Lab: Code Coverage With Intellij

Creating And Running Unit Tests

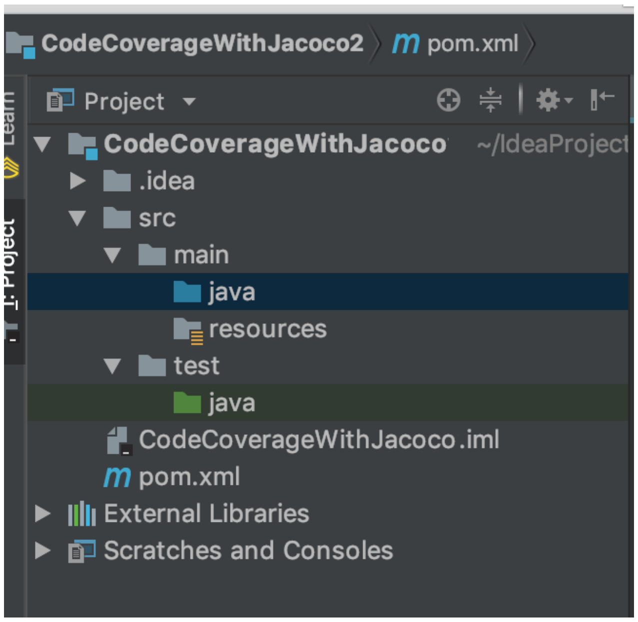
Introduction

Now that you know how to use JUnit, and have learned about code coverage, it's time to learn how to use code coverage in a Java project to help you determine if your project has sufficient testing.

Setup

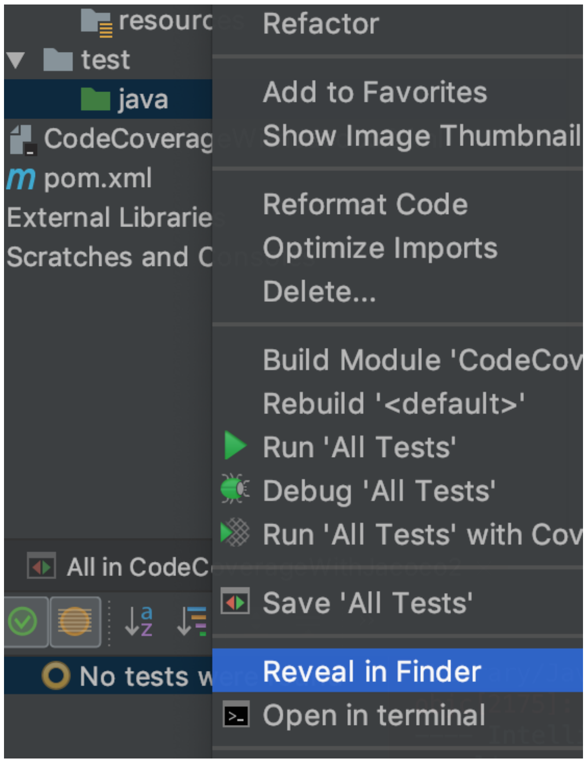
Complete the following steps to create and setup a project for the lab.

1. Make a new Maven or Gradle project with an artifact ID of "CodeCoverageWithIntelliJ". Once you have built your project, it should have this structure if you used Maven (the structure will be similar if you used Gradle):



