

탈잉 개발자와 프로그래밍 무작정 배워보기



5. 배열과 에러처리

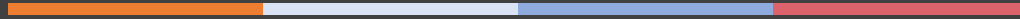
강대규

목차



1. 데이터객체와 null
2. Array 와 ArrayList
3. try - catch

1. 데이터 객체와 null



데이터 객체



DTO (Data Transfer Object)

데이터 객체



DTO (Data Transfer Object)

데이터가 포함된 객체

데이터 객체

```
class Student {  
  
    private String name;  
    private int age;  
    private String school;  
    private boolean graduated;  
  
}
```

데이터 객체 - Getter / Setter

Getter / Setter

DTO 내의 데이터를 set 혹은 get 할 수 있는 메소드

데이터 객체 - Getter / Setter

```
public class Student {  
  
    private String name;  
    private int age;  
    private String school;  
    private boolean graduated;  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    ...  
}
```


데이터 객체 - Getter / Setter 사용예시

```
Student student1 = new Student();
```

```
student1.setName("강대규");
```

```
student1.setAge(27);
```

```
student1.setSchool("한양대학교");
```

```
student1.setGraduated(false);
```

null



(사전적 의미) Null : 없음;

null



(사전적 의미) Null : 없음;

→ 어떠한 값도 가지고 있지 않다.

null



(사전적 의미) Null : 없음;

→ 어떠한 값도 가지고 있지 않다.

→ 0이나 공백과는 다른 개념

객체가 null



객체가 null 이라면?

객체가 null



객체가 null 이라면?

그 안의 method 를 실행시키거나, 변수를 가져올 때 에러가 발생함

객체가 null 인 경우



1. 객체를 선언만 하고 생성자를 통해 메모리 공간을 할당하지 않은 경우
2. null 이라고 선언한 경우

객체가 null 인 경우

```
Student student1 = new Student();
```

```
student1.setName("강대규");  
student1.setAge(27);  
student1.setSchool("한양대학교");  
student1.setGraduated(false);
```

```
Student student1;
```

```
student1.setName("강대규");  
student1.setAge(27);  
student1.setSchool("한양대학교");  
student1.setGraduated(false);
```


객체가 null 인 경우

생성자를 통해 메모리 공간을 할당해줌

```
Student student1 = new Student();
```

```
student1.setName("강대규");  
student1.setAge(27);  
student1.setSchool("한양대학교");  
student1.setGraduated(false);
```

객체 선언만 해놓은 상태 → null

```
Student student1;
```

```
student1.setName("강대규");  
student1.setAge(27);  
student1.setSchool("한양대학교");  
student1.setGraduated(false);
```

객체가 null 인 경우

생성자를 통해 메모리 공간을 할당해줌

```
Student student1 = new Student();
```

```
student1.setName("강대규");  
student1.setAge(27);  
student1.setSchool("한양대학교");  
student1.setGraduated(false);
```

객체 선언만 해놓은 상태 → null

```
Student student1;
```

```
student1.setName("강대규");  
student1.setAge(27);  
student1.setSchool("한양대학교");  
student1.setGraduated(false);
```

에러

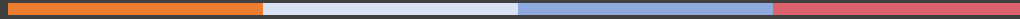
객체가 null 인 경우



```
Student student1 = null;
```

```
String helloWorld = null;
```

2. Array 와 ArrayList



배열 (Array)



같은 자료형을 가진 연속된 메모리 공간으로 이루어진 자료구조

배열 (Array)

0 -> 1 -> 2 -> 3 -> ...

같은 자료형을 가진 연속된 메모리 공간으로 이루어진 자료구조

데이터의 타입

배열 (Array)


	0번째	1번째	2번째	3번째	4번째
Class (Data Type)	Instance 1 or Value 1	Instance 2 or Value 2	Instance 3 or Value 3	Instance 4 or Value 4	Instance 5 or Value 5

배열의 선언



```
int[] intArray = new int[10];
```


배열의 선언



The diagram shows the code `int[] intArray = new int[10];` with three annotations:

- A red oval around `intArray` with the label **변수명** (Variable Name) above it.
- An orange oval around `int` with the label **데이터의 타입** (Data Type) below it.
- A blue oval around `10` with the label **배열의 크기
사이즈가 10** (Array Size
Size is 10) below it.

배열의 선언



```
Student[] students = new Student[10];
```

배열의 선언

`Student[] students = new Student[10];`

데이터의 타입

변수명

배열의 크기
사이즈가 10

배열의 초기화



```
intArray[0] = 10;  
intArray[1] = 20;  
intArray[2] = -3;  
...  
intArray[9] = 10;
```

배열의 선언 및 초기화

```
int[] intArray = {1, 2, 3, 4};
```

```
String[] stringArray = {"a", "b", "c"};
```

배열의 선언 및 초기화

`int[] intArray = {1, 2, 3, 4};` → 크기 4인 배열

`String[] stringArray = {"a", "b", "c"};` → 크기 3인 배열

→ 배열의 선언과 초기화를 동시에 해주는 방식

배열의 선언 및 초기화 예시

```
Student[] students = new Student[5];
```

```
Student student1 = new Student("강대규", 27, "한양대학교", false);
```

```
Student student2 = new Student("권영훈");
```

```
students[0] = student1;
```

```
students[2] = student2;
```

배열의 선언 및 초기화 예시

	0번째	1번째	2번째	3번째	4번째
Student	student1	null	student2	null	null

배열의 선언 및 초기화 예시

실습시간!

