

소프트웨어프로젝트

Project 4

소프트웨어학과 학과
서준상 이름
20201672 학번
2021/05/08(토) 날짜

1. 소스 프로그램

- GenericStackLimitedCapacity

```
class GenericStackLimitedCapacity {
    protected int top = -1;
    private final int INIT_CAPACITY = 2;
    protected Object[] list;

    GenericStackLimitedCapacity() {
        list = new Object[INIT_CAPACITY];
    }

    GenericStackLimitedCapacity(int capacity) {
        list = new Object[capacity];
    }

    public boolean isEmpty() {
        return (top < 0);
    }

    public Object pop() {
        if (top >= 0) {
            return list[top--];
        } else {
            return null;
        }
    }

    public void push(Object o) {
        list[++top] = o;
    }

    public String toString() {
        String ss = "";
        for (int i=0; i<=top; i++) {
            ss += list[i] + " ";
        }
        return ss;
    }
}
```

- GenericStack

```
public class GenericStack extends GenericStackLimitedCapacity {

    private final int INIT_CAPACITY = 2;

    GenericStack() {
        list = new Object[INIT_CAPACITY];
    }

    GenericStack(int capacity) {
        list = new Object[capacity];
    }

    public boolean isEmpty() {
        return super.isEmpty();
    }

    public Object pop() {
        return super.pop();
    }

    public void push(Object o) {
        int size = list.length;

        if ( top < size-1 ) {
            super.push(o);
        } else {
            int arraySize = list.length;
            Object[] copyList = new Object[ arraySize*2 ];
            for (int i = 0; i < arraySize; i = i+1 ) {
                copyList[i] = list[i];
            }
            list = new Object[arraySize*2];
            for (int i = 0; i < arraySize; i = i+1 ) {
                list[i] = copyList[i];
            }
            super.push(o);
        }
    }

    public String toString() {
        return super.toString();
    }
}
```

- ParaStack

```
public class ParaStack extends GenericStack {

    private final int INIT_CAPACITY = 2;

    ParaStack() {
        list = new Object[INIT_CAPACITY];
    }

    ParaStack(int capacity) {
        list = new Object[capacity];
    }

    public boolean isEmpty() {
        return super.isEmpty();
    }

    public Object pop() {
        return super.pop();
    }

    public void push(Object o) {
        if ( o instanceof String ) {
            super.push(o);
        } else {
            String token;
            token = "" + o;
            super.push(token);
        }
    }

    public String toString() {
        return super.toString();
    }
}
```

2. 설계 노트

- GenericStackLimitedCapacity

이 부분은 과제설명서에 주어졌으므로, 거의 다 똑같습니다.

과제설명서의 코드와 다른 부분이 있다면,

과제설명서엔 private로 되어있는 변수 top과, Stack을 저장하는 배열 list[]를 다른 class에서도 참조할 수 있도록 하기 위해 protected로 수정하였습니다.

```
protected int top = -1;  
private final int INIT_CAPACITY = 2;  
protected Object[] list;
```

- GenericStack

클래스를

`public class GenericStack extends GenericStackLimitedCapacity { }`로 선언하여 클래스 `GenericStackLimitedCapacity`를 상속하였습니다.

Stack의 저장 공간 (`list[]`)이 가득 차서 더 이상 새로운 값을 `push`할 수 없을 경우, 배열의 크기를 두 배 늘리는 method를 구현하기 위해 수정해야할 부분은 저장 공간이 부족함을 바로 파악할 수 있는 `push()` 메소드라 판단했습니다.

따라서

`push()` 외의 다른 메소드들은 전부 `return super.(메소드);` 를 통해 그대로 유지하고 `push()`는 overriding을 통해 새로 작성하였습니다.

```
public void push(Object o) {
    int size = list.length;

    if ( top < size-1 ) {
        super.push(o);
    } else {
        int arraySize = list.length;
        Object[] copyList = new Object[ arraySize*2 ];

        for (int i = 0; i < arraySize; i = i+1 ) {
            copyList[i] = list[i];
        }

        list = new Object[arraySize*2];
        for (int i = 0; i < arraySize; i = i+1 ) {
            list[i] = copyList[i];
        }
        super.push(o);
    }
}
```

