Refactoring Proposal

cs6015
Jeremy Rodgers

Prior to starting the MSD program, I dabbled in coding through an online boot camp. Every now and again I would get stuck and ask a friend (a U of U computer science graduate) for help. He and I would spend hours fixing my code and after the first few times of going through this exercise, a habit began to emerge that started to slightly annoy me. We would write a function, talk about how to make it efficient and correct, we'd implement it, and then he'd spend a few minutes "making the code look pretty". I was obviously tremendously grateful for the help, but at the time, I wanted to get things done and I thought he was just being a neat freak. Now, after hours and hours of being punished by messy, unclear and poorly labeled code, I realize just how important that step in his development process was.

I now try and do the same as my friend when I write code and set aside a minute or two after writing a function to clean things up a bit. Even after all of this, as is the nature of development, approaches change and slowly I creep towards confusing and ugly code. My Event Simulator is no exception and a good refactor could definitely help.

In my simulator design, I tried to use inheritance to avoid code duplication. I created a "Establishment" class which holds shared code between the Bank and Supermarket classes. I think this worked out well for the most part but I didn't fully utilize the rules of inheritance. I wouldn't want anybody to make an instance of the Establishment class without implementing a nextEvent() method which holds most of the logic for queuing and dequeuing Events. I didn't put this method into the Establishment class because the Bank and Supermarket implement them differently but I should at very least make it abstract. That will be a small but important change.

In addition, my nextEvent() class has turned into a bit of a mega function. It holds code that handles arrivals, begin service, and end service events. I think it would be much nicer if I broke this out into smaller functions and just had nextEvent be a driver method to call methods like handleArrival, handleBeginService, handleEndService and so on.

My main has also gotten a little unwieldy. Test cases have been commented out because the print to Std::out which garbled up my output for graphing. Obviously I still want to be able to run tests, so I could either create a separate executable for my testing or gather all my tests into one test method so I can just write test() in main to run my tests. Time permitting, I'd like to just figure out how to create the separate executable for testing while still having everything work in Xcode.

As I refactor, I'll also try and make name changes that make the code more readable if I want to come back to this code later. I think that should give me enough to work on over the weekend. Let me know if you have any questions or concerns.