06장. 배열

1. 1 차원 배열

먼저 배열의 사용법에 대해 간단하게 확인한다. 예를 들어 5 명의 성적을 변수에 저장하는 경우를 생각해 보자

```
int jumsu01, jumsu02, jumsu03, jumsu04, jumsu05;

jumsu01= 85;
jumsu02= 78;
jumsu03= 92;
jumsu04= 62;
jumsu05= 69;
```

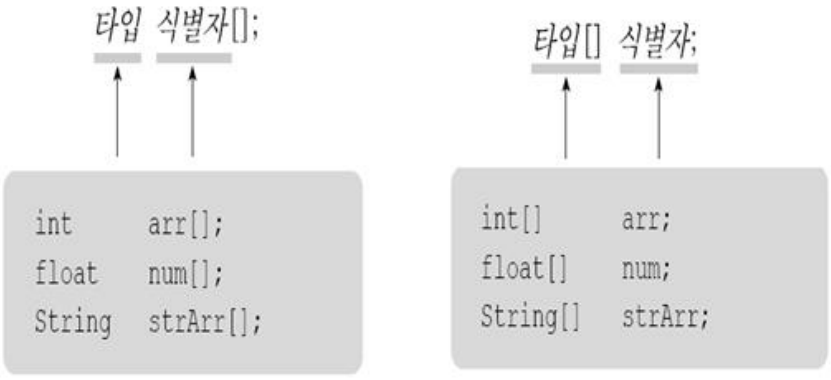
변수 하나에 하나의 값만 저장할 수 없기 때문에 5 명의 성적을 저장하려면 5 개의 변수가 필요하다. 만일 50 명, 500 명이 있다면 변수를 해당 인원수에 맞게 선언해야 하는 단점이 있다. 이를 해결하기 위해 만든 것이 바로 배열이다.

배열이란? 같은 이름과 같은 데이터 타입을 갖는 연속적인 메모리의 집합이다. 같은 데이터 타입을 갖는 여러 개의 데이터를 저장하는 것을 목적으로 한다. 원소의 수의 변화에 따라 동적으로 배열의 크기가 변화하지 못하는 정적의 타입이다. 배열의 크기는 변화시킬 수 있으나 이것은 어떤 명령문을 사용해야 가능한 것이고 스스로 크기가 커지지는 않는다

배열을 사용하는 경우 다음과 같이 선언한다.

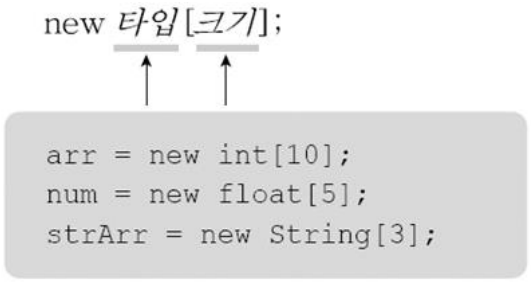
1. 먼저 배열의 개별 요소에 담을 수 있는 데이터 형과 배열을 참조하는 변수 이름을 지정한다.

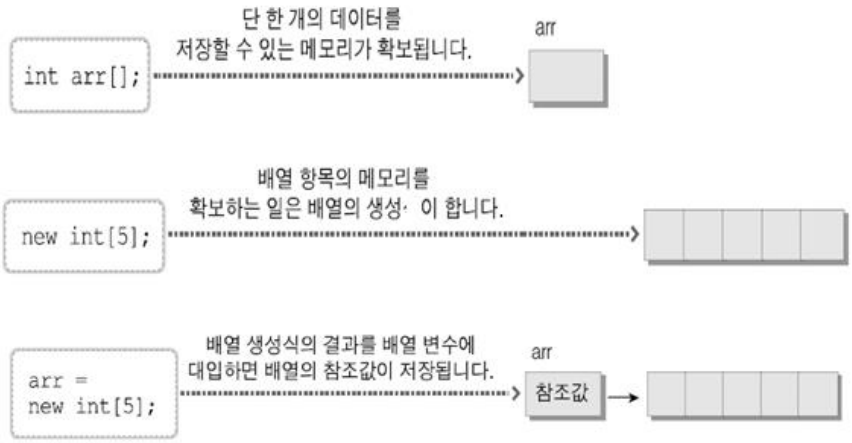
data type [ ] 배열 이름; 또는 data type 배열 이름 [];



