The Priority Queue implementation in the Java is a default min heap Priority Queue

In order to turn it into a max one I create a custom comparator object like below or Use Java's Collections.reverseOrder() comparator.

Comparator<Integer> cmp = new Comparator<Integer>()

{

public int compare( Integer x, Integer y )

{

return y - x;

}

};

* [Queue.add](http://docs.oracle.com/javase/7/docs/api/java/util/Queue.html#add%28E%29)

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and **throwing an IllegalStateException if no space is currently available**...  
Returns: true (as specified by [Collection.add(E)](http://docs.oracle.com/javase/7/docs/api/java/util/Collection.html" \l "add%28E%29))  
Throws: IllegalStateException - if the element cannot be added at this time due to capacity restrictions...

* [Queue.offer](http://docs.oracle.com/javase/7/docs/api/java/util/Queue.html#offer%28E%29)

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions. **When using a capacity-restricted queue, this method is generally preferable to add(E), which can fail to insert an element only by throwing an exception**...  
Returns: true if the element was added to this queue, else false...

since PriorityQueue is unbounded (as stated in javadocs: "unbounded priority queue based on a priority heap..."), preference of API designers expressed above do not apply.