5주차

목차

문자열

변수는 큰따옴표로 나타내고

자료형은 `String` 이다.

문자열을 비교 → 비교 연산자가 아닌 메소드를 이용함

문자열 메소드

equals

`a.equals(b)`

값을 비교해서 같으면 `true` 다르면 `false` 리턴

```
String a = "java";
String b = new String("java");

System.out.println(a==b);
System.out.println(a.equals(b));
```

`equals` 메소드와 `==` 비교 연산자의 차이 → 과제

equalsIgnoreCase

`a.equalsIgnoreCase(b)`

값을 비교해서 같으면 `true` 다르면 `false` 리턴

* **대소문자를 구분하지 않음**

compareTo

`a.compareTo(b)`

값을 비교하여 정수값 리턴

숫자의 비교 같은 경우는 단순히 크다(1), 같다(0), 작다(-1) 의 관한 결과값을 리턴해주는 반면 문자열의 비교 같은 경우는같다(0), 그 외 양수/음수값 같이 결과를 반환해준다.

compareToIgnoreCase

`a.compareToIgnoreCase(b)`

대소문자를 무시하고 비교하는 메소드

charAt

`a.charAt(index)`

문자열에서 하나의 문자를 index로 지정하고, 지정한 값을 문자형(`char`)으로 리턴

toLowerCase

`a.toLowerCase()`

문자열의 전부를 소문자로 변환

toUpperCase

`a.toUpperCase()`

문자열의 전부를 대문자로 변환

substring

`a.substring(index)`

문자열의 일부 변환

trim

`a.trim()`

앞뒤 공백 제거 후 반환

contains

`a.contains(”특정문자열")`

대상 문자열에 특정 문자열이 포함되는지 확인

대/소문자 구분하고, 공백도 체크

startsWith

`a.startsWith(”특정문자열")`

특정 문자열이 시작하는 문자열, 끝나는 문자열에 포함되는지 확인

isEmpty

`a.isEmpty()`

문자열의 길이가 0이면 true 반환

```
String str1 = "Hi,";
String str2 = " Java";
String str3 = "Hi, Java";
String str4 = "Java";

System.out.println(str4.isEmpty()); // false
str4 = "";
System.out.println(str4.isEmpty()); // true
```

배열

타입이 같은 데이터를 다룰 때 배열을 통해 동일한 데이터 타입으로 집합을 쉽게 처리할 수 있음

배열의 선언과 생성

```
[데이터타입[] 변수or배열이름]

int[] arr1 = new int[5];
int arr2[] = {10, 100, 90, 50, 40};


// 배열 요소에 접근하는 방법 1

int a = arr2[2]
System.out.println(a);



// 배열 요소에 접근하는 방법 2

for(int i=0; i<arr1.length; i++) {
  arr1[i] = sc.nextInt();
  System.out.println(str1[i] + " ");
}
```

1차원 배열

```
int[] arr1 = new int[5];
int arr2[] = {10, 100, 90, 50, 40};
```

2차원 배열

```
int[][] arr3 = new int[3][5];

-> 3행 5열의 배열 생성
```

과제

- 백준 10039
- 백준 2845

추가적으로 개념 찾기

- String b = “JAVA”와 String c = new String(”JAVA”) 차이점
- equals 와 == 의 차이점 (최소 2가지)
- 다차원 배열

오늘 배운거 정리에 위 두개도 작성해서 제출

String b = “JAVA” 와 String c = new String(”JAVA”)의 차이점

문자열 String은 다른 기본 자료형과는 다르게 참조 자료형이다

String 타입의 변수를 사용하는 방법은 아래와 같이 두 가지로 나뉜다

`String b = “JAVA”`

`String c = new String(”JAVA”)`

참조 자료형의 경우에는 객체를 생성할 때 `heap` 영역에서 메모리 관리를 하게 되는데,

따라서 `stack` 영역에는 `heap` 영역 안에 있는 문자열을 가리키는 주소값인 형태로 데이터가 만들어진다

그 중에서도 String 타입의 경우, 메모리를 효율적으로 사용하기 위한 `String pool`이 존재한다

`String pool`은 String 타입의 변수를 생성할 때, 문자열 값이 같은 객체가 이미 존재한다면

객체를 새로 생성하지 않고 원래 있던 객체와 같은 래퍼런스를 가지게 하여 메모리를 효율적으로

사용할 수 있게 도와준다

결론

`String b = “JAVA”`

→ 기본적인 변수 선언 방법

`String c = new String(”JAVA”)`

→ 문자열 값이 같은 객체가 있는지 여부와 관계없이 새로운 객체 생성

예제)

```
String a = "Java";
String b = "Java";
String c = new String("Java");


// "==" 비교 연산자는 stack 영역의 값을 서로 비교한다
// 따라서 주소값이 동일한 a, b는 true
System.out.println(a == b); // true
```

```
// 새로운 객체를 생성해서 주소값이 다른 a와 c는 false가 출력된다.
System.out.println(a == c); // false
```

equals 와 == 의 차이점

위에 참조 연산자에서 설명했던 것 처럼 String 타입의 경우 `stack` 영역의 주소값과 `heap` 영역의 문자열 값으로 나누어진다.

여기서 비교연산자 “==” 의 경우 `stack` 영역의 값을 비교하기 때문에
문자열을 서로 비교할 때에는 부정확한 결과가 나올 수 있으며,
메소드 `equals()` 의 경우에는 heap 영역에 있는 문자열 값을 확실하게 비교한다.

```
// 새로운 객체를 생성했기 때문에 문자열 값은 같지만 주소값은 다름
String a = "Java";
String b = new String("Java");


// 주소값이 서로 다르기 때문에 "=="의 결과는 false
System.out.println(a == b); // false


// 문자열 값은 동일하므로 equals()의 결과는 true
System.out.println(a.equals(b)); // true
```

다차원 배열

이론적인 부분은 1,2차원 배열과 다른게 없기 때문에, 예제 하나 적겠습니다

```
int[][][] arr = new int[2][5][2]

// 직육면체모양(x축,y축,z축) 3차원 배열 생성
int[][][] arr = {
  {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}},
  {{11, 12, 13, 14, 15}, {16, 17, 18, 19, 20}}
};
```

백준 정답 확인

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
47320518	tureve842	10039	맞았습니다!!	17656 KB	204 ms	Java 11 / 수정	376 B	24초 전

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
47321111	tureve842	4 2845	맞았습니다!!	18252 KB	224 ms	Java 11 / 수정	399 B	5초 전