2. 배열에 지정된 개수만큼만 요소를 확보한다. 데이터 유형에 따라 사용하는 메모리의 크기가 변화하기 때문에, 여기에서도 데이터 유형 이름을 지정한다. 배열은 객체 형이며, 선언은 요소의 수만큼 메모리에 데이터 형의 크기만큼 확보한 다음 데이터 형의 초기값으로 초기화된다. 선언과 동시에 배열의 인스턴스를 생성하려면 다음과 같이합니다.

data type [ ] 배열 명 = new data type [요소 수];





일반 데이터 형의 변수의 경우 명시 적으로 값을 할당하지 않으면 사용할 수 없다. 디폴트 값이 자동으로 할당 되거나 하지 않는다. 배열은 객체의 하나의 형태로 객체를 참조하는 참조 형식 변수의 경우 명시 적으로 값을 할당하지 않아도 디폴트 값이 미리 정해져 있으며 자동으로 할당된다. 참조 형 변수의 디폴트 값 (기본값)은 `null` 이다. 아무것도 참조하지 않는 것을 명시합니다. 다음은 각 데이터 형의 기본값을 표시한 내용이다.

byte	0
short	0
int	0
long	0L
float	0.0F
double	0.0
char	'\ u0000'
boolean	false
참조 형	null

또한 선언하는 경우에만 다음과 같이하여 초기 값을 할당 할 수 있다. 자동으로 배열 인스턴스가 만들어 값이 할당된다.

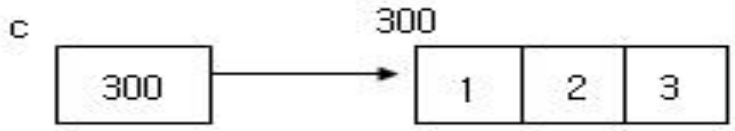
data type [ ] 배열 이름 = {값 1, 값 2, 값 3, ..., n 값};

.

(예) int 형 배열에서 배열 이름을 c 로, 1, 2, 3 의 3 개의 값으로 초기화 할 때

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

배열은 선언과 동시에 주소가 생성되어 메모리 할당을 요소의 크기만큼 이룬다.
메모리 확보한 시작 주소가 배열 변수에 대입되어 요소로 값을 리턴 받는 참조 구조를 이룬다.



배열의 각 요소는 다음과 같이 지정한다.

배열 이름 [첨자];

값의 대입은 일반 변수와 같은 것이다. 그러나 지정하는 인덱스 (요소 번호)는 0 부터 시작한다. 따라서, 배열을 new 할 때 지정한 요소의 개수를 N 개라고하면 인덱스에서 지정 가능한 수치는 0 에서 N-1 이다.

다음과 같은 코드를 살펴 본다.

```
int [] c={1,3,5,7,9,10,11,13,15,17};

// 배열 값의 값  취득
int x = c [6]; ----------→ 11 이 대입된다.
```

첨자	c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]	c[8]	c[9]
값	1	3	5	7	9	10	11	13	15	17

배열의 선언은 효율적으로 다음과 같은 형태로 정의되어 선언될 수 있다.

```
int arr[] =
    new int[10];
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;
arr[4] = 50;
arr[5] = 60;
arr[6] = 70;
arr[7] = 80;
arr[8] = 90;
arr[9] = 100;
```

[상수의 크기에 의해 메모리 확보]

```
int arr[] = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };
```

[동적 할당 시 상수의 크기에 의해 메모리 확보]

```
int arr[] = new int[] { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };
```

**[배열 변수를 선언한 후
동적 할당 시 상수의 크기에 의해 메모리 확보]**

```
int arr[];
arr = new int[] { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };
```

배열에서 확보 된 요소의 수를 조사하고 싶은 경우가 있다. 요소 수는 배열의 길이라고도 하지만 다음의 형식으로 취득 할 수 있다.

```
배열 변수 이름 .length
```

조사하고 싶은 배열 변수 뒤에 점 (.) + length 라고 기술하여 배열의 길이를 얻을 수 있다. 실제로는 다음과 같이 사용한다.

```
int n [] = {18, 29, 36, 12};
System.out.println (n.length);
```

배열에는 4 가지 요소가 확보되어 있기 때문에 화면에 4 가 출력된다.

