

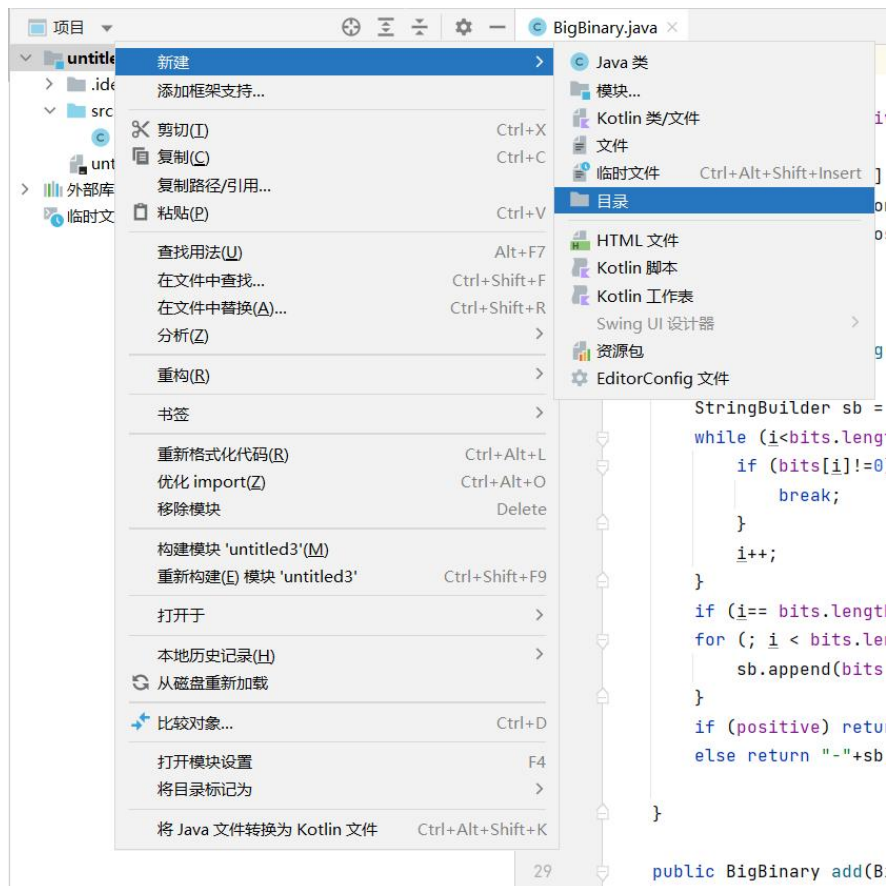
How to use JUnit in IDEA

1. 准备工作:

首先将根据 Assignment4 题目自己写的待测试文件放在 src 目录下，再新建两个目录 test 和 testcase

1. Preparation work:

First put the file to be tested written according to the Assignment4 under the src directory, and then create two new directories called test and testcase.



