Java Web Programming 입문 05

(Java #05)

오늘의 키워드

- 변수 (revisit) – 선언 위치 별
  - 인스턴스 변수
  - 클래스 변수
  - 지역 변수
- 메소드 (method)
  - 함수
  - return
  - 매개변수 (기본형, 참조형)
  - 오버로딩
- 문자열 클래스 (String Class)

선언 위치별 변수

- 선언 위치별 변수

  [접근 제어자] [일반 제어자] 데이터타입 변수명;

```
private static int num;
```

```
class Variables {

    int iv;           // 인스턴스변수
    static int cv;    // 클래스변수(static변수, 공유변수)

    void method(){

        int iv = 0; // 지역변수

    }

}
```

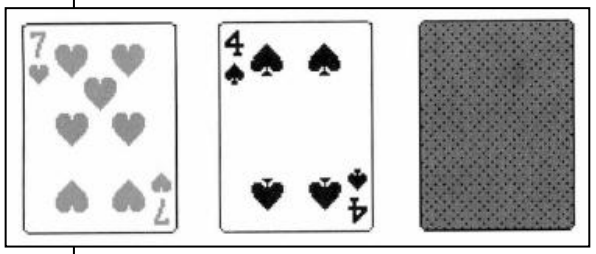
클래스영역

메서드영역

변수의 종류	선언위치	생성시기
클래스변수 (class variable)	클래스 영역	클래스가 메모리에 올라갈 때
인스턴스변수 (instance variable)		인스턴스가 생성되었을 때
지역변수 (local variable)	클래스 영역 이외의 영역 (메서드, 생성자, 초기화 블럭 내부)	변수 선언문이 수행되었을 때

선언 위치별 변수

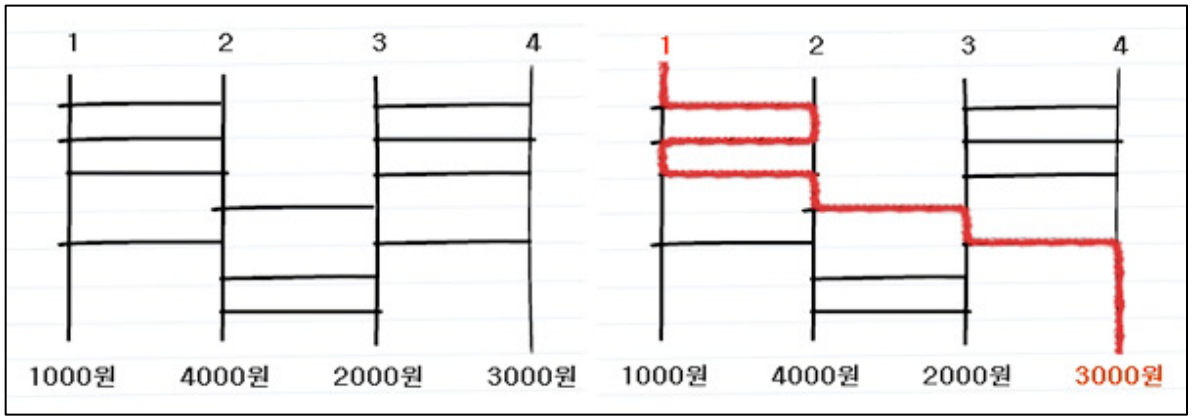
```
class CardTest{
    public static void main(String[] args){

        System.out.println("Card.width : " + Card.width);
        System.out.println("Card.Height : " + Card.Height);

        Card c1 = new Card();
        c1.kind = "Heart";
        c1.number = 7;

        Card c2 = new Card();
        c2.kind = "Spade";
        c2.number = 3;

        System.out.println( c1.kind   + " : " + c1.number + " : "
                          + c1. width + " : " + c1.height);
        System.out.println( c2.kind   + " : " + c2.number + " : "
                          + c2. width + " : " + c2.height);

        c1.width = 30;
        c1.height = 300;

        System.out.println( c1.kind   + " : " + c1.number + " : "
                          + c1. width + " : " + c1.height);
        System.out.println( c2.kind   + " : " + c2.number + " : "
                          + c2. width + " : " + c2.height);
    }
}
```

