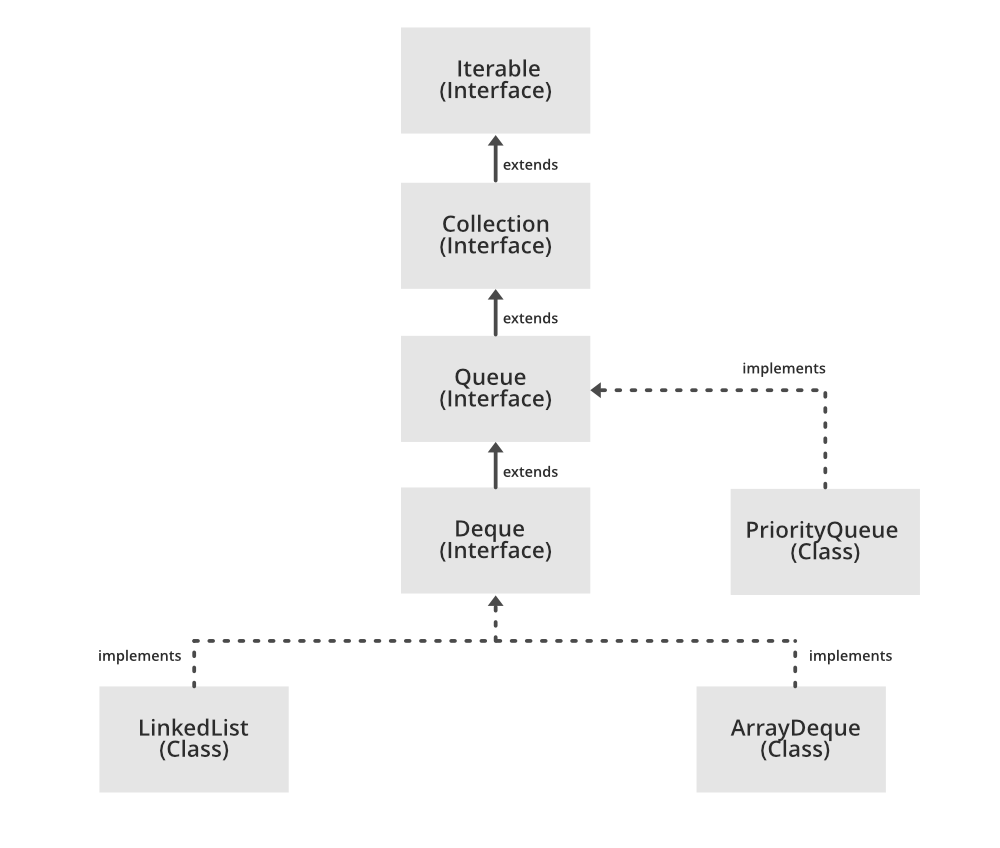
Queue

The Queue interface present in the [java.util](https://www.geeksforgeeks.org/java-util-package-java/) package and extends the [Collection interface](https://www.geeksforgeeks.org/collections-in-java-2/) is used to hold the elements about to be processed in FIFO(First In First Out) order. It is an ordered list of objects with its use limited to insert elements at the end of the list and deleting elements from the start of the list, (i.e.), it follows the FIFO or the First-In-First-Out principle.



Attention reader! Don’t stop learning now. Get hold of all the important DSA concepts with the [**DSA Self Paced Course**](https://practice.geeksforgeeks.org/courses/dsa-self-paced?utm_source=geeksforgeeks&utm_medium=article&utm_campaign=gfg_article_dsa_content_bottom) at a student-friendly price and become industry ready. To complete your preparation from learning a language to DS Algo and many more, please refer [**Complete Interview Preparation Course**](https://practice.geeksforgeeks.org/courses/complete-interview-preparation?utm_source=GeeksforGeeks&utm_medium=Text&utm_campaign=GFG_Article_Bottom_Text_CIP)**.**

In case you wish to attend **live classes** with experts, please refer [**DSA Live Classes for Working Professionals**](https://practice.geeksforgeeks.org/courses/geeks-classes-live?utm_source=geeksforgeeks&utm_medium=article&utm_campaign=gfg_article_dsa_content_bottom) and [**Competitive Programming Live for Students**](https://practice.geeksforgeeks.org/courses/competitive-programming-live?utm_source=geeksforgeeks&utm_medium=article&utm_campaign=gfg_article_cp_content_bottom).

Being an interface the queue needs a concrete class for the declaration and the most common classes are the [PriorityQueue](https://www.geeksforgeeks.org/priority-queue-class-in-java-2/) and [LinkedList](https://www.geeksforgeeks.org/linked-list-in-java/) in Java. Note that neither of these implementations are thread safe. [PriorityBlockingQueue](https://www.geeksforgeeks.org/priorityblockingqueue-class-in-java/) is one alternative implementation if thread safe implementation is needed.

