CS 1632 - DELIVERABLE 2: Unit Testing and Code Coverage

16 February 2016

Evan Sheen
Cyrus Ramavarapu

GitHub: https://github.com/teirm/cs1653-deliverable2

Coding and Testing Memo:

Modelling the original CoffeeMakerQuest as a finite state machine enabled the rapid design of the sequel, CoffeeMakerQuest2. In this model, each room represents a unique state and the transition function between states depends on the player's choice of north or south. Rooms with searchable items have an additional transition upon a research request which would alter the player state to now include the item in the inventory.

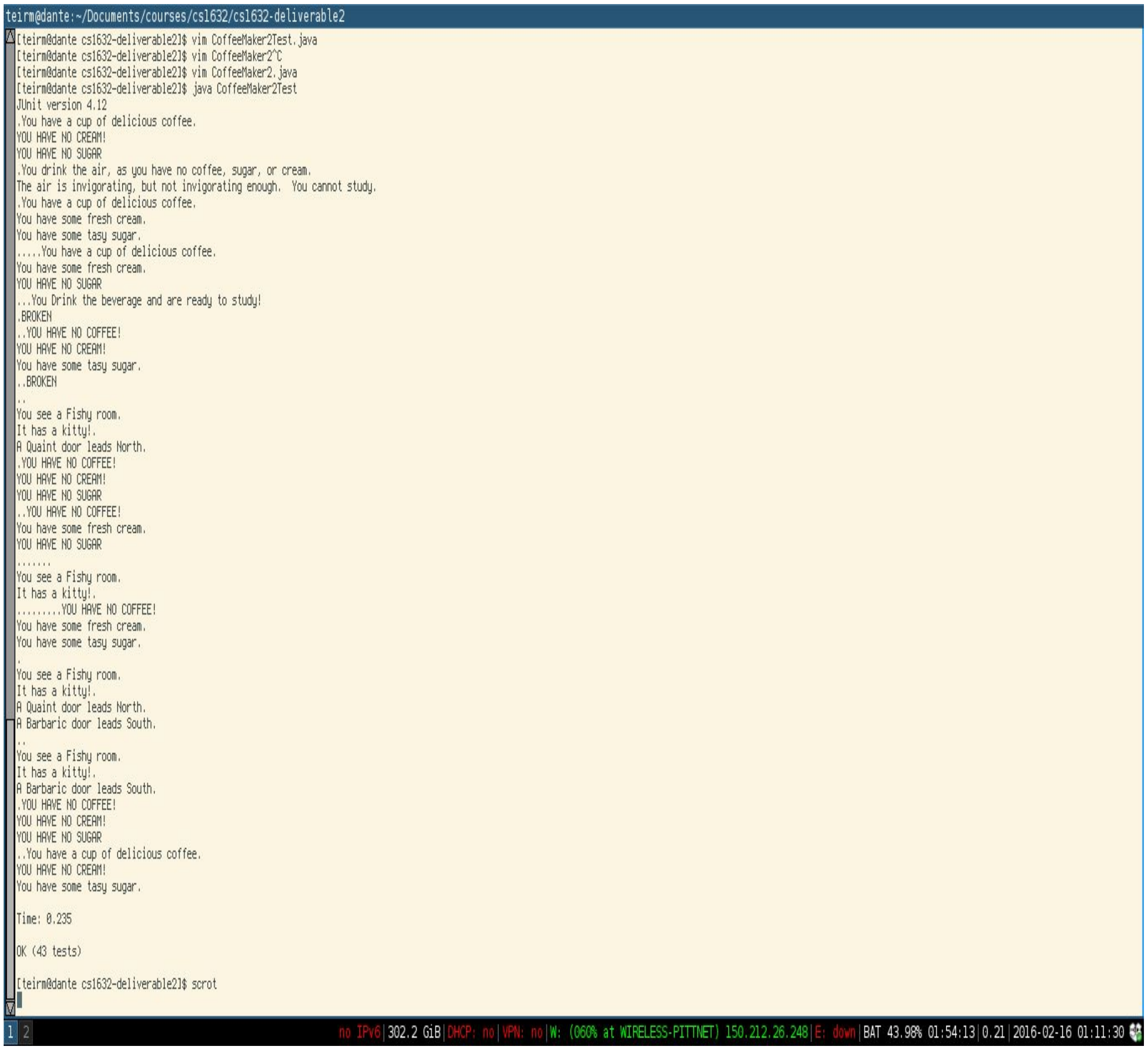
Using the available java data structures, the above model was easily implemented. An array of room objects represents the finite set of states and transitions between states are limited to only contiguous states. The array implementation also allows for the end points to be easily fixed. Additionally, implementing the user inventory as an ArrayList enabled checks to prevent duplicates. Each player input was then mapped to either a transition between states or a modification to the player or inventory.