배열의 길이는 배열과 반복을 함께 사용하는 경우에 자주 사용되며 배열의 모든 요소를 순서대로 처리하고 싶을 때, 요소의 수를 가져 반복하는 횟수를 결정한다.

다음 구문을 살펴 보자

```
int n [] = {18, 29, 36, 12};
for (int i = 0; i <n.length; i + +) {
  System.out.println (n [i]);
}
```

조건식 부분에서 "배열 변수 .length"을 사용 반복 횟수를 결정하고 있다.배열의 요소 수를 확정하고 있으면 수치를 직접 작성하는 것이 효율이 좋지만 배열 변수 .length 로두면 나중에 배열 요소의 수가 변경된 경우에도 수정할 필요가 없다. 또한 수치가 기술되어 있으면 왜이 숫자인지 나중에 모르게 될 가능성도 있지만 숫자 대신 배열 변수 .length 이 기술되어 있으면 배열의 길이만 처리하고 싶었던 것으로 분명히 알 수 있다.

5명의 점수를 대입하여 출력하는 구문을 배열로 작성해 보자.

```
public class Array01 {
   public static void main(String[] args) {

      int jumsu [] = {85, 78, 92, 62, 69};
      // 배열의 요소의 개수는 length라는 속성으로 구현할 수 있다.
      for (int i = 0; i <jumsu.length; i++) {
        System.out.println (jumsu[i]);
      }
   }

}
```

[실행결과]

85

78

92

62

69

2. 가변배열

자바에서는 2차원 이상의 배열에 대해서 "배열의 배열"의 형태로 처리한다는 사실을 이용하여 보다 자유로운 형태의 배열을 구성할 수 있다. 이를 가변 배열이라고 한다.

다차원 배열의 선언은 다음과 같이 사용한다.

```
data type  [ ] [ ] ... 배열 이름;
```

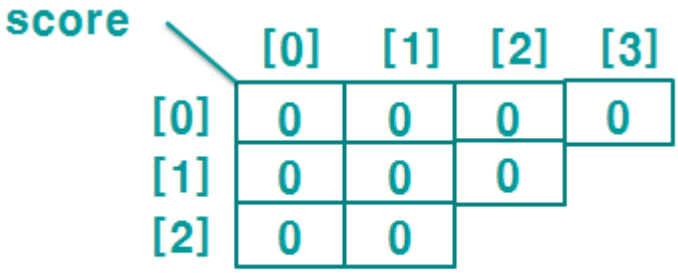
또한 초기화를 할 경우는 다음과 같이 사용한다.

```
data type [] [] ... 배열 이름 = {{값 11 값 12, ...}, {값 21 값 22, ...} ...};
```

자바에서는 2차원 이상의 배열에 대해서 "배열의 배열"의 형태로 처리한다는 사실을 이용하면 보다 자유로운 형태의 배열을 구성할 수 있다.

2차원 배열을 생성하면 다음과 같이 각 열마다 다른 크기의 배열이 생성하는 것이 가능하다

```
int[][] score = new int[3][];
score[0] = new int[4];
score[1] = new int[3];
score[2] = new int[2];
```



다음은 2차원 배열의 객체를 이용한 프로그램이다.

```
public class Array02 {

   public static void main(String[] args) {
      int intary[][] = { { 11, 12, 13 },
                         { 21, 22, 23 },
                         { 31, 32, 33 } };
      int i, j;
      for (i = 0; i < 3; i++) {
         for (j = 0; j < 3; j++) {
            System.out.println("[" + i + "" + j + "] =" + intary[i][j]);
         }
      }
   }
}
```

[실행결과]

[00] =11

[01] =12

[02] =13

[10] =21

[11] =22

[12] =23

[20] =31

[21] =32

[22] =33

다차원 배열의 경우에도 요소의 크기를 리턴 받을 수 있다.

```
int num [][] = {{1, 3, 5}, {2, 4, 6}};

System.out.println (num.length);
```