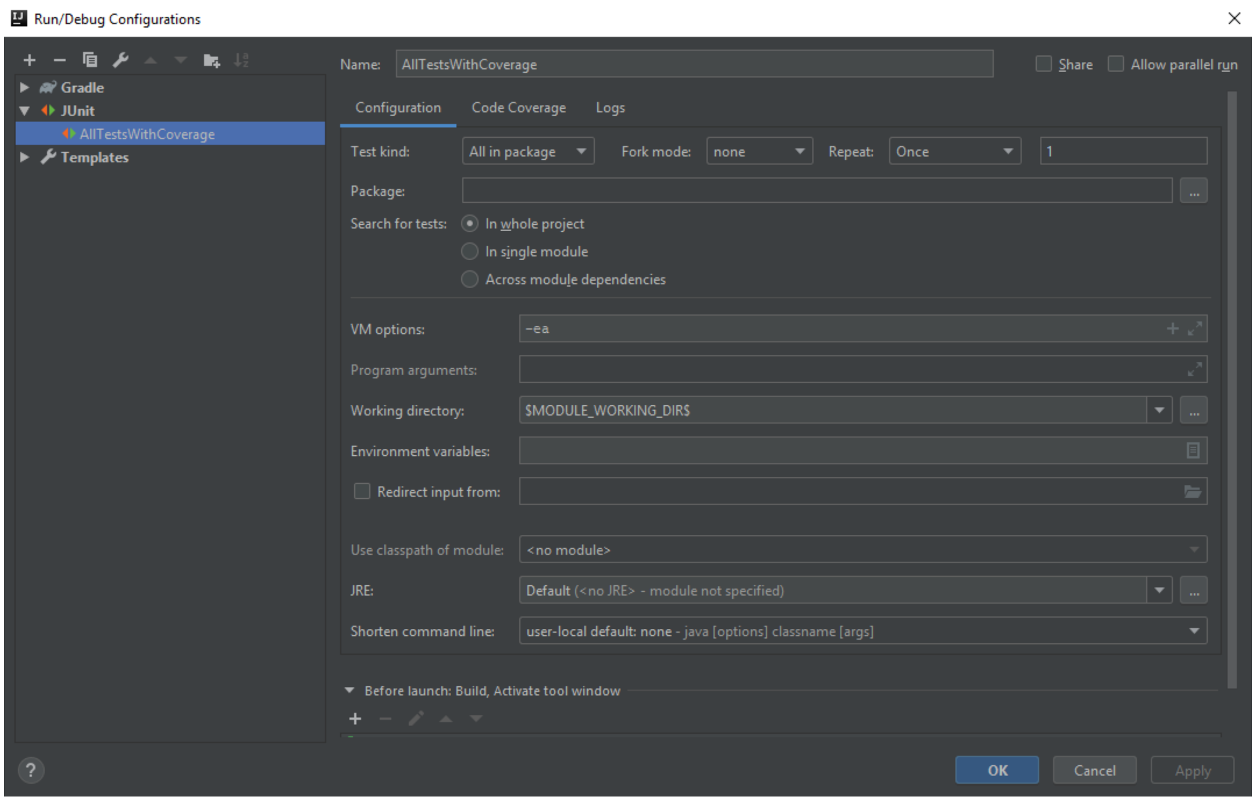
```
CodeCoverageWithJacoco2 〉 pom.xml 〉
Project ▾
▼ CodeCoverageWithJacoco  ~/IdeaProject
  ▶ .idea
  ▼ src
    ▼ main
        java
        resources
    ▼ test
        java
    CodeCoverageWithJacoco.iml
    pom.xml
▶ External Libraries
▶ Scratches and Consoles
```

2. Download these files to your computer.
3. Add the files SimpleCoverageFunctions.java and HarderCoverageFunctions.java to /src/main/java and add SimpleCoverageFunctionsTest.java and HarderCoverageFunctionsTest.java to /src/test/java. The easiest way to do this is to right click on the folder and find the option "Reveal in finder" or "Reveal in file manager" and move the files into the correct folders. You can also simply drag the files straight into the correct folder in IntelliJ's UI.



4. After you do that, make sure the files have no errors in them and you are able to run both tests. You will probably have to add a JUnit5 dependency to your pom.xml file (for Maven) or your build.gradle file (for Gradle). Refer to the “Unit Testing with JUnit” tutorial if you need help adding the JUnit5 dependency.
5. Once you have confirmed that you can run the tests, right-click on the green folder /src/test/java and click “Run ‘All Tests’ with Coverage”. A sidebar with your results should appear. If you open SimpleCoverageFunctions.java you should see that to the left of the code there is green and red highlighting. Currently you are only testing for line or statement coverage. We will next enable branch coverage as well.
    a. If building a Gradle Project you may find that on running all your tests with coverage that they will fail or produce no results. If this is accompanied with a message similar to “Test events were not received” then this is because IntelliJ is attempting to run your tests through Gradle, not through JUnit. To fix this problem, do the following to create a run a JUnit test configuration:

        i. Go to Run > Edit Configurations in the top toolbar. Click on the “+” to add a new configuration and select a JUnit configuration. Name it “AllTestsWithCoverage” or whatever you want to help you remember what it is. Select “Test kind” as “All in package” with “Search for tests” as “In whole project”. Click on the Code Coverage tab and make sure that you select Tracing as we did before. See the screenshots below:


