Exercise 4 (10 points) – can be done in pair or individually

- The first lines of all source files must be comments containing names & IDs of all members. Also create file readme.txt containing names & IDs of all members

- Put all files (source, input, readme.txt) in folder Ex4_xxx where xxx = ID of the group representative, i.e. your source files must be in package Ex4_xxx (assumedly in Maven's src/main/java). Input files must be read from this path

- The group representative zips Ex4_xxx & submits it to Google Classroom. The other members submit only readme.txt. Email submission is not accepted

==

1. Complete classes <u>Customer</u> and <u>Restaurant</u>. Add variables and methods as needed but the given variables must remain <u>private</u>.

```
class Customer {
  private static int runningID = 1;   // for running customer ID
  private int ID;
  private int priority;               // 1 = vvip, 2 = vip, 3 = normal
  private int seats;                  // booked seats
}

class Restaurant {
  private int maxSeats, maxTime;                      // from user input
  private PriorityQueue<Customer> waitingQueue;       // sorted by priority, then by ID
  private ArrayDeque<Customer>    diningQueue;

  public void simulation()        { /* implement restaurant simulation */ }
}
```

2. Get max seats in the restaurant & max time for simulation from user

3. Create Restaurant object. Create 5 <u>normal</u> Customers and put them in waitingQueue. Simply run customer IDs 1, 2, 3, ... Random booked seats between 1-10. Call method simulation.

4. The simulation runs in a loop from t=1 to t=maxTime. In each round:
   4.1 New arrival: create a new Customer with random priority & random seats (1-10). Put new customer in waitingQueue

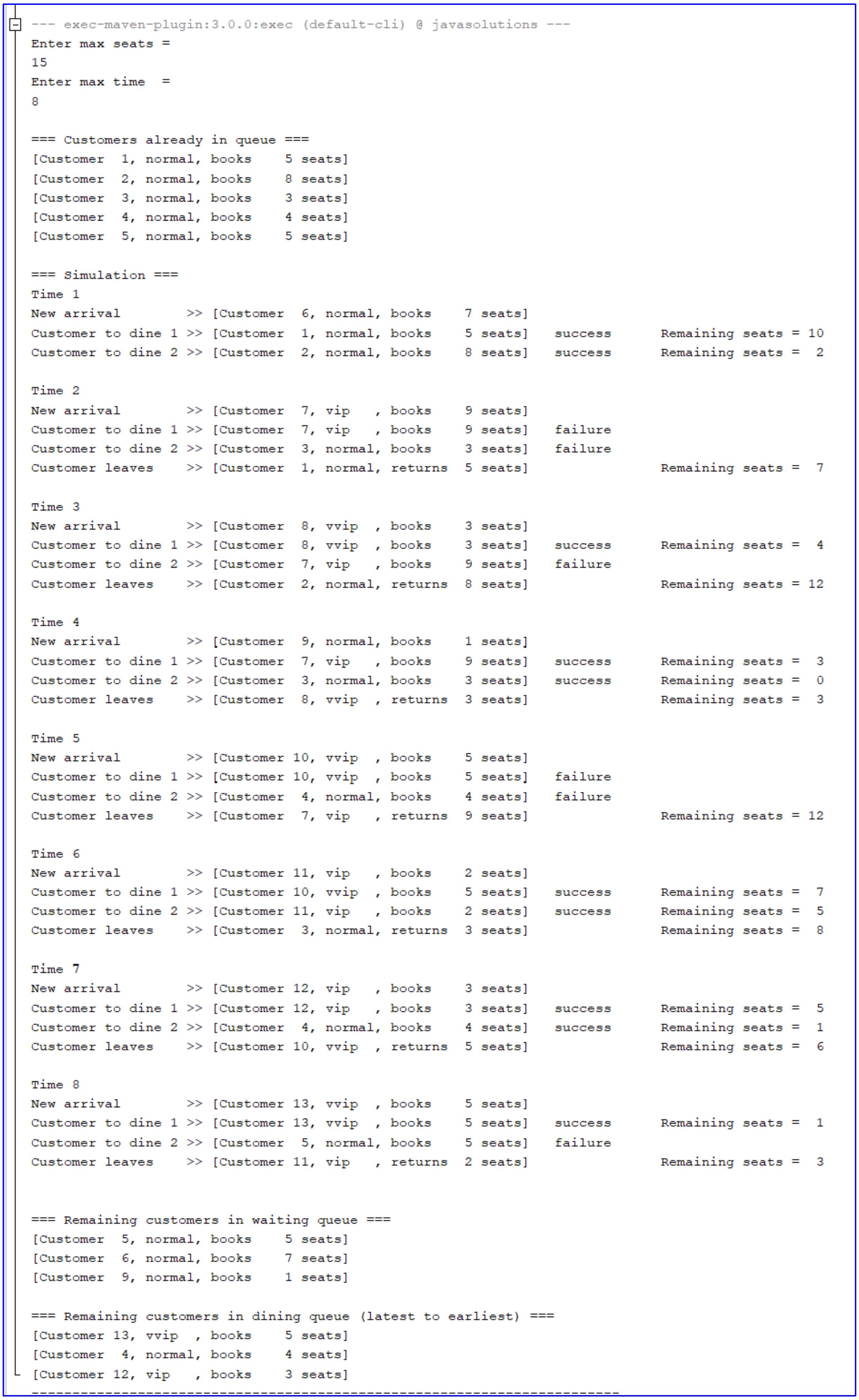
   4.2 Customer to dine: take first 2 customers from waitingQueue.
   If seat requested <= remaining seats, print success message & update remaining seats.
   Otherwise, print failure message.

