0X05 스택

상태	완료
담당자	㉮ 서연

스택이란?

LIFO

> 💡 가장 최근에 들어온 값이 먼저 나가게 되는 한 쪽 방향만으로 삽입 삭제가 이루어지는 자료구조.

Restricted Structure

> 💡 삽입과 삭제의 제한이 있다는 자료구조임. 여기에 스택, 큐, 덱이 속함

스택의 성질

1. 삽입과 삭제

시간복잡도 O(1)이 소요됨.

- 스택에 가장 나중에 쌓인 값을 빼고 삭제하기 때문에 추가적인 연산이 필요하지 않음.
- 배열로 보면 가장 마지막 위치에 있는 값을 삭제하고 삽입하는 개념임.

2. 상단 값 확인

시간복잡도 O(1)이 소요됨.

- 스택에 가장 상단 값을 확인하기 때문에 크기만 가지고 있다면 언제든 가장 위의 값을 알 수 있음.
- 배열로 보면 마지막 인덱스에 직접 접근하여 검색하는 개념임

3. 스택의 개념 상 가장 상단의 값이 아닌 값은 조회 및 변경이 불가능

- 그러나. 직접 자료구조를 구현할 경우에는 필요에 의해 정의할 수 있음.

스택의 구현

배열

필요 변수

1. 원소를 담은 큰 배열
2. 인덱스를 저장할 변수 한 개 = 현재 가장 탑 값의 인덱스를 기억하고자.

삽입

- 삽입할 원소를 받고 스택의 top에 삽입하는 메소드
- 삽입 알고리즘

> 1. 현재 top 값의 위치를 가지고 있는 변수(=pos)를 1 증가시킴
> 2. 배열로 구현한 스택에 pos 위치에 삽입할 데이터를 저장함.

```
private static void push(int data) {
        dat[++pos]= data;
    }
```

삭제

- 현재 스택의 상단의 원소를 삭제하는 메소드
- 삭제 알고리즘

1. 현재 배열에서 pos 위치를 1 감소 시킴.

이게 삭제? → 삭제가 맞음. 현재 pos의 값을 1 감소시키면 본래의 top값은 더 이상 스택의 원소가 아니게 되기 때문임. 그 다음 삽입 연산이 일어나면 덮어씌워짐

```
private static void pop() {
        pos--;
    }
```

top 확인

- 현재 스택의 상단의 원소를 확인하는 메소드.
- 확인 알고리즘

1. 배열의 인덱스로 pos를 집어넣어서 값 확인

```
private static int top() {
        return dat[pos];
    }
```

연결리스트

- 구현은 가능하지만 직접해보지는 않을 거라고 함. → 근데 저는 해봤음...
- push, pop 연산을 수행하는데 걸리는 시간은 맨 앞의 노드를 저장하고 삭제하기 때문에 **O(1)**시간이 소요됨.

  ~~(노드 클래스가 정의 되었다고 가정)~~
- 스택 top 항목을 가진 노드를 가리키는 레퍼런스 변수 top
- 스택의 항목 수를 가진 size
- 메인에서 얘를 테스팅 해보면 성공적

```
public class ListStack<E> implements IListStack<E>{
    private Node<E> top;
    private int size;
    public ListStack(){
        top=null;
        size=0;
    }
    public int size() {
        return size;
    }
    public boolean isEmpty(){
        return size == 0;
    }
    //peek(), push(), pop()선언
    public E peek(){
        if(isEmpty()){
            throw new EmptyStackExecption();
        }
        return top.getItem();
    }
    public void push(E newItem){
        Node<E> newNode = new Node(newItem, top);
        top= newNode;
        size++;
    }
    public E pop(){
        if(isEmpty()){
            throw new EmptyStackExecption();
        }
        E topItem = top.getItem();
        top=top.getNext();
        size--;
        return topItem;
    }
}
```

▼ 전체코드

```
public class stack_test {

    static final int MX= 1000005;
    static int dat[]= new int[MX];
    static int pos=-1;

    private static void push(int data) {
        dat[++pos]= data;
    }

    private static void pop() {
        pos--;
    }

    private static int top() {
        return dat[pos];
    }

    private static void test() {
        push(5);

        push(4);

        push(3);
        System.out.println(top());
        pop();
        pop();
        System.out.println(top());
        push(10);
        push(12);
        System.out.println(top());
        pop();
        System.out.println(top());


    }
```

```
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        test();
    }

}
```

추가적인 꿀팁

현재 자바에서는 스택 클래스를 지원함.

코테에서는 이걸 스택처럼 사용하는게 유리하다고 하네요~

```
Deque<Integer> stack = new ArrayDeque<>();
```