Although replicating original CoffeeMakerQuest game was not challenging, testing the code proved much harder and revealed several problems. Initially, many of the methods in CoffeeMakerQuest2 were impure, such as methods that simply display the contents of an object. To make these methods easier to test, an integer value was returned upon completion which, within the test suite, could be asserted against to determine success or failure.

Having to create mocks and stubs was an additional challenge, but forced the code to be rewritten in a more modular fashion and with objects being passed into methods. All of these methods were also given a return value to be asserted against. A small source of concern during the creation of mocks was the use of an array of objects to represent the states of the game, which was not mocked as an aggregate; however, every method using room objects and associated methods were mocked, stubbed, and tested.

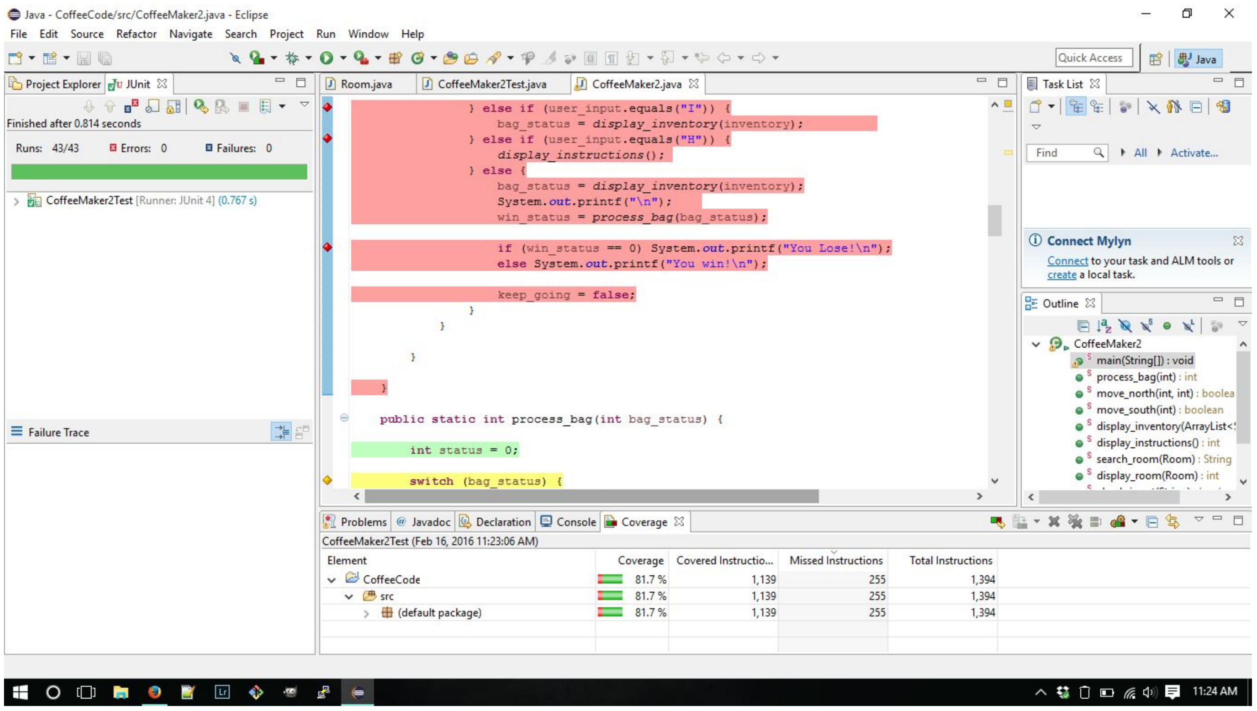
Lastly, when calculating the code coverage for the test suite the test code was included in the calculation, possibly skewing the coverage percentage. However, every method except main was tested, and the code within main is primarily declarations and a control loop for the game.