다차원 배열의 요소의 수 이므로 이 경우 "2"가 출력된다. 만약에 요소로 할당 된 각 배열의 요소 수를 가져오고 싶은 경우에는 다음과 같이 요소에 대해 ".length"으로 리턴받는다.

```
배열 변수 이름 [인덱스].length
```

구체적으로는 다음과 같이 사용한다.

```
int num [] [] = {{1, 3, 5}, {2, 4, 6}};

System.out.println (num[0] .length);→1,3, 5 요소의  개수를 리턴
System.out.println (num[1] .length); -→ 2,4,6 의 요소의 개수를 리턴
```

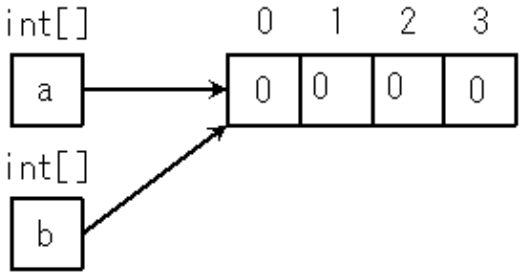
이 경우 다차원 배열의 요소에 할당되는 배열의 요소 수를 각각 출력하고 있기 때문에 모두 "3"과 출력됩니다.

다차원 배열은 다음 페이지에서 설명하도록 요소에 대입하는 배열의 길이는 같아야가 없기 때문에 각 요소에 할당되고 있는 배열의 길이를 개별적으로 취득해야하는 경우이 같이 "다차원 배열 이름 [인덱스] .length"로 리턴 받는다

3. 배열의 복사와 할당

배열도 배열간의 배열 변수에 복사를 할 수 있다. 일반적으로 변수를 대입하면 오른쪽에서 왼쪽으로 값이 복사된다. 그러나 일반적인 자료형의 경우는 대입 된 값이 복사되지만 참조 형식의 경우 할당되는 것은 참조이며, 참조 된 배열이 복사되는 것은 아닙니다. 즉, 주소가 대입되어 같은 곳의 메모리를 참조하게 된다.
다음과 같이 a의 배열 변수가 받은 메모리 할당 주소를 b에게 대입하게 되면 b는 a가 참조하는 주소를 대입받아 같은 주소값을 참조하는 배열변수가 된다.
즉 a[0]와 b[0]번지 주소와 요소의 값은 같은 주소, 같은 값이다.

```
int[] a = new int[4];
int[] b = a;
```



배열 변수의 유형 정의는 int 나 String 과 같은 형태와 대괄호 ([및])이 지정되어 있다.
요소 수를 지정하여 실제로 배열이 생성되는 것은 다음이다. 즉, 변수 자신은 요소 수에 대한 정보는 없고, 요소에 저장할 수 있는 형태와 배열 뿐이다. 따라서 일단 배열로 선언 된 변수에 다른 배열 변수를 할당 할 때 요소의 개수가 다른 경우에도 대입 호환된다.

```
int [] array1 = new int [3];
int [] array2 = new int [10];
array1 = array2; // OK!
```

단, 배열 변수는 같은 데이터 형태와 차원이 동일한 경우, 요소 수에 관계없이 대입 호환된다.

배열은 한번 생성하면 그 크기를 변경할 수 없다. 더 많은 저장공간이 필요하다면 보다 큰 배열을 새로 만들고 이전 배열로부터 내용을 복사해야한다.
배열 간의 내용을 복사하려면 for문을 사용하거나 System클래스의 arraycopy메서드를 사용한다

[형식]

System.arraycopy(원본배열, 원본인덱스,
대상배열 , 대상인덱스, 갯수)

다음은 arraycopy메소드를 이용한 배열 복사를 살펴 보자.

```
public class Array03 {
  public static void main(String[] args) {

    int [] FromInt = {1, 2, 3, 4, 5, 6, 7};   // (1)
    int [] ToInt = new int [10];   // (2)
    System.arraycopy (FromInt, 1, ToInt, 3, 5);   // (3)

    for (int i = 0; i <ToInt.length; i++) {   // (4)
     System.out.printf("%3d",ToInt [i]);
    }

  }

}
```

[해설]

(1) 원본 배열 FromInt 을 선언 · 생성

(2) 대상 배열 ToInt 을 선언 · 생성

(3) arraycopy 메서드를 사용하여 배열의 카피를 한다.. 여기에 FromInt 배열의 인덱스 번호 1 에서 5 인덱스 분을 ToInt 배열의 인덱스 번호 3 부터 순차적으로 복사하는 것을 의미한다.

