Simulation

In class we went over program “QueueDemo.java” which illustrates a simple and not-very-realistic simulation of a store with arriving and departing customers.

In class we performed experiments that showed:

- assuming that at each time unit either somebody enters the queue or somebody exits the queue, the simulation time (time until the queue becomes empty) varied wildly, sometimes taking only a few dozen time units, sometimes hundreds, sometimes hundreds of thousands
- the average length of the queue (we added some code in class to compute this) also varied wildly, sometimes fewer than 10, but sometimes several hundred or more

There are many problems with this simulation, but here are two specific ones that we will be concerned with. First, it does not take into account the possibility of simultaneous events. For instance, someone could arrive at the same time that someone else is leaving. Second, it doesn’t allow for the possibility that there could be several check-out lines, giving the customers the choice of standing in a shorter line. (Other problems that we will not deal with include variations in the amount of time each customer needs to spend in the store, variations in the amount of time needed to process each customer in the check-out line, variation in the number of people who can enter the store simultaneously, and so on.)

Create a more realistic simulation of a store. Use two ArrayLists to represent two separate check-out lines in the store. Just as in the sample program, assume customers are labelled with consecutive integers as they arrive at the store; the customer’s number is what gets saved in the ArrayList.

In the first ten time units, ten customers arrive and are immediately added to the two check-out lines. No one leaves the store during the first ten time units (this is the same as in the sample program shown in class). The rule for choosing which check-out line a customer goes to is very simple: if check-out 1 has size less than or equal to check-out 2, the customer gets in line 1; otherwise the customer gets in line 2. Therefore, each check-out line will have five people after the first ten time units have passed.

At each time step beginning with 11, there is a 2/3 probability of a new customer arriving. Also at each time step beginning with 11, there is a 1/2 probability for each of the two check-out lines that a customer will leave. (NOTE: you will need to generate three random numbers — one to determine whether or not a new customer arrives, a second one to determine whether or not
someone leaves the first check-out line, and a third to determine whether or not someone leaves the second check-out line.) Obviously, if a check-out line is empty then nobody can leave it, even if the random number says it’s okay — you will need to use the “isEmpty” method to make sure there’s something in the queue before you try to remove anything from it.

Here is some sample output to show you what could happen:

```
Time 10: Customer 10 enters checkout 2
Time 12: Customer 2 departs from checkout 2
Time 13: Customer 11 enters checkout 2
Time 14: Customer 1 departs from checkout 1
Time 14: Customer 4 departs from checkout 2
Time 15: Customer 12 enters checkout 1
Time 15: Customer 3 departs from checkout 1
```

Observe that there may be times when nothing happens (such as time 11) and there may be times when several things happen (at time 14 two customers departed; at time 15 a customer arrived and a customer departed). This is because three different random numbers were used to determine when people arrive and when people leave. Observe that it is no longer the case that customers leave the store in the same order that they arrived — in the above example, customer 1 had to wait in check-out line 1 due to an unlucky draw of the random number, while customer 2 got to leave the check-out line 2 because it got lucky with the random numbers.

Your Java program should keep track of six things and print them out at the end:

- Total amount of time elapsed until both lines are empty
- Total number of customers served
- Average length of the first check-out line, measured starting from time 11
- Average length of the second check-out line, measured starting from time 11
- Maximum length that is ever reached by the first check-out line over the course of the entire simulation
- Maximum length that is ever reached by the second check-out line over the course of the entire simulation

**Sample Output** (obviously you will not get the same numbers due to the random number generator):

```
$ java CheckOut
Time 1: Customer 1 enters checkout 1
Time 2: Customer 2 enters checkout 2
Time 3: Customer 3 enters checkout 1
Time 4: Customer 4 enters checkout 2
```
Time 87: Customer 63 enters checkout 2
Time 87: Customer 62 departs from line 1.
Time 87: Customer 63 departs from line 2.

Elapsed time: 87
Customers processed: 63
Line 1 average: 2.0641025641025643
Line 2 average: 1.7307692307692308
Line 1 max: 8
Line 2 max: 7

**Important:** Your program should not consist of merely a `main` method. You should create instance variables, a constructor, and other useful methods as you have been doing for other assignments. Your program should use methods for any pieces of code that get used in more than one place, and it should use methods with parameters for any sections of code that are almost identical except for the names of the variables used. See file `Lab9Skeleton.java` for suggestions.

Run the program once inside the `script` environment to show me the output. If the output is very long, terminate the script and start over — don’t waste paper printing out a long, boring output.

Run the program ten times and create a table showing, for each of the runs, the six values I asked for. What is the average of the average queue length over all ten experiments? What is the average value of the maximum queue length over all ten experiments?

**If You Want to Do More...**

The simulation can be made more realistic if we do the following: start with just one check-out line. If its length ever becomes greater than 10, open a second check-out line and evenly divide the elements in the first line between the two lines (you will have to remove integers from the first check-out line and add them to the second). Assume that this redistribution of customers all occurs in a single time step. Print out a message saying when a second queue opens up. If there are two check-outs open and one of them becomes empty, close down the empty check-out and go back to just one check-out line. Print out a message saying when you close down a queue. You may need to play around with the random number probabilities to get this to produce reasonable output (if we make the probability of leaving the queue too large, then we will never need a second queue; but if we make it too small then the simulation may run for a very long time).

**Checklist:**

- [ ] Hand in printouts of all code. Header comments should include your name, the lab number, the date, and a description of what the program does.
- [ ] Be sure to include descriptive comments about all code, including comments for each method and instance variable that describe what the method or variable does.
- [ ] Run your program once and print the output. To do this, type:
Run ten experiments (don’t print them out) and create a table showing, for each experiment, the elapsed time, the number of customers served, the average length of each queue over the course of the simulation, and the maximum length of each queue over the course of the simulation. Find the averages for each of these over the ten experiments and hand in your table and the averages.

Sign your name on the programs and the output for the honor code.