Your Own Maven Project*](https://favour-link.feishu.cn/docs/doccnpJUpcabopOtxYNVyMaf7l7#BFKIUo) carefully. According to the instructions and steps illustrated in this file, do the following subtasks:
   1. Configure your own maven project by adding JUnit 4 and JUnit 5 dependencies;
   2. Copy all of the Java class and test code to this project;
   3. Run all of the test methods and snapshot the testing results.



* 1. NOTE: You should submit your maven project and your running results.

1. Now, please illustrate why we manage JUnit 4 and JUnit 5 dependencies by maven, rather than manually?

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, Maven deals with several areas of concern:

* Making the build process easy
* Providing a uniform build system
* Providing quality project information
* Encouraging better development practices

1. **Assert** is very important for Making Tests. Please read and run the test codes, [CalculatorTest.java](https://favour-link.feishu.cn/file/boxcnNkM1q4YVp0NxA5XSDMGc9d), and illustrate why? That is, if we don’t use Assert functions, what will happen in our testing code?

* The program should crash because a condition that was supposed to be true became false, that shouldn’t change this intended behaviour by catching the exception with a try…except block.

**3.3 Task 03: Parameterized Test.**

[This video](https://youtu.be/srJ91NRpT_w) introduced the Roman numeral problem. We provide its implementation in [**RomanNumeral.java**](https://favour-link.feishu.cn/file/boxcnw1IS4tPJ4kJ5HTudCTFQqc) and its corresponding test class in [**RomanNumeralTest.java**](https://favour-link.feishu.cn/file/boxcnqrAqWZuJv9vtORulKAXCXc). The method *singleDigit* in [**RomanNumeralTest.java**](https://favour-link.feishu.cn/file/boxcnqrAqWZuJv9vtORulKAXCXc) tries to check if the 7 Roman numerals: I, V, X, L, C, D, and M, can be correctly mapped by *singleDigit* to their corresponding Arabic numbers, 1, 5, 10, 50, 100, 500, and 1000. However, the testing codes seem very duplicated and repetitive. We can greatly simplify the testing codes by using ***Parameterized Test*** provided by JUnit 5. Please refer to user guide [**here**](https://favour-link.feishu.cn/docs/doccnpJUpcabopOtxYNVyMaf7l7#3Ft24O) to rewrite the test method *singleDigit* by using *Parameterized Test*. Give your answer in *singleDigitParameterizedTest* in [**RomanNumeralTest.java**](https://favour-link.feishu.cn/file/boxcnqrAqWZuJv9vtORulKAXCXc).

Note that if you're using Python and perform unit testing by using pytest, you should rewrite both [**RomanNumeral.java**](https://favour-link.feishu.cn/file/boxcnw1IS4tPJ4kJ5HTudCTFQqc) and [**RomanNumeralTest.java**](https://favour-link.feishu.cn/file/boxcnqrAqWZuJv9vtORulKAXCXc) in Python.

**3.4 Task 04: Test your own code with JUnit 5.0**

**Basic Employee Compensation Problem.** For each week, hourly employees are paid a standard wage per hour for the first 40 hours worked, 1.5 times their wage for each hour after the first 40 hours, and 2 times their wage for each hour worked on Sundays and Holidays. Table 1 gives some test cases of this.

Please write a Java class, WageCalculator, to solve the wage problem in the following and a test class WageCalculatorTest to test your code by using the test cases in Table 1.

