COSC 311, Project #3
Due Date: October 26, 2023

A research center provides remote access to its computing facilities via a bank of modems. A modem (from this modem bank) is accessed by dialing a telephone number which is made public to the research community. Suppose that the research center has installed a system that queues phone calls when all modems are busy. Write a Java program to simulate the modem bank using the following simulation parameters: length of simulation, average time between dial-in attempts, number of modems in the bank, average connection time, and the size of the queue.

In this project, you will have two different queues: a priority queue for organizing the events (dial-in or hang-up), and a regular queue to queue the users who are waiting for modems to become free. The program must maintain a clock, and at each time unit the event queue is checked to determine if any events occur at that time. There are two possible events: dial-in (users dialing), or hang-up (users leaving). All the events scheduled for the current time unit are removed from the event queue and processed. A dial-in event is placed in the waiting queue until a modem becomes available. A hang-up event signals that a modem becomes free. When a user is finally given a modem, the connection time is calculated and added to the dial-in time. This gives the hang-up time for that user. A hang-up event for that time will be added to the event queue. A key step in this simulation is to determine, at each unit of time, how many users dial in and the length of the connection for each. Empirical studies have shown that the arrival patterns of dial-ins as well as the connection times are random following a Poisson distribution.

PROGRAM INPUT: The simulation parameters are entered interactively and include:

- the length of the simulation,
- the average time between dial-in attempts (a number between 0 and 1),
- the average connection time (a positive integer less than 10),
- the number of modems in the bank, and
- the size of the waiting queue or -1 for an infinite queue.

The program must prompt the user for each input, and each input is entered on a separate line. At the end of each simulation, the program asks if another simulation is desired. If the answer is yes, the program prompts the user to input a new set of simulation parameters.

PROGRAM OUTPUT: The outputs are printed both on the screen and to a report file (report.txt). The screen output for the current simulation is: the percentage of time the modems were busy, the average wait time, and the number of users left in the waiting queue when the simulation ends. Following the execution of the last simulation, the following message is printed to the screen: “Simulation is complete: Summary in file report.txt”. Additionally, you must print a message when an incoming call is dropped because the waiting queue is full. This might occur when the waiting queue size is finite.

The output written to the file “report.txt” consists of the set of simulation parameters, the percentage of the time the modems were busy, the average wait time, and the number of users waiting in the waiting queue when the current simulation runs out. The output for each simulation should be printed on a single line, resulting in a tabular listing. A header line should be written to the file at the beginning of the program to label the columns, and possibly a trailer section should be written to the end of the file, describing the meaning of the columns in the table.

PROCESSING REQUIREMENTS: Make allowance for an infinite queue. If a dial-in comes when the waiting queue is full, discard the request and print an error message on the screen (the error message must include the user id as well as the simulation time). Real-number output should be specified to two decimal places.

IMPLEMENTATION REQUIREMENTS:
- This project must have at least four classes: an event class, a regular queue class that is implemented using a **circular array**, a priority queue class that is implemented using a **singly linked list**, and a class that has your driver program.
- Both queue classes must be **generics.**
- The program must be robust and not crash if the input data type is incorrect.

SUBMISSION: Please submit a zip file containing the following to the class website:
- Source codes (well documented)
- A file containing a sample run that shows you have run the program, and shows the resulting screen output for wo sets of simulation parameters
- The report file (report.txt) for the sample run above
- Compress your Eclipse project folder, including all the material listed above, and submit it to the class website (see below)

Important note: Test your program carefully prior to submitting it, because you **can only submit once.** Compress your Eclipse project folder (the entire folder) containing all the source codes (all the classes), the file containing sample screen output, and the simulation output file "report.txt" and then submit that zip file on the class website. Once your submitted file is unzipped, it should compile and run in Eclipse as is.

How to compute the percentage of the time that modems are busy: Assuming that there are 'n' modems in the modem bank and the simulation runs for 't' units of time, then n*t defines the maximum time for modems to be busy. Every time a modem is assigned to a client, add the length of the time that modem is in use to a running total (total), and when the simulation ends compute the percentage as: (total /(n*t))*100