Department of Information and Communications Technology

Higher Diploma in Software Engineering (IT114105)
AY 2024/2025

ITP4510 Data Structure and Algorithms

ASSIGNMENT
Deadline: 13 Apr 2025 (Sun 23:59)

Cashier Counter Simulation

This is a simulation of a waiting line of customers at a supermarket. The purpose of the simulation is to provide a store manager with the means for predicting the effects of varying the number of cashier counter serving customers on the overall level of customer service.

Assume at least one teller is ready for serving customer. The number of tellers can vary, and it stored in the variable **maxTellers**. When a customer arrives at the cashier counter and all counter are busy serving other customers, the new arrival enters a single waiting line, which is a Queue called **waitLine**. The implementation of the Queue is using the **LinkedList** given in the laboratory exercise. For simplicity, we assume that no more than one customer can arrive at the counter during the same minute (that means in each minute only have ONE customer come or none).

Before initiating the simulation, it needs to input the duration of the simulation in minutes and specify the number of counters to be opened for customer service. Once the simulation commences, for every minute, you will be prompted to enter the maximum service time (in minutes) for any arriving customer (assuming a maximum of one customer per minute). In case no customer arrives within a minute, then input a value of 0.

The program should provide the statistical data after simulation completed, for example:

- How many minutes of the simulation?
- How many tellers exist in the simulation?
- The total number of the customers has served.
- How many customers wait in the queue in average?
- Maximum queue length during the simulation.

Sample of simulation

Here is a sample of one simulation. User inputs are in bold and underlined.

```
------------ SETUP SIMULATION ENVIRONMENT----------------
Input simulation length (min):6
Input number of counter:2

------------ START SIMULATION ----------------

At the beginning of iteration 1 ...
Input serving time for a new customer:5
After 1 minute ##
    Teller_1[6]         Teller_2[0]      Waiting Queue:[ ]

At the beginning of iteration 2 ...
Input serving time for a new customer:0
After 2 minute ##
    Teller_1[6]         Teller_2[0]      Waiting Queue:[ ]

At the beginning of iteration 3 ...
Input serving time for a new customer:1
After 3 minute ##
    Teller_1[6]         Teller_2[4]      Waiting Queue:[ ]

At the beginning of iteration 4 ...
Input serving time for a new customer:3
After 4 minute ##
    Teller_1[6]         Teller_2[7]      Waiting Queue:[ ]

At the beginning of iteration 5 ...
Input serving time for a new customer:2
After 5 minute ##
    Teller_1[6]         Teller_2[7]      Waiting Queue:[ 2 ]

At the beginning of iteration 6 ...
Input serving time for a new customer:0
After 6 minute ##
    Teller_1[8]         Teller_2[7]      Waiting Queue:[ ]

------------ END OF SIMULATION ----------------
Total minute simulated: 6 minutes
Number of Tellers: 2
Number of customer served: 4 customers
```

Program Logic

(One iteration/cycle represents one minute)

---------- SETUP SIMULATION ENVIRONMENT----------
Input simulation length (min):6
Input number of counter:2

Iteration	Event Happened
#1	At the beginning of iteration 1 ... Input serving time for a new customer:**5** → A new customer arrived and need 5 minutes service time → The new customer will use **Teller_1**, and this customer will leave while the 6th minute. → Leave time = current time + serving time, → such that the leave time is 1 + 5 = 6
	After 1 minute ## Teller_1[6] Teller_2[0] Waiting Queue:[ ]
#2	At the beginning of iteration 2 ... Input serving time for a new customer:**0** → value of **0** that means no customer came.
	After 2 minute ## Teller_1[6] Teller_2[0] Waiting Queue:[ ]
#3	At the beginning of iteration 3 ... Input serving time for a new customer:**1** → As **Teller_1** is served a customer, the newly arrived customer will be served by **Teller_2**. → Leave time = 3 + 1 = 4
	After 3 minute ## Teller_1[6] Teller_2[4] Waiting Queue:[ ]
#4	At the beginning of iteration 4 ... Input serving time for a new customer:**3** → Original customer at **Teller_2** has left at this iteration → A new customer arrived and need 3 minutes service time → Leave time = 4 + 3 = 7
	After 4 minute ## Teller_1[6] Teller_2[7] Waiting Queue:[ ]

#5	At the beginning of iteration 5 ... Input serving time for a new customer:**2** → Since two counters also occupied, the new arrived need to wait in the waiting queue. → A customer with serving time 2 minutes waits in the queue
	After 5 minute ## Teller_1[6] Teller_2[7] Waiting Queue:[2]
#6	At the beginning of iteration 6 ... Input serving time for a new customer:**0** → No customer arrived → The customer in **Teller_1** has already left. → Next, the customer in waiting queue go to **Teller_1** with serving time 2 minutes → Leave time = 6 + 2 = 8
	After 6 minute ## Teller_1[8] Teller_2[7] Waiting Queue:[ ]

Final result

```
Total minute simulated: 6 minutes
Number of Tellers: 2
Number of customer served: 4 customers
… …
```

Task Specification

1. You should use the provided java classes LinkedList and ListQueue for the implementation of the program using Java.
2. You are not allowed to use the Queue and LinkedList from "java.util".
3. Submit **listings of all programs.**
4. You should create your own Exception class for handling the Exceptional/abnormal cases. Submit a brief description of all such cases (e.g**. invalid input**) handled by the program.
5. Provide evidence of testing, included the test program and the logged **listing of run samples**.

Mark Allocationdd

0. Compilation -- success compilation and execute the program	10%
1. Design -- correct using Queue	10%
2. Implementation -- input parameter -- program simulation -- result print out	40%
3. Statistical data and extra functions provided	10%
4. Error handling -- create own Exception class -- handle exception	10%
5. Report -- executable results (screen dumps)	10%
6. Coding standard -- proper indentation -- proper naming -- consistency coding style -- appropriate comments	10%
Total	**100%**

**** This assignment is counted for your EA. The weighting of this assignment is 40% of EA.**

Instructions to Students

This assignment is an individual assignment. Each student has to submit his/her own work. Plagiarism will be treated seriously. All assignments that have been found involved wholly or partly in plagiarism (no matter these assignments are from the original authors or from the plagiarists) will score ZERO marks. Further, disciplinary action will be followed.

You are not allowed to use any AI tools for generating coding. If teacher is in doubt, he/she may conduct interview with you and ask you to explain the code written.

Adequate in-program comments should be placed as appropriate. All user-defined names should be descriptive as much as possible. Marks are given based on correctness, programming quality, and style.

You are required to hand in:

a) All classes (programs) written and well commented by you for this assignment.
b) The evidence of testing - You should submit at least 4 dump screens (similar to the sample output given above) to show the run results with your own input files (that means you should design your own test cases).

All work is to be submitted to Moodle on/before 13th April 2025 11:59 pm. Late submission without valid reasons will be given ZERO mark.