Jung Hwan Brandon Bae

CS 472

JPacman Write-up

This write-up will consist of a report on the code coverage summary that JaCoCo and IntelliJ provide. In addition, further input will be given on the style and format of both these tools give their summary.

First and foremost is the difference in code coverage reports between JaCoCo and IntelliJ. When we created unit tests to increase code coverage, IntelliJ would often report that the tests were completely successful in raising code coverage for that method. However, if we peruse the JaCoCo report, we can see that this code coverage tool gives us a different story. Take the package *PacManUiBuilder* as an example. We created a unit test for the *addButton* method. If we look at line 1, we can see the following code 'assert caption != null'. While IntelliJ gives us 4 hits for the code coverage on that line, JaCoCo tells us we've missed ¼ branches. This difference, while minute, can probably be attributed to how these two tools calculate code coverage. While I'm not sure of the specifics, this is the only reasonable explanation as to why different reports could be given.

JaCoCo provides a visual html overview of your project. In addition, there is a colored representation of coverage results for each package, method, and branches inside of said method. This is an extremely useful tool that allowed me to easily pick methods to create unit tests for. For this lab assignment, I was incentivized to create unit tests for methods that did not have one and was not covered in the code coverage report. In comparison, IntelliJ does something similar. However, instead of an HTML webpage you view your project from, you view the code coverage results in a separate project window in the IDE. IntelliJ provides you with class, method, and line coverage results for each layer of your project. In addition, IntelliJ also colors the line green or red depending on whether that line has been covered by some snippet of code elsewhere. Between these two, I prefer the report that JaCoCo provides because it's easier to grab information from a short glance. In addition, I believe that the difference in code coverage results imply that JaCoCo is more stringent on what it considers "covered" code.

Link to forked JPacman repository: https://github.com/baej12/jpacman

Link to lab write-up in team repository: https://github.com/TheBenKnee/CS-472-Senior-Design-Project