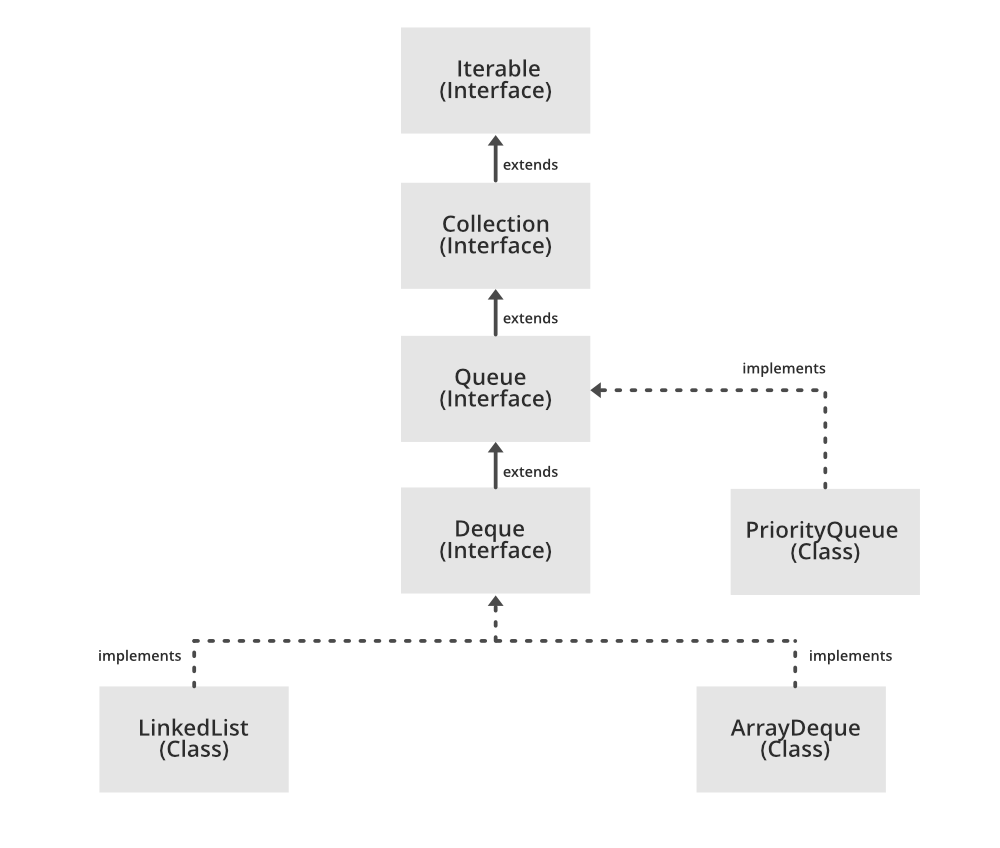
**Queue Interface In Java**

The Queue interface is present in [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.



**Priority Queue using Queue Interface:**

**import** java.util.\*;

**public** **class** QueueOpertions {

**public** **static** **void** main(String args[])

    {

//Creating a Queue

        Queue<String> pq = **new** PriorityQueue<>();

//adding values into Queue

        pq.add("Lakshmi");

        pq.add("Nagalakshmi");

        pq.add("Mahalakshmi");

        System.out.println(pq);

// removing an element

pq.remove("Nagalakshmi");

System.out.println("After Remove " + pq);

        System.out.println("Poll Method " + pq.poll());

        System.out.println("Final Queue " + pq);

// Iterating over the Queue

  Iterator iterator = pq.iterator();

**while** (iterator.hasNext()) {

        System.out.print(iterator.next() + " ");

}

    }

}

**LinkedList using Queue Interface:**

**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);

**int** size = q.size();

        System.out.println("Size of queue-" + size);

    }

}

**Queue implementation using an Array:**

class Queue

{

    private int[] arr;      // array to store queue elements

    private int front;      // front points to the front element in the queue

    private int rear;       // rear points to the last element in the queue

    private int capacity;   // maximum capacity of the queue

    private int count;      // current size of the queue

    // Constructor to initialize a queue

    Queue(int size)

    {

        arr = new int[size];

        capacity = size;

        front = 0;

        rear = -1;

        count = 0;

    }

    // Utility function to dequeue the front element

    public int dequeue()

    {

        // check for queue underflow

        if (isEmpty())

        {

            System.out.println("Underflow\nProgram Terminated");

            System.exit(-1);

        }

        int x = arr[front];

        System.out.println("Removing " + x);

        front = (front + 1) % capacity;

        count--;

        return x;

    }

    // Utility function to add an item to the queue

    public void enqueue(int item)

    {

        // check for queue overflow

        if (isFull())

        {

            System.out.println("Overflow\nProgram Terminated");

            System.exit(-1);

        }

        System.out.println("Inserting " + item);

        rear = (rear + 1) % capacity;

        arr[rear] = item;

        count++;

    }

    // Utility function to return the front element of the queue

    public int peek()

    {

        if (isEmpty())

        {

            System.out.println("Underflow\nProgram Terminated");

            System.exit(-1);

        }

        return arr[front];

    }

    // Utility function to return the size of the queue

    public int size() {

        return count;

    }

    // Utility function to check if the queue is empty or not

    public boolean isEmpty() {

        return (size() == 0);

    }

    // Utility function to check if the queue is full or not

    public boolean isFull() {

        return (size() == capacity);

    }

}

class Main

{

    public static void main (String[] args)

    {

        // create a queue of capacity 5

        Queue q = new Queue(5);

        q.enqueue(1);

        q.enqueue(2);

        q.enqueue(3);

        System.out.println("The front element is " + q.peek());

        q.dequeue();

        System.out.println("The front element is " + q.peek());

        System.out.println("The queue size is " + q.size());

        q.dequeue();

        q.dequeue();

        if (q.isEmpty()) {

            System.out.println("The queue is empty");

        }

        else {

            System.out.println("The queue is not empty");

        }

    }

}