CITY SIM 9003

CS 1632 – DELIVERABLE 2: Unit Testing CitySim9003

Author:
Anthony Poerio (adp59@pitt.edu)

Github URL:
https://github.com/adpoe/CitySim9003

For our second deliverable in CS1632, we were asked to write code and tests for a program called CitySim9003, given only its formal requirements list.

Concretely, CitySim9003 is a simulation program that details the actions of a driver moving through a small city (roughly modeled on Pittsburgh), until the driver leaves the area. When the driver leaves the city we've modeled, we output how many cups of coffee she has consumed, based on how many times her car has stopped at the location named "Coffee."

The program works using Java's Random Number Generator (RNG), and the RNG is seeded using an Integer that the user has passed in via the command line.

The core challenge of the project was to model our simulation program using object oriented design patterns which allow for easily testable code. I decided to model the city itself as a Graph, consisting of Nodes (locations), and Edges (streets). I then wrote a Driver class which outputs the details of what has happened at each simulation step for the user.

Modeling the program in this object oriented way was helpful, and it allowed me to write a separate testing class for each Object I used to compose the simulation model (Node, Edge, Graph, and Driver). Working incrementally, I was able to create an object and use mocks and stubs to ensure that each object itself worked correctly, assuming that the other pieces existed and gave me back the values I expected. I found this to be a pleasant way to work, and this test-driven development style was very satisfying and also gave me confidence that I was going in the right direction, and everything was working correctly before moving on.

The **biggest problem** I encountered was in testing the actual Driver itself. Because the simulation's goal is to output Strings to the console, I modeled the actual simulation method as a void method that printed output to System.out directly. I tried a variety of ways to test that the print statements were constructed as expected, but realized afterwards that there's not any way to test the validity of console output using JUnit (at least that I could find). Moreover, when I tried to test internal values of the City (Graph) object before and after each execution step, I began receiving errors and it caused my test to fail, even when the values were correct. It seems like if you print something to System.out, JUnit returns a 'null' value, no matter what else happens.

To remedy, I broke the simulation method into a smaller parts and tested that I was correctly identifying the expected 'next' state. This worked, and it's reasonable solution, I think. But I'd rather test all of the console outputs directly, to ensure that the user is actually seeing what I am expecting them to.

I also encountered issues with setting up Mockito, because I was unable to find the Jar file necessary from the Mockito site. But later, I was able to find it on https://code.google.com/archive/p/mockito/downloads, and once I got it setup, using mocks and stubs was straightforward, in my opinion. I think going forward, it would be helpful to have a walkthrough or example of how to setup Mockito from the command line for new users, as a suggestion for the assignment itself. Beyond that, I found the experience valuable and feel like it was a great introduction to using JUnit.


deliverable02 git:(master) ✗ sh run.sh
ALL TESTS PASSED