양방향 큐 는 큐의 양쪽 끝에서 데이터를 삽입하거나 삭제할 수 있는 선형 자료구조이다.
스택과 큐의 기능을 모두 갖고 있어서 다양한 알고리즘 문제 해결에 유용하게 활용된다.

양방향 큐의 특징

1. 양방향 연산 : 앞 또는 뒤에서 데이터를 추가하거나 삭제할 수 있다.
2. 스택과 큐의 결합 : 스택처럼 후입선출 방식으로 데이터를 처리하거나 큐처럼 선입선출 방식으로 처리가 할 수 있다.
3. 유연성 : 다양한 알고리즘 문제에 적용이 가능하다.

자바에서 Deque 사용

자바에서는 **java.util.Deque** 인터페이스를 통해 **Deque**를 사용할 수 있으며 이 인터페이스를 구현한 클래스로는 **ArrayDeque**와 **LinkedList** 를 사용한다.

ArrayDeque: 배열을 기반으로 구현되어 있어 랜덤 접근이 빠르지만 크기 조정 시 오버헤드(어떤 작업을 수행하는 데 있어 실제로 필요한 작업 외에 추가적으로 소모되는 자원이나 시간을 의미)가 발생할 수 있다.
LinkedList: 연결 리스트를 기반으로 구현되어 있어 삽입/삭제 연산이 빠르지만 랜덤 접근이 느리다.

결론: 대부분의 연산에서 ArrayDeque가 LinkedList보다 빠르지만 배열의 크기 조정이 자주 발생하는 경우에는 LinkedList가 더 효율적일 수 있다.

자바에서 양방향 큐 선언시

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

public class DequeExample {
    public static void main(String[] args) {
        Deque<Integer> deque = new ArrayDeque<>();
        // 데이터 추가
        deque.addFirst(10);
        deque.addLast(20);
        deque.addFirst(5);

        // 데이터 출력
        while (!deque.isEmpty()) {
            System.out.print(deque.removeFirst() + " ");
        }
    }
}
```

메소드:

addFirst(e) : 큐의 앞에 요소 e를 추가
addLast(e) : 큐의 뒤에 요소 e를 추가
removeFirst() : 큐의 앞에서 요소를 제거하고 반환
removeLast() : 큐의 뒤에서 요소를 제거하고 반환
peekFirst() : 큐의 앞에 있는 요소를 반환하지만 제거하지는 않음
peekLast() : 큐의 뒤에 있는 요소를 반환하지만 제거하지는 않음