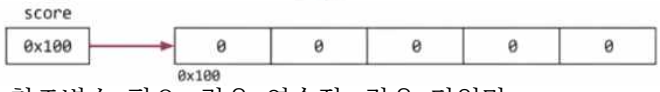
chapter 5. 배열

ch5-1 배열이란

같은 타입의 여러 변수를 하나의 묶음으로 다루는 것

```
int[] score = new int[5];
```



-참조변수 필요, 값은 연속적, 같은 타입만

ch5-2 배열의 선언과 생성

배열도 사용 전 반드시 선언과 생성을 해야 한다.

```
타입[] 변수이름;              // 배열을 선언(배열을 다루기 위한 참조변수 선언)
변수이름 = new 타입[길이];    // 배열을 생성(실제 저장공간을 생성)
```

(참조변수)

ch5-3 배열의 인덱스

각 저장공간에 자동으로 붙는 일련번호

ch5-4 배열의 길이

배열이름.length-배열의 길이를 알 수 있다.

배열은 한번 생성하면 실행동안 그 길이를 바꿀 수 없다.

-왜? 메모리가 있을 수도 있고 없을 수도 있기 때문

-부족하면? 새로 만들고 기존 값을 복사한다.

ch5-5 배열의 초기화

배열의 각 요소에 처음으로 값을 저장하는 것 (자동은 0)

```
int[] score = { 50, 60, 70, 80, 90};  // new int[]를 생략할 수 있음
```

ch5-6 배열의 출력

-프린트()에 참조변수 이름만 입력해서는 출력X

-char[]은 예외.. 출력됨

1.
```
for(int i=0;i < iArr.length; i++) {  // 배열의 요소를 순서대로 하나씩 출력
    System.out.println(iArr[i]);
}
```

2.
```
// 배열 iArr의 모든 요소를 출력한다. [100, 95, 80, 70, 60]이 출력된다.
System.out.println(Arrays.toString(iArr));
```

ch5-7 배열의 활용 (실습)

-배열의 모든 요소의 총합과 평균 구하기

-배열의 요소중 제일 큰값과 작은값 구하기

-배열안의 숫자 섞기

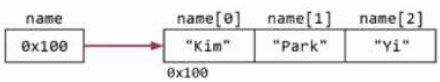
-로또 번호 생성

ch5-8 String배열의 선언과 생성

-여러개의 문자열을 저장 할 수 있는 배열

-String 은 참조형이기 때문에 기본값이 null

```
name[0] = "Kim";
name[1] = "Park";  ▶ String[] name = { "Kim", "Park", "Yi" };
name[2] = "Yi";
```



//참조형이기 때문에 값 집적입력X

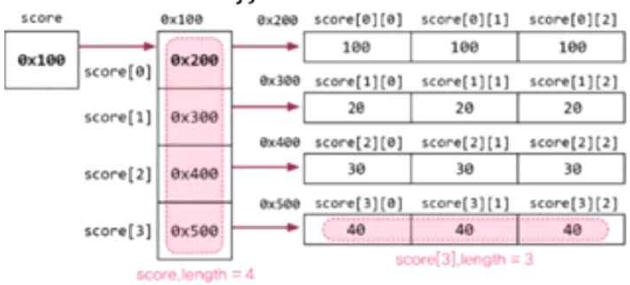
//주소값이 들어간다!

ch5-9 2차원 배열, 초기화

테이블 형태의 데이터를 저장하기 위한 배열

<2차원 배열 생성과 초기화>

```
int[][] score = {
                 {100, 100, 100},
                 {20, 20, 20},
                 {30, 30, 30},
                 {40, 40, 40}
                };
```



1차원 배열의 배열이 2차원 배열..!

ch5-10 String클래스

1.char[]와 메서드(기능)을 결합 한 것->String 클래스로 쓰는 것이 편리해서 String클래스로 씀..!

2.String클래스는 내용을 변경 할 수 없다. (read only)

->"a"+"b"="ab"로 바뀌는 것이 아니라 새로운 공간에 "ab"가 생성되는 것이다.

ch5-11 String클래스의 주요 메서드

메서드	설명
char charAt(int index)	문자열에서 해당 위치(index)에 있는 문자를 반환한다.
int length()	문자열의 길이를 반환한다.
String substring(int from, int to)	문자열에서 해당 범위(from~to)의 문자열을 반환한다.(to는 포함 안 됨)
boolean equals(Object obj)	문자열의 내용이 같은지 확인한다. 같으면 결과는 true, 다르면 false
char[] toCharArray()	문자열을 문자배열(char[])로 변환해서 반환한다.

substring()에서 to를 생략하면 from부터 끝까지 ~

ch5-12 Arrays 클래스로 배열 다루기

배열의 비교와 출력 - equals(), toString()

<toString()>

```
int[]   arr   = {0,1,2,3,4};
int[][] arr2D = {{11,12}, {21,22}};

System.out.println(Arrays.toString(arr)); // [0, 1, 2, 3, 4]
System.out.println(Arrays.deepToString(arr2D)); // [[11, 12], [21, 22]]
```

<equals()>

```
String[][] str2D  = new String[][]{{"aaa","bbb"},{"AAA","BBB"}};
String[][] str2D2 = new String[][]{{"aaa","bbb"},{"AAA","BBB"}};

System.out.println(Arrays.equals(str2D, str2D2));     // false
System.out.println(Arrays.deepEquals(str2D, str2D2)); // true
```

배열의 복사 - copyOf(), copyOfRange()

```
int[] arr =  {0,1,2,3,4};
int[] arr2 = Arrays.copyOf(arr, arr.length); // arr2=[0,1,2,3,4]
int[] arr3 = Arrays.copyOf(arr, 3);          // arr3=[0,1,2]
int[] arr4 = Arrays.copyOf(arr, 7);          // arr4=[0,1,2,3,4,0,0]
int[] arr5 = Arrays.copyOfRange(arr, 2, 4);  // arr5=[2,3] ← 4는 불포함
int[] arr6 = Arrays.copyOfRange(arr, 0, 7);  // arr6=[0,1,2,3,4,0,0]
```

배열의 정렬 - sort()

```
int[] arr = { 3, 2, 0, 1, 4 };
Arrays.sort(arr);  // 배열 arr을 정렬한다.
System.out.println(Arrays.toString(arr)); // [0, 1, 2, 3, 4]
```