CSCI 3353 Object Oriented Design

Homework Assignment 1
Due Friday, September 9

Download the file *DiningHallSimulation.java*. This program performs a simulation of a dining hall. It makes the following assumptions:

- The dining hall has four cash registers.
- Each second, there is a 18% chance that a customer will be ready to check out.
- A customer will always enter the shortest checkout line.
- Each customer has between 1 and 10 items. Each item takes two seconds to scan, and payment takes another 10 seconds. Thus if a customer has n items, it will take 2n+10 seconds to check out.

The simulation runs for 50,000 seconds. Afterwards, it prints the number of customers serviced and the average wait time per customer for each register.

This code does its job reasonably elegantly. However, it has a fundamental design flaw: It is organized in a non-object-oriented way. There is only one class. Your job is to refactor it using the best object-oriented style you know. In particular, I want you to focus on creating modular, well-designed classes. What is the responsibility of each class? What methods should each class support? What work should each method do? Your design should follow these basic principles:

- Work should be done by the class that knows best how to do it.
- A class should divulge as little as possible to other classes.

I do not want you to come up with a new way to do the simulation. Your revised code should generate the same output as before. All I want you to do is refactor the existing code – that is, to find a way to do essentially the same thing, but in a more object-oriented way. You need to **thoroughly** understand what my code does; otherwise, you will not be free enough to make the necessary changes. If you have any questions, please ask me.

WARNING: This assignment is not as simple as it might seem. The easy part is to realize that you need to create the classes *Customer* and *CashRegister*. The hard part is to figure out what responsibilities to assign to each class. This is the sort of thing that you need to mull over, which means that it is unlikely you will come up with a really good solution in a single sitting. Think about what makes my code so poorly designed, and then figure out how to do it better. You might be surprised at the dramatic change that a good refactoring can produce.

WHAT TO SUBMIT:

You should have three files, named *BetterDiningHallSimulation.java*, *Customer.java*, and *CashRegister.java*. Please do NOT place them in a package. Compress the three files into a single zip file, and submit the zip file to Canvas.