String Class의 Instance Method 관찰

String Class의 대표적인 Instance Method

문자열의 길이 반환	public int length()	문자열의 길이를 계산해서 int형으로 반환
두 문자열의 결합	public String concat(String str)	두 문자열을 연결해서 새로운 문자열 Instace를 만들어서 반환한다
두 문자열의 비교	public int compareTo(String anotherString)	문자열이 동일하면 0, 동일하지 않으면 0이 아닌 값,
두 문자열이 같은가	public boolean equals(Object anObject)	원래는 instance의 비교용이지만 문자열도 instace이므로 비교가능, 같으면 true, 다르면 false

```
class StringMethod
     public static void main(String[] args) {
          String str1 = "Smart";
          String str2 = " and ";
          String str3 = "Simple";

          String str4 = str1.concat(str2).concat(str3);
                        // 새로운 인스턴스     //새로운 인스턴스가 생성됐으므로 이런 식의 문장 구성이 가능한 것이다!


          System.out.println(str4);
          System.out.println("문자열 길이 : "+str4.length());

          if(str1.compareTo(str3)<0) // 사전 편찬순서상 str1이 앞서면 음수, str3가 앞서면 양수
               System.out.println("str1이 앞선다");
          else
               System.out.println("str3이 앞선다");

               public boolean equals(Object anObject)
               Object Class에 미리 정의해 놓은 Method로써
               Instance간의 내용 비교를 할 수 있는데, 문자열은 Instance화 되므로 equals로 같은지 비교가능!

          if(str1.equals(str3))
```

```
            System.out.println("str1이 앞선다");
        else
            System.out.println("str3이 앞선다");
    }
}
```

자바에서의 문자열 복사

String Instance의 문자열은 변경이 불가능한 상수 인스턴스

```
String str1 = "ABC";
String str2 = "ABC";

// str1과 str2는 동일한 "ABC" Instance 참조!
// (str1 == str2)
```

문자열 복사라는 것은 동일한 instace를 하나 더 생성하는 것이다.

```
String str1 = "ABC";
String str2 = new String(str1);

// String Class에는 public String(String original) 생성자가 있다.
// str1과 str2는 서로 다른 "ABC" instance를 각각 참조한다.
// (str1 != str2)
```

참조값(메모리주소) 비교

참조변수끼리의 비교연산은 내용 비교가 아니라, **참조변수가 참조하고 있는 메모리 주소(인스턴스)가 같은지 비교**한다.

```
String str1 = "ABC";
String str2 = "ABC";

if(str1 == str2)
    System.out.println("동일한 instance를 참조하고 있습니다");
else if(str1 != str2)
    System.out.println("서로 다른 instance를 참조하고 있습니다");
```

"문자열"+"문자열"의 진실

+ 연산자는 숫자 연산에 사용되는데 어떻게 문자열을 더하는 연산에 사용될 수 있었을까?

```
Public static void main(String[] args) {

    System.out.println("Hello " + "Java");
```

"Hello".concat("Java")로 컴파일러가 문장을 바꿔버림

i. "Hello " 문장에 의해 새로운 String instance 생성
ii. Instance의 concat method를 호출하면서 "Java"의 instance를 인자로 전달
iii. 두 문자열을 더해서 새로운 "Hello Java"의 String Instamce 생성

```
    String str1 = "lemon" + "ade";

    String str2 = "lemon" + 'A';
    String str3 = "lemon" + 3;
    // 문자열 + 문자,숫자, 이 경우에는 concat Method를 호출할 수 없다..
```

String class의 valueOf() Method : 인자를 문자열 instance로 만들어 주는 Method

valueOf() Method는 Overloading되어 있으므로 숫자,문자 아무거나 인자로 전달할 수 있다.

```
    "lemon".concat(String.valueOf('A'));
    "lemon".concat(String.valueOf(3));
```

```
    String str4 = 1 + "lemon" + 2;
```

1 + "lemon"의 경우 concat Method를 사용하려는데 왼편에 문자열이 아닌 숫자 1이 있다.

```
    String str4 =     String.valueOf(1).concat("lemon").concat(String.valueOf(2));

                    // Method 호출 과정에서 너무 많은 String instance가 생성이 되서 비효율적이다.

    Str4 += '!';

    System.out.println(str1);
    System.out.println(str2);
    System.out.println(str3);
```

```
        System.out.println(str4);
    }
```