Run/Debug Configurations

Gradle
JUnit
AllTestsWithCoverage
Templates

Name: AllTestsWithCoverage
Share
Allow parallel run

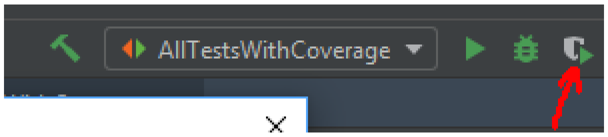
Configuration | Code Coverage | Logs

Test kind: All in package
Fork mode: none
Repeat: Once
1
Package:
Search for tests: (●) In whole project ( ) In single module ( ) Across module dependencies
VM options: -ea
Program arguments:
Working directory: $MODULE_WORKING_DIR$
Environment variables:
Redirect input from:
Use classpath of module: <no module>
JRE: Default (<no JRE> - module not specified)
Shorten command line: user-local default: none - java [options] classname [args]

Before launch: Build, Activate tool window

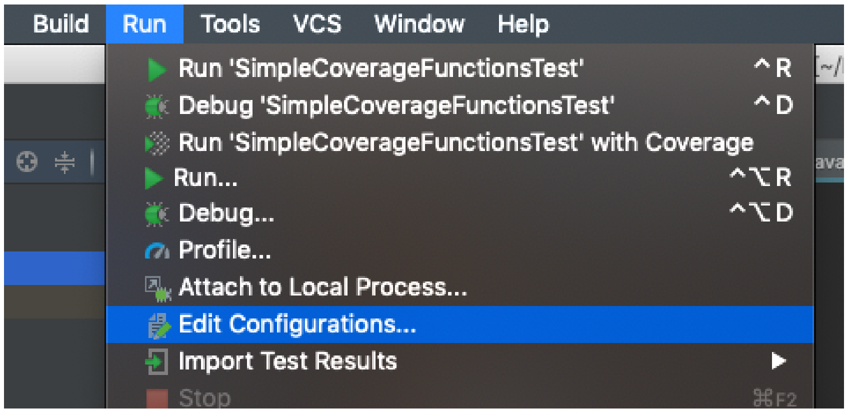
OK | Cancel | Apply

Now all you need to do is select this test configuration in the upper right hand corner and select the “Run with Coverage” button.

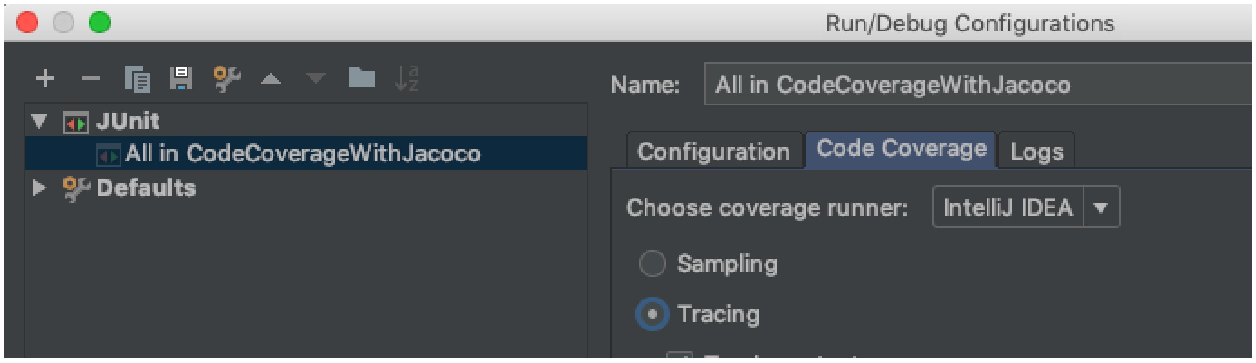


To enable Branch coverage, not just statement coverage, follow the directions below:

