📖 Java Study Skill Up Team Review

객체 지향 프로그래밍

문자열 잘라내기

```
package com.chapter.ch05.example.ex01;

public class Substring {
public static void main(String[] args) {
String ssn = "111111-2222222";
String firstNum = ssn.substring(0,6);
String secondNum = ssn.substring(7);

        System.out.println(firstNum);   // 111111
        System.out.println(secondNum);  // 2222222
    }
}
```

- 특정 문자열을 다른 문자열로 대체하고 싶다면 replace() 메서드를 사용할 수 있다.

문자열 찾기

```
package com.chapter.ch05.example.ex01;

public class IndexOf {
    public static void main(String[] args) {
        String subject = "자바 프로그래밍";
        int index = subject.indexOf("프로그래밍");
        int index2 = subject.indexOf("플밍");

        System.out.println(index);  // 3
        System.out.println(index2); // -1
    }
}
```

- 문자열에서 특정 문자열의 위치를 찾고자 할 때에는 indexOf() 메서드를 사용할 수 있다.

문자열 분리

```
package com.chapter.ch05.example.ex01;

public class Split {
    public static void main(String[] args) {
        String board = "번호, 제목, 내용, 성명";
```

```
        String[] arr = board.split(", ");

        for (int i = 0; i < arr.length; i++) {
            System.out.println(i + " index value : " + arr[i]);
            // 0 index value : 번호
            // 1 index value : 제목
            // 2 index value : 내용
            // 3 index value : 성명
        }
    }
}
```

- 문자열이 구분자를 사용하여 여러 개의 문자열로 구성되어 있을 경우, 따로 분리해서 얻고 싶다면 split() 메서드를 사용할 수 있다.

배열(Array) 타입

- 변수는 하나의 값만 저장할 수 있다.
- 많은 양의 값을 다루기 위해 효율적인 방법은 배열을 사용하는 것이다.
- 배열은 같은 타 입의 값만 관리한다. (int 배열은 int 타입의 값만 관리함, String 배열은 문자열만 관리함)
- 배열의 길이는 늘리거나 줄일 수 없다. (배열은 생성과 동시에 길이가 결정됨)

[배열 변수 선언]

- 배열 변수 선언은 다음과 같은 두 가지 형태로 작성이 가능하다.

```
package com.chapter.ch05.example.ex02;

public class Array {
    // 1
    String[] stringArray = null;
    // 2
    String stringArray2[] = null;
}
```

- 관례적으로는 첫 번째 방법을 주로 사용한다.
- 배열 변수는 참조 변수이다. 배열도 객체이므로 힙 영역에 생성되고 배열 변수는 힙 영역의 배열 주소를 저장한다.
- 참조할 배열이 없다면 배열 변수도 null로 초기화 할 수 있다. 즉 그냥 null 로 초기화 할 수 있으니 해주자라는 내용

[읽기, 수정]

```
package com.chapter.ch05.example.ex02;

public class Array {
    public static void main(String[] args) {
        String[] stringArray = {"영", "일", "이", "삼"};
```

```
        System.out.println(stringArray[0]); // 읽기: 출력값 '0'

        stringArray[0] = "Zero";
        System.out.println(stringArray[0]); // 수정 후 출력값 'Zero'

        String[] stringArray2 = {"일", "영"};
        // stringArray2 = {"영", "일"};  // ❌(불가능) : 배열 변수를 미리 선언한 후 값 목록을 변수에 대입할 수 없다.
        stringArray2 = new String[] {"영", "일"}; // ⭕(가능) : 값 목록이 대입되는 시점이 다르면 new 타입[] 을 이용하면 대입할 수 있다.
    }
}
```

[메서드의 매개변수가 배열 타입일 경우]

```
package com.chapter.ch05.example.ex02;

public class MethodArray {
    void numberList (int[] numbers) {
        System.out.println(numbers);
    }
    public void main(String[] args) {
        // numberList({1, 2, 3, 4, 5});         // ❌
        numberList(new int[] {1, 2, 3, 4, 5});  // ⭕
    }
}
```

- 메서드의 매개변수가 배열 타입일 경우에도 마찬가지로 매개변수로도 이미 선언되어 있다면 new int[]를 사용해야 한다.

배열 길이

- 가변 길이: 매개변수의 개수와 상관없이 매개값을 줄 수 있는 것

```
package com.chapter.ch05.example.ex02;

public class Varargs {
    static int sum(int... values) {
        int sum = 0;
        for (int i = 0; i < values.length; i++) {
            sum = sum + values[i];
        }
        return sum;
    }
}
```

- int 타입으로 ... values로 가별길이 매개변수를 선언

```
package com.chapter.ch05.example.ex02;

public class Varargs {
    static int sum(int... values) {
        int sum = 0;
        for (int i = 0; i < values.length; i++) {
            sum = sum + values[i];
        }
        return sum;
    }

    public static void main(String[] args) {
        int result = sum(1, 2, 3, 4, 5);
        System.out.println(result);
    }
}
```

- 이후 총 5개의 숫자(1,2,3,4,5)를 매개값으로 지정하여 sum 메서드를 호출하였고 결괏값은 15로 출력되었다.

```
int result = sum(1, 14);
```

- 그리고 같은 결괏값이 출력되는지 총 2개의 숫자(1,14)를 매개값으로 다시 한번 메서드를 호출해 보았다.

```
역시나 결괏값은 15로 출력이 되었다.
즉 넘겨받는 매개값의 개수와 상관없이 무조건 더한다거나 혹은 가변으로 동작하는 메서드의 경우 가변길이 매개변수(가변인자)를 사용할 수 있겠다.
```

배열 길이

- 배열변수.length
- 배열의 length 필드는 읽기만 가능하기 때문에 값을 변경하는 것을 불가능하다.
- for 문을 사용해서 전체 배열 항목을 반복할 때 많이 사용된다.