**CS1699: Software Testing**

**Ryan McDonald**

**Deliverable 5:**

**Developing Game Using TDD**

**Game Title: Escape the Game**

Given the opportunity to have free reign over the final deliverable and the fact that I decided to be adventurous and code up a test-based adventure game for another class (EngLit0255) I decided to use this very same project in conjunction with this class. While I could have done some sort of testing on code that I had already written, I decided it would be an interesting opportunity to embark on a quest, not for coffee, but for money, fame and power by implementing an amazing text-based adventure game that everybody will want to play through the art of Test Driven Development.

The plot of the game is genius: You’re a frat-bro who gets roofied at a party and wakes up in a sorority house and must escape. But, this amazing plot is not the focus here because, with Test Driven Development, the Tests come before the infrastructure which comes before the story. Focusing on the TDD aspect of creating the game, it lent itself well to what I was developing since I was coding it from scratch (admittedly borrowing ideas from CoffeeMakerQuest because its implementer is such a boss that he should be a professor or something).

As I trotted along on my quest to create test cases that fail first and then build actual methods that cause them to succeed, I found that it was flat out annoying. Typically when I code, I find my creativity and spontaneousness take over and code out large ideas before fully testing them. While this is certainly not good practice, TDD had me changing my entire way of coding. Not only did this go against how I do me, it also restricted creativity in terms of how I write my game in a sense. Every time I had another idea for something to implement in the game, it was a world of hurt in terms of how many more test cases I needed to write. But I persevered.

Eventually I managed to finish the underlying infrastructure of the game and left the story elements out for this stage of the project. I am planning to take the lazy coder’s approach and read in my game data from a file in order to build a map. Since the complexity of that would have involved a million more test cases amongst all of the time constraints with other obligations, I decided that 93 test cases of fun was enough fun for me.

Seeing as how I tested the code as I developed it, I was able to create test cases that would always pass. Just so I can sleep at night without the worry of nightmares due to bad code, I decided to run coverage tests with all of the Junit test cases that I implemented and I essentially verified that I had 100% test coverage in all of my classes (or really close to that). As opposed to when we wrote tests for Coffee Maker Quest and were aiming for 100% code coverage, I simply had 100% code coverage from the get-go and all I needed to do was maintain that coverage. Thinking about that was kind of interesting but it only held my interest for about 3 or 4 test cases. I’d rather code first and test later!

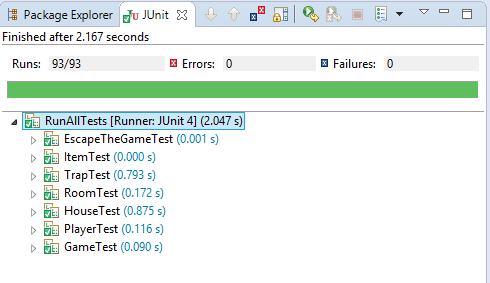
If I were to gauge whether or not this product can be released or not, I would certainly have to say that it is not quite ready yet. This is not due to the test coverage but due to the number of things I still have to implement such as a working main() and a well thought out story. If I were to gauge this in terms of reliability at this moment, I would say that I am fairly confident that the code written thus far is so sound that not even three angels descending from heaven humming the Barney theme song can match its soundness (don’t ask how I came up with that, I don’t even know).

Although, in just a few days, this revolutionary text-based adventure game will put even the famed Coffee Maker Quest to shame.

**Current Working Title :** Escape the Game

**Repository Location :** <http://github.com/ram757/Deliverable5>

**Unit Test Results:**



**Code Coverage Results:**