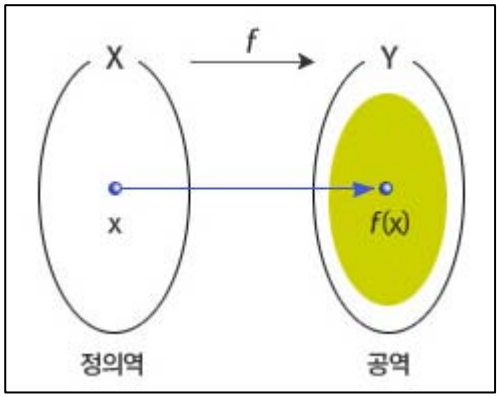


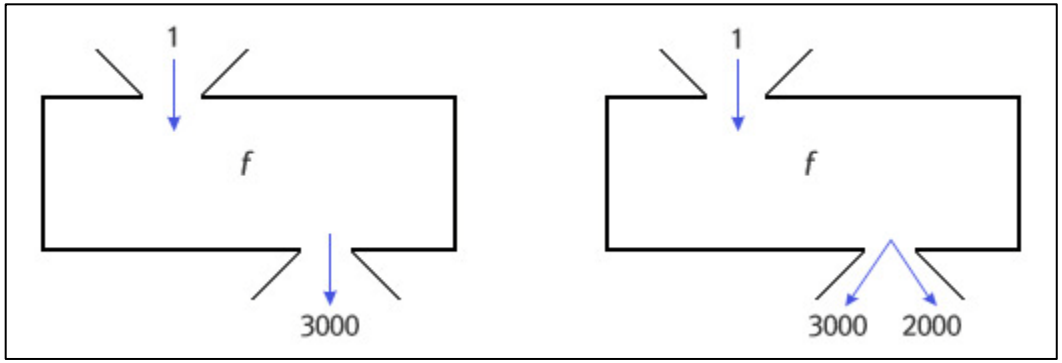
```
class Card{
    String kind;              // 카드의 무늬 - 인스턴스변수
    int nuber;                // 카드의 숫자 - 인스턴스변수
    static int width = 100;   // 카드의 폭   - 클래스 변수
    static int height = 250; // 카드의 높이 - 클래스 변수
}
```

메소드 (method)

- 함수 (Function)







메소드 (method)

- 어떤 작업/기능을 수행하기 위한 명령문의 집합
  - 특정한 값을 입력 받아(Parameter:매개변수)
    처리 후(Logic) 결과값을 반환(Return)
  - 입력 값은 없을 수도 있다.
  - 결과 반환도 없을 수 있다.
- 코드 경량화, 유지보수 용이
  - 동일한 코딩이 3번 이상 들어가면 메소드 추출을 고려

```
- 하나의 메소드는 한 가지 기능만 수행하도록 작성하는 것이 좋음
- 반복적으로 수행되어야 하는 여러 문장을 하나의 메소드로 정의해 놓으면 좋음
- 관련된 여러 문장을 하나의 메소드로 만들어 놓는 것이 좋음
```

메소드 (method)

```
리턴타입 메서드이름 (타입 변수명, 타입 변수명, ... ) {
    // 메서드 호출시 수행될 코드
}
```

```
int add (int a, int b){
    int result = a + b;
    return result;  // 호출한 메서드로 결과를 반환
}
```

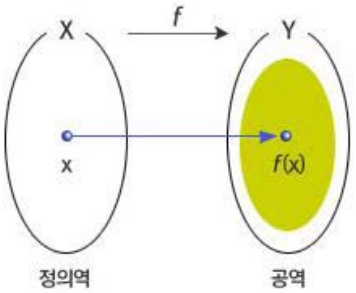
선언부

구현부

```
// 반환값이 없는 경우 리턴타입 대신 void 사용
void power() {
    poser = !power;
}
```

메소드 (method)

- modeling, extract, behavior, function
  - 값(Input Parameter, 입력 파라미터, 매개변수)을 입력 받고,
  - 처리 (programmatic logic) 후,
  - 결과값을 반환 (return)



- 입력하는 값 / 결과값 반환 있을 수도 있고 없을 수도 있다

```
[접근 제어자] [일반 제어자] 리턴타입 메소드명 ( 입력 파라미터 ) {
    // Statement
}
```

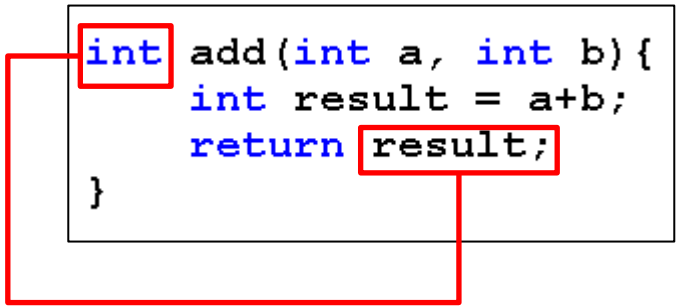
```
public int add ( int num1, num2 ){
    return num1 + num2;
}
```

```
public static void main (String[] args){
    // Statement
}
```

메소드 (method)

- 메소드(Method) 의 종료
  - { }
    - 메소드의 블록{}내에 있는 모든 문장이 수행되었을 때
  - return
    - 메소드의 블록{}내에 있는 문장 수행 중 return 문을 만났을 때

- return
  - 반환 값이 없는 경우 : return문만 사용 `return;`
  - 반환 값이 있는 경우 : return문 뒤에 반환 값 지정 `return 반환값;`



메소드 (method)