현재의 `list`의 크기를 `size`에 저장하여,

`top` 값이 `size-1` 보다 작을 경우엔, 저장 공간이 가득 차지 않은 것이므로,

`super.push(o);`를 통해 `GenericStackLimitedCapacity`의 메소드를 그대로 실행하도록 하고,

만약 아닐 경우엔, 저장 공간이 부족하단 뜻이므로,

복사를 위한 배열인 `copyList[arraySize*2]`를 통해 2배 더 큰 배열을 만든 뒤,

0부터 `top`까지의 저장된 값을 `copyList`에 복사하고,

`list = new Object[arraySize*2];`를 통해 크기가 2배인 새로운 `list`를 생성한 뒤,

`copyList`에 저장한 값을 다시 `list`로 복사하는 방식을 통해 구현하였습니다.

- ParaStack

```
public void push(Object o) {  
    if ( o instanceof String ) {  
        super.push(o);  
    } else {  
        String token;  
        token = "" + o;  
        super.push(token);  
    }  
}
```

마찬가지로 push() 메소드만 수정하여 구현하였습니다.

Object o 의 type를 검사하여, 만약 o가 String일 경우,
super.push(o);를 통해 상속받은 기존 클래스의 메소드를 그대로 실행하도록 했고,

o가 String이 아닐 경우엔
token = "" + o를 통해 Object o를 스트링 타입으로 형변환한 뒤,
super.push(token)을 통해 스트링 타입으로 Stack에 저장되도록 하였습니다.

강의계획서에 나온 질문인

- 4) 다양한 GenericStackLimitedCapacity/GenericStack/ParaStack 테스트
 - 예를 들어, int형과 String형 데이터를 같은 스택에 넣는 것 실험
 - 이 기능은 장점? 단점