**Declaration:** The Queue interface is declared as:

*public interface Queue extends Collection*

**Creating Queue Objects**  
**Since** *Queue* is an [interface](https://www.geeksforgeeks.org/interfaces-in-java/), objects cannot be created of the type queue. We always need a class which extends this list in order to create an object. And also, after the introduction of [Generics](https://www.geeksforgeeks.org/generics-in-java/) in Java 1.5, it is possible to restrict the type of object that can be stored in the Queue. This type-safe queue can be defined as:

*// Obj is the type of the object to be stored in Queue*   
*Queue<Obj> queue = new PriorityQueue<Obj> ();*

**Example of a Queue:**

* Java

|  |
| --- |
| // Java program to demonstrate a Queue    **import** java.util.LinkedList;  **import** java.util.Queue;    **public** **class** QueueExample {    **public** **static** **void** main(String[] args)  {  Queue<Integer> q  = **new** LinkedList<>();    // Adds elements {0, 1, 2, 3, 4} to  // the queue  **for** (**int** i = 0; i < 5; i++)  q.add(i);    // Display contents of the queue.  System.out.println("Elements of queue "  + q);    // To remove the head of queue.  **int** removedele = q.remove();  System.out.println("removed element-"  + removedele);    System.out.println(q);    // To view the head of queue  **int** head = q.peek();  System.out.println("head of queue-"  + head);    // Rest all methods of collection  // interface like size and contains  // can be used with this  // implementation.  **int** size = q.size();  System.out.println("Size of queue-"  + size);  }  } |

**Output:** Elements of queue [0, 1, 2, 3, 4]removed element-0[1, 2, 3, 4]head of queue-1Size of queue-4

### Operations on Queue Interface

Let’s see how to perform a few frequently used operations on the queue using the [Priority Queue class](https://www.geeksforgeeks.org/priority-queue-class-in-java-2/).  
1. Adding Elements: In order to add an element in a queue, we can use the [add() method](https://www.geeksforgeeks.org/queue-add-method-in-java/). The insertion order is not retained in the PriorityQueue. The elements are stored based on the priority order which is ascending by default.

* Java

|  |
| --- |
| // Java program to add elements  // to a Queue    **import** java.util.\*;    **public** **class** GFG {    **public** **static** **void** main(String args[])  {  Queue<String> pq = **new** PriorityQueue<>();    pq.add("Geeks");  pq.add("For");  pq.add("Geeks");    System.out.println(pq);  }  } |

**Output:**

[For, Geeks, Geeks]

**2. Removing Elements:** In order to remove an element from a queue, we can use the [remove() method.](https://www.geeksforgeeks.org/queue-remove-method-in-java/) If there are multiple such objects, then the first occurrence of the object is removed. Apart from that, poll() method is also used to remove the head and return it.

* Java

|  |
| --- |
| // Java program to remove elements  // from a Queue    **import** java.util.\*;    **public** **class** GFG {    **public** **static** **void** main(String args[])  {  Queue<String> pq = **new** PriorityQueue<>();    pq.add("Geeks");  pq.add("For");  pq.add("Geeks");    System.out.println("Initial Queue " + pq);    pq.remove("Geeks");    System.out.println("After Remove " + pq);    System.out.println("Poll Method " + pq.poll());    System.out.println("Final Queue " + pq);  }  } |

**Output:** Initial Queue [For, Geeks, Geeks]After Remove [For, Geeks]Poll Method ForFinal Queue [Geeks]

**3. Iterating the Queue:** There are multiple ways to iterate through the Queue. The most famous way is converting the queue to the array and traversing using the for loop. However, the queue also has an inbuilt iterator which can be used to iterate through the queue.

* Java

|  |
| --- |
| // Java program to iterate elements  // to a Queue    **import** java.util.\*;    **public** **class** GFG {    **public** **static** **void** main(String args[])  {  Queue<String> pq = **new** PriorityQueue<>();    pq.add("Geeks");  pq.add("For");  pq.add("Geeks");    Iterator iterator = pq.iterator();    **while** (iterator.hasNext()) {  System.out.print(iterator.next() + " ");  }  }  } |

**Output:** For Geeks Geeks

**Characteristics of a Queue:** The following are the characteristics of the queue:

* The Queue is used to insert elements at the end of the queue and removes from the beginning of the queue. It follows FIFO concept.
* The Java Queue supports all methods of Collection interface including insertion, deletion, etc.
* [LinkedList](https://www.geeksforgeeks.org/linked-list-in-java/), ArrayBlockingQueue and [PriorityQueue](https://www.geeksforgeeks.org/priority-queue-class-in-java-2/) are the most frequently used implementations.
* If any null operation is performed on BlockingQueues, NullPointerException is thrown.
* The Queues which are available in java.util package are Unbounded Queues.
* The Queues which are available in java.util.concurrent package are the Bounded Queues.
* All Queues except the Deques supports insertion and removal at the tail and head of the queue respectively. The Deques support element insertion and removal at both ends