- return 사용 주의

Error

```
int max (int a, int b){

    if ( a > b ){   return a; }
}
```

```
int max (int a, int b){

    if ( a > b ){   return a;   }
    else        {   reutrn b;   }
}
```

```
int max (int a, int b){

    if ( a > b){
        return a;
    } else {
        return b;
    }
}
```

```
int max (int a, int b){

    int result = 0;
    if ( a > b){
        result = a;
    } else {
        result = b;
    }
    return result;
}
```

메소드 (method)

- 매개변수 사용

```
// 메소드에 선언된 매개변수가 없는 경우
참조변수.메소드이름()

// 메소드에 선언된 매개변수가 있는 경우
참조변수.메소드이름(값1, 값2, ...);
```

- 기본형 데이터타입 매개변수 VS 참조형 데이터 타입 매개변수
  - 기본형 매개변수
    - 변수의 값을 읽기만 할 수 있음 (Read Only)
  - 참조형 매개변수
    - 변수의 값을 읽고 변경 가능 (Read & Write)

메소드 (method)

- 기본형 데이터타입 매개변수

```
class Data {    int x;  }

class ParameterTest {
    public static void main(String[] args){
        Data d = new Data();
        d.x = 10;
        System.out.println("main() : x = " + d.x);

        change(d.x);
        System.out.println("After change(d.x)");
        System.out.println("main() : x = " + d.x);
    }

    static void change( int x ){    // 기본형 매개변수
        x = 1000;
        System.out.println("change() : x = " + x);
    }
}
```

```
main() : x = 10
change() : x = 1000
After change(d.x)
main() : x = 10
```

메소드 (method)

- 참조형 데이터타입 매개변수 (배열)

```
class Data { int x; }
```

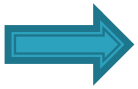
```
class ParameterTeset3 {
    public static void main(String[] args){
        int[] x = {10};
        System.out.println("main() : x = " + x[0]);

        change(x);
        System.out.println("After change(x)");
        System.out.println("main() : x = " + x[0]);
    }

    static void change(int[] x){
        x[0] = 1000;
        System.out.println("change() : x= " + x[0]);
    }
}
```

```
main() : x = 10
change() : x = 1000
After change(x)
main() : x = 1000
```

메소드 (method)

- 참조형 데이터타입 매개변수

```
class ParameterTest2 {
    public static void main(String[] args){

        Data d = new Data();
        d.x = 10;
        System.out.println("main() : x = " + d.x);

        change(d);
        System.out.println("After change(d)");
        System.out.println("main() : x = " + d.x);
    }

    static void change (Data d) { // 참조형 매개변수
        d.x = 1000;
        System.out.println("change() : x = " + d.x);
    }
}
```



```
main() : x = 10
change() : x = 1000
After change(d)
main() : x = 1000
```

메소드 (method)

- 메소드 오버로딩 (Method Overloading)
  - 한 클래스 내 동일한 이름의 메소드 여러개
  - 입력 매개변수 (input parameter) 의
    - 개수가 다르다.
    - 데이터 타입이 다르다
  - return 데이터 타입의 영향을 받지 않는다.

```
void println(                )
void println( boolean    x )
void println( char       x )
void println( char[]     x )
void println( double     x )
void println( float      x )
void println( int        x )
void println( long       x )
void println( Object     x )
void println( String     x )
```

```
int add(int a, int b) { return a+b; }

int add(int x, int y) { return x+y; }
```

```
int  add(int a, int b) { return a+b;          }

long add(int a, int b) { return (long)(a + b); }
```

```
long add(int a, long b) { return a+b; }

long add(long a, int b) { return a+b; }
```

```
int add(int a, int b) {return a+b; }

long add(long a, long b) {return a+b;}

int add(int[] a){
    int result = 0;
    for( int i = 0 ; i < a.length ; i++ ){
        result += a[i];
    }
    return result;
}
```

문자열 클래스 (String Class)