Test Screenshot



```
teirm@dante:~/Documents/courses/cs1632/cs1632-deliverable2
[teirm@dante cs1632-deliverable2]$ vim CoffeeMaker2Test.java
[teirm@dante cs1632-deliverable2]$ vim CoffeeMaker2^C
[teirm@dante cs1632-deliverable2]$ vim CoffeeMaker2.java
[teirm@dante cs1632-deliverable2]$ java CoffeeMaker2Test
JUnit version 4.12
.You have a cup of delicious coffee.
YOU HAVE NO CREAM!
YOU HAVE NO SUGAR
.You drink the air, as you have no coffee, sugar, or cream.
The air is invigorating, but not invigorating enough.  You cannot study.
.You have a cup of delicious coffee.
You have some fresh cream.
You have some tasy sugar.
.....You have a cup of delicious coffee.
You have some fresh cream.
YOU HAVE NO SUGAR
...You Drink the beverage and are ready to study!
.BROKEN
..YOU HAVE NO COFFEE!
YOU HAVE NO CREAM!
You have some tasy sugar.
..BROKEN
..
You see a Fishy room.
It has a kitty!.
A Quaint door leads North.
.YOU HAVE NO COFFEE!
YOU HAVE NO CREAM!
YOU HAVE NO SUGAR
..YOU HAVE NO COFFEE!
You have some fresh cream.
YOU HAVE NO SUGAR
.......
You see a Fishy room.
It has a kitty!.
.........YOU HAVE NO COFFEE!
You have some fresh cream.
You have some tasy sugar.
.
You see a Fishy room.
It has a kitty!.
A Quaint door leads North.
A Barbaric door leads South.
..
You see a Fishy room.
It has a kitty!.
A Barbaric door leads South.
.YOU HAVE NO COFFEE!
YOU HAVE NO CREAM!
YOU HAVE NO SUGAR
..You have a cup of delicious coffee.
YOU HAVE NO CREAM!
You have some tasy sugar.

Time: 0.235

OK (43 tests)

[teirm@dante cs1632-deliverable2]$ scrot

1 2          no IPv6|302.2 GiB|DHCP: no|VPN: no|W: (060% at WIRELESS-PITTNET) 150.212.26.248|E: down|BAT 43.98% 01:54:13|0.21|2016-02-16 01:11:30
```

Test Coverage



Java - CoffeeCode/src/CoffeeMaker2.java - Eclipse

Finished after 0.814 seconds

Runs: 43/43 Errors: 0 Failures: 0

CoffeeMaker2Test [Runner: JUnit 4] (0.767 s)

```java
                } else if (user_input.equals("I")) {
                    bag_status = display_inventory(inventory);
                } else if (user_input.equals("H")) {
                    display_instructions();
                } else {
                    bag_status = display_inventory(inventory);
                    System.out.printf("\n");
                    win_status = process_bag(bag_status);

                    if (win_status == 0) System.out.printf("You Lose!\n");
                    else System.out.printf("You win!\n");

                    keep_going = false;
                }
            }

        }

    }

    public static int process_bag(int bag_status) {

        int status = 0;

        switch (bag_status) {
```

CoffeeMaker2Test (Feb 16, 2016 11:23:06 AM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
CoffeeCode	81.7 %	1,139	255	1,394
src	81.7 %	1,139	255	1,394
(default package)	81.7 %	1,139	255	1,394