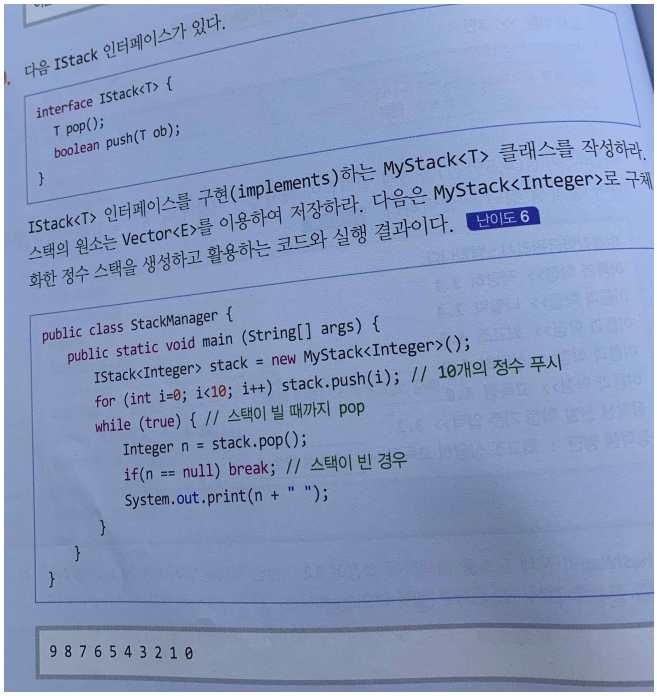
과목명	객체지향프로그래밍	분반	x	담당교수	김화성 교수님
학과	전자통신공학과	학번	2016707079	이름	하상천
과제명: Generic class 작성					

1, 과제설명 (사용자 요구사항 기술: 과제에 대한 설명 및 목표)



2, 사용자 요구사항을 정형적 방법으로 기술 (UML, Pseudo code, 그림등을 이용하여 기술)

Vector<E>

* List<E> 인터페이스를 구현한 클래스로서 가변 개수의 배열이 필요할 때 적합함.
* 벡터에 삽입되는 요소의 수가 많아지면 자동으로 크기가 조절됨.
* 요소는 벡터의 맨 마지막이나 중간에 삽입 가능. 이 경우 벡터는 삽입되는 요소의 뒤에 있는 모든 요소들을 한자리씩 뒤로 이동시킴.
* 요소를 삭제하면 뒤의 요소들을 앞으로 하나씩 이동시킴.

벡터 생성

정수 값만 삽입 가능한 벡터를 만들고자 하면 E에 Integer를 삽입하여

```
Vector<Integer> v = new Vector<Integer>();
```

* 용량을 처음에 설정하고 싶다면

```
Vector<Integer> v = new Vector<Integer>(5); // 초기 용량이 5인 벡터 생성.
```

3. 알고리즘 및 자료구조 설계 내용

for문과 push 메소드를 통해 0부터 9까지의 숫자를 벡터에 넣어주었다. 그리고 pop 메소드를 통해 마지막에 들어간 숫자부터 나오도록 코드를 구성하였다. size() 메소드와 remove 메소드를 통해 맨 마지막에 들어있는 숫자를 가리키는 레퍼런스를 리턴하고, 그 값을 바로 지우는 방식으로 코드를 구성하였다. 만약 벡터안의 요소가 없을 시 size 메소드가 0을 리턴하고, 그럴 경우에 pop 메소드에서 null을 리턴하도록 하였다. break문을 통해 while문을 탈출 하도록 구성하였다.

4. 소스코드 설명 (직접 작성한 소스코드중에 핵심 부분을 발췌하여 설명)

```
Vector<T> v = new Vector<T>(10);
```

초기 용량이 10인 벡터를 생성하는 코드이다.