이 부분에 대해 생각해보면, 상황에 따라 장단점이 있을 것 같습니다

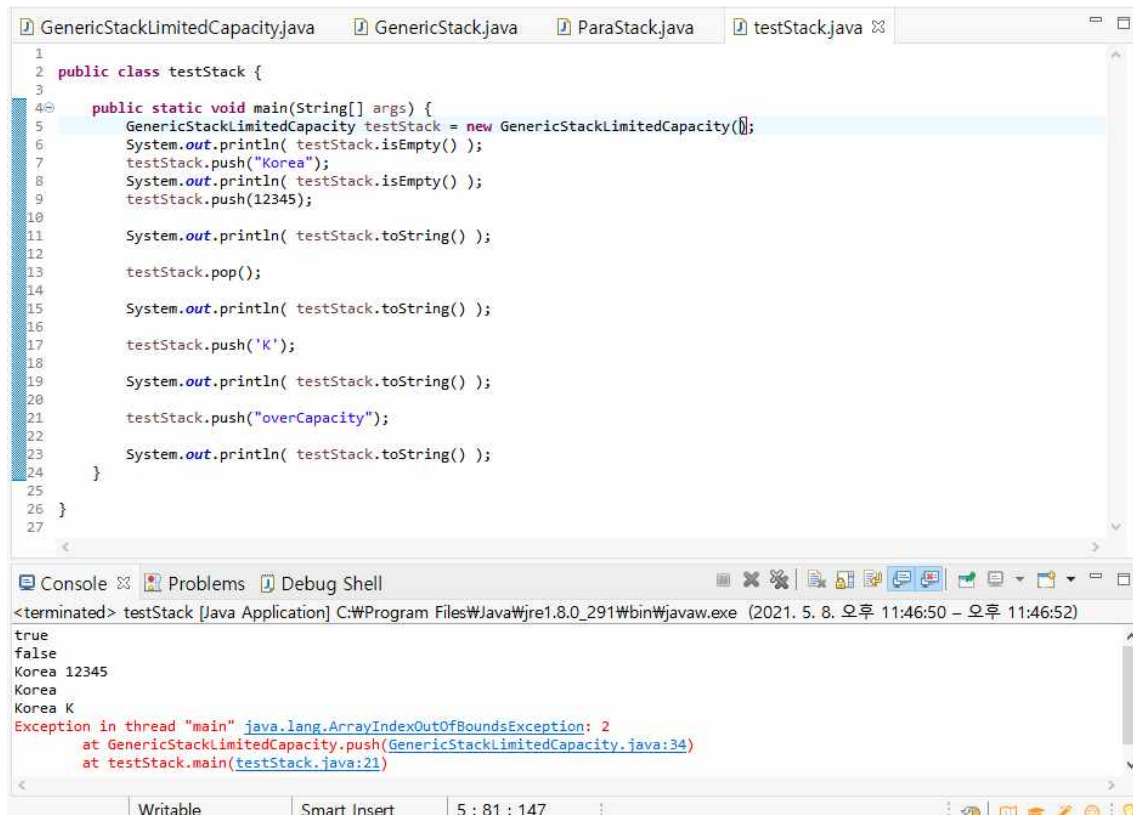
강의에서 여러번 언급된 것처럼

여러 사람이 참여하는 프로젝트의 경우엔, Stack에 저장할 때,
push되는 값들의 datatype이 하나로 통일되도록 하는 것이
프로그래머 간의 코드 충돌을 예방할 수 있을 것이므로,
이 점에선 같이 넣는 것이 안 좋다고 생각합니다.

하지만, 예를 들어, 여러 학생들의 성적을 Stack에 저장하였다가,
나중에 Stack 안에 저장된 값들을 이용하여 전체 학생의 평균 점수를 구하는 등의 작업에선
int나 double처럼 연산이 가능한 datatype으로 저장되는 것이 편리하지만,
제가 작성한 코드처럼 모든 값을 String으로 저장하는 Parameterized Generic Stack은
저장된 값을 꺼내도 String type이기 때문에
다시 형변환을 거쳐야하므로, 이 부분에선 하나로 통일하는 것이 불편할 것이라 생각합니다.

3. 테스트 과정

- GenericStackLimitedCapacity



```
1 public class testStack {
2
3
4     public static void main(String[] args) {
5         GenericStackLimitedCapacity testStack = new GenericStackLimitedCapacity(2);
6         System.out.println( testStack.isEmpty() );
7         testStack.push("Korea");
8         System.out.println( testStack.isEmpty() );
9         testStack.push(12345);
10
11         System.out.println( testStack.toString() );
12
13         testStack.pop();
14
15         System.out.println( testStack.toString() );
16
17         testStack.push('K');
18
19         System.out.println( testStack.toString() );
20
21         testStack.push("overCapacity");
22
23         System.out.println( testStack.toString() );
24     }
25 }
26
27
```

Console Output:

```
<terminated> testStack [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw.exe (2021. 5. 8. 오후 11:46:50 - 오후 11:46:52)
true
false
Korea 12345
Korea
Korea K
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 2
    at GenericStackLimitedCapacity.push(GenericStackLimitedCapacity.java:34)
    at testStack.main(testStack.java:21)
```

우선 testStack를 생성한 다음에 isEmpty() 결과를 한번 출력한 뒤,
String인 "Korea"를 push하고, 다시 한번 isEmpty() 결과를 출력하였습니다.

그 뒤, int인 12345를 push,
12345를 pop()을 통해 제거,
다시 'K'를 push 하는 동작을 실행했습니다.

그 이후, 이미 2개가 들어간 상태에서 하나를 더 push 할 경우를 실험해보기 위해
"overCapacity"를 push 했습니다. (list 크기 실험을 위해 고의로 오류를 발생시킴)
=> 3번째 push이기 때문에 "overCapacity"가 들어가야 할 공간이
INIT_CAPACITY인 2보다 큰 위치인 list[2] 이므로, 오류가 발생하였습니다.

The screenshot shows an IDE with four tabs: GenericStackLimitedCapacity.java, GenericStack.java, ParaStack.java, and testStack.java. The testStack.java tab is active, displaying the following code:

```
1 public class testStack {
2
3
4     public static void main(String[] args) {
5         GenericStackLimitedCapacity testStack = new GenericStackLimitedCapacity(3);
6         System.out.println( testStack.isEmpty() );
7         testStack.push("Korea");
8         System.out.println( testStack.isEmpty() );
9         testStack.push(12345);
10
11         System.out.println( testStack.toString() );
12
13         testStack.pop();
14
15         System.out.println( testStack.toString() );
16
17         testStack.push('K');
18
19         System.out.println( testStack.toString() );
20
21         testStack.push("overCapacity");
22
23         System.out.println( testStack.toString() );
24     }
25 }
26
27 }
```