(4) 복사가 제대로되어 있는지, ToInt 배열 데이터의 내용을 순서대로 표시한다.

[실행 결과]

```
 2. 0   0   2   3   4   5   6   0   0
```

4. 커맨드라인을 통해 입력 받기

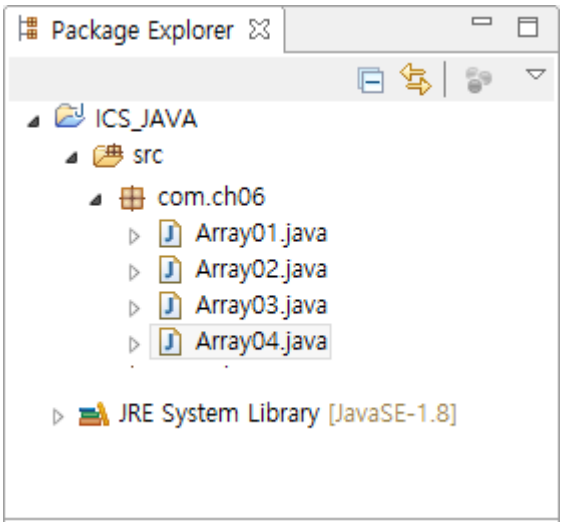
Sring [] args - main () 메소드의 인수로 지정되는 배열 Srintg [] args 는 무엇을 위한 것일까 ?
대답은 명령 줄 인수이다. 명령 줄에서 Java 응용 프로그램에 매개 변수를 전달할 수 있다.
명령 줄 인수는 main () 메소드에서 사용할 수 있는 매개 변수를 주는 구조이다. 명령 줄에서 실행할 때 전달 된 문자열이 String 형 배열 args에 할당된다. 예를 들어, 명령 줄에서 전달 된 문자열의 첫 번째는 args[0] 에서 볼 수 있다.

```
public class Array04 {
        public static void main(String[] args) {
                System.out.print ( args [0] );
        }
}
```