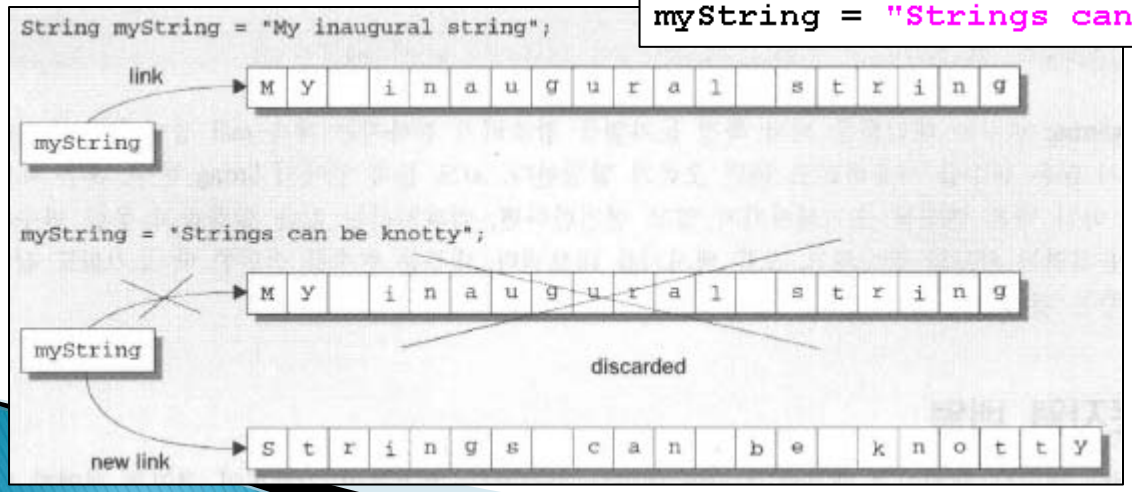
- String 클래스

```
public final class String implements java.io.Serializable, Comparable {
    /** The value is used for character storage. */
    private char[] value;
    ...
```

```
String a = "a";
String b = "b";
String ab = a + b;
```

□ 문자열
◇ 문자열은 문자상수의 집합이며, 이중 따옴표를 이용해서 문자열을 만든다.
◇ 자바에서 문자열은 String 클래스의 객체이다.

```
String myString = "My inaugural string";

myString = "Strings can be knotty";
```



문자열 클래스 (String Class)

- String 선언
  - 문자열의 생성

```
String str1 = "Hello World! ";
```

  - 문자열의 결합

```
String ntr3 = "Hello" + 100 +  "World!" + "새로운 문자열";
```

  - 문자열 내 Escape 문자 사용

```
String rmyString = "c:\\Program Files\\Java";
```

  - 문자열 생성 방법

```
String str1 = "Hello World!";
String str2 = "Hello World!";
```

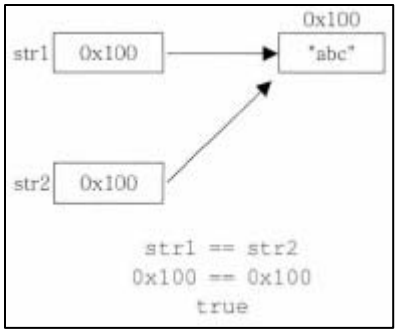
```
String ntr1 = new String("Hello World!");
String ntr2 = new String("Hello World!");
```

문자열 클래스 (String Class)

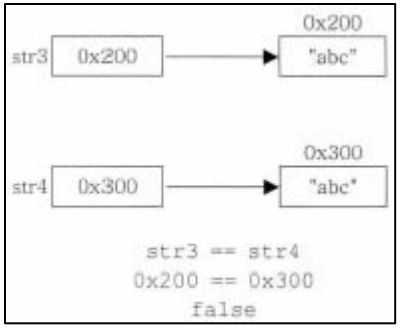
- String 선언

```
/** 문자열 생성과 결합 **/
public class StringBasicMain{

    public static void main(String[] args){

        //문자열을 생성하는 방법 I
        String str1 = "Hello World!";
        String str2 = "Hello World!";
        System.out.println("str1==str2 : " + (str1==str2));

        //문자열 생성하는 방법 II
        String ntr1 = new String("Hello World!");
        String ntr2 = new String("Hello World!");
        System.out.println("ntr1==ntr2 : " + (ntr1==ntr2));

        //문자열 결합하기
        String ntr3 = "Hello" + 100 +  "World!" + "새로운 문자열";
        System.out.println(ntr3);

        //문자열 내에 에스케이프(Escape) 문자의 사용
        String myString = "c:\\Program Files\\Java";
        System.out.println("에스케이프문자의 사용 : " + myString);
    }
}
```

```
String str1 = "Hello World!";
String str2 = "Hello World!";
str1 == str2 //결과는 true
```



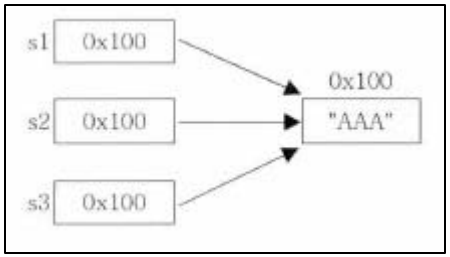

```
String ntr1 = new String("Hello World!");
String ntr2 = new String("Hello World!");
ntr1 == ntr2 //결과는 false
```



문자열 클래스 (String Class)