학생 클래스를 만들어 3명의 학생을 차례로 입력받아 배열로 저장하고

입력을 다 받았다면 한번에 출력하시오.

(학생 클래스에는 이름, 학교, 학번, 나이가 포함되어있다.)

배열의 단점

1. 배열의 사이즈를 미리 정해놔야 한다.
 - 데이터가 추가적으로 필요할때는 다시 선언을 해야한다.
2. 배열의 빈 부분은 Null 로 되어있다.
 - Null 은 에러를 유발시키기 때문에 쓰지 않도록 한다.
3. 데이터를 추가 / 삭제하는데 유동적이지 못하다.

배열의 장점



1. 속도가 빠르다.
 - 메모리의 구성이 연속되어 있기 때문에

ArrayList



내부적으로 배열을 사용하여 “리스트”를 구현한 것

ArrayList



내부적으로 배열을 사용하여 “리스트”를 구현한 것

→ Java 내의 API로 배열의 단점을 극복하여 사용할 수 있는 것

ArrayList



내부적으로 배열을 사용하여 “리스트”를 구현한 것

→ Java 내의 API로 배열의 단점을 극복하여 사용할 수 있는 것

- 미리 사이즈를 지정해야 한다.
- 추가 / 삭제에 유동적이지 않다.

ArrayList 사용 예시

```
ArrayList<Student> students = new ArrayList<Student>();  
  
Student student1 = new Student("강대규", 27, "한양대학교", false);  
Student student2 = new Student("권영훈");  
  
students.add(student1);  
students.add(student2);
```

ArrayList 사용 예시

```
ArrayList<Student> students = new ArrayList<Student>(); ← size : 0
```

```
Student student1 = new Student("강대규", 27, "한양대학교", false);  
Student student2 = new Student("권영훈");
```

```
students.add(student1); ← size : 1
```

```
students.add(student2); ← size : 2
```


배열의 선언 및 초기화 예시

실습 1

학생을 차례로 입력받아 배열로 저장하고

입력을 다 받았다면 한번에 출력하시오.

(반복문에서 1을 입력하면 학생 추가, 2를 입력하면 현재 학생 현황 출력, 0은 종료)

CRUD



CRUD

Create, Read, Update, Delete

CRUD



CRUD

Create, Read, Update, Delete

데이터를 다룰 때 기본적으로 작동하는 기능

배열의 선언 및 초기화 예시

실습 2

학생을 차례로 입력받아 배열로 저장하고

입력을 다 받았다면 한번에 출력하시오.

반복문에서 1을 입력하면 학생 추가

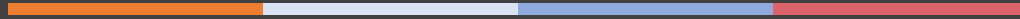
2를 입력하면 현재 학생 현황 출력

3을 입력하고 학번을 적으면 제거

4를 입력하고 학번을 적으면 학생 정보 다시 기입

0은 종료

3. try - catch



에러



프로그램에서의 에러란?

에러



프로그램에서의 에러란?

프로그램이 의도치 않게 멈추는 것

에러



프로그램에서의 에러란?

프로그램이 의도치 않게 멈추는 것

→ Exception (예외사항)

에러



프로그램에서의 에러란?

프로그램이 의도치 않게 멈추는 것

→ Exception (예외사항)

에러 - 예시

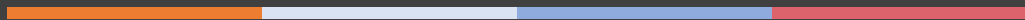
```
public static void main(String[] args) {  
    String[] a = new String[10];  
    System.out.println(a[13]);  
}
```

에러 - 예시

```
public static void main(String[] args) {  
    String[] a = new String[10];  
    System.out.println(a[13]);  
}
```

<terminated> Loop (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk-1.8.0_112.jdk/Contents/Home/bin/java
Exception in thread "main" [java.lang.ArrayIndexOutOfBoundsException: 13](#)
at Loop.main([Loop.java:7](#))

에러



프로그램의 에러를 막기 위해선?

에러



프로그램의 에러를 막기 위해선?

예외처리가 필요하다

예외처리

```
try {
```

에러발생

...

```
} catch(exception 1) {
```

```
} catch(exception 2) {
```

```
}
```

예외처리 - 예시

```
String[] a = new String[10];  
...  
try {  
    System.out.println(a[13]);  
} catch (ArrayIndexOutOfBoundsException e) {  
    System.out.println("에러발생");  
}
```

예외처리 - 실습

실습시간!

학생 클래스를 만들어 10명의 학생을 차례로 입력받아 배열로 저장하고

입력을 다 받았다면 한번에 출력하시오.

(학생 클래스에는 이름, 학교, 학번, 나이가 포함되어있다.)

(에러처리도 같이 해보세요!)