Dr Businge, Project Coordinator

Garrett Prentice, Project Manager

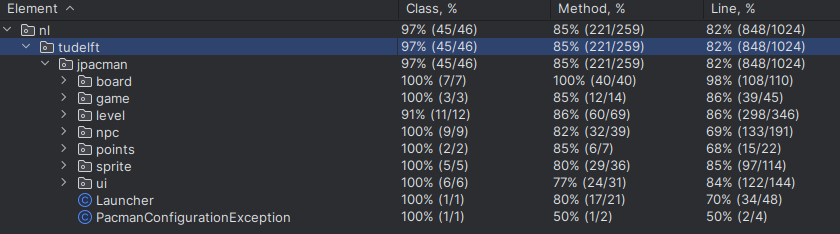
**RE: Code Coverage Evaluation (JaCoCo v IntelliJ)**

I believe that the coverage results from IntelliJ are better for uses that prefer integrated environments. When I use IntelliJ, I use the built in code coverage as it provides me with all that I need and allows me to jump directly to editable source code. JaCoCo is a great substitute for users that do not use IntelliJ; VSCode, or Neovim.

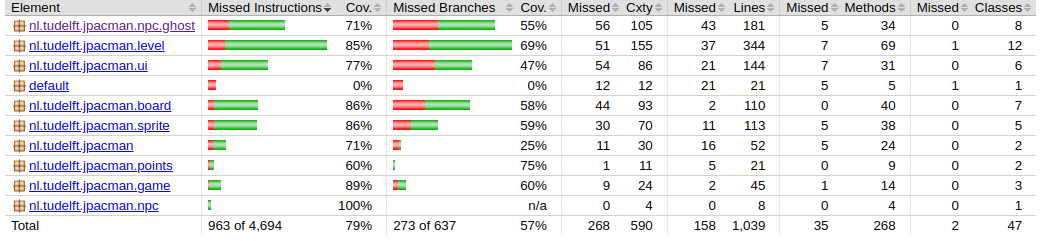
**Similarity**

I have similar results between both JaCoCo and IntelliJ’s coverage reporting values. However, IntelliJ gives me a line based breakdown of the code. JaCoCo gives me functional breakdown of the code. For example, in Figure 1 and 2, IntelliJ reports by line, method, and class. These are things that only require touching to change. But JaCoCo coverage is more focused on code branching and combination of input. For this reason, JaCoCo might be better for quality of test.

*Figure 1: IntelliJ coverage report for JPacman repo*



*Figure 2: JaCoCo coverage report for JPacman repo*

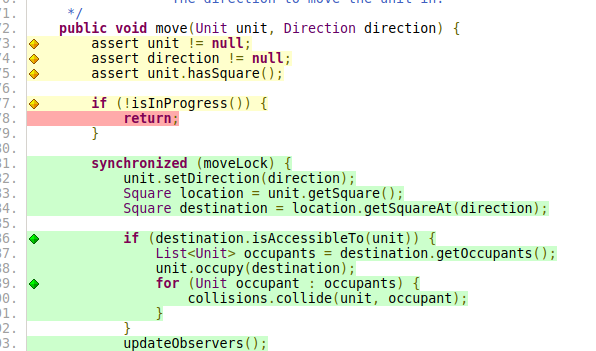


As shown above, JaCoCo reports a lower code coverage than IntelliJ. In JaCoCo, it reports only 79% for missed instructions, which is closest to line coverage in IntelliJ, which is reported as 82%.

**Usefulness**

I believe that seeing the source code in the browser is the most useful feature of JaCoCo. I can match this directly to my test cases without opening the project. For example, in Figure 3, I can navigate to a specific section of code that was not covered and then write test cases to cover it.

*Figure 3: JaCoCo coverage of Level.java#move()*



As shown above, the code in yellow is assertions and branches that were checked. The section shown in red is branch that has not yet been covered, and the green is what has been covered.

**Preference**

I have taken this course at a different school, we used JaCoCo, however, when paired with IntelliJ it is not as powerful. But, when developing a report to senior developer or even as the project lead, it can make my life easy. Receiving a zip file with these test case reports is sometimes easier than going through the process of opening GitHub or IntelliJ to view test results.

Now that I use Neovim, a text editor IDE, it is much lighter weight, but test cases are harder to view. JaCoCo bundled with Gradle would be a solution to this problem without needing to integrate with a plugin or write any code. From a developers standing, I prefer IntelliJ. From a project manager perspective, I prefer JaCoCo.