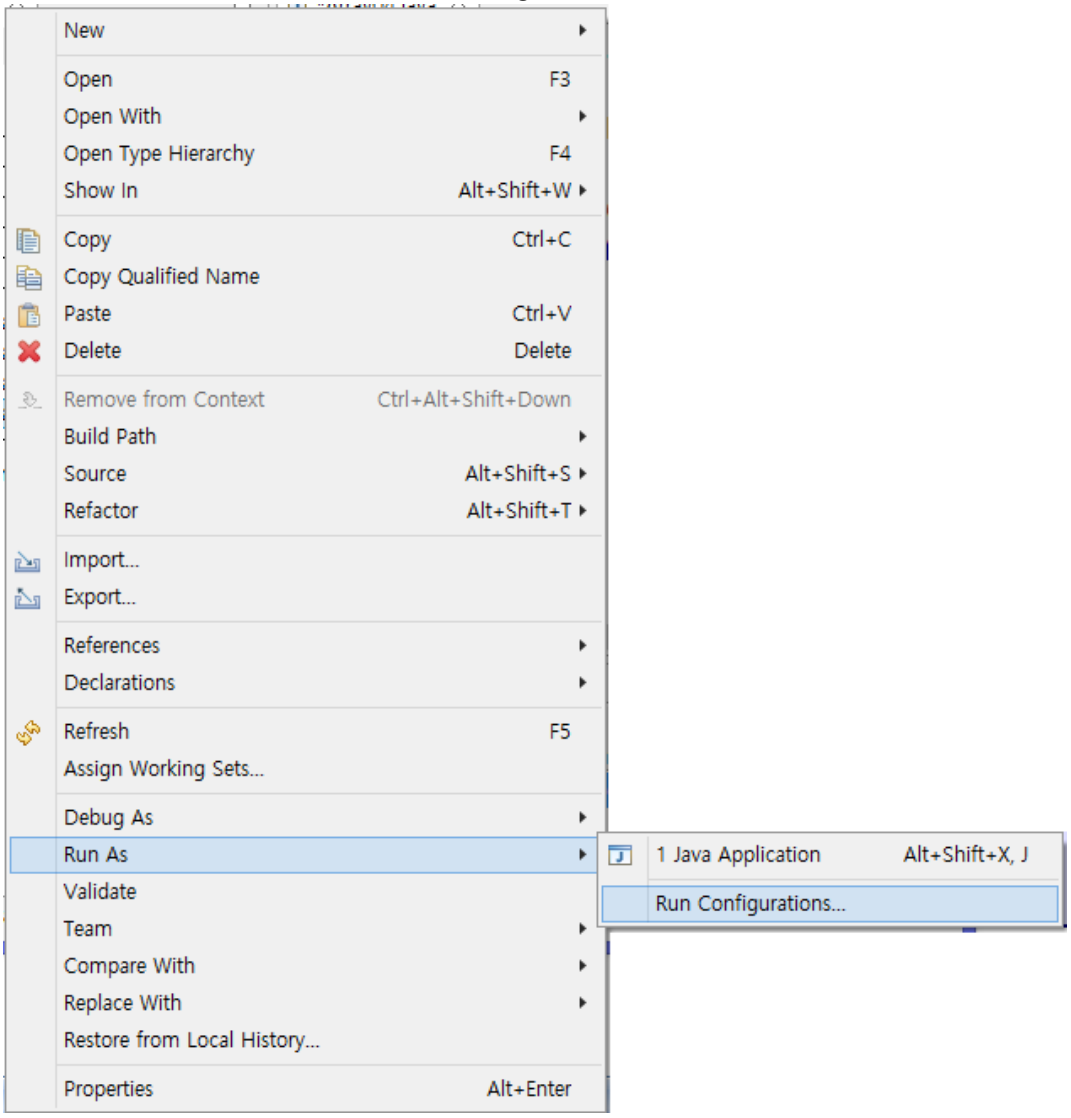
위 코드 명령 줄 인수의 첫 번째 것을 출력한다.
실행 예는 다음과 같습니다. 명령 줄 인수 `Hello!`을 주고 있다.

이클립스에서 실행 시에 커맨드 창으로 값을 대입하려면 다음과 같이 실행한다.

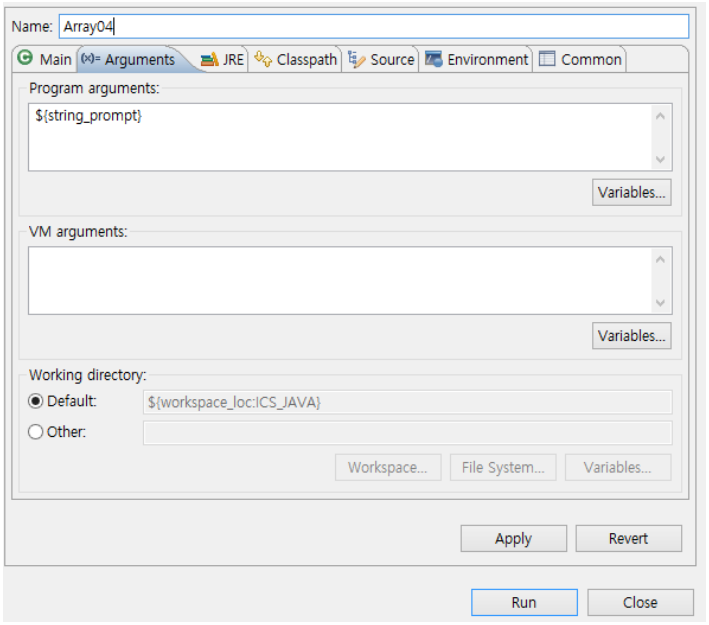
[1 단계] 패키지 익스플러로 창에서 해당 클래스 파일을 클릭한 다음 오른쪽 버튼을 클릭한다.



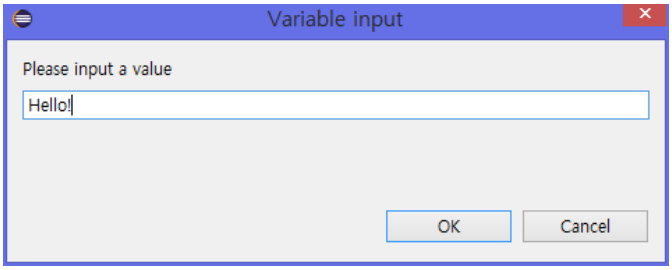
[2 단계]창이 뜨면 Run As -> Run Configurations 를 클릭한다.



