|  |
| --- |
|  |
| CS1632 – DELIVERABLE 2 |
| Unit Testing and Code Coverage |

|  |
| --- |
| Philip Ni  https://github.com/phil-nye/CS1632/tree/master/CoffeeQuest |

1. Summary and Difficulties

This program fixes the problems with the original Coffee Quest game. Functionality is verified and validated through unit testing and test-driven development techniques. Throughout the process of developing this rewrite for the original program, I encountered several difficulties.

First, it was difficult to determine exactly what should be tested. Because certain aspects of my code improved of functions of the original that did not work, I had to determine what the expected behavior should be on my own. For example, the FUN-HELP functionality was not included in the original program. So, determining how the information should be presented took more time to implement than functionality that I could reference in the original. The requirements were helpful for giving me a guideline of what functions the components I added were expected to do.

Second, coming up mocks, doubles and stubs was difficult. I often had to restructure my program to accommodate the test methods. I had to very carefully plan out how the output of each function I was implementing should return. For example, I did not use any of my own objects as arguments in my methods. So, coming up with a way to include mocks in my program was a difficult task. I had to rewrite some of my methods to allow for stubbing mocks. This made the process of developing the game take much longer, as I had to make sure that return types were all consistent with what the test methods were expecting to receive. In these cases, it would have been helpful to write the test cases first.

Finally, because my main method (in CoffeeQuest.java) contains the driver for the game, I could not find a way to the main without actually having to play the game. In hindsight, the functionality I implemented in main could have been included in Player.java instead.