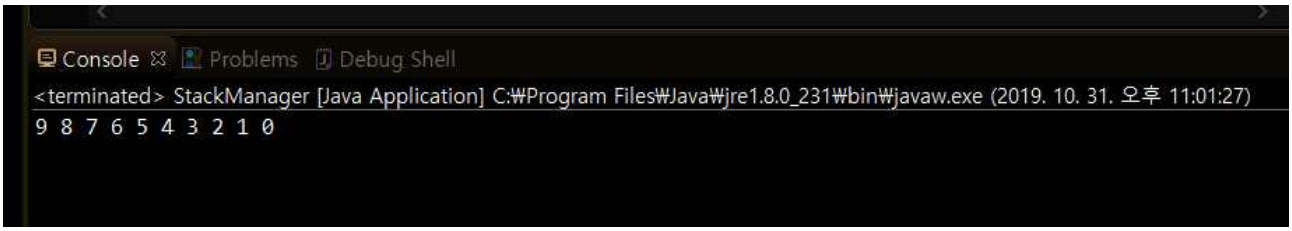
```
int num = v.size();
```

벡터에 들어있는 요소의 개수를 num에 대입한다.

```
if(n==null) {break;}
```

레퍼런스 변수 n이 null 이라면 break를 통해 while문을 탈출한다.

5. 실행결과 및 설명 (실행 결과를 캡쳐하여 첨부한 후 설명)



(그림을 문서에 포함, 글자처럼 취급 옵션, 잉크 절약과 잘 보이게 하기위해 그림 반전)

작성된 프로그램을 실행한 결과 위와 같은 화면이 출력 되었다.

6. 고찰 (과제를 진행하면서 배운점 이나, 시행 착오 내용, 기타 느낀점)

제네릭 클라스를 수업시간에 듣고 배운 것과 이렇게 직접 코드를 작성해보며 문제를 풀어보니 또 다른 느낌이었다. 처음에 pop 메소드와 push 메소드를 오버라이딩 할 때 접근지정자를 그냥 default로 하고 작성 하였는데 오류가 발생했었다. 다시 한번 생각해보니 interface는 접근지정자가 모두 public이기 때문에 오버라이딩 하는 메소드도 접근지정자가 public 이어야만 했던 것이었다. 그래서 접근지정자를 바꿔주니 오류가 발생하지 않았다. 또한 pop 메소드에서 값을 가리키는 레퍼런스를 리턴하고, 그 값을 지우고 싶었는데 이것을 어떻게 코드를 짜야 되는지 고민을 많이 하다가 remove 메소드를 리턴자리에 넣어주었더니 해결 되었다. 과제를 진행하면서 다시 한번 interface의 접근지정자가 public 이라는 것을 몸소 배웠고, 제네릭에 대해 조금 더 친근하게 다가갈 수 있을 것 같다.

7. 전체 소스코드 (글자크기 9에 줄간격을 120%로 유지하고 한 줄이 너무 길지 않게 작성)

```
package HA_homework;
import java.util.Vector;

interface IStack<T> {
T pop();
boolean push(T ob);
}

class Mystack<T> implements IStack<T>{
Vector<T> v = new Vector<T>(10);
public T pop() { // 맨 마지막에 있는 값을 가리키는 레퍼런스를 하나씩 리턴해주는 메소드
int num = v.size();
if(num == 0) {
return null;
}
else {
return v.remove(num-1); //리턴해주고 remove 메소드를 통해 값 삭제
}
```

```
}
public boolean push(T ob) { //새로운 값을 넣어주는 메소드
v.add(ob);
return true;
}
}
public class StackManager {

public static void main(String[] args) {
IStack<Integer> Stack = new Mystack<Integer>();    //Mystack객체생성
for(int i=0; i<10; i++) {                          //0부터9까지 값을 넣어주는 반복문
Stack.push(i);
}
while(true) {                                      //더 이상 값이 없을 때 까지 반복
Integer n = Stack.pop();
if(n==null) {
break;
}
System.out.print(n+" ");                           //값출력
}

}

}
```

(글자크기는 10으로 유지하고 줄간격도 160%를 유지할 것)