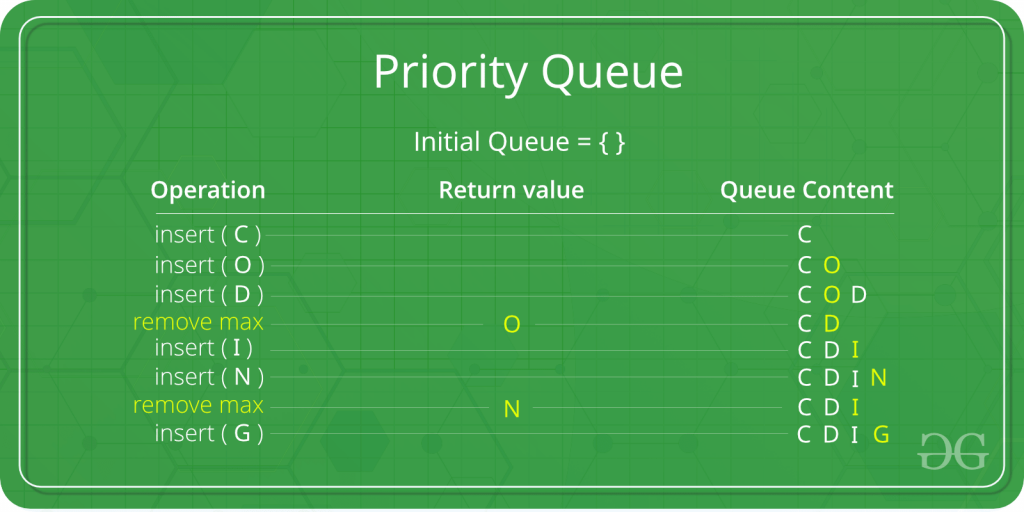
**PriorityQueue in Java**

A Priority Queue is used when the objects are supposed to be processed based on the priority. It is known that a queue follows First-In-First-Out algorithm, but sometimes the elements of the queue are needed to be processed according to the priority, that’s when the PriorityQueue comes into play. The PriorityQueue is based on the priority heap. The elements of the priority queue are ordered according to the natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used.

In the below priority queue, element with maximum ASCII value will have the highest priority.  


Few important points on Priority Queue are as follows:

* PriorityQueue doesn’t permit null.
* We can’t create PriorityQueue of Objects that are non-comparable
* PriorityQueue are unbound queues.
* The head of this queue is the least element with respect to the specified ordering. If multiple elements are tied for least value, the head is one of those elements — ties are broken arbitrarily.
* The queue retrieval operations poll,  remove,  peek, and element access the element at the head of the queue.
* It inherits methods from AbstractQueue, AbstractCollection, Collection and Object class.

**Constructors of PriorityQueue class**

* **PriorityQueue():** Creates a PriorityQueue with the default initial capacity (11) that orders its elements according to their natural ordering.
* **PriorityQueue(Collection<E> c):** Creates a PriorityQueue containing the elements in the specified collection.
* **PriorityQueue(int initialCapacity)**: Creates a PriorityQueue with the specified initial capacity that orders its elements according to their natural ordering.
* **PriorityQueue(int initialCapacity, Comparator<E> comparator):** Creates a PriorityQueue with the specified initial capacity that orders its elements according to the specified comparator.
* **PriorityQueue(PriorityQueue<E> c)**: Creates a PriorityQueue containing the elements in the specified priority queue.
* **PriorityQueue(SortedSet<E> c)**: Creates a PriorityQueue containing the elements in the specified sorted set.

**Basic Operations on PriorityQueue**:

* [boolean add(E element):](https://www.geeksforgeeks.org/priorityqueue-add-method-in-java/) This method inserts the specified element into this priority queue.
* [public peek():](https://www.geeksforgeeks.org/priorityqueue-peek-method-in-java/) This method retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
* [public poll():](https://www.geeksforgeeks.org/priorityqueue-poll-method-in-java/) This method retrieves and removes the head of this queue, or returns null if this queue is empty.

Below program illustrates the above methods:

|  |
| --- |
| // Java program to demonstrate working of  // priority queue in Java  import java.util.\*;    class GfG {      public static void main(String args[])      {          // Creating empty priority queue          PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>();            // Adding items to the pQueue using add()          pQueue.add(10);          pQueue.add(20);          pQueue.add(15);            // Printing the top element of PriorityQueue          System.out.println(pQueue.peek());            // Printing the top element and removing it          // from the PriorityQueue container          System.out.println(pQueue.poll());            // Printing the top element again          System.out.println(pQueue.peek());      }  } |

**Output:**

10

10

15

**Methods in PriorityQueue class:**

1. [boolean add(E element):](https://www.geeksforgeeks.org/priorityqueue-add-method-in-java/) This method inserts the specified element into this priority queue.
2. [public remove():](https://www.geeksforgeeks.org/priorityqueue-remove-method-in-java/) This method removes a single instance of the specified element from this queue, if it is present
3. [public poll():](https://www.geeksforgeeks.org/priorityqueue-poll-method-in-java/) This method retrieves and removes the head of this queue, or returns null if this queue is empty.
4. [public peek():](https://www.geeksforgeeks.org/priorityqueue-peek-method-in-java/) This method retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
5. [Iterator iterator():](https://www.geeksforgeeks.org/priorityqueue-iterator-method-in-java/) Returns an iterator over the elements in this queue.
6. [boolean contains(Object o):](https://www.geeksforgeeks.org/priorityqueue-contains-method-in-java/) This method returns true if this queue contains the specified element
7. [void clear():](https://www.geeksforgeeks.org/priorityqueue-clear-method-in-java/) This method is used to remove all of the contents of the priority queue.
8. [boolean offer(E e):](https://www.geeksforgeeks.org/priorityqueue-offer-method-in-java/) This method is used to insert a specific element into the priority queue.
9. [int size():](https://www.geeksforgeeks.org/priorityqueue-size-method-in-java/) The method is used to return the number of elements present in the set.
10. [toArray():](https://www.geeksforgeeks.org/priorityqueue-toarray-method-in-java/) This method is used to return an array containing all of the elements in this queue.
11. [Comparator comparator()](https://www.geeksforgeeks.org/priorityqueue-comparator-method-in-java/): The method is used to return the comparator that can be used to order the elements of the queue.