자료구조 실습 보고서

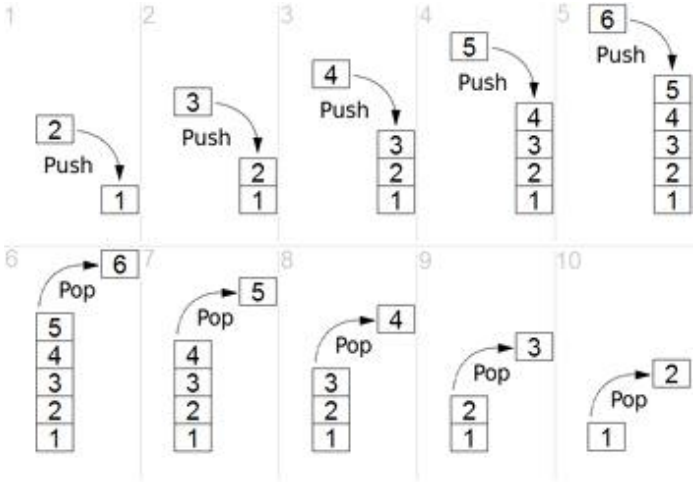
[제 6 주]

제출일 : 2018.4.11.수요일

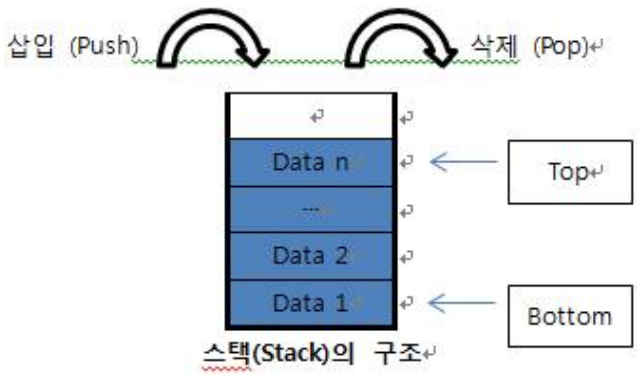
학번 : 201604137

이름 : 김예지

[프로그램 설명서]



Stack이란 모든 원소들의 삽입(insert)과 삭제(delete)가 리스트의 한쪽 끝에서만 수행되는 제한 조건을 가지는 선형 자료 구조(linear data structure)로서, 삽입과 삭제가 일어나는 리스트의 끝을 top이라 하고, 다른 한쪽 끝을 bottom이라 한다.
스택(Stack)에 저장된 데이터 항목들 중에서 먼저 삽입된 데이터가 나중에 삭제되고, 나중에 삽입된 데이터가 먼저 삭제된다는 뜻에서 후입 선출 LIFO (Last In First Out)이라고 부른다.



스택에 새 데이터를 삽입하는 것을 푸시(Push), 삭제하는 것을 팝(Pop)이라 하는데, 가장 최근에 삽입된 데이터부터 삭제한다.

[실행 결과 분석]

```
public interface IntStack {
    public boolean isEmpty();
    public int pop();
    public int popBottom();
    public void push(int i);
    public int size();

}
```

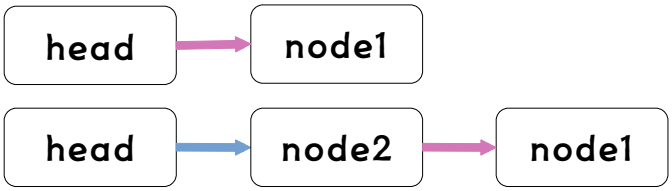
스택이 비었는지 확인해주는 isEmpty, 가장 나중에 들어온 데이터를 삭제해주는 pop, 가장 처음에 넣은 데이터를 삭제해주는 popBottom, 데이터를 입력하는 push, 사이즈를 출력해주는 size로 구성되어있다.

```
public class Node {
    private int data;
    private Node next;


    public Node(int data) {
        this.data = data;
        this.next = null;
    }

    public int getData() {
        return data;
    }

    public void setData(int data) {
        this.data = data;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }


}
```