   4.3 Customer to leave: take 1 first customer from diningQueue. Update remaining seats.

   4.4 Update queues: if dining in (4.2) succeeds, put customer(s) in diningQueue.
   Otherwise, put customer(s) back to waitingQueue

5. After completing maxTime, print remaining customers in both queues
   5.1 Customers remaining in waitingQueue, in the order they would have been served (if the simulation continues)

   5.2 Customers remaining in dinningQueue, starting from latest to earliest customers



```
--- exec-maven-plugin:3.0.0:exec (default-cli) @ javasolutions ---
Enter max seats =
15
Enter max time  =
8

=== Customers already in queue ===
[Customer  1, normal, books    5 seats]
[Customer  2, normal, books    8 seats]
[Customer  3, normal, books    3 seats]
[Customer  4, normal, books    4 seats]
[Customer  5, normal, books    5 seats]

=== Simulation ===
Time 1
New arrival        >> [Customer  6, normal, books    7 seats]
Customer to dine 1 >> [Customer  1, normal, books    5 seats]   success      Remaining seats = 10
Customer to dine 2 >> [Customer  2, normal, books    8 seats]   success      Remaining seats =  2

Time 2
New arrival        >> [Customer  7, vip   , books    9 seats]
Customer to dine 1 >> [Customer  7, vip   , books    9 seats]   failure
Customer to dine 2 >> [Customer  3, normal, books    3 seats]   failure
Customer leaves    >> [Customer  1, normal, returns  5 seats]                Remaining seats =  7

Time 3
New arrival        >> [Customer  8, vvip  , books    3 seats]
Customer to dine 1 >> [Customer  8, vvip  , books    3 seats]   success      Remaining seats =  4
Customer to dine 2 >> [Customer  7, vip   , books    9 seats]   failure
Customer leaves    >> [Customer  2, normal, returns  8 seats]                Remaining seats = 12

Time 4
New arrival        >> [Customer  9, normal, books    1 seats]
Customer to dine 1 >> [Customer  7, vip   , books    9 seats]   success      Remaining seats =  3
Customer to dine 2 >> [Customer  3, normal, books    3 seats]   success      Remaining seats =  0
Customer leaves    >> [Customer  8, vvip  , returns  3 seats]                Remaining seats =  3

Time 5
New arrival        >> [Customer 10, vvip  , books    5 seats]
Customer to dine 1 >> [Customer 10, vvip  , books    5 seats]   failure
Customer to dine 2 >> [Customer  4, normal, books    4 seats]   failure
Customer leaves    >> [Customer  7, vip   , returns  9 seats]                Remaining seats = 12

Time 6
New arrival        >> [Customer 11, vip   , books    2 seats]
Customer to dine 1 >> [Customer 10, vvip  , books    5 seats]   success      Remaining seats =  7
Customer to dine 2 >> [Customer 11, vip   , books    2 seats]   success      Remaining seats =  5
Customer leaves    >> [Customer  3, normal, returns  3 seats]                Remaining seats =  8

Time 7
New arrival        >> [Customer 12, vip   , books    3 seats]
Customer to dine 1 >> [Customer 12, vip   , books    3 seats]   success      Remaining seats =  5
Customer to dine 2 >> [Customer  4, normal, books    4 seats]   success      Remaining seats =  1
Customer leaves    >> [Customer 10, vvip  , returns  5 seats]                Remaining seats =  6

Time 8
New arrival        >> [Customer 13, vvip  , books    5 seats]
Customer to dine 1 >> [Customer 13, vvip  , books    5 seats]   success      Remaining seats =  1
Customer to dine 2 >> [Customer  5, normal, books    5 seats]   failure
Customer leaves    >> [Customer 11, vip   , returns  2 seats]                Remaining seats =  3


=== Remaining customers in waiting queue ===
[Customer  5, normal, books    5 seats]
[Customer  6, normal, books    7 seats]
[Customer  9, normal, books    1 seats]

=== Remaining customers in dining queue (latest to earliest) ===
[Customer 13, vvip  , books    5 seats]
[Customer  4, normal, books    4 seats]
[Customer 12, vip   , books    3 seats]
------------------------------------------------------------------------
```