Git/Java Setup Guide

20-3-2024 – Max L. Wilson

Ultimately, you only want the source files to sync in the background when you are working with Java and git. So when you finish some work, you commit and push, and the java files go up. And when you pull, only the java files come down to you to merge. Perhaps projects might want some other things, like a folder of test data, or build scripts (in more advanced projects) to sync. Indeed, eclipse lets you edit and view markdown. **Handy!**

But, you do not want your team-mates java project file to sync, such that the file paths etc are wrong for your computer. And you don’t want to sync the compiled class files in the /bin/ folder.

So the ideal setup is that your java project workspace is NOT in the git repository, but your java project edits the files in your git folder. Thus, when you save files, it updates the files in the git folder (without you e.g. copying or moving them there). This is a separation between Java project, and source files, that you should become aware of.

There are many ways to achieve this, but the following steps are Max’s recommendation.

Stage 0: One person do this for the team

On a dev branch:

Step 1: Move the CW2 java files into the /src/ .

- One person should do this, commit and push, and then everyone should pull it down.

Stage 1: Each person should do this

Everyone make sure you are on a dev branch. Follow the steps below to configure the java project for your computer.

Step 2: Each person should create a new Java project. This could be stored anywhere in your computer. I recommend (to reduce confusion) that you store it in the root Coursework folder of the git project if you want them to be closely related (Java projects expect a src folder below them anyway), and then you can access the src and docs folders etc from the java project.

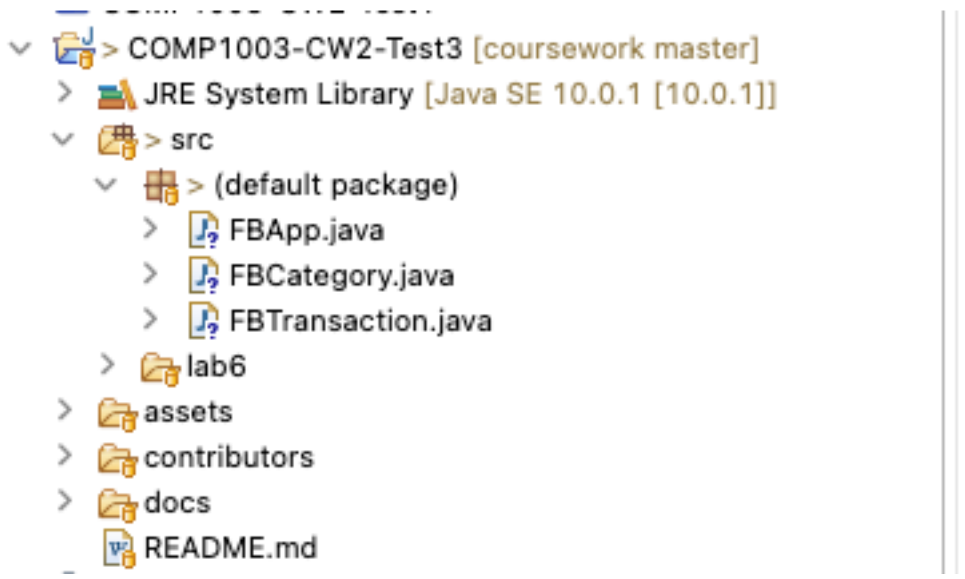
1. Open Eclipse
2. Create a new Java Project
3. IMPORTANT: Untick ‘store in the default location’
4. Navigate to the git coursework folder on your computer
5. IMPORTANT: check what version of java you are using. The whole team should use the same to avoid problems. This can be changed later if someone does this wrong.
6. Press Finish (with default settings etc.)
7. If asked, click ‘don’t create’ for a module-info file.

- Check straight away that you can then see your existing folder structure in the java project. You can delete the TempApp.java file now too.
- You should be able to right click on the main java file, and click 'run' to see it run in the console part of eclipse.
- Check with a `git status` that its not going to try and sync any unwanted files.

Step 2b Exclude other java projects: If you have previously stored java projects within the repository (and its subfolders), then you will see a lot of java files in your project that you probably do not want (if not, then move to step 3). You can clean up your Java project at this stage, to only include the files you want:

1. Right click on the project, go to build path > configure build path
2. In the Source tab, select the source folders you do not want to include, and press 'remove'. This remove them from your project, but not delete them from the folder. This should just leave you with just the src folder with the 3 CW2 java files in it.
3. You may be left with a number of 'packages' listed below 'default package'. You can right click on these and press build path > exclude.