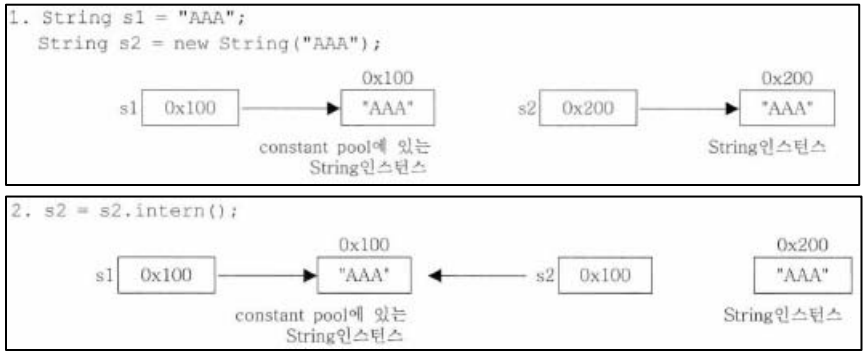
- String 선언

```
class StringEx2 {
    public static void main(String args[]){
        String s1 = "AAA";
        String s2 = "AAA";
        String s3 = "AAA";
        String s4 = "BBB";
    }
}
```




```
class StringEx3 {
    public static void main(String args[]){
        String s1 = "AAA";
        String s2 = new String("AAA");

        if (s1 == s2){
            System.out.println("s1==s2 ? true");
        } else {
            System.out.println("s1==s2 ? false");
        }

        s2 = s2.intern();
        System.out.println("s2에 intern()을 호출한 후");

        if (s1 == s2){
            System.out.println("s1==s2 ? true");
        } else {
            System.out.println("s1==s2 ? false");
        }
    }
}
```

```
s1==s2 ? false
s2에 intern()을 호출한 후
s1==s2 ? true
```



문자열 클래스 (String Class)

- String 클래스의 생성자와 메소드 (책/검색)

메서드 / 설명	예 제	결 과
String(String s) 주어진 문자열(s)을 갖는 String인스턴스를 생성한다.	String s = new String("Hello");	s = "Hello"
String(char[] value) 주어진 문자열(value)을 갖는 String인스턴스를 생성한다.	char[] c = {'H','e','l','l','o'} String s = new String(c);	s = "Hello"
String(StringBuffer buf) StringBuffer인스턴스가 갖고 있는 문자열과 같은 내용의 String인스턴스를 생성한다.	StringBuffer sb = new StringBuffer("Hello"); String s = new String(sb);	s = "Hello"

char charAt(int index)
지정된 위치(index)에 있는 문자를 알려준다. (index는 0부터 시작)
	String s = "Hello";
	String n = "0123456";
	char c = s.charAt(1);
	char c2 = n.charAt(1);
		c = 'e'
		c2 = '1'

String concat(String str) 문자열(str)을 뒤에 덧붙인다.	String s = "Hello"; String s2 = s.concat(" World");	s2 = "Hello World"
boolean endsWith(String suffix) 지정된 문자열(suffix)로 끝나는지 검사한다.	String file = "Hello.txt"; boolean b =file.endsWith("txt");	b = true

boolean equals(Object obj)
매개변수로 받은 문자열(obj)과 String인스턴스의 문자열을 비교한다. obj가 String이 아니 거나 문자열이 다르면 false를 반환한다. (대소문자 구분한다)
	String s = "Hello";
	boolean b = s.equals("Hello");
	boolean b2 = s.equals("hello");
		b = true
		b2 = false

boolean equalsIgnoreCase(String str)
문자열과 String인스턴스의 문자열을 대소문자 구분없이 비교한다.
	String s = "Hello";
	boolean b =
	s.equalsIgnoreCase("HELLO");
	boolean b2 =
	s.equalsIgnoreCase("heLLo");
		b = true
		b2 = true

int indexOf(int ch)
주어진 문자(ch)가 문자열에 존재하는지 확인하여 위치(index)를 알려준다. 못 찾으면 -1을 반환한다.(index는 0부터 시작)
	String s = "Hello";
	int idx1 = s.indexOf('o');
	int idx2 = s.indexOf('k');
		idx1 = 4
		idx2 = -1

int indexOf(String str) 주어진 문자열이 존재하는지 확인하여 그 위치(index)를 알려준다. 없으면 -1을 반환한다.(index는 0부터 시작)	String s = "ABCDEFG"; int idx = s.indexOf("CD");	idx = 2

String intern()
문자열을 constant pool에 등록한다. 이미 constant pool에 같은 내용의 문자열이 있을 경우 그 문자열의 주소값을 반환한다.
	String s = new String("abc");
	String s2 = new String("abc");
	boolean b = (s==s2);
	boolean b2 = s.equals(s2);
	boolean b3 =
	(s.intern()==s2.intern());
		b = false
		b2 = true
		b3 = true

int lastIndexOf(int ch)
지정된 문자 또는 문자코드를 문자열의 오른쪽 끝에서부터 찾아서 위치(index)를 알려준다. 못 찾으면 -1을 반환한다.
	String s = "java.lang.Object";
	int idx1 = s.lastIndexOf('.');
	int idx2 = s.indexOf('.');
		idx1 = 9
		idx2 = 4

