In the concrete class implementing PriorityQueue interface, a new field named as Comparator<E> comparator will be included.

Changes to the patient queue design:
The class of UrgentCareQueueOfPatient will be added as a subclass of AbstractQueueOfPatient. In this class, the Comparator of the priority queue will be initialized as the PatientOrderDecreaseComparator. And the patient with the smallest order number will be added to front of the queue. When, trying to get the next patient, we just need to get the patient with index 0. So the running time of the nextPatient() will be O(1).
And when to remove the next patient, we need remove the patient at index of 0. This will also take the time of O(1).

Comparator

ⓜ compare(T, T)	int
ⓜ equals(Object)	boolean
ⓜ reversed()	Comparator\<T>
ⓜ thenComparing(Comparator\<? super T>)	Comparator\<T>
ⓜ thenComparing(Function\<? super T, ? extends U>, Comparator\<? super U>)	Comparator\<T>
ⓜ thenComparing(Function\<? super T, ? extends U>)	Comparator\<T>
ⓜ thenComparingInt(ToIntFunction\<? super T>)	Comparator\<T>
ⓜ thenComparingLong(ToLongFunction\<? super T>)	Comparator\<T>
ⓜ thenComparingDouble(ToDoubleFunction\<? super T>)	Comparator\<T>
ⓜ reverseOrder()	Comparator\<T>
ⓜ naturalOrder()	Comparator\<T>
ⓜ nullsFirst(Comparator\<? super T>)	Comparator\<T>
ⓜ nullsLast(Comparator\<? super T>)	Comparator\<T>
ⓜ comparing(Function\<? super T, ? extends U>, Comparator\<? super U>)	Comparator\<T>
ⓜ comparing(Function\<? super T, ? extends U>)	Comparator\<T>
ⓜ comparingInt(ToIntFunction\<? super T>)	Comparator\<T>
ⓜ comparingLong(ToLongFunction\<? super T>)	Comparator\<T>
ⓜ comparingDouble(ToDoubleFunction\<? super T>)	Comparator\<T>

Serializable

PatientUrgencyComparator

ⓜ compare(Patient, Patient)	int
ⓜ equals(Object)	boolean
ⓜ hashCode()	int
ⓜ toString()	String

PatientOrderComparator

ⓜ compare(Patient, Patient)	int
ⓜ equals(Object)	boolean
ⓜ hashCode()	int

PatientOrderDecreaseComparator

ⓜ compare(Patient, Patient)	int
ⓜ equals(Object)	boolean
ⓜ hashCode()	int