스택 은 가장 최근에 추가된 데이터가 가장 먼저 삭제되는 후입선출(LIFO) 방식의 선형 자료구조이다.

스택의 특징

1. 후입선출 (LIFO) : 가장 나중에 들어온 데이터가 가장 먼저 나간다.
2. 한쪽 끝에서만 접근 : 데이터의 삽입과 삭제이 스택의 한쪽 끝에서만 이루어진다.
3. 스택의 시간 복잡도 : push, pop 연산은 일반적으로 O(1)의 시간 복잡도를 가진다.

자바에서 스택 선언시

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

public class StackExample {
    public static void main(String[] args) {
        Deque<String> stack = new ArrayDeque<>();
        //스택처럼 사용
        stack.push("aaa");
        stack.push("bbb");
        stack.push("ccc");

        //스택의 맨 위 요소 확인
        System.out.println("맨 위의 요소: " + stack.peek());
        //ccc

        //스택의 맨 위 요소 제거
        while (!stack.isEmpty()) {
            System.out.println(stack.pop());
        }
        //스택에 특정 요소가 있는지 확인
        System.out.println(stack.contains("bbb"));
        //false
    }
}
```

java.util.Stack 클래스는 더 이상 권장되지 않고 ArrayDeque 클래스로 스택과 큐의 기능을 구현한다. ArrayDeque클래스는 Deque 인터페이스를 구현한 클래스이다.

주요 메소드

메소드	설명
push(e)	스택의 맨 위에 요소 추가

메소드	설명
pop()	스택의 맨 위 요소 제거 및 반환
peek()	스택의 맨 위 요소 확인
isEmpty()	스택이 비어 있는지 확인
contains(Object o)	스택에 특정 객체가 있는지 확인 (정확한 위치는 제공하지 않음)

ArrayList vs LinkedList vs ArrayDeque

특징	ArrayList	LinkedList	ArrayDeque
삽입/삭제	O(n)	O(1)	O(1)
랜덤 접근	O(1)	O(n)	O(1)
메모리 효율성	보통	좋음	매우 좋음
스택/큐	스택으로 사용 가능하지만 비효율적	스택으로 사용 가능	스택과 큐 모두 효율적으로 사용 가능