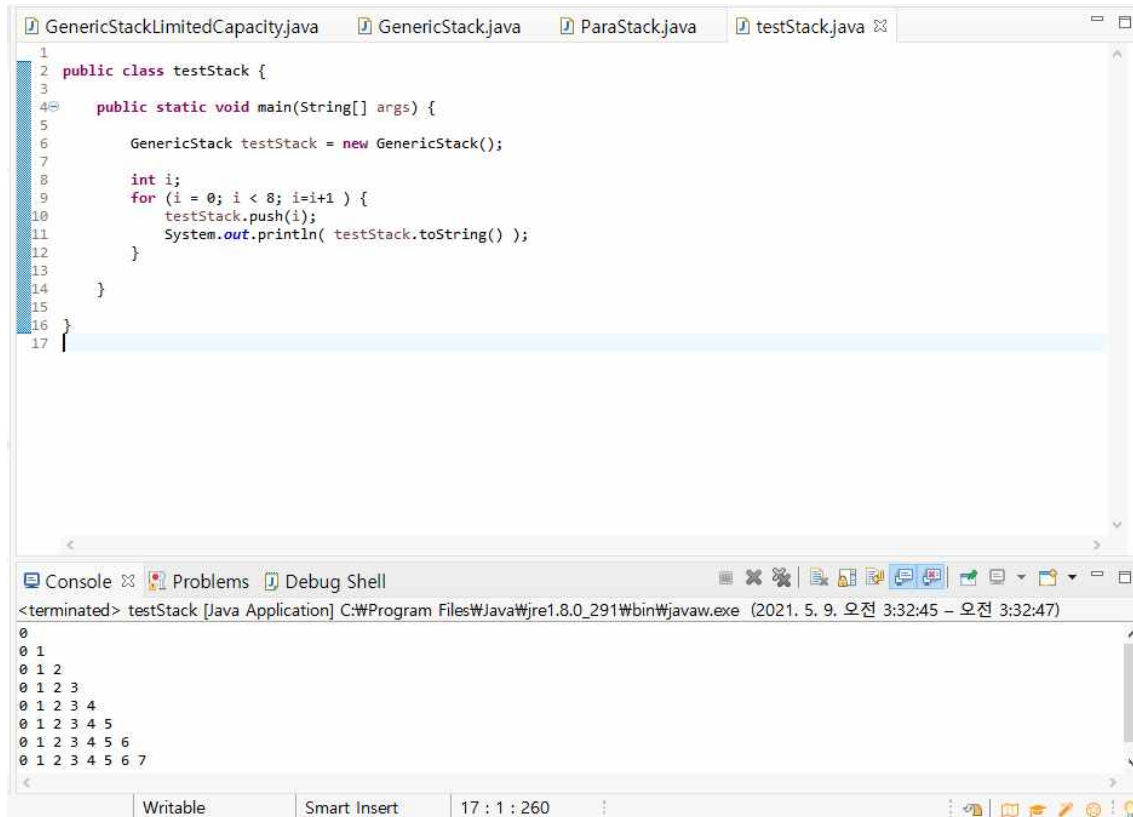
Below the code editor, the Console window shows the output of the program:

```
<terminated> testStack [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw.exe (2021. 5. 8. 오후 11:47:18 - 오후 11:47:20)
true
false
Korea 12345
Korea
Korea K
Korea K overCapacity
```

test class의 다른 부분은 앞에서 한 test와 같은 상태에서
GenericStackLimitedCapacity testStack = new GenericStackLimitedCapacity(); 부분만
GenericStackLimitedCapacity testStack = new GenericStackLimitedCapacity(3); 으로
수정하여 최대 3개까지 push할 수 있도록 만든 뒤 실행해보았습니다.

Stack의 크기가 3으로 늘어났으므로, "overCapacity" 도 정상적으로 push됩니다.

