Katie Robinson
Assignment 17
Refactoring

Project I will refactor: Simulators

Goals:

The biggest portion of my refactoring project will be reducing code duplication. I found the project unexpectedly time-consuming and wasn't able to spend as much time as I usually do writing careful code; instead, I did what Code Complete says *not* to do and started copying and pasting material from the bank simulation into the supermarket simulation. The result is difficult to read and redundant, and has already resulted in difficulty fixing errors. I plan to improve it by pulling code out from the two simulation methods into helper methods that I then call in the simulateBank() and simulateSupermarket() driver methods (as suggested in the Code Complete chapter on writing good methods). There will be a few key differences between the bank and supermarket simulations (such as the data structures I used and the manner in which I queued up customers) that I will have to isolate in the driver methods. I anticipate that isolating these differences in a way that minimizes redundancy and does not change the logic will be the trickiest part.

Of course, I also plan on reducing unnecessary whitespace, improving variable and function names, and improving comments (removing unnecessary ones and making good ones more clear). I ran out of time and did not make nice comments explaining each function, which I usually do while writing them; I plan to go back and do this in my refactoring. I will also add a little bit of simple input validation since I am taking command-line arguments.

Lastly, I want to add some sanity checks and assertions; I wish I had programmed more of them in as I developed the project, because there were a lot of pitfalls that smart assertions could have prevented. For example, I spent probably 45 minutes tracking down an error caused by accidentally modifying a parameter in a function when I passed it in by reference instead of constant reference; an assertion there that I wasn't modifying the parameter would have allowed me to fix it in under 30 seconds. Overall, there was some painful debugging for me in this project because I didn't follow the best coding practices, which is a good reminder that time put in up front generally saves a lot more time at the end!

Since this project is fairly simple, I am not going to write a big test suite to make sure my refactoring doesn't change the meaning of the code. However, I will record results from a variety of different inputs before I complete the refactoring and then make sure that at each point in my refactoring the numbers I get are the same; I feel that this is sufficient given the small scope and purpose of the project.