1. Click on run in the menu bar and select Edit Configurations.


Build
Run
Tools
VCS
Window
Help
Run 'SimpleCoverageFunctionsTest' ^R
Debug 'SimpleCoverageFunctionsTest' ^D
Run 'SimpleCoverageFunctionsTest' with Coverage
Run... ^⌥R
Debug... ^⌥D
Profile...
Attach to Local Process...
Edit Configurations...
Import Test Results
Stop ⌘F2

2. Click on the tab that says code coverage and check the Tracing radio button, then click apply.


Run/Debug Configurations

JUnit

All in CodeCoverageWithJacoco

Defaults

Name: All in CodeCoverageWithJacoco

Configuration | Code Coverage | Logs

Choose coverage runner: IntelliJ IDEA

Sampling

[✓ Tracing]

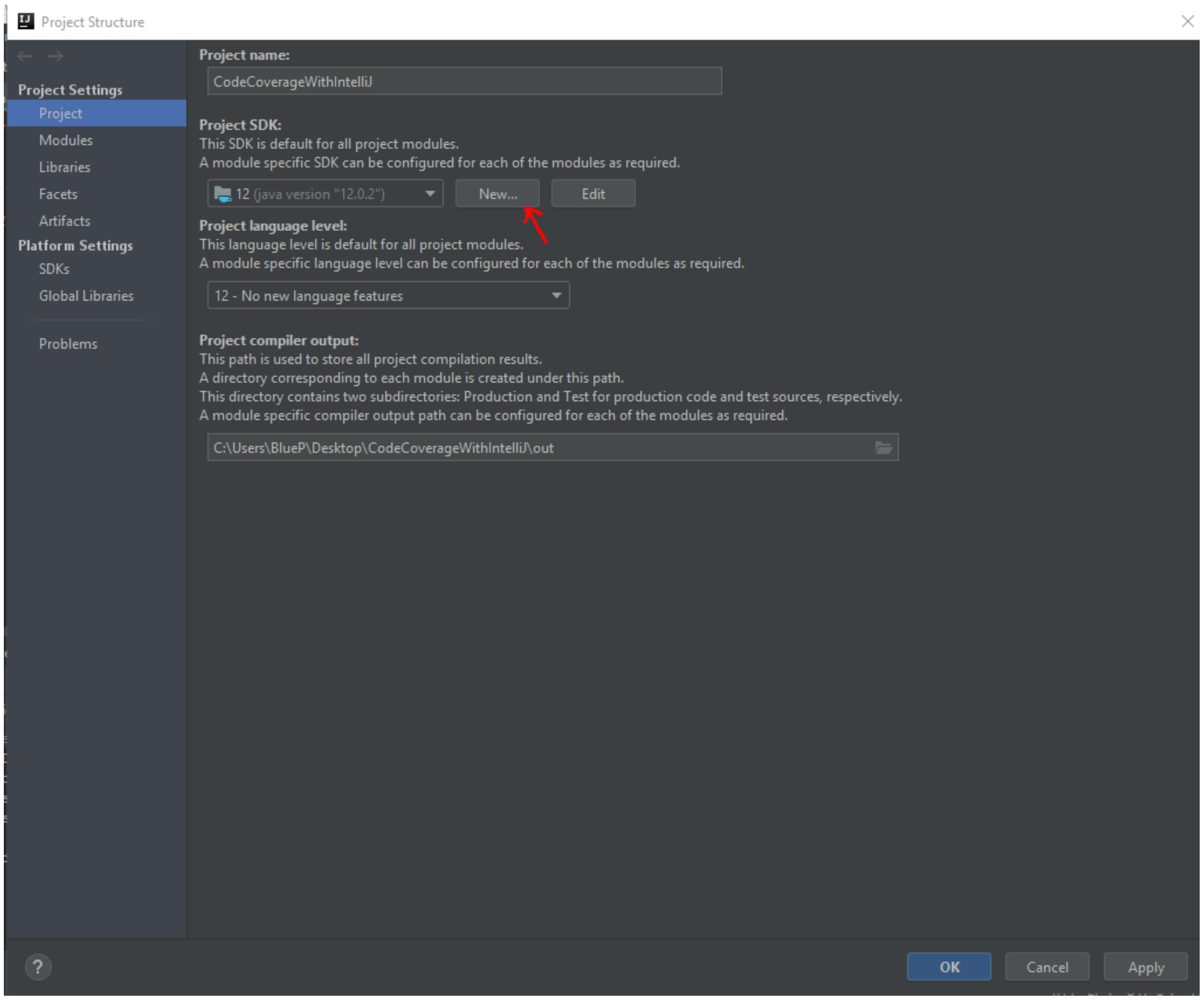
Note: if you wanted to run just one of the tests with branch coverage, you would enable Tracing for that specific test.