int lastIndexOf(String str)
지정된 문자열을 인스턴스의 문자열 끝에서부터 찾아서 위치(index)를 알려준다. 못 찾으면 -1을 반환한다.
	String s = "java.lang.java";
	int idx1 =s.lastIndexOf("java");
	int idx2 = s.indexOf("java");
		idx1 = 10
		idx2 = 0

int length() 문자열의 길이를 알려준다.	String s = "Hello"; int length = s.length();	length = 5
String replace(char old, char nw) 문자열 중의 문자(old)를 새로운 문자(nw)로 바꾼 문자열을 반환한다.	String s = "Hello"; String s1 = s.replace('H', 'C');	s1 = "Cello"
String replaceAll(String regex, String replacement) 문자열 중에서 지정된 문자열(regex)과 일치하는 것을 새로운 문자열(replacement)로 모두 변경한다.	String ab = "AABBAABB"; String r = ab.replaceAll("BB","bb");	r = "AAbbAAbb"
String replaceFirst(String regex, String replacement) 문자열 중에서 지정된 문자열(regex)과 일치하는 것 중, 첫째 것만 새로운 문자열(replacement)로 변경한다.	String ab = "AABBAABB"; String r =	
String[] split(String regex) 문자열을 지정된 분리자(regex)로 나누어 문자열 배열에 담아 반환한다.	String anima String[] arr	
String[] split(String regex, int limit) 문자열을 지정된 분리자(regex)로 나누어 문자열배열에 담아 반환한다. 단, 문자열 전체를 지정된 수(limit)로 자른다.	String anima String[] arr	

boolean startsWith(String prefix)
주어진 문자열(prefix)로 시작하는지 검사한다.
	String s = "java.lang.Object";
	boolean b =s.startsWith("java");
	boolean b2=s.startsWith("lang");
		b = true
		b2 = false

String substring(int begin) **String substring(int begin, int end)** 주어진 시작위치(begin)부터 끝 위치(end) 범위에 포함된 문자열을 얻는다. 이 때, 시작위치의 문자는 범위에 포함되지만, 끝 위치의 문자는 포함되지 않는다. (begin ≤ x < end)	String s = "java.lang.Object"; String c = s.substring(10); String p = s.substring(5,9);	c = "Object" p = "lang"
String toLowerCase() String인스턴스에 저장되어있는 모든 문자열을 소문자로 변환하여 반환한다.	String s = "Hello"; String s1 = s.toLowerCase();	s1 = "hello"
String toString() String인스턴스에 저장되어 있는 문자열을 반환한다.	String s = "Hello"; String s1 = s.toString();	s1 = "Hello"
String toUpperCase() String인스턴스에 저장되어있는 모든 문자열을 대문자로 변환하여 반환한다.	String s = "Hello"; String s1 = s.toUpperCase();	s1 = "HELLO"

String trim() 문자열의 왼쪽 끝과 오른쪽 끝에 있는 공백을 없앤 결과를 반환한다. 이 때 문자열 중간에 있는 공백은 제거되지 않는다.	String s = " Hello World "; String s1 = s.trim();	s1 = "Hello World"
static String valueOf(boolean b) **static String valueOf(char c)** **static String valueOf(int i)** **static String valueOf(long l)** **static String valueOf(float f)** **static String valueOf(double d)** **static String valueOf(Object o)** 지정된 값을 문자열로 변환하여 반환한다. 참조변수의 경우, toString()을 호출한 결과를 반환한다.	String b = String.valueOf(true); String c = String.valueOf('a'); String i = String.valueOf(100); String l = String.valueOf(100L); String f = String.valueOf(10f); String d = String.valueOf(10.0); java.util.Date dd = new java.util.Date(); String date = String.valueOf(dd);	b = "true" c = "a" i = "100" l = "100" f = "10.0" d = "10.0" date = "Sun Jan 27 21:26:29 KST 2008"

문자열 클래스 (String Class)

- String 클래스의 메소드

```
/**
문자열 메서드의 사용
**/
import static java.lang.System.out;
public class StringMethodMain{
    public static void main(String[] args){
        String str1 = new String("www.jabook.org");
        String str2 = new String("www.jabook.org");

        System.out.println("str1의 길이:" + str1.length());
        System.out.println("str2의 길이:" + str2.length());
        System.out.println("str1.equals(str2):" + str1.equals(str2));
        System.out.println("str1.compareTo(str2):" + str1.compareTo(str2));

        out.println("str1.concat(str2):" + str1.concat(str2));
        out.println("str1+str2:" + str1+str2);
        out.println("str1.indexOf(\"jabook\"):" + str1.indexOf("jabook"));
        out.println("str1.lastIndexOf(\"o\"):" + str1.lastIndexOf("o"));

        out.println("str1.substring(4,10):" + str1.substring(4,10));
        out.println("str1.replace(\"o\", \"t\"):" + str1.replace("o", "t"));
    }
}
```

```
str1의 길이:14
str2의 길이:14
str1.equals(str2):true
str1.compareTo(str2):0
str1.concat(str2):www.jabook.orgwww.jabook.org
str1+str2:www.jabook.orgwww.jabook.org
str1.indexOf("jabook"):4
str1.lastIndexOf("o"):11
str1.substring(4,10):jabook
str1.replace("o", "t"):www.jabttk.trg
```

