CHARGELAB

Coding Challenge

Java developer position

Confidential

Problem Definition

You are expected to design a custom priority queue. The following is the required logic for this queue:

- Each item in the queue has Generic Type (T) along with a priority.
- Priority is a positive integer. The highest priority is “1”.
- Multiple items might have the same priority.
- The following logic determines the deque process:
    - Items with higher priority are dequeued first.
    - For every two (2) items dequeued with priority "x", the next item must be of priority "x+1". We call this the Throttle Rate of a priority class.
    - The two items dequeued must **not** necessarily be dequeued **consecutively** for this constraint to apply. See examples below.
    - Items follow FIFO order in their own priority class.
- The queue must be thread-safe as it is used exclusively in multi-threaded environments.
- The queue has a fixed, pre-determined size.
- On a full queue, the producer must be blocked till capacity is available.
- On an empty queue, the consumer must be blocked till queue items are available.

Deliverables

1. Identify a scenario where lower priority items may never exit the queue. Provide a solution.
2. Implement this data structure and incorporate your answer to the deliverable (1). Please add test coverage to your comfort level.
3. Please use **Java 8**, **Maven**, and **IntelliJ** (alternatively Eclipse). Please submit the solution as **a single** zip file, not individual files. If attaching to email message didn't work, please share via GDrive/DropBox/etc.

Examples

Example 1:

Please consider the sequence below:

- Enqueue (left to right): 4 1 3 2 1 2
- Dequeue: 1
- Dequeue: 1
- Enqueue: 1
- Dequeue: 2 (Constraint B)
- Dequeue: 1
- Dequeue: 2
- Dequeue: 3

Example 2:

- Enqueue (left to right): 4 1 3 2 2
- Dequeue: 1
- Dequeue: 2
- Enqueue: 1
- Dequeue: 1
- Dequeue: 2
- Dequeue: 3 (Constraint B)
- Dequeue: 4

Example 3:

Consider the following sequence of numbers for a single producer (spaces are only for readability purposes):

4 1 3 2 1 4 2 3 2 4 1 3 3 5 2 1 3 6 1 2 4 2 4 1 3 2 1 5 2 1 1 2 3 1 1

The desired dequeue sequence will be the following for a single consumer:

1 1 2 1 1 2 3 1 1 2 1 1 2 3 4 1 1 2 1 2 3 2 2 3 4 5 2 3 3 4 3 4 5 6 4

Final Notes:

- The items can be enqueued/dequeued at any time in a **multi-threaded** environment.
- Correctness, simplicity, reusability, and efficiency are the main concerns here.
- Adequate test coverage and code documentation are required.
- **Bonus**: Generalize the throttle Rate. Please comment on how the data structure in Problem A can be generalized to accommodate an arbitrary Throttle Rate for each priority class.