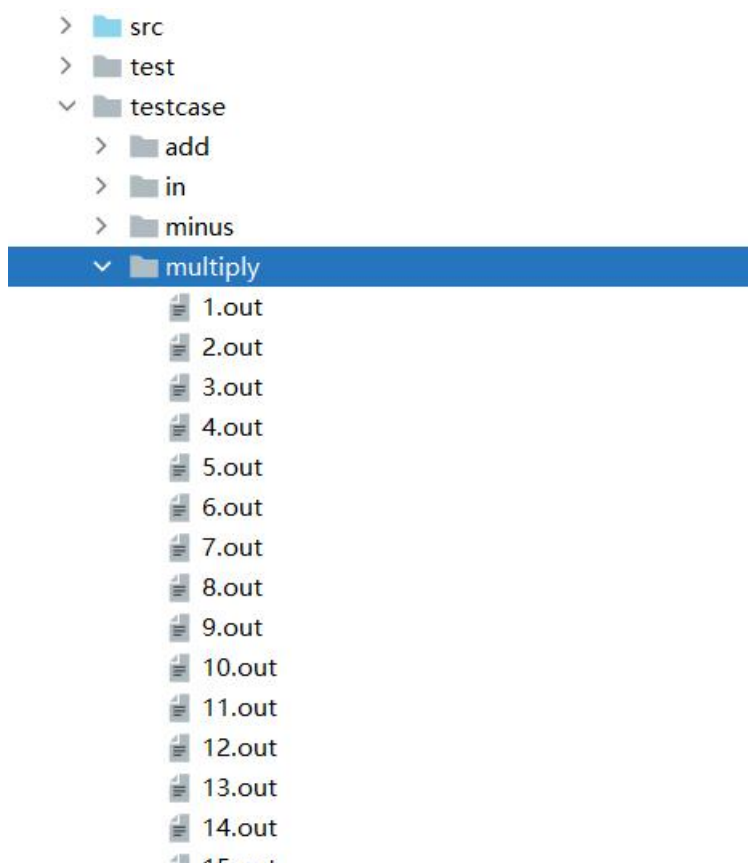
完成后如下图


After this operation, it's like this




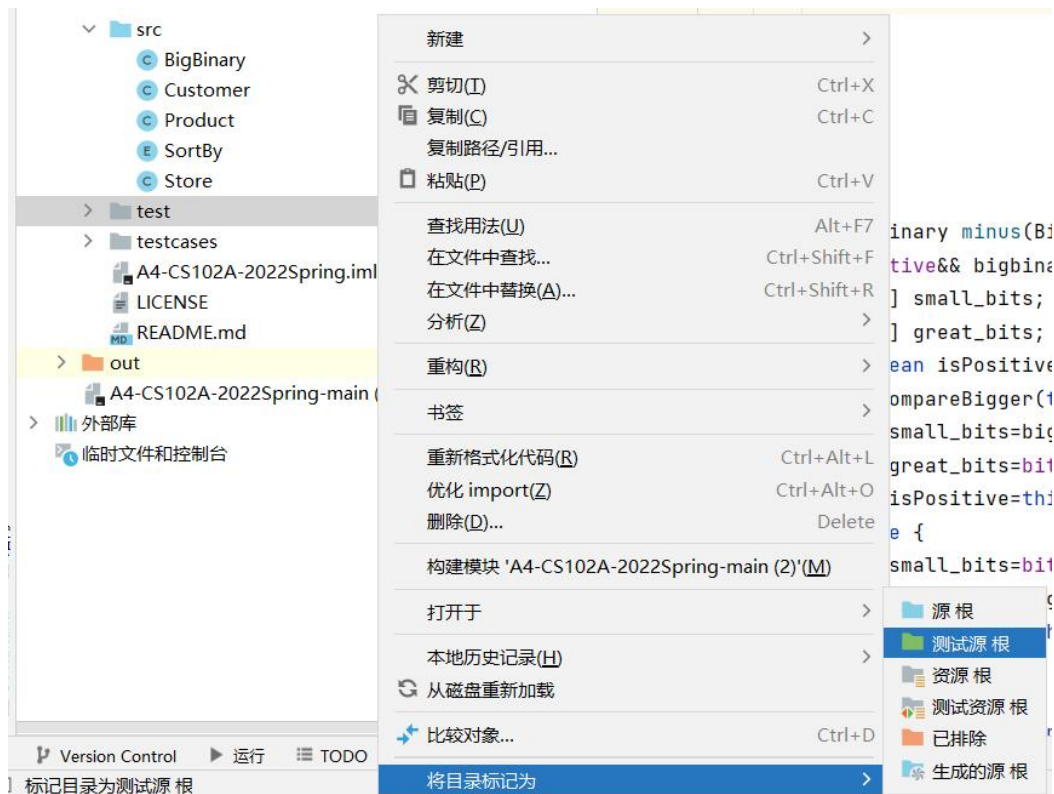
接下来将测试样例导入 testcase 目录

Next, import test cases into the testcase directory



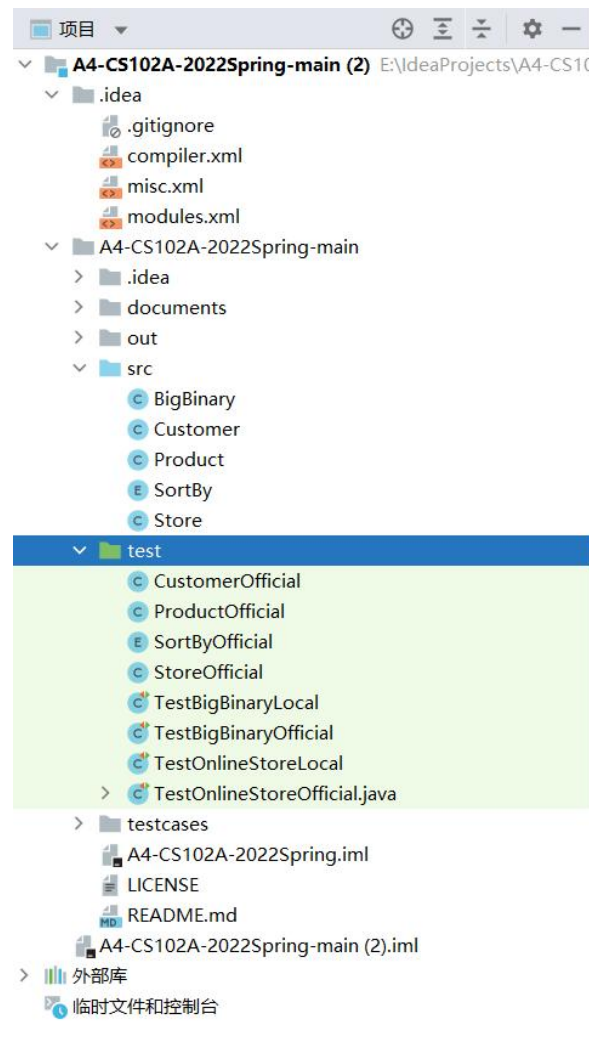
然后将 test 目录标记为测试源目录，变成  test

Then mark the test directory as the test source directory and it becomes  test



将 junit 测试文件导入 test 目录，最终情况类似下图

Import junit test files into the test directory. Ultimately it is similar to the figure below



注意：要保证这三个目录（src, test, testcase）处于同一级目录下

Note: Ensure that these three directories (src, test, testcase) are in the same level directory.

2. 导入 JUnit 文件

接下来，我们以 TestBigBinaryLocal.java 为例 讲解如何修改导入 JUnit 文件后的报错。打开 TestBigBinaryLocal.java 后会看到红色的报错如下

2. Import JUnit files

