Queue Interface:

- Queue follows **FIFO (First In First Out)** principle (except PriorityQueue which orders by priority).
- Commonly used for **task scheduling, buffering, and resource management**.
- Key methods (from Queue interface):
    - `offer(E e)` → Inserts element.
    - `poll()` → Retrieves & removes head (returns `null` if empty).
    - `peek()` → Retrieves but does not remove head (returns `null` if empty).

(I) PriorityQueue

- Stores elements based on **priority (natural ordering or custom comparator)**, **not insertion order**.
- By default, it orders elements in **ascending order (min-heap)**.
- Doesn't allow `null`.
- Useful in scenarios like **job scheduling, Dijkstra's algorithm, Huffman coding**.

Important Methods

- `add(E e)` / `offer(E e)` → Insert element.
- `peek()` → View the highest priority element (smallest in min-heap).
- `poll()` → Remove + return the highest priority element.
- `remove(Object o)` → Remove specific element.
- `size()` → Number of elements.
- `iterator()` → Traverse (but order is not guaranteed).

Example

```
import java.util.PriorityQueue;

public class PriorityQueueDemo {
    public static void main(String[] args) {
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        pq.add(30);
```

```
        pq.add(10);
        pq.add(20);

        System.out.println("PriorityQueue: " + pq);
        System.out.println("Peek: " + pq.peek());   // 10 (smallest
element)
        System.out.println("Poll: " + pq.poll());   // removes 10
        System.out.println("After Poll: " + pq);    // [20, 30]
    }
}
```

(II) ArrayDeque

- Implements **Deque (Double-Ended Queue)**.
- Allows insertion and deletion at **both ends** (head & tail).
- Faster than `Stack` and `LinkedList` for stack/queue operations.
- Doesn't allow `null`.
- Can be used as **Queue (FIFO)** or **Stack (LIFO)**.

Important Methods

- `addFirst(E e)` / `addLast(E e)` → Insert at head/tail.
- `removeFirst()` / `removeLast()` → Remove head/tail.
- `peekFirst()` / `peekLast()` → View head/tail without removal.
- `pollFirst()` / `pollLast()` → Retrieve & remove head/tail.
- `offerFirst(E e)` / `offerLast(E e)` → Queue-style insertions.
- `push(E e)` / `pop()` → Use as Stack.

Example

```
import java.util.ArrayDeque;

public class ArrayDequeDemo {
    public static void main(String[] args) {
        ArrayDeque<String> dq = new ArrayDeque<>();
        dq.addFirst("A");
        dq.addLast("B");
```

```
        dq.addLast("C");

        System.out.println("ArrayDeque: " + dq);   // [A, B, C]
        dq.removeFirst();
        System.out.println("After removeFirst: " + dq); // [B, C]

        dq.push("Z");  // acts like stack push
        System.out.println("After push: " + dq);   // [Z, B, C]
        dq.pop();      // acts like stack pop
        System.out.println("After pop: " + dq);    // [B, C]
    }
}
```