CSCI 3353 Object Oriented Design
Homework Assignment 3
Due Friday, September 25

In this assignment you will add the following two features to the HW 1 dining hall simulation program:

- The dining hall currently has 2 cash registers, and the cashiers are assumed to be equally competent. Let's call these cashiers "normal speed" cashiers. You must revise the code so that the dining hall can also have "fast" cashiers, who work twice as quickly. Moreover, a user should be able to configure the program so that it simulates a dining hall with N normal speed cashiers and M fast cashiers.

- Customers currently choose between 1 and 10 items, equally likely. This is known as a "uniform" distribution of size 10. In your revised code, a user should be able to specify that the distribution is uniform of size N, for any N>0. In addition, a user also should be able to specify that the distribution is a "bimodal" distribution of size N. In a bimodal distribution, customers choose either a small number or a large number of items, but nothing in the middle. In particular, let's say that the number of items will be uniformly distributed, but only between the ranges [1 .. N/4] and [3N/4 .. N].

You should use the strategy pattern to implement each of these features. For the first feature, you need to have a strategy interface (call it *Cashier*) with two strategy classes (call them *NormalSpeed* and *Fast*). The *CashRegister* class will call its strategy object when it needs to "elapse a second". The normal speed cashier does what the current program does, and a fast cashier will do twice as much during that second. (For simplicity, if the cashier finishes a customer during the first half second, it need not move to the next customer until the beginning of the next second.)

For the second feature, you need to have a strategy interface (call it *ItemDistribution*) with two strategy classes (call them *Uniform* and *Bimodal*). The constructor for each strategy class will have an argument N that indicates the size of the distribution.

In order to simplify my grading task, I want everyone to begin from the same HW 1 solution, namely my "even better" solution to HW 1. You can download a zip file from Canvas that contains the code.

Your revised *main* method should request the number of normal and fast cashiers from the user, as well as the type of customer distribution and its size. It can then pass this information into the *DiningHall* constructor. For example, the method should have code that looks something like this:

```
int normal = ...        // get from user
int fast   = ...        // get from user
String distType = ...   // get from user
int distSize = ...      // get from user

ItemDistribution id;
if (distType.equals("uniform"))
     id = new Uniform(distSize);
else
     id = new Bimodal(distSize);

DiningHall h = new DiningHall(normal, fast, id);
```

Note that the *main* method creates the requested *ItemDistribution* object and passes it to the *DiningHall* constructor. But why does the method pass that strategy object to the *DiningHall* object? After all, *DiningHall* doesn't use that object, *Customer* does. The reason is that *main* doesn't know about customers—instead, customers are mediated through the *DiningHall* and *CashRegister* classes. Consequently, *main* has to pass the strategy object to the *DiningHall* object, who then passes it to each *CashRegister* object it creates, who in turn pass it to each *Customer* object they create.

Which class should create the *Cashier* strategy objects and when? You need to make that decision.

You may have to make significant changes to some of the classes in order to include these strategies. This is typical in software design. Do whatever you feel is appropriate. If you do make a significant change, please use a comment to indicate what you did.

When you are testing your code, you might want to vary the customer arrival percentage. A value of 19% works well for two normal-speed cashiers, but it won't tell you anything interesting for a simulation of a lot of cashiers. Try bumping up the percentage until you get interesting results. I certainly will when I test your code.

You should submit files for the following ten classes to Canvas:

- The main class, which you should call *HW3Simulation*
- The revised versions of the HW1 classes: *DiningHall*, *CashRegister*, and *Customer*
- The cashier strategy interface and classes: *Cashier*, *NormalSpeed*, and *Fast*
- The distribution strategy interface and classes: *ItemDistribution*, *Uniform*, and *Bimodal*.