Next, we use TestBigBinaryLocal.java as an example to explain how to modify the reported errors in a JUnit file after it is imported. When you open the TestBigBinaryLocal.java, you will see the red letters.

```

import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

import java.io.BufferedReader;
import java.io.FileReader;
import java.lang.reflect.Constructor;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.time.Duration;

import static org.junit.jupiter.api.Assertions.*;

```

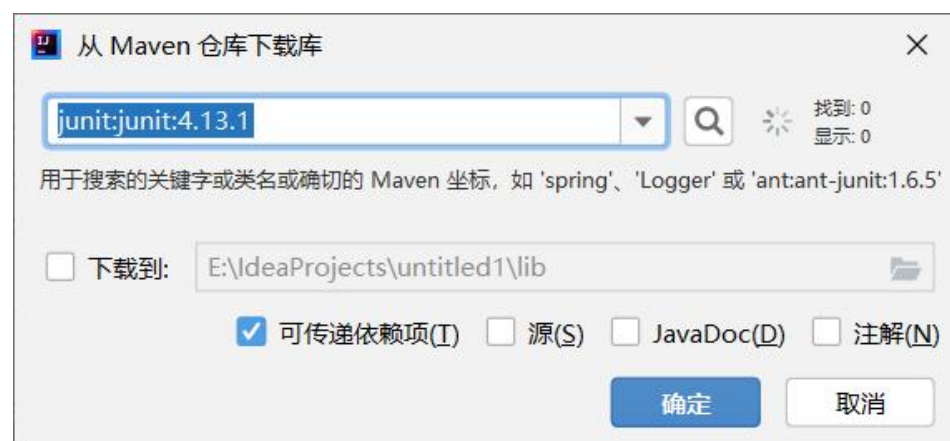
接下来将鼠标移到红色的 junit 上会看到这样

Next, move the mouse over the red “junit” and you will see this



点击“将 ‘JUnit4’ 添加到类路径”，然后出现下图的情况

Click "Add 'JUnit4' to the classpath" and then the situation in the figure below appears



点击“确定”。接下来发现 jupiter 爆红，我们又重复上述的类似操作，将鼠标放在标红的 jupiter 上，添加 JUnit5.8.1 到类路径

Click "ok". We find that “jupiter” become red and we repeat similar operations above.

