Problem 2

Manideepika Reddy Myaka

a. **List all of the input parameters, including the state variables.**
   The input parameters for the methods in the BoundedQueue class include:
   1. Constructor: BoundedQueue(int capacity)
      - Input parameter: capacity (integer) representing the maximum number of elements the queue can hold.
   2. State variables:
      - Current size of the queue: Represents how many elements are currently in the queue.
      - Capacity of the queue: The maximum number of elements that can be in the queue.
   3. Methods:
      - enQueue(Object X): Adds an object X to the queue.
        - ➢ Input parameter: X (object) representing the element to be added to the queue.
      - deQueue(): Removes and returns an object from the queue.
      - isEmpty(): Returns a boolean indicating whether the queue is empty.
      - isFull(): Returns a boolean indicating whether the queue is full.

b. **Define characteristics for the input parameters. Make sure you cover all input parameters.**
   1. Capacity (capacity):
      Characteristic 1: capacity
      - Block 1: capacity = 0 (invalid, no space for any element)
      - Block 2: capacity > 0 (valid, allows the queue to store at least one element)
   2. Current Size of Queue (size):
      Characteristic 2: size (current size of the queue)
      - Block 1: size = 0 (empty queue)
      - Block 2: 0 < size < capacity (partially filled queue)
      - Block 3: size = capacity (full queue)
   3. Input Object (X):

Characteistic 3: X (the object being added)

- Block 1: X = null (invalid input, no object to add)
- Block 2: X ≠ null (valid input)

4. State after operations:
   - isEmpty(): The queue can be empty (true) or not empty (false).
   - isFull(): The queue can be full (true) or not full (false).

c. **Partition the characteristics into blocks. Choose one block per partition as the "Base" block.**

1. Capacity:
   - Block 1: capacity = 0 (Base block: capacity > 0).
   - Block 2: capacity > 0.
2. Current Size of Queue:
   - Block 1: size = 0 (empty).
   - Block 2: 0 < size < capacity (partially filled).
   - Block 3: size = capacity (full queue).
3. Input Object:
   - Block 1: X = null (invalid input).
   - Block 2: X ≠ null (valid input).

d. **Define values for each block.**

1. Capacity:
   - Block 1: capacity = 0 → BoundedQueue(0)
   - Block 2: capacity > 0 → BoundedQueue(3) (for example, a queue of capacity 3)
2. Current Size of Queue:
   - Block 1: size = 0 → Empty queue
   - Block 2: 0 < size < capacity → Queue with 1 or 2 elements, e.g., {1}
   - Block 3: size = capacity → Full queue, e.g., {1, 2, 3}
3. Input Object:
   - Block 1: X = null
   - Block 2: X = "A" (or any valid object)

e. **Define a set of Test cases. That is, write test inputs for test cases, together with the expected output of those inputs on the enQueue method. You are not required to write JUnit tests for this problem.**

Test Case	**Capacity**	**Current Size**	**Input Object**	**Expected Outcome**
TC1	0	0	"A"	Error or exception due to zero capacity (invalid)
TC2	3	0	"A"	"A" is added to the queue, queue size becomes 1
TC3	3	1	"B"	"B" is added to the queue, queue size becomes 2
TC4	3	2	"C"	"C" is added to the queue, queue size becomes 3
TC5	3	3	"D"	No action or error (queue is already full)
TC6	3	2	null	Error or no action (cannot add null to the queue)
TC7	3	3	null	No action or error (queue full and null input)