Melissa Jones, Nicholas Benyo, Scott Moser
Professor McKee
CPSC 5210-01
2 May 2019

Milestone #1

I. Project Description

The project chosen by the D'Buggers is the classic game of Battleship written in Java. In this variant, a user sets his or her ships on the board and then plays against the computer via command-line interaction. Hits and misses are recorded under the hood as state is maintained by the program until a winner is declared. We are using the Eclipse IDE for development and the built-in JUnit framework for testing. Our project was forked from GitHub user ymarcus93 who committed the code back in 2013 with MIT licensing.

GitHub Project URL: https://github.com/mosers1/Java-Battleship

II. Test Approach

For the purposes of this project, we are defining a software unit as a class. After inspecting the code and playing the game for a while to familiarize ourselves with the application, we needed to choose a unit to test.

We started by identifying the various software units and evaluating their testability. The Battleship project has a total of six classes. We quickly ruled out the Battleship class which is used to drive the program and interact with the user. This left us with five testable classes. We evaluated each class and determined the Location class would be ideal for the first milestone. Our simple evaluation can be found in our GitHub repository under /docs/UnitMapping.xlsx.

Once the unit was identified, the individual public functions were assigned to testers and the tests were created. The Location class encapsulates its private data well and exposes numerous public accessors and mutators to manipulate its data. This results in paired getter/setter functions. Half of the mutators take no parameters or only a Boolean value. This made for relatively simple test cases. The other half take integer parameters. Equivalence class partitioning was performed for these functions and their corresponding accessor functions.

Location unit test code URL:
https://github.com/mosers1/Java-Battleship/blob/master/LocationTest.java

III. Test Results

A total of 10 test cases were created to test the Location software unit. Upon executing them using the JUnit framework built into the Eclipse IDE, all ten tests were able to run and pass. The results of the test run can be seen below in Figure 1. These results inspire confidence in the software unit.

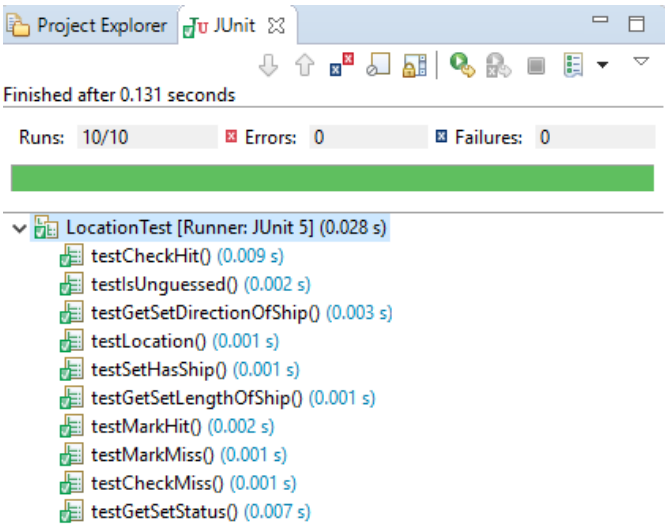

Figure 1. JUnit test run results.

IV. Code Coverage

For the software unit under test (UUT), 100% code coverage was achieved. This was measured using the code coverage tool built into the Eclipse IDE. Detailed results of the code coverage analysis can be seen below in Figure 2.

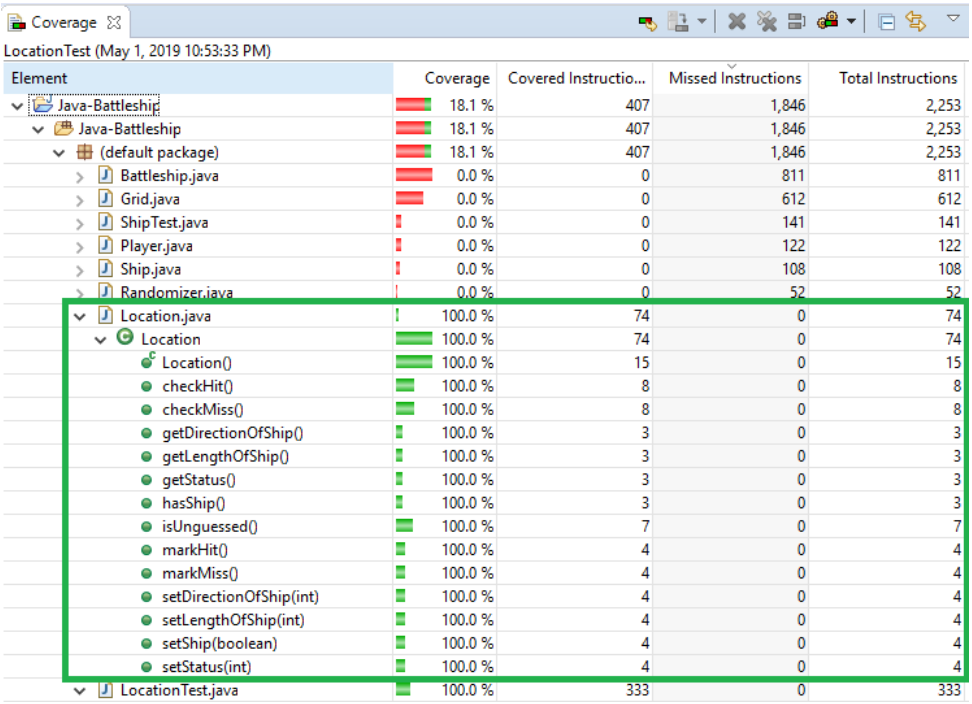

Figure 2. Code coverage for Location.java.

V. Experience to Date

The first two projects we picked turned out to be untestable without refactoring or a significant amount of mocking. Given that all group members work full-time, our estimate for this level of work was deemed unrealistic and so these projects were scrapped. Unfortunately, it was not until we learned more about unit testing that we figured this out.

However, the overall experience has been positive since we found a testable project. The Battleship project we chose proved to be easy to understand, easy to import and build, and fairly easy to test. While several of our test cases are simple, we still get the benefit of practicing writing test cases, reducing the number of test cases using techniques learned in class, and evaluating code coverage. Additionally, we get to exercise our teamwork skills, use source control tools, and debug our program and test code along the way. These skills are critical in the workplace so exercising them in an academic environment is valuable.