**Lab 5b – A Bank Simulation**

In this lab, you will be creating a program simulating the operation of a bank: Customers enter the bank, and there are one or more tellers; if a teller is free, that teller serves the customer. Otherwise, the customer enters the queue and waits until a teller is free. Your program should accept the following inputs:

* The total simulation time as an *integer*
* The average processing time for a customer as an *integer*. You may assume that a given customer will take anywhere from 40% to 160% of the average time at the window/teller.
* The arrival rate for customers which is a *double* value indicating the expected number of customer arrivals per unit of time. E.g., a rate of 1 means a new customer arrives every unit of time, 2 means that two new customers arrive every unit of time, 0.5 means 1 new customer arrives every two units of time (i.e., half a customer arrives every unit of time, etc...)
* The number of tellers at the bank as an *integer*

In a later lab, you’ll modify your created simulation to explore how data structures might help in serving “more important” customers before “less important” ones. So please make sure you complete your simulation soon.

### *Part 1: Studying the airport service counter simulation program*

### There are three Java files in your lab folder that make up a complete airport service counter simulation. Use these files as a model for your bank simulation program. *The most challenging part of this lab is understanding the provided simulation so, before you start working on this lab, make sure you run it many times with a variety of inputs* *and dedicate PLENTY of time to understanding the code.*

### *Part 2: Creating the Customer and CustomerQueue classes*

Once you've finished studying the model simulation, create two Java files: a *Customer* class modeled on the *Passenger* class and a *CustomerQueue* class modeled on the *PassengerQueue* class with all necessary changes. Replace the *ZHLinkedQueue* with your own queue implementation (or Java’s queue implementation if you prefer). Both files should compile without errors.

***Part 3: Creating the bank simulation program***

Complete the bank simulation by creating an appropriate simulation class modeled on the supplied *AirlineCheckinSim* class and using the *Customer* and *CustomerQueue* classes you developed earlier in this lab. Follow the directions below, debug your code and run the program several times with different input values. Make sure to read and understand the remainder of the write-up before completing the bank simulation.

Note that in the bank simulation, we're requesting the user to input the *average* processing time (let us call it *avgProcessingTime*) and not the *maximum* processing time (*maxProcessingTime*) as in the airline simulation. Java has a class called *Random* which can generate various types of random values. (Check the Java API online). In the airline simulation, we used the following line to generate a random processing time less than or equal to *maxProcessingTime*:

**processingTime = 1 + (new Random()).nextInt(maxProcessingTime);**

The method call nextInt(int n) generates a random integer between 0 and *n* − 1 (inclusive), so adding one to that makes the generated random number fall between 1 and *n* (inclusive).

In the bank simulation, we need to generate a random integer between two double values: 0.4 × *avgProcessingTime* and 1.6 × *avgProcessingTime* so the nextInt method won't be very helpful here. Let's refer to the range bounds as *lower* (which is equal to 0.4 × *avgProcessingTime*) and *upper* (which is equal 1.6 × *avgProcessingTime*). We can use method nextDouble to generate a value between 0.0 and 1.0. If we multiply that value by the range (*upper* − *lower*) we will get a double value between 0.0 and (*upper* − *lower*); adding *lower* to the result will finally give us a value between *lower* and *upper*. Finally, we cast the result to integer as shown below:

**processingTime = (int) ((new Random()).nextDouble() \* (upper - lower) + lower);**

At the end of the simulation, your program should print the statistics of the simulation including:

(1) total number of customers served,

(2) average wait time for customers,

(3) number of customers in the queue at the end of the simulation,

(4) the number of customers served by each teller, and

(5) the total amount of idle time for each teller

As with the airport simulation program, your program should also have a boolean variable that will print details of the simulation if set. Note, however, that the bank simulation program requires only one queue (so you'll need to remove one of the two queues from the airport simulation and modify the methods accordingly) but might have more than one teller to serve customers (number of tellers is input by the user).

Try to change one thing at a time; first, rename the files and change comments and code appropriately to simulate the bank; second, change the method enterData to request the desired information from the user; third, reduce the number of queues to one; and finally, allow for multiple tellers. Make sure you finish this lab by the specific deadline.

Below I am including a couple of sample input/outputs scenarios (with all details of the simulation printed out). Your output should be formatted exactly as shown below but please remember that this is a simulation which uses random numbers so your output might differ for the same inputs.

|  |
| --- |
| **INPUT FOR SCENARIO I**    **OUTPUT FOR SCENARIO I** |
| **INPUT FOR SCENARIO II**    **OUTPUT FOR SCENARIO II** |