Place the mouse over the marked red jupiter and add JUnit5.8.1 to classpath.

```
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

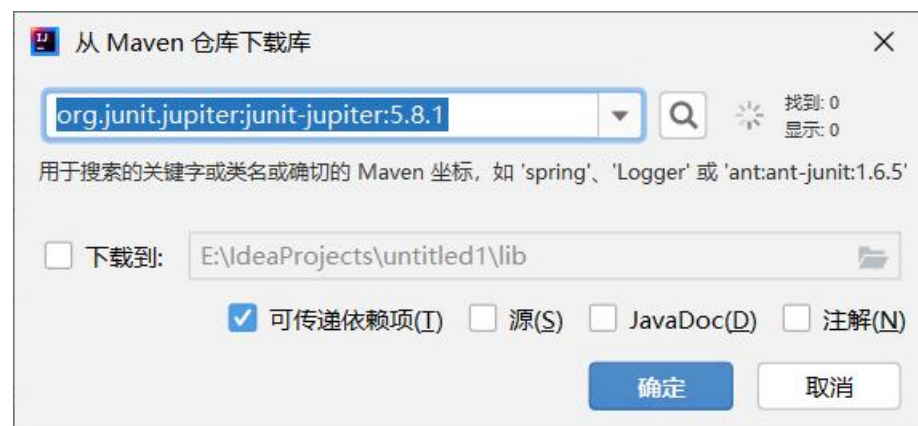
import java.io.BufferedReader;
import java.io.FileReader;
import java.lang.reflect.Method;
import java.lang.reflect.Modifier;
import java.time.Duration;

import static org.junit.jupiter.api.Assertions.*;
```

无法解析符号 'jupiter'
 将 'JUnit5.8.1' 添加到类路径中 Alt+Shift+Enter 更多操作... Alt+Enter
 未找到任何文档。

点击“确认”

Click “ok”



然后红色的报错就没有了，文件可以正常运行。

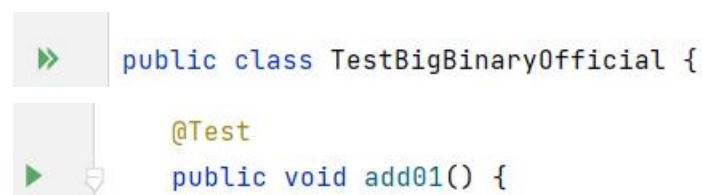
Then red letters disappear and the test file can run fine.

3. 运行：

通过点击类和方法旁边的绿色三角形符号运行测试文件。可以通过类旁边的三角形测试所有样例，也可以通过方法旁边的三角形测试单个样例

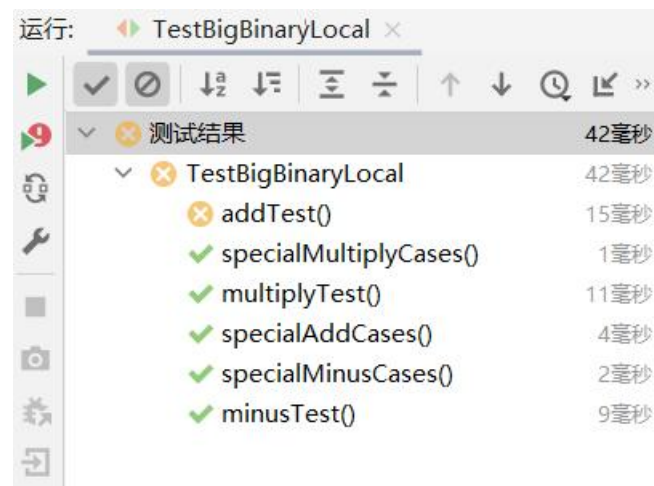
3. Run:

Run the test file by clicking the green triangle symbol next to the class and methods. You can test all tests by the triangle next to the class, or a single test by the triangle next to the method



运行后在下面可以看到结果, 我们可以点击相应的样例查看详情

You can see the results after running



✓ : 通过了这个测试样例 Pass this test

✗ : 未通过这个测试样例 Fail in the test

4. 提醒:

通过了全部的测试样例，也不能保证提交到 OJ 上后得到满分噢。如果关于 junit 测试还有什么疑惑，可以询问一下助教~

4. Reminder:

Although passing all test cases, there is no guarantee to get full marks in OJ. If you have any other question about the junit test, you can ask SAs for help.