Queue in Java

A **Queue** is a **linear data structure** that follows the **FIFO (First-In, First-Out)** principle.

- The **element inserted first** is removed first.
- Think of it like a real-life queue at a ticket counter.

In Java, **Queue** is an **interface** present in java.util package:

public interface Queue<E> extends Collection<E>

Since it’s an **interface**, you cannot directly create a Queue object. Instead, you use classes that implement Queue, such as:

- LinkedList
- PriorityQueue
- ArrayDeque

Common Queue Methods

Method	**Description**
add(E e)	Inserts element, throws exception if fails
offer(E e)	Inserts element, returns false if fails
remove()	Removes head, throws exception if empty
poll()	Removes head, returns null if empty
element()	Returns head, throws exception if empty
peek()	Returns head, returns null if empty

Example 1: Using Queue with LinkedList

```
import java.util.*;


public class QueueExample {
    public static void main(String[] args) {
        // Creating a Queue using LinkedList
        Queue<String> queue = new LinkedList<>();


        // Adding elements
        queue.add("A");
        queue.add("B");
        queue.add("C");
        queue.add("D");

        System.out.println("Queue: " + queue);

        // Removing the first element (FIFO)
        String removed = queue.remove();
        System.out.println("Removed: " + removed);

        // Checking the head without removing
        String head = queue.peek();
        System.out.println("Head: " + head);

        // Removing head safely
        queue.poll();
        System.out.println("Queue after poll: " + queue);
    }
}
```

Output:

```
Queue: [A, B, C, D]
Removed: A
Head: B
Queue after poll: [C, D]
```

Example 2: Using PriorityQueue

Unlike LinkedList, a **PriorityQueue** orders elements based on natural ordering or a custom comparator (not strictly FIFO).

```
import java.util.*;
```

```
public class PriorityQueueExample {
    public static void main(String[] args) {
        // Min-heap (default natural ordering)
        Queue<Integer> pq = new PriorityQueue<>();

        pq.add(40);
        pq.add(10);
        pq.add(30);
        pq.add(20);

        System.out.println("PriorityQueue: " + pq);

        // Elements are retrieved in sorted order
        while (!pq.isEmpty()) {
            System.out.println("Removed: " + pq.poll());
        }
    }
}
```

Output:

```
PriorityQueue: [10, 20, 30, 40]
Removed: 10
Removed: 20
Removed: 30
Removed: 40
```

Example 3: Using ArrayDeque (Double-ended Queue)

```
import java.util.*;

public class ArrayDequeExample {
    public static void main(String[] args) {
        Queue<String> adq = new ArrayDeque<>();
        adq.offer("One");
        adq.offer("Two");
        adq.offer("Three");

        System.out.println("ArrayDeque: " + adq);

        adq.poll();  // removes first element
        System.out.println("After poll: " + adq);

        System.out.println("Peek: " + adq.peek());
    }
}
```

✅ **Summary**

- Queue = FIFO data structure.
- Implemented by LinkedList, PriorityQueue, ArrayDeque.
- Key methods: add(), offer(), remove(), poll(), peek(), element().