

**CENG 112 - Data Structures**  
**Homework 2: Car Rental Simulation**

This homework will cover the topics given below:

- Queue ADT
- Deque ADT
- List ADT
- Generics

*Please read the whole document carefully.*

In this homework, you are expected to implement a “Car Rental Simulation” in Java.

A car rental system uses waiting lists for both customers and cars to rent cars. According to the rental policy, when a possible match between the customer and the car is found, the customer must be contacted and decide whether to accept the car. If the applicant does not accept, he or she is placed back at the beginning of the waiting list for the next car, and must wait again. In this system, cars are rented according to their quality rating instead of their features. Each customer has a quality threshold. Assume that once accepted, each car will be occupied for a random number of days between 1 and 5.

- Assume  $k$  customers apply for rent. Each customer has a randomly generated quality threshold (a random double between 1 and 3), and an ID number (or String) to keep track of.
- Create a list of  $N$  cars. Each car should store its ID number (or String) and two pieces of information: its quality score (a random double between 1 and 3), and how many days are left in its occupancy (0 means it is available and ready to be rented).

Get  $N$  and  $k$  from the user via command line arguments.

- The renting process should dequeue a car then dequeue a customer (or visa versa), and check if the customer “accepts” the renting of that car (is the car’s quality value above the customer’s threshold?). If the customer accepts, set the car to rent for a random period of between 1 and 5 days and delete the customer. If the customer rejects the car, enqueue the customer, dequeue the next customer, and repeat.
- If all customers reject a car, move on to the next car. This means that even if customers are on the waiting list, there may still be unrented cars. The rejected car must be inserted into the end of the car queue.
- The renting process repeats until there are no available cars or all waiting customers have seen all cars. Once either of these occurs, move to the next day.
- At the end of each day, decrement the number of days remaining for each rented car, and run the renting process for each available car. If a car becomes available (if occupancy becomes 0) for rent, must be inserted beginning of the car queue.
- Each time a customer is put back on the waiting list, multiply his or her threshold by 0.9. This will simulate the customer growing desperation.
- Stop the simulation when all customers rent a car.

Output statistics after each day. Display the available cars that day, and the number of cars newly rented. An example output is provided below. Your output must be in this format.

### Example Output:

```
Enter available car count, N=5
Enter customer count, k=8
|*****Day1*****
Current car0 quality=1,78 is offering to
    Current cust0 threshold=1,21          ---accepted
Current car1 quality=2,32 is offering to
    Current cust1 threshold=1,28          ---accepted
Current car2 quality=1,43 is offering to
    Current cust2 threshold=2,22          ---not accepted
    Current cust3 threshold=1,36          ---accepted
Current car3 quality=1,17 is offering to
    Current cust2 threshold=2,00          ---not accepted
    Current cust4 threshold=2,12          ---not accepted
    Current cust5 threshold=2,29          ---not accepted
    Current cust6 threshold=2,12          ---not accepted
    Current cust7 threshold=2,14          ---not accepted
    ---not accepted by any customer----
Current car4 quality=1,80 is offering to
    Current cust2 threshold=1,80          ---accepted
All cars have seen
But there are still customers waiting.
Rented cars:
    car0 by cust0 occupancy=1
    car1 by cust1 occupancy=5
    car2 by cust3 occupancy=2
    car4 by cust2 occupancy=5
Available cars:
    car3
*****End of Day*****;

*****Day2*****
Current car0 quality=1,78 is offering to
    Current cust4 threshold=1,91          ---not accepted
    Current cust5 threshold=2,06          ---not accepted
    Current cust6 threshold=1,91          ---not accepted
    Current cust7 threshold=1,92          ---not accepted
    ---not accepted by any customer----
Current car3 quality=1,17 is offering to
    Current cust4 threshold=1,72          ---not accepted
    Current cust5 threshold=1,85          ---not accepted
    Current cust6 threshold=1,72          ---not accepted
    Current cust7 threshold=1,73          ---not accepted
    ---not accepted by any customer----
All cars have seen
But there are still customers waiting.
Rented cars:
    car1 by cust1 occupancy=4
    car2 by cust3 occupancy=1
    car4 by cust2 occupancy=4
Available cars:
    car0
    car3
*****End of Day*****;
```

```

*****Day3*****
Current car2 quality=1,43 is offering to
    Current cust4 threshold=1,54          ---not accepted
    Current cust5 threshold=1,67          ---not accepted
    Current cust6 threshold=1,55          ---not accepted
    Current cust7 threshold=1,56          ---not accepted
    ---not accepted by any customer----
Current car0 quality=1,78 is offering to
    Current cust4 threshold=1,39          ---accepted
Current car3 quality=1,17 is offering to
    Current cust5 threshold=1,50          ---not accepted
    Current cust6 threshold=1,39          ---not accepted
    Current cust7 threshold=1,40          ---not accepted
    ---not accepted by any customer----
All cars have seen
But there are still customers waiting.
Rented cars:
    car1 by cust1 occupancy=3
    car4 by cust2 occupancy=3
    car0 by cust4 occupancy=3
Available cars:
    car2
    car3
*****End of Day*****;

*****Day4*****
Current car2 quality=1,43 is offering to
    Current cust5 threshold=1,35          ---accepted
Current car3 quality=1,17 is offering to
    Current cust6 threshold=1,25          ---not accepted
    Current cust7 threshold=1,26          ---not accepted
    ---not accepted by any customer----
All cars have seen
But there are still customers waiting.
Rented cars:
    car1 by cust1 occupancy=2
    car4 by cust2 occupancy=2
    car0 by cust4 occupancy=2
    car2 by cust5 occupancy=2
Available cars:
    car3
*****End of Day*****;

*****Day5*****
Current car3 quality=1,17 is offering to
    Current cust6 threshold=1,13          ---accepted
All cars have seen
But there are still customers waiting.
Rented cars:
    car1 by cust1 occupancy=1
    car4 by cust2 occupancy=1
    car0 by cust4 occupancy=1
    car2 by cust5 occupancy=1
    car3 by cust6 occupancy=4
*****End of Day*****;

```

\*\*\*\*\*Day6\*\*\*\*\*


Current car2 quality=1,43 is offering to

Current cust7 threshold=1,14

---accepted

All cars have seen

All customer rent a car.

- This is a 2-person group assignment. However, inter-group collaboration is not allowed!
- All assignments are subject to plagiarism detection and the suspected solutions (derived from or inspired by the solution of other groups) will be graded as zero.
- It is not allowed to use Java Collections Framework. You can use the interface, methods, and classes included in the lecture slides.
- Your code should be easy to read and test: **Keep your code clean. Avoid duplication and redundancy. Follow Java Naming Conventions.** Use *relative paths* instead of absolute ones. 

### Submission Rules

All submissions must:

- be performed via **Microsoft Teams** by only one of the group members,
- be exported as an Eclipse Project and saved in ZIP format,
- include all necessary data files (if any TXT, CSV, JSON, etc.) in the right directory,
- follow a specific naming convention such that CENG112\_HW2\_*groupID*.

**Eclipse Project:** CENG112\_HW2\_*G05*

**Exported Archive File:** CENG112\_HW2\_*G05*.zip

Submissions that do not comply with the rules above are penalized.