Lab 9
Due 12 November

CSCI 136: Fall 2017
Handout L9
05 November

Simulating Businesses

1 Notes

The assignment this week is to complete **13.7 Laboratory: Simulating Business** from *Bailey pg. 341*. In prior labs we have developed programs to process data or solve problems using the data structures and programming concepts we discussed in lecture. This week, we will explore the ways that data structures appear in the physical world, comparing two scenarios:

1. In many supermarkets and retail stores, customers select and wait in one of several available queues to purchase items. Once in a queue, they wait until the cashier has finished serving the customers ahead of them.
2. In many banks, customers wait in a single queue. When a teller becomes available, the customer at the front of the queue is serviced by that teller.

We will build a simulation for both scenarios and then measure the performance of each approach.

2 Lab Program

Instructions for this assignment can be found on pages 341–342 in *Bailey*. We have posted some suggestions below. We have also posted a sample abstract base class (`Simulation.java`) to help you design your program to simulate the two scenarios.

Please modify the sample files in any way(s) that you wish. You may find the need to add additional methods or variables. You may choose to implement functionality common to the single-queue and multiple-queue simulations in the abstract base class, or you may choose to place functionality in one or more sub-classes. The appropriate location for the implementation of individual methods will depend on your program design.

Suggestions. You should feel free to deviate from any of these suggestions. The only requirement is that you have a well-organized, well-commented program that simulates the two scenarios described above.

- The approach suggested by the assignment is to maintain an event queue, where each event is a customer arrival, a teller becoming free, etc. You may find it easiest to generate a single `Customer` class and store only `Customer` objects in the event queue.
  - A `Customer` must implement the `Comparable` interface to be stored in a Priority Queue. `Customers` should be ranked by their "arrival" time.
  - Think about what information is necessary to answer the thought questions. You may need to know when a `Customer` "arrives", when a `Customer` begins receiving service from a teller, how long it takes a teller to service a `Customer`, etc.
- You should maintain an `Integer` counter that keeps track of the current time step of your simulation and your simulation should proceed in rounds. During each time step:
  - Available `Customer` objects may potentially join a teller queue. (You may want to pre-populate your event queue with `Customer` objects, but you should ignore them until the time step matches their designated "arrival" time.)
  - Each teller should make progress towards serving the first `Customer` in their line (if one exists).
  - You may also need to update simulation statistics in order to answer the though questions.

- You may find it useful to implement `toString()` methods for each class so that you can monitor the state of your queues and debug.
- Your simulation should end when (1) there are no more `Customer` objects in your event queue, and (2) all `Customer` objects have been serviced by all tellers.
- You may want a separate class or classes that do nothing but run your simulation (e.g., a main method that accepts command-line arguments and runs the chosen simulation with the chosen variables).

2.1 Thought questions

Please answer thought questions 1–4, and place your answers in the comments at the top of `BusinessSimulation.java`. Some notes:

- Questions 1 and 2 ask you to compare simulation statistics. Please give the parameters that you used for your simulations (number of customers, number of tellers, min/max service times, seed, etc.) in addition to the statistics the question asks.
- For thought question 2, the *average wait time* is the difference between the time that a `Customer` appears in the simulation and the time that the `Customer` first begins receiving service from a teller. The wait time should not include the time that a `Customer` is actively receiving service.
- For thought question 4, you should consider the multiple-queue simulation (e.g., the supermarket model) and the single-queue simulation (e.g., the bank model) separately. For the single-queue simulation, imagine that the set of tellers is partitioned so that Customers with short service times cannot be helped by tellers that serve Customers with long service times and vice versa.

2.2 Deliverables

Create a lab directory named `unix-lab9` (or `unix1-unix2-lab9` if working with a partner) that includes the following:

1. Your lab solution in appropriately named `.java` files.
2. Your answers to thought questions 1-4 in a file named `README.txt`.

As in all labs, you will be graded on design, documentation, style, and correctness. Be sure to document your program with appropriate comments, including a general description at the top of the file, a description of each method with pre- and post-conditions where appropriate. Also use comments and descriptive variable names to clarify sections of the code which may not be clear to someone trying to understand it.