- GenericStack



```
1 public class testStack {
2
3
4     public static void main(String[] args) {
5
6         GenericStack testStack = new GenericStack();
7
8         int i;
9         for (i = 0; i < 8; i=i+1 ) {
10             testStack.push(i);
11             System.out.println( testStack.toString() );
12         }
13
14     }
15
16 }
17 }
```

Console Output:

```
<terminated> testStack [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw.exe (2021. 5. 9. 오전 3:32:45 - 오전 3:32:47)
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
```

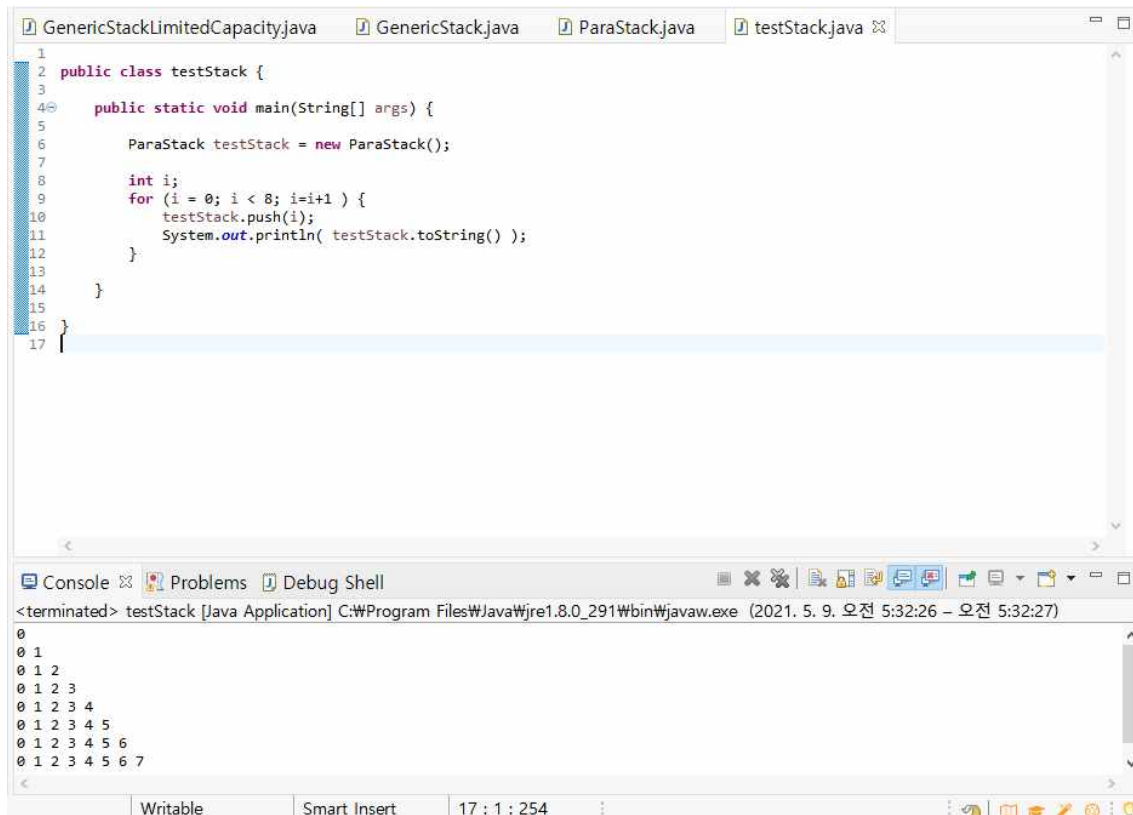
GenericStack testStack = new GenericStack(); 을 통해 생성된 최초 스택 testStack의 크기는 INIT_CAPACITY인 2 입니다.

그 뒤 for문을 통해 0부터 7까지의 자연수를 하나씩 push하고, 하나를 push할 때 마다 스택에 저장된 내용을 testStack.toString()을 출력하여 확인하면,

최초 공간의 크기는 2임에도 0부터 7까지, 2보다 큰 8개의 값 모두 스택에 정상적으로 push 되었음을 확인할 수 있습니다.

수정이 필요한 push 이외의 pop 등 다른 동작들은 전부 super.pop() 형식으로 가져왔고, override 하지 않은 pop(), isEmpty() 등의 method의 작동은 이미 GenericStackLimitedCapacity 테스트에서 확인했으므로, 따로 실험하지 않았습니다.

