MostRecentlyInsertedQueue

Implement a class called `MostRecentlyInsertedQueue` which implements the interface `java.util.Queue<E>`. To reduce the amount of boilerplate code you have to write you may extend `java.util.AbstractQueue<E>`. The purpose of this queue is to store the N most recently inserted elements. The queue should have the following properties:

1. The queue implements the interface `java.util.Queue<E>`
2. The queue is bounded in size. The total capacity of the queue must be passed into the constructor.
3. New elements are added to the tail of the queue
4. The queue is traversed from head to tail
5. The queue must always accept new elements. If the queue is already full (`Queue#size() == capacity`), the oldest element that was inserted (the head) should be evicted, and then the new element can be added at the tail.

The following code demonstrates the desired behavior:

```
Queue<Integer> queue = new MostRecentlyInsertedQueue<Integer>(3);
// queue.size(): 0, contents (head -> tail): [ ]
queue.offer(1);              // queue.size(): 1, contents (head -> tail): [ 1 ]
queue.offer(2);              // queue.size(): 2, contents (head -> tail): [ 1, 2 ]
queue.offer(3);              // queue.size(): 3, contents (head -> tail): [ 1, 2, 3 ]
queue.offer(4);              // queue.size(): 3, contents (head -> tail): [ 2, 3, 4 ]
queue.offer(5);              // queue.size(): 3, contents (head -> tail): [ 3, 4, 5 ]
int poll1 = queue.poll();    // queue.size(): 2, contents (head -> tail): [ 4, 5 ], poll1 = 3
int poll2 = queue.poll();    // queue.size(): 1, contents (head -> tail): [ 5 ], poll2 = 4
queue.clear();               // queue.size(): 0, contents (head -> tail): [ ]
```

The primary evaluation criteria are correctness in the behavior of the queue as specified in the problem description, design, and clarity. Secondary criteria include performance and memory efficiency.

Bonus #1: Implement `ConcurrentMostRecentlyInsertedQueue`, a thread-safe version of `MostRecentlyInsertedQueue`

Bonus #2: Implement `MostRecentlyInsertedBlockingQueue`, a thread-safe variant of `MostRecentlyInsertedQueue` that implements `java.util.concurrent.BlockingQueue<E>`