문자열 클래스 (String Class)

- String 클래스의 메소드

문자열 생성

```
String str1 = new String("www.jabook.org");
String str2 = new String("www.jabook.org");
```

문자열의 비교

```
System.out.println("str1.equals(str2):" + str1.equals(str2));
System.out.println("str1.compareTo(str2):" + str1.compareTo(str2));
```

문자열의 길이 구하기

```
System.out.println("str1의 길이:" + str1.length());
```

인덱스를 이용한 문자열 검색

```
str1.indexOf("jabook", 3)
//인덱스가 3인 위치부터 "jabook"이라는 단어를 검색하라
str1.lastIndexOf("o", 10)
//인덱스가 10인 위치부터 꺼꾸로 "o"라는 단어를 검색하라
```

문자열의 결합

```
out.println("str1.concat(str2):" + str1.concat(str2));
out.println("str1+str2:" + str1+str2);
```

문자열 추출

```
out.println("str1.substring(4,10):" + str1.substring(4,10));
```

문자열 검색

```
out.println("str1.indexOf(\"jabook\"):" + str1.indexOf("jabook"));
out.println("str1.indexOf(\"jabook\"):" + str1.lastIndexOf("o"));
```

문자열 교체(자바 5.0)

```
out.println("str1.replace(\"o\", \"t\"):" + str1.replace("o", "t"));
```

문자열 클래스 (String Class)

- StringBuffer 클래스
  - String 인스턴스는 한번 생성되면
    - 가지고 있는 값(문자열)을 읽기만 가능 (변경이 불가)
  - StringBuffer 클래스는
    - 문자열의 추가, 삭제, 수정, 검색 등의 기능을 가지고
    - 메모리 상에서 직접 문자열을 편집할 수 있는 클래스

String 클래스는 수정이 불가능한(Immutable) 클래스이다.

```
1  String str1 = "www.";
2  String str2 = "jabook.";
3  String str3 = "org";
4  String str4 = str1 + str2 + str3; //새로운 문자열을 만들어서 리턴한다.
```

StringBuffer 클래스는 수정이 가능한 클래스이다.

```
1  StringBuffer sb = new StringBuffer();
2  sb.append("www."); //문자열 추가
3  sb.append("jabook.");
4  sb.append("org");
5  sb.replace(11, 14, "net"); //문자열 교체
6  String str = sb.toString(); //문자열 얻어내기
```

```
www.jabook.org
www.jabook.net
www.jabook.net
```

```
/**
StringBuffer 클래스의 사용
**/
public class StringBufferMain{
    public static void main(String[] args){
        //1. String 클래스는 수정이 불가능한(Immutable) 클래스이다.
        String str1 = "www.";
        String str2 = "jabook.";
        String str3 = "org";
        String str4 = str1 + str2 + str3;
        System.out.println(str4);

        String str5 = str4.replace("org", "net");
        System.out.println(str5);

        //2. StringBuffer 클래스는 수정이 가능한 클래스이다.
        StringBuffer sb = new StringBuffer();
        sb.append("www.");
        sb.append("jabook.");
        sb.append("org");
        sb.replace(11, 14, "net");
        String str = sb.toString();
        System.out.println(str);
    }
}
```

문자열 클래스 (String Class)

- StringBuffer 클래스의 메소드 (책/검색)

메서드 / 설명	예 제 / 결 과
StringBuffer()	StringBuffer sb = new StringBuffer();
16문자를 담을 수 있는 버퍼를 가진 StringBuffer 인스턴스를 생성한다.	sb = ""
StringBuffer(int length)	StringBuffer sb = new StringBuffer(10);
지정된 개수의 문자를 담을 수 있는 버퍼를 가진 StringBuffer인스턴스를 생성한다.	sb = ""
StringBuffer(String str)	StringBuffer sb = new StringBuffer("Hi");
지정된 문자열 값(str)을 갖는 StringBuffer 인스턴스를 생성한다.	sb = "Hi"

StringBuffer append(boolean b)
StringBuffer append(char c)
StringBuffer append(char[] str)
StringBuffer append(double d)
StringBuffer append(float f)
StringBuffer append(int i)
StringBuffer append(long l)
StringBuffer append(Object obj)
StringBuffer append(String str)
	StringBuffer sb = new StringBuffer("abc");
	StringBuffer sb2 = sb.append(true);
	sb.append('d').append(10.0f);
	StringBuffer sb3 =
	sb.append("ABC").append(123);

