ENGI 7894/9869 Assignment 2

Due: June 14, 2016

This assignment will require you to implement a concurrent program in Java. Please make sure that you have the computing environment for building and testing your program properly configured on your computer before beginning.

In this assignment, you are asked to implement two different types of concurrent queues to determine which is most efficient.

The first queue should be implemented in a class called `BankQueue`, and is modeled after the lineup found in most banks. In such a queue implementation, there is only one lineup, and as soon as one teller becomes free, the first person in the queue proceeds to be served.

The second type of queue should be implemented in a class called `GroceryQueues`, which will be array of type `GroceryQueue`, and is modeled after the lineups in many grocery stores and supermarkets. In this queue implementation, there will be several lineups, one for each cashier. When a customer arrives, they join the queue with the fewest number of people waiting. If there are two such queues, it chooses one at random.

For the purposes of this assignment, you may assume that it takes between 60 and 300 seconds to serve any customer (uniform distribution), and that new customers arrive between every 20 and 60 seconds (also uniformly distributed).

The constructor for the `GroceryQueues` class should take the number of `GroceryQueue`'s and the maximum length of each queue (excluding the customer being served) as input parameters.

Similarly, the constructor for `BankQueue` should take the number of tellers and the maximum length of the single queue as parameters.

In the case of `GroceryQueues`, if a customer arrives, and no queue exists with any space, then he/she waits until a queue becomes available until a maximum of 10 seconds and then promptly leaves. For the `BankQueue`, a customer departs immediately if the queue is full.

You should also implement a `Customer` class which at the very minimum keeps track of the customer's arrival time, and the time needed for it to be served. If it was not served because all queues were at maximum capacity, then this fact should also be recorded. There should also be a `QueueSimulator` class which accepts events from customers, contains a clock which ticks every second, and processes any events that occurred in the last clock tick.

Your main program should take as parameter the number of minutes you wish to simulate the two queue types, and for each type of queue, output the total number of customers that arrived, the total number of customers who were forced to leave without being served, the total number of customers served, and the average amount of time taken to serve each customer.

You are expected to submit your source code, along with the program's output for 2 hours of the simulation when there are exactly 3 tellers in the case of the `BankQueue` class and 3 cashiers (and thus 3 queues) in the case of the `GroceryQueues` class, and when the maximum length of a `BankQueue` is 5, while the maximum length of a `GroceryQueue` is just 2 (this excludes the customer being served).

You should use synchronization mechanisms such as locks, semaphores and monitors to implement your solution.

2. In the client-server example given in the article on Proof Outline Logic by Dr. Norvell, a proof outline is given. Suppose we are using Java, and we let $f(x) = x^4$ and $g(x) = \sqrt{x}$. Will the proof outline be correct in this case? Why or why not? If so, say why, and if not, say how the precondition/postcondition may be changed to make it correct.