Assignment 2

ENGI 9869/7894 Adv. Concurrent Programming
Electrical and Computer Engineering
Memorial University
Due on June 15 , 2019 11:59 PM

Note that the work that you turn in for this assignment must represent your individual effort. You are welcome to help your fellow students to understand the material of the course and the meaning of the assignment questions, however, the answer that you submit must be created by you alone. Assignments will be assessed on your efforts and approach you are using for particular solution.

1 Concurrent Programming

This assignment will require you to implement a concurrent program (in Java or any of your prefer language that supports Multithreading). Please make sure that you have the computing environment properly configured for building and testing your program before begin. You are required to simulate two different types of queuing systems found in daily life, namely: “Drive Through” and a “Grocery Store”.

1.1 Drive Through [50]

The ”Drive Through” queueing system should be implemented in a class called DriveThrough and is modeled after the lineup procedure you observe in most drive through services. In such a queue, there are one or more lines to dispatch/place the order whereas only one line for making payment and order pick ups. Multiple actions required to serve one customer by crew member(s) i.e. taking order, taking payments, delivering ordered items. One crew member can work on three stations sequentially or three different crew members can stay on their assigned stations. You are required program the following specifications in your solutions.

1. You should implement a simulator that will simulate the drive through handling scenario, where each station (Dispatching, Payment, Delivery)of is simulated by its own thread(crew member). The process of arriving customers (joining a queue) will be processed by a separate thread. And your queue should be thread safe as the queue will be accessed by multiple threads receive the orders and dispatch them (ensure the safe writes to

your queue dispatching orders). For this example if we have two order dispatchers then primarily two queues are merging into one queue. [10]

2. Create a customer class for this implementation (and any other classes if you need to) that will keep track of the customer, at the very minimum keeps track of the customers 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. [10]

3. You should use synchronization mechanisms such as locks, semaphores and monitors to implement your solution to successfully join two queues without conflict.[10]

4. Make your solution fair with both dispatchers. Both should same amount of customers. (hint: create a separate thread assigning customers to two different dispatchers) [10]

5. Your simulation need a simulated clock (to reduce the run time) for each N minutes on your simulated clock. Your program should receive the value of N as the command line parameter at the start of the execution. It should output the total number of customers that arrived, the total number of the total number of customers served, and the average amount of time taken to serve each customer. [10]

1.2 Grocery Stores [50]

The second type of queue should be implemented in a class called GroceryQueue, which is modeled after the customer checkout queue system in the popular grocery stores e.g. Dominion, Walmart etc.. Here we have multiple checkout station in the store and each checkout station has its own queue. When a new customer proceed to checkout, it is the customers choice to select a queue among the available ones. For this assignment you can consider, the customer will always select the queue that has the lowest number of customer waiting. If there are multiple queue with lowest number of customer waiting, the new customer can choose one randomly.

1.2.1 Implementation Requirements

You may assume that it takes between 300 and 600 seconds to serve any customer (uniform distribution) on both of **1.2** , and that new customers arrives between every 50 and 100 seconds (also uniformly distributed). Also, both type of the queues has limited capacity. If a customer arrives and see the capacity of the queue(s) full, customer waits for a maximum of 20 seconds. If no space become available within 20 seconds, he/she will move around and and come again after 600 seconds and again try to join the queue, second time he/she waits for 40 seconds and keeping in consideration of his/her second try he/she leave the store without service if queue is still full.

(a) You should implement a simulator that will simulate the customer handling situations for both of the cases mention above, where each of the serving stations and checkout stations are simulated by its threads. The process of arriving customers (joining a queue) will be processed by a separate thread. And your queue should be thread safe as the queue will be accessed by multiple threads to add and remove customers (ensure the safe writes to your queue). [20]

(b) Your simulation need a simulated clock (to reduce the run time) and run each simulation separately, for each N minutes on your simulated clock. Your program should receive the value of N as the command line parameter at the start of the execution. 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. [20]

(c) Create a customer class for this implementation (and any other classes if you need to) that will keep track of the customer, at the very minimum keeps track of the customers 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. [20]

(d) You should use synchronization mechanisms such as locks, semaphores and monitors to implement your solution. You can create two separate programs to simulate two separate queue, or you can put both in one program and use another command line parameter to execute one of the simulation at a time. [10]

(e) In case of 1.2 create a self-service queue with capacity of serving two customers at once(two self checkout terminals) and consider the queue for customers unable to join DominionQueue. Assume the self service takes 400 to 600 seconds for one customer. How many customers you can save with the implementation of this queue(consider the simulation details 1.4)? [30]

1.3 Submission Details

You are required to submit your source code, along with the programs simulation for 4 hours (240 minutes or 14400 seconds) of the simulation when there are exactly 3 checkout stations (and thus 3 queues) in the case of the DominionQueues class, and when the maximum length of a DominionQueue is just 2 (this excludes the customer being served). You must ensure you add the N as command line parameter so that while testing we can run it for any number of minutes we want.