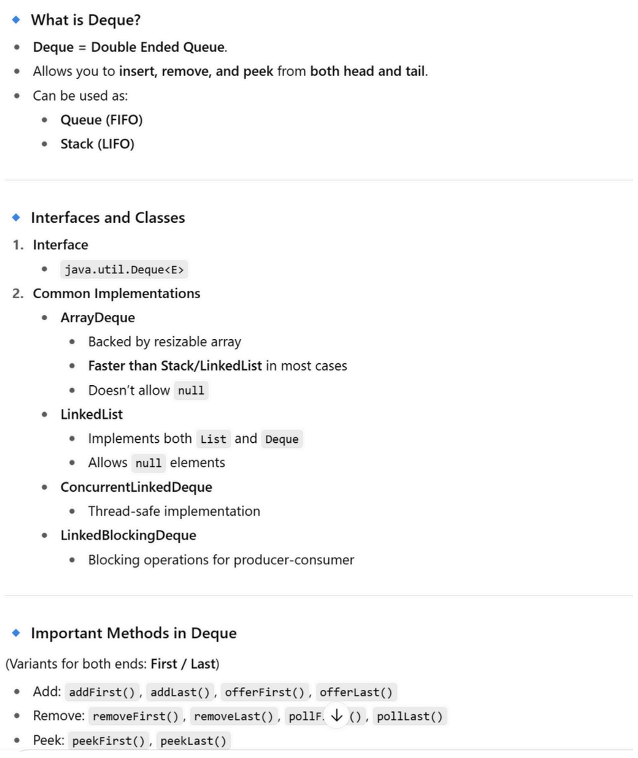

🔹 **What is Deque?**

- **Deque = Double Ended Queue.**
- Allows you to **insert, remove, and peek** from **both head and tail**.
- Can be used as:
  - **Queue (FIFO)**
  - **Stack (LIFO)**

🔹 **Interfaces and Classes**

1. **Interface**
   - `java.util.Deque<E>`
2. **Common Implementations**
   - **ArrayDeque**
     - Backed by resizable array
     - **Faster than Stack/LinkedList** in most cases
     - Doesn't allow `null`
   - **LinkedList**
     - Implements both `List` and `Deque`
     - Allows `null` elements
   - **ConcurrentLinkedDeque**
     - Thread-safe implementation
   - **LinkedBlockingDeque**
     - Blocking operations for producer-consumer

🔹 **Important Methods in Deque**

(Variants for both ends: **First / Last**)

- Add: `addFirst()`, `addLast()`, `offerFirst()`, `offerLast()`
- Remove: `removeFirst()`, `removeLast()`, `pollF ()`, `pollLast()`
- Peek: `peekFirst()`, `peekLast()`

🔹 Example: Using Deque as Stack

```java
import java.util.ArrayDeque;
import java.util.Deque;

public class DequeExample {
    public static void main(String[] args) {
        Deque<String> stack = new ArrayDeque<>();

        stack.push("A"); // addFirst
        stack.push("B");
        stack.push("C");

        System.out.println(stack.pop()); // removes C
        System.out.println(stack.pop()); // removes B
    }
}
```

✅ Output:

```css
C
B
```

🔹 Example: Using Deque as Queue

java

```java
Deque<String> queue = new ArrayDeque<>();
queue.offer("A");
queue.offer("B");
queue.offer("C");

System.out.println(queue.poll()); // removes A
System.out.println(queue.poll()); // removes B
```

✅ Output:

css

```css
A
B
```

🔹 Use Cases

1. **Stack replacement**
   - Instead of `Stack` (legacy, synchronized, slower).
   - Use `ArrayDeque` → better performance.
2. **Queue replacement**
   - Instead of `LinkedList` when you don't need random access.
3. **Sliding Window Problems** (in algorithms)
   - Example: Max element in every window of size K.
4. **Undo/Redo functionality**
   - One end for undo, other end for redo.
5. **Browser history**
   - Navigate back/forward efficiently.
