Step 1: Identify - (a)

1. **BoundedQueue (int capacity)** - set maximum size of queue and initialize parameters(constructor)
   Exception: IllegalArgumentException → if capacity is negative number

2. **void enQueue (Object o)** - add new object in queue
   Exception: NullPointerException -> if o is null
   Exception: IllegalStateException -> if queue is full

3. **Object deQueue ()** - return first element in queue
   Exception: IllegalStateException -> if queue is empty

4. **boolean isEmpty()** - return true if queue is empty

5. **boolean isFull()** - return true if queue is full

6. **String toString()** - return all elements in queue (ex: [1, 2, 3])

7. **Parameters:**
   (1) Object[] - store all elements in queue
   (2) size - total number of elements
   (3) front - index of first element
   (4) back - index of last element
   (5) capacity - maximum limit of size

Step2: Develop Characteristics - (b)

Method	Params	Returns	Values	Exception	ChID	Characteristic	Covered by
Bounded Queue	1,2,3,4,5				C1	positive integer of arguments	
				IllegalArgument Exception			C1
enQueue	1,2,4,5				C2	add non-null value	
				NullPointer Exception			C2
				IllegalState Exception	C3	constraint satisfied	
deQueue	1,2,3,5	Object	Object		C4	return non-null value	
				IllegalState Exception			C3
isEmpty	2	boolean	True or false				C4
isFull	2,5	boolean	True or false				C4

Step3: Design a partitioning - (c)

ID	Characteristic	BoundedQueue()	enQueuer()	deQueuer()	isEmpty()	isFull()
C1	positive integer of arguments	✓				
C2	add non-null value		✓			
C3	constraint satisfied		✓	✓		
C4	return non-null value			✓	✓	✓

Step4: Define Test Requirements - (d)

Method	Characteristics	Test Requirements	Infeasible TRs	Revised TRs	Number of TRs
BoundedQueue	C1	{T, F}			2
enQueue	C2, C3	{TT, FT, TF}	FT	FT→FF	3
deQueue	C3, C4	{TT, FT, TF}	FT	FT→FF	3
isEmpty	C4	{T, F}			2
isFull	C4	{T, F}			2