**CS1699 – DELIVERABLE 2: Unit Testing and Code Coverage**

**Project:** Coffee Maker Quest 1.0

**Authors:** Sheridan Zivanovich & Jordan McAleer

Some of the issues we faced stemmed from the fact that we were writing unit tests for code we did not create. For example, some methods do nothing besides print out some messages and return a constant value (e.g. CoffeeMaker.runArgs); is this something that should be captured in a unit test? We made the decision that, since the return value is unused and the console output is not verifiable (with the way the code is currently structured), it is not necessary to unit test methods such as this. We believe that methods like these present a good opportunity for refactoring, perhaps by changing the return type to *void*, for example.

Additionally, since we did not write the code ourselves, we found it was easy to fall into the trap of writing tests to match the code, rather than writing tests that capture what the code *should* do. We resolved this issue by remaining cognizant of the requirements specified in Deliverable 1, and translating these requirements from system requirements to requirements of particular classes and methods. Due to this, we ended up having a few failing unit tests.

* *HouseTests.moveNorthDoesNotChangeRoomWhenRoomHasNoNorthernDoor*This unit test fails because the current room description changes after *House.moveNorth* is called when the current room has no northern door.
* *HouseTests.moveSouthDoesNotChangeRoomWhenRoomHasNoSouthernDoor*This unit test fails because the current room description changes after *House.moveSouth* is called when the current room has no southern door.
* *GameTests.gameMovesPlayerNorthWhenGivenStringN*

This unit test fails because the *Game* object does not call *moveNorth* on the *House* object when the string “n” is passed to the *doSomething* method. That is, the command to move north (“N”) is not case insensitive.

However, we were at a loss as to how we should capture functionality that was completely missing from the code. For example, the Help command (by providing the game with the string “H”) is not implemented in the code. In all likelihood, this particular command would have simply written a message to the console, and thus would not have been easily testable anyway. We could apply test-driven development, writing an (initially failing) unit test to capture the functionality, then writing the code to support it. However, we felt that modifying the game code would have been outside of the scope of the deliverable.

The code is available at <https://github.com/sdzivanovich/deliverable2_repo>.

**NOTE:** The executed tests are split into two screenshots. The coverage report is captured in a single screenshot.