You should now look something like this (except the file names may vary from year to year):



```
v > COMP1003-CW2-Test3 [coursework master]
  > JRE System Library [Java SE 10.0.1 [10.0.1]]
  v > src
    v > (default package)
      > FBApp.java
      > FBCategory.java
      > FBTransaction.java
    > lab6
  > assets
  > contributors
  > docs
    README.md
```

NOTE: (file names may be different)

Stage 2: You may need to do this – not everyone will need to.

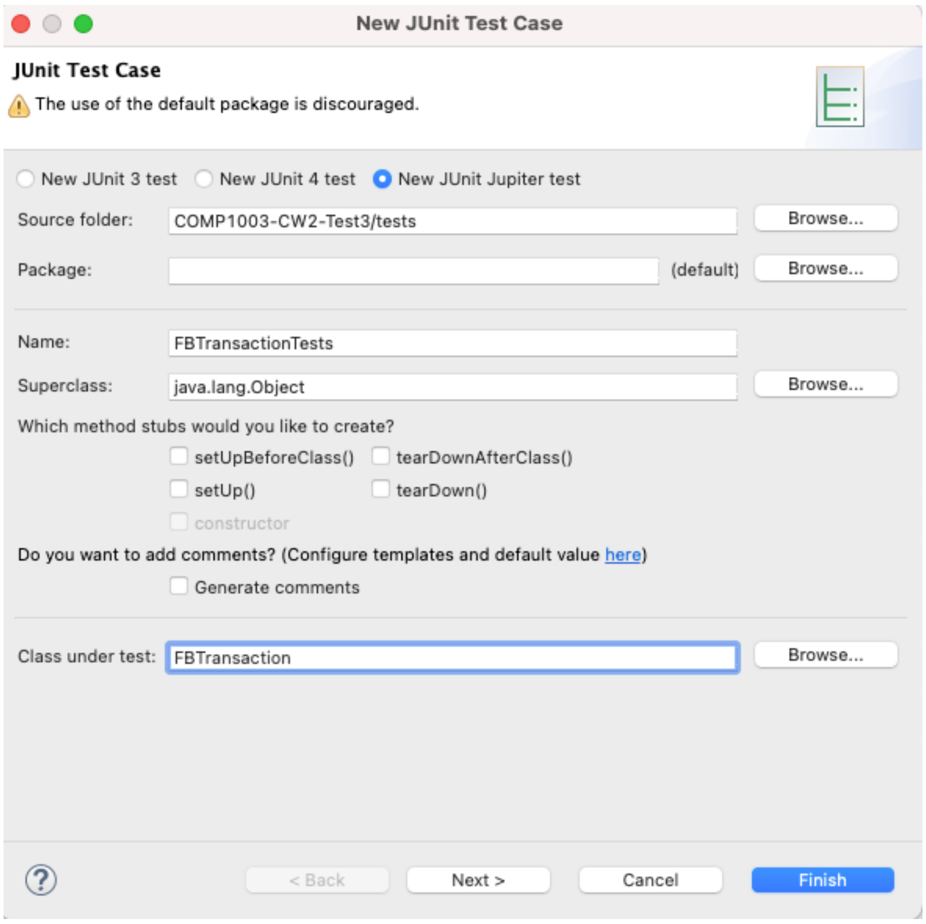
If someone has created the tests folder already, or has created a JUnit file already for the tests you are contributing to. i.e. if you are writing some tests for a java class, then someone else may have already created the tests folder, and may have already created a JUnit test file to hold the tests for this class. In which case you don't need to do these steps below, as there will already be a folder called tests, and already a file that you can contribute to. See alternate stage 2 at the end of the document.

Step 3: Right click on the project and add a new **source folder**, call it 'tests'

- Only one person needs to do this really, but it won't sync with git until it has a file. But you certainly don't want everyone to have a different tests folder with a different name. If you all do this step, and all call it the same thing, you'll be fine.

Step 4: Right click on the tests source folder. And add a JUnit Test case (making sure it is testing the relevant class in the 'class under test' box). Hit Finish and you should be able to take it from here. You can right click on the test file and run as JUnit test, to see it run and fail (as it should).

- A `git status` now will probably offer you to add this new test file.
- Of course, you want to give this test a name that's specific to the class you are working on. And if two of you are working on the same test file, then you probably only one of you wants to do this. Then use `git add`, `git commit`, `git push`, and have them `git pull` to be able to work on the same file with you.


New JUnit Test Case

JUnit Test Case

The use of the default package is discouraged.