- ParaStack



The screenshot shows an IDE with four tabs: GenericStackLimitedCapacity.java, GenericStack.java, ParaStack.java, and testStack.java. The testStack.java file is active, showing the following code:

```
1 public class testStack {  
2  
3     public static void main(String[] args) {  
4  
5         ParaStack testStack = new ParaStack();  
6  
7         int i;  
8         for (i = 0; i < 8; i=i+1 ) {  
9             testStack.push(i);  
10            System.out.println( testStack.toString() );  
11        }  
12    }  
13  
14  
15  
16 }  
17 }
```

The console output shows the following sequence of numbers:

```
<terminated> testStack [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw.exe (2021. 5. 9. 오전 5:32:26 - 오전 5:32:27)  
0  
0 1  
0 1 2  
0 1 2 3  
0 1 2 3 4  
0 1 2 3 4 5  
0 1 2 3 4 5 6  
0 1 2 3 4 5 6 7
```

ParaStack testStack = new ParaStack(); 을 통해 테스트를 진행하였습니다.

우선, String만을 지원하는 Stack이라는 것 외엔 GenericStack과 차이가 없으므로, GenericStack 테스트와 동일한 test 코드를 실행했을 때에 나오는 결과 역시 똑같습니다.

```

31
32 public void push(Object o) {
33     if ( o instanceof String ) {
34         super.push(o);
35     } else {
36         String token;
37         token = "n" + o;
38         super.push(token);
39     }
40 }
41

```

위의 테스트만으론 차이점을 직관적으로 알기 어려울 것 같아 약간의 차이를 만들었습니다.

class ParaStack에서 만약 push할 object가 String 타입이 아닐 경우,
 token = "" + o; 를 통해 Casting(형변환)하는 부분을
 token = "n" + o; 로 수정하여, push 과정의 차이를 알 수 있도록 하였습니다.

The screenshot shows an IDE with four tabs: GenericStackLimitedCapacity.java, GenericStack.java, ParaStack.java, and testStack.java. The testStack.java file is active, showing the following code:

```

1 public class testStack {
2
3
4 public static void main(String[] args) {
5
6     ParaStack testStack = new ParaStack();
7
8     int i;
9     for (i = 0; i < 8; i=i+1 ) {
10         testStack.push(i);
11         System.out.println( testStack.toString() );
12     }
13
14     testStack.push("8");
15     System.out.println( testStack.toString() );
16
17 }
18
19 }
20

```

The console output shows the following sequence of strings:

```

<terminated> testStack [Java Application] C:\Program Files\Java\jre1.8.0_291\bin\javaw.exe (2021. 5. 9. 오전 5:41:32 - 오전 5:41:34)
n0
n0 n1
n0 n1 n2
n0 n1 n2 n3
n0 n1 n2 n3 n4
n0 n1 n2 n3 n4 n5
n0 n1 n2 n3 n4 n5 n6
n0 n1 n2 n3 n4 n5 n6 n7
n0 n1 n2 n3 n4 n5 n6 n7 8

```

이 경우, for문과, int i 를 통해 push할 때,
 i 는 String 타입이 아닌 int타입이므로, Casting 과정에서 앞에 "n"이 붙지만,
 처음부터 String 타입인 "8"은 push할 때 앞에 "n"이 붙지 않습니다.
 이를 통해 ParaStack 클래스는 Stack에 object를 push할 때,
 String 타입으로만 저장된다는 것을 확인할 수 있습니다.

4. 자체평가표

평 가 표

평가 항목	학생 자체 평가 (리포트 해당 부분 표시 및 간단한 의견)	평가 (빈칸)	점수 (빈칸)
- GenericStack 구현? * super class 개념 이용해서 * array 용량 제한을 해결 * inheritance 이용 필수 - 실험으로 검증? * 검증 사항? generic, 용량	GenericStack 구현하였습니다. 용량 이론상 unLimited임을 확인 실험 내용은 보고서의 '3. 테스트 과정'의 '- GenericStack' 파트에 첨부		
- ParaStack 구현? * parameterized coding * type-safe 작업 - 실험으로 검증? * parameterizing * type-safety	ParaStack 구현하였습니다. String type만 저장되도록 구현 과제설명서의 예시엔 [예) ParaStack<String> s;]로 나왔지만, 2) GenericStack를 참조하라는 설명에 따라 Array로 구현하였습니다. 테스트 내용은 위 항목과 동일한 3번의 '- ParaStack' 파트에 첨부하였습니다.		
기타			
총평/계			

* 학생 자체 평가는 점수에 반영되지 않음.

* 학생 스스로 자신의 보고서를 평가하면서, 체계적으로 프로젝트를 마무리하도록 유도하는 것이 목적임.