노드는 기본적으로 데이터를 저장하고, 다음 노드의 주소값을 담는 것으로 구성되어있다.

```
//데이터 삽입
@Override
public void push(int data) {
    Node newNode = new Node(data);
    newNode.setNext(this.head);
    this.head=newNode;
    this.size++;

}
```

push: 맨 처음 값을 입력하는데, 새로운 노드를 생성하여 그 안에 데이터를 넣고 헤드가 가리키고 있던 주소값을 가지게 설정한다. 그리고 헤드는 새로운 노드의 주소값을 가진다.



```java
//마지막에 들어온 데이터 삭제
@Override
public int pop() {
    int target=0;
    Node currNode = this.head; // currentNode

    if(isEmpty()) return -1;
    //변수 target에 마지막 노드 데이터값 저장
    //새로운노드.getNaxt를 head에 넣어줌(연결끊김)
    //target반환
    target=currNode.getData();
    this.head=currNode.getNext();
    size--;
    return target;


}
```

pop: 마지막에 들어온 데이터의 주소값은 헤드가 가지고 있기 때문에 현재노드를 헤드로 설정해주고 값을 변수에 저장 해준뒤에 헤드가 두 번째 노드의 주소값을 가지게 설정해주고 사이즈를 줄이면 연결이 끊기면서 값이 삭제된다.

```java
//처음으로 들어온 데이터 삭제
@Override
public int popBottom() {
    Node prevNode=null;
    Node currNode = this.head;
    int target=0;

    if(isEmpty()) return -1;

    while(currNode!=null) {//널일 때 까지 (끝까지)탐색
        if(currNode.getNext()==null) { //마지막 바로 전 노드에 도착하면
            target=currNode.getData(); //현재 노드의 데이터를 target에 저장하고
            prevNode.setNext(null); // 현재노드를 null로 만듦으로써 prev노드를 마지막 노드로 만들어줌
            break;
        }
        //마지막 노드의 전노드가 아닐 경우에는

        prevNode=currNode;
        currNode=currNode.getNext();
    }
    size--;
    return target;
}
```

popBottom: 현재노드가 null일 때 까지 탐색을 시작하다가 현재노드의 다음노드가 없다면(null) 마지막 노드인 것을 의미한다. 마지막노드까지 왔다면 변수에 현재 값을 저장하고 prevNode의 다음값을 null로 세팅해준다.(=마지막 노드가 되었음을 의미) 만약, 마지막 노드가 아니라면 prevNode를 현재로 해주고 현재노드에 다음노드를 담는다.(순서 이동) 그리고 사이즈를 줄이고 변수(삭제한 노드의 데이터)를 반환한다.
prevNode.setNext(null); 이 currNode=null;이면 절대 안되는데, 이유는

이렇게 값을 넣었다고 가정했을 때, 마지막 if문 순서에서 prevNode=3 , currNode=3 이 되고 currNode=null; 한다고 해도 prevNode는 여전히 3이다.
(a=1; b=a; 하면 b=1이 된다. 그 후에 a=3; 이라고 지정해도 b는 여전히 1인 것처럼)

```
1.스택에 내용 추가
2.스택에 마지막으로 들어온 내용 삭제
3.스택에 처음 들어온 내용 삭제
4.사이즈 출력
5.내용 출력
6.종료
5
5 4 3 2 1
1.스택에 내용 추가
2.스택에 마지막으로 들어온 내용 삭제
3.스택에 처음 들어온 내용 삭제
4.사이즈 출력
5.내용 출력
6.종료
2
삭제한 노드 값:5
삭제 한 결과: 4 3 2 1
1.스택에 내용 추가
2.스택에 마지막으로 들어온 내용 삭제
3.스택에 처음 들어온 내용 삭제
4.사이즈 출력
5.내용 출력
6.종료
3
삭제한 노드 값:1
삭제 한 결과: 4 3 2
```