매개변수로 입력된 값을 문자열로 변환하여 StringBuffer인스턴스가 저장하고 있는 문자열의 뒤에 덧붙인다.	sb = sb2 = sb3 =
int capacity()	String sb.appe int buf int str
StringBuffer인스턴스의 버퍼크기를 알려준다. length()는 버퍼에 담긴 문자열의 크기를 알려준다.	buffers strings
char charAt(int index)	String char c
지정된 위치(index)에 있는 문자를 반환한다.	c='c'
StringBuffer delete(int start, int end)	String String String
시작위치(start)부터 끝 위치(end) 사이에 있는 문자를 제거한다. 단, 끝 위치의 문자는 제외.	sb = "0126" sb2 = "0126"
StringBuffer deleteCharAt(int index)	StringBuffer sb = new StringBuffer("0123456"); sb.deleteCharAt
지정된 위치(index)의 문자를 제거한다.	sb = "012456"
StringBuffer insert(int pos, boolean b) **StringBuffer insert(int pos, char c)** **StringBuffer insert(int pos, char[] str)** **StringBuffer insert(int pos, double d)** **StringBuffer insert(int pos, float f)** **StringBuffer insert(int pos, int i)** **StringBuffer insert(int pos, long l)** **StringBuffer insert(int pos, Object obj)** **StringBuffer insert(int pos, String str)**	StringBuffer sb StringBuffer("0 sb.insert(4,'.
두 번째 매개변수로 받은 값을 문자열로 변환하여 지정된 위치(pos)에 추가한다. pos는 0부터 시작한다.	sb = "0123.456"
int length()	StringBuffer sb = new StringBuffer("0123456"); int length = sb.length();
StringBuffer인스턴스에 저장되어 있는 문자열의 길이를 반환한다.	length = 7
StringBuffer replace(int start, int end, String str)	StringBuffer sb = new StringBuffer("0123456"); sb.replace(3, 6, "AB");
지정된 범위(start~end)의 문자들을 주어진 문자열로 바꾼다. end위치의 문자는 범위에 포함 되지 않음. (start ≤ x < end)	sb = "012AB6" "345"를 "AB"로 바꿨다.

StringBuffer reverse()	StringBuffer sb =
	new StringBuffer("0123456");
	sb.reverse();

StringBuffer인스턴스에 저장되어 있는 문자열의 순서를 거꾸로 나열한다.	sb = "6543210"
void setCharAt(int index, char ch)	StringBuffer sb = new StringBuffer("0123456"); sb.setCharAt(5, 'o');
지정된 위치의 문자를 주어진 문자(ch)로 바꾼다.	sb = "01234o6"

void setLength(int newLength)	StringBuffer sb = new
	StringBuffer("0123456");
	sb.setLength(5);
	StringBuffer sb2 = new
	StringBuffer("0123456");
	sb2.setLength(10);
	String str = sb2.toString().trim();

지정된 크기로 문자열의 길이를 변경한다. 크기를 늘리는 경우에 나머지 빈 공간을 널문자 '₩u0000'로 채운다.	sb = "01234" sb2 = "0123456 " str = "0123456"
String toString()	StringBuffer sb = new StringBuffer("0123456"); String str = sb.toString();
StringBuffer인스턴스의 문자열을 String으로 반환한다.	str = "0123456"
String substring(int start) **String substring(int start, int end)**	StringBuffer sb = new StringBuffer("0123456"); String str = sb.substring(3); String str2 = sb.substring(3, 5);
지정된 범위 내의 문자열을 String으로 뽑아서 반환한다. 시작위치(start)만 지정하면 시작위치부터 문자열 끝까지 뽑아서 반환한다.	str = "3456" str2 = "34"

오늘 숙제

- 구구단 메소드를 제작
  - 정수를 입력 받아 해당 정수를 1~9까지 곱한 결과가 저장된 1차원 배열을 return 하는 메소드
    - 배열 제작 시 for 반복문을 사용하는 메소드1
    - 배열 제작 시 while 반복문을 사용하는 메소드2
  - 1차원 배열을 입력 받아, 짝수 인덱스를 가진 값을 역순으로 출력하는 메소드3
  - [실행]: 2~9까지 구구단을 짝수단일 경우 메소드1 사용, 홀수단일 경우 메소드2 사용하여, 반환받은 1차원 배열을 메소드3을 통해 출력
  - 메소드1을 오버로딩하는 메소드4 : while 반복문을 사용
  - 메소드2를 오버로딩하는 메소드5 : for 반목문을 사용
  - 메소드3을 오버로딩하는 메소드6 : 홀수 인덱스를 가진 값을 정순으로 출력하되, 마지막 인덱스는 출력하지 않는 메소드
  - [실행]: 2~9까지 구구단을 짝수단일 경우 메소드4 사용, 홀수단일 경우 메소드5 사용하여, 반환받은 1차원 배열을 메소드6을 통해 출력