[3 단계]창이 아래와 같이 뜨면 Arguments 탭에 Program arguments:탭메뉴에 ${string_prompt}을 입력하고 Run 실행 버튼을 클릭한다.



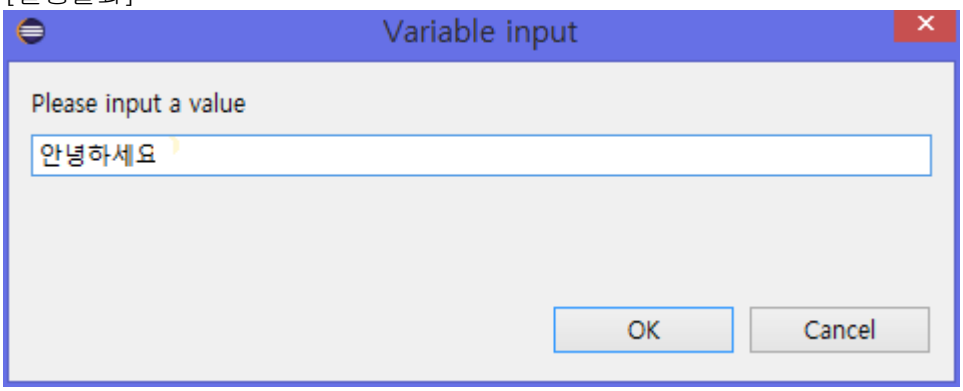
[4 단계] 실행을 하게 되면 다음과 같은 프롬프트 창에 Hello! 라고 입력한다.



[실행결과]

Hello!

다음의 실행 예는 명령 줄 인수에 안녕하세요!를 입력한 결과이다.
[실행결과]



안녕하세요

실행 결과를 본대로 인수를 변경하면 소스 코드를 다시 작성하지 않고도 실행 결과가 달라진다. 일반적으로 인수는 프로그램 실행 시 프로그램에 매개 변수를 동적으로 제공 하는 구조이다.

다음은 실행시 메인으로 두개의 값을 받아 곱을 구하는 구문이다.

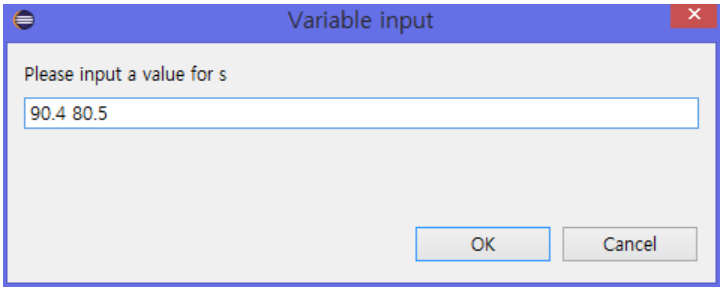
```
public class Array05 {

	public static void main(String[] args) {
		double a = Double.parseDouble ( args [0] );
		double b = Double.parseDouble ( args [1] );
		System.out.println (a * b);

	}

}
```

명령 줄 인수를 두 번째 것이 args [1]에서 참조되어 그 값을 double 형으로 해석 한 것이 b 에 대입 된다. 실행 예는 다음과 같다.여러 명령 줄 인수는 공백으로 구분한다.



[실행결과]

7277.200000000001

위 프로그램 실행시 아무런 인수를 주지 않고 실행하게 되면 다음과 같은 Exception이 발생한다.

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
0
	at com.ch06.Array05.main(Array05.java:6)
```

아무것도 인수를 제공하지 않기 때문에, 배열이 만들어져 없지만, 첫 번째 인수를 출력하려고 했기 때문에 발생한 상황이다. 없는 것을 내놓으라고 하는 의미이다.

"ArrayIndex Out Of Bounds Exception"유형의 오류가 보고 될 때는 "배열의 인덱스가 한계를 넘었습니다"라는 의미이다.