New JUnit 3 test / New JUnit 4 test / [✓ New JUnit Jupiter test]

Source folder: COMP1003-CW2-Test3/tests

Package: (default)

Name: FBTransactionTests

Superclass: java.lang.Object

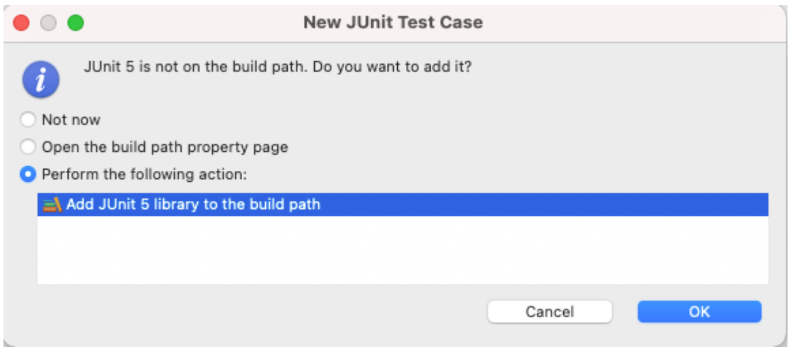
Which method stubs would you like to create?
setUpBeforeClass() tearDownAfterClass() setUp() tearDown() constructor

Do you want to add comments? (Configure templates and default value here)
Generate comments

Class under test: FBTransaction

NOTE: (file names may be different)

This image shows adding the junit library to your project. This should pop up when you first add a jUnit file.

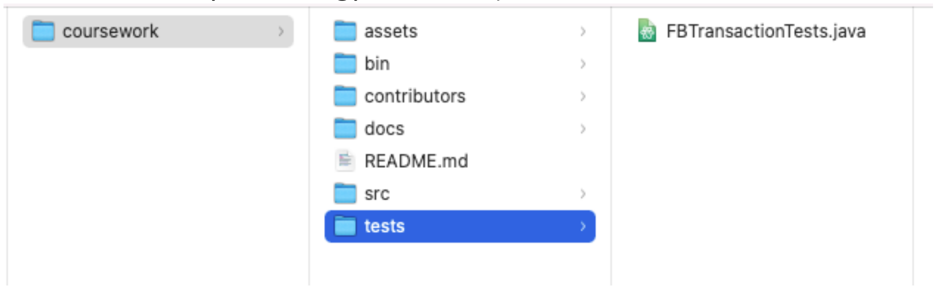

New JUnit Test Case

JUnit 5 is not on the build path. Do you want to add it?

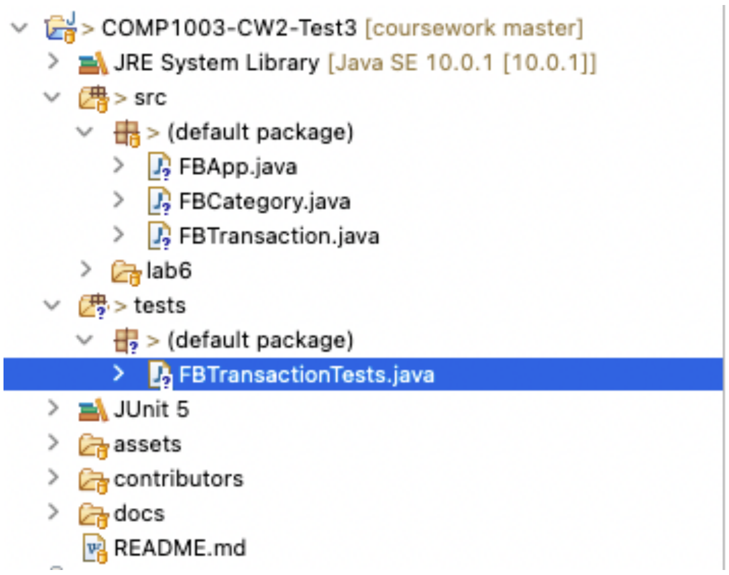
Not now
Open the build path property page
[✓ Perform the following action:]

Add JUnit 5 library to the build path

Your folder structure should end up like this (except, hopefully you've given it the name relevant to this years dodgy code files).



Your java project may look something like this:



NOTE: (file names may be different)

Alternate Stage 2:

Note: If someone else created the JUnit test file before you, you will need to manually add the JUnit library to your project before you can run (and write) JUnit

1. Right click on project in eclipse > build path > add libraries
2. Choose JUnit, and then JUnit 5, and add to the library.

Then you can right click on a JUnit file, and run as > JUnit test