3. If you run your tests with coverage again you should see in SimpleCoverageFunctions.java that there is now yellow highlighting as well to the left of your code (This will only show up if you have actual tests that execute lines of code for those classes). The green, yellow, and red highlighting indicates which lines are tested, partially tested, and not tested. If the line is partially tested that usually indicates that only one path of the branch was executed.

   NOTE - You may run into a few issues when trying to enable Branch Coverage. Here are the issues and the fixes for them

   - When you first run branch coverage you might already be told that you have 100% branch coverage (100% 0/0). The 0/0 indicates zero out of zero branches are covered. This happens because branch coverage only applies to branches in the code that get executed. If you are running empty tests then the code is never executed therefore it is 100% branch coverage. As you write your code this issue will resolve itself and give you proper statistics.

   - In Maven you may not be able to select Tracing to enable Branch Coverage. This is a rare case that occurs on some systems when they are using later versions of Java. If this does occur you may visit https://www.oracle.com/technetwork/java/javase/downloads/index.html
     And download a different version of Java. Install it on your computer. Then in IntelliJ navigate to File > Project Structure on top toolbar. Here you can set the different version of Java you just downloaded by changing the Project SDK in the Project tab. Click on “New…” > “JDK” and IntelliJ will point you to your Java directory where you can select the Java version you just downloaded. (In TA experience the version we used to fix this problem was version 10)


Project Structure

Project name:
CodeCoverageWithIntelliJ

Project SDK:
This SDK is default for all project modules.
A module specific SDK can be configured for each of the modules as required.
12 (java version "12.0.2")

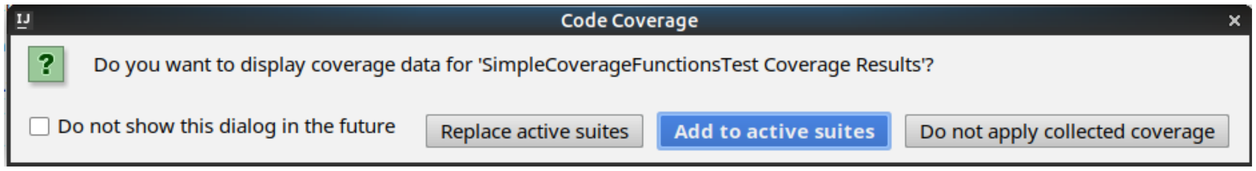
Project language level:
This language level is default for all project modules.
A module specific language level can be configured for each of the modules as required.
12 - No new language features

Project compiler output:
This path is used to store all project compilation results.
A directory corresponding to each module is created under this path.
This directory contains two subdirectories: Production and Test for production code and test sources, respectively.
A module specific compiler output path can be configured for each of the modules as required.
C:\Users\BlueP\Desktop\CodeCoverageWithIntelliJ\out

Lab

Now that your project is setup, you are ready to complete the lab. Add additional tests (you can add them to the existing test methods and/or you can create new tests methods), until you have 100% **line and branch** coverage. When you are through you should only have green highlighting (no red or yellow) to the left of your code in the code editor.

NOTE: if you aren't able to run all tests at the same time you can add several tests to the same report by selecting "Add to active suites" after running more than one test)


Code Coverage

Do you want to display coverage data for 'SimpleCoverageFunctionsTest Coverage Results'?

Do not show this dialog in the future
Replace active suites
Add to active suites
Do not apply collected coverage

Submission

Once you have 100% line coverage and 100% branch coverage create a new folder on your desktop (or some other convenient location). Submit a screenshot of your coverage, and both of your testing files to Canvas (as three separate files).