**Coffee Maker Quest**

**Ginger Mulky & Justin Papez**

**CS 1699 - DELIVERABLE 2: Unit Testing and Code Coverage**

For deliverable 2 we used JUnit and Mockito to write unit tests for Coffee Maker Quest. We based our tests on the requirements and test plan from deliverable 1 in addition to the provided source code for Coffee Maker Quest.

We initially faced some issues synching our development environments. For example, the input string “¡¢£¤¥¦§¨©ª«¬®µ¶” initially used in testRunArgsOutputStream() and testRunArgsReturnValue() passed for one of us and returned a Null Pointer Exception for the other. This could be an encoding issue in JUnit or Netbeans, or some other error in importing the code. It was solved by retyping the string instead of trying to copy it. We also changed the input string to a more common text string in the equivalence class in an attempt to determine the issue. This error demonstrates the importance of a unified development framework for testers and developers working together. In general, the Null Pointer Exceptions found during testing also indicate that the application should have more robust error handling, especially when errors may be inconsistent.

We also faced issues when testing equivalence classes. Our first group of tests missed some of the errors found in manual testing, such as the lowercase “n” not working correctly. To solve this, we rewrote the tests using parameters to pass in more values for equivalence classes and edge cases automatically. However, this did not change the result because the tests only checked that the function handles the input somehow, not that it handles it as expected. More specific tests were required to check functionality and confirm problems such as the inability to move north using lowercase “n”. The testMoveNorthLower() fails because the check for an “N” command in Game.java’s doSomething() method does not ignore case. Because the behavior is different for “n” and “N”, they are not being treated as equivalent by the program.

Another issue with unit testing is that it misses many of the defects found by manual testing because the code has not been written to meet all requirements. Without code to implement a help function, there can be no unit test to check that entering “H” or “h” works correctly. Thus unit testing alone is no replacement for verifying requirements. We wrote unit tests for these missing requirements, but all they do is fail automatically to show that code has not been implemented for that functionality.

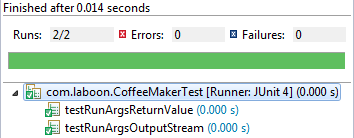
At least one method, run() in Game.java, was not easily testable via unit testing because of its external dependency on the Scanner. Because the method covers the whole game, it cannot be easily tested even with stubbing of the win and lose conditions. The run() method could call a different method to check win conditions which would be possible to test without initializing the Scanner.

A couple of other tests, testOutOfBoundsNorth() and testOutOfBoundsSouth(), also failed because they did not meet expected behavior based on the requirements. Because we based our tests on the requirements rather than just the code itself, we determined that the ability to move to a magical land should be considered a failure rather than a success. The requirements stated that the player should not be able to move north or south without an available door. Although the programmer handles the case differently than a move through an available northern or southern door, as testers we decided to count the unexpected message as a failure rather than a success. This conflict demonstrates the interpretability of requirements by different parties, and should be followed up by a meeting to decide the appropriate handling of the requirement.

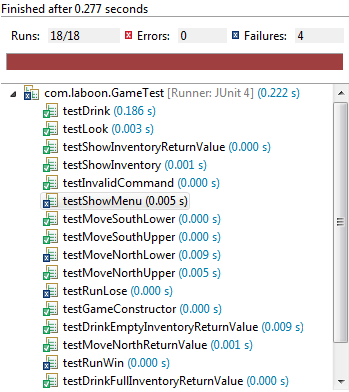
Our complete source code is located at <https://github.com/jjp75/CS1699-Deliverable2>

**Unit Test Screenshots**

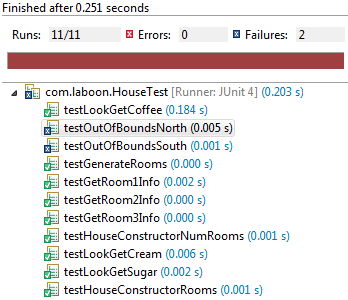
Coffee Maker Test



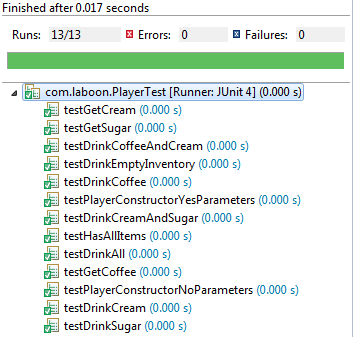
Game Test



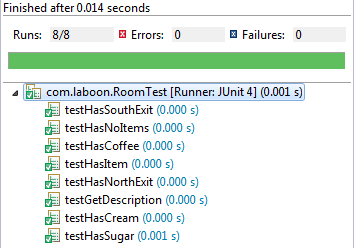
House Test



Player Test

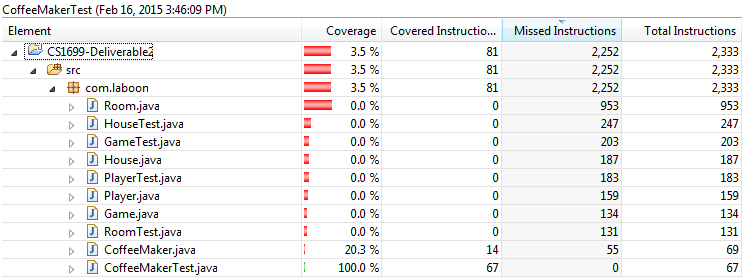


Room Test

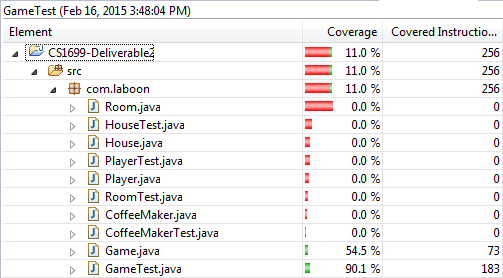


**Code Coverage**

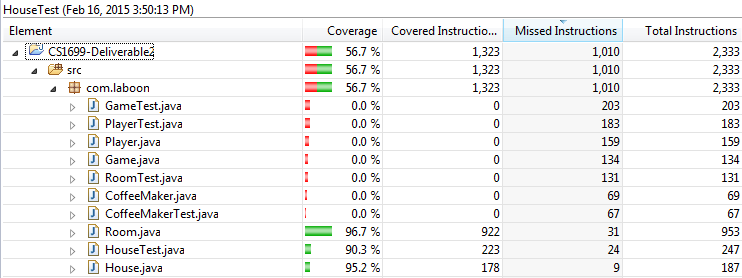
Coffee Maker Test



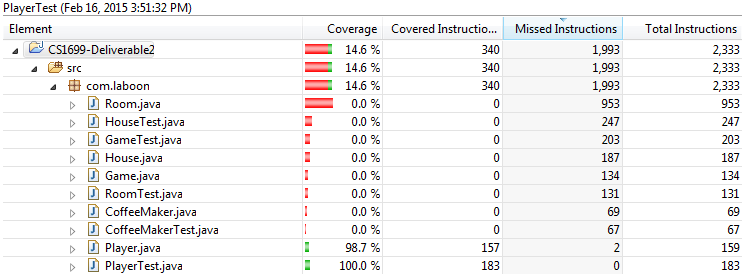
Game Test



House Test



Player Test



Room Test

