제 6 장
유용한 참조타입과 배열
3부 - Array

참조타입

- 참조(reference)타입, 객체(object)타입의 변수
  - 정보가 저장된 (위치를 가리키는) 주소를 저장하고 있는 유형의 변수 (cf. 기본 데이터 타입)
  - 모든 객체는 참조타입의 변수로써 할당된다
- 자주 사용하는 참조 타입의 객체
  - **String**
  - **Number**
    - wrapper 클래스 : 기본 데이터 타입을 참조타입의 객체로 변환
    - 자동박싱(Autoboxing)과 언박싱(Unboxing)
  - **Math** : 각종 연산 메소드를 포함하고 있는 클래스
  - **배열 객체**
  - **Enum** 객체

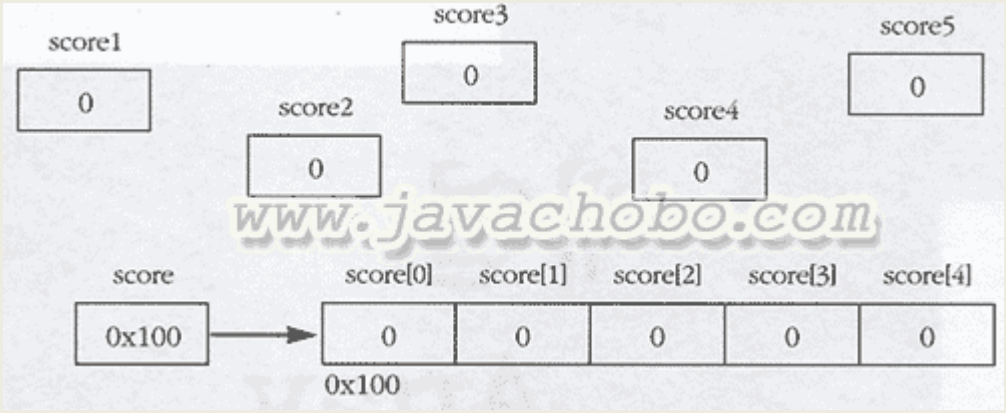
배열(Array)

- 예를 들어, 학생이 10명이고 이들의 평균 성적을 계산하라
- 기존의 기본 타입 변수 개념을 사용한다면, 학생이 10명이므로 10개의 변수가 필요하다

```
int    s0, s1, s2, s3, s4, s5, s6, s7, s8, s9;
```

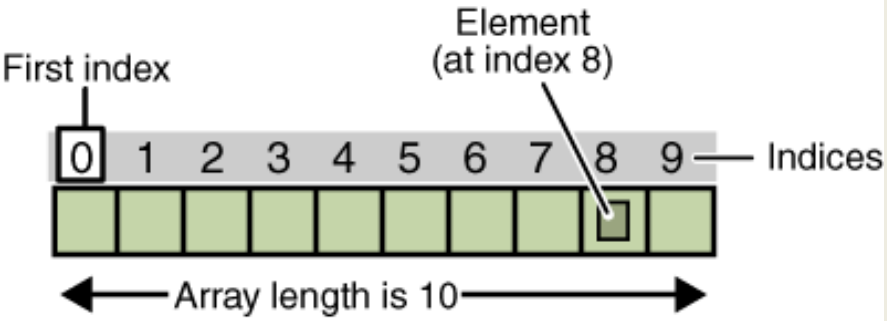
- 만약 학생이 100명이라면? 아래처럼 해야 하는가?

```
int   s0, s1, s2, s3, s4, s5, s6, s7, s8, s9,...,s99;
```



배열의 개념

- 배열(array): 1개 이상의 동일한 타입의 특정 개수 변수들의 선형 순차적 모임을 수용하는 컨테이너 **객체 – 기본 타입이 아니다**
- 하나의 참조 변수로 액세스 되는 다수의 동일유형의 데이터 컨테이너 객체
- 객체 저장 메모리 영역에서 연속적 저장 형태를 가진다
- 인덱스를 통해 액세스가 이루어진다
- 1차원, 2차원, 3차원 등 다차원을 가질 수 있다

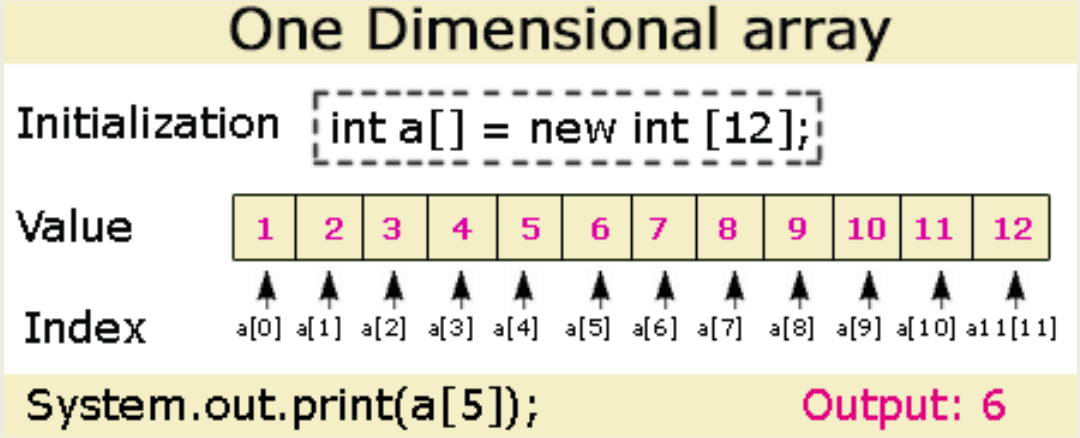

First index
Element
(at index 8)
0 1 2 3 4 5 6 7 8 9
Indices
Array length is 10

1차원 배열


수학에서 행렬
2차원 배열
http://hunit.tistory.com
1행 1,1 1,2 1,3 1,4 1,5
2행 2,1 2,2 2,3 2,4 2,5
3행 3,1 3,2 3,3 3,4 3,5
1열 2열 3열 4열 5열
0행 0,0 0,1 0,2 0,3 0,4
1행 1,0 1,1 1,2 1,3 1,4
2행 2,0 2,1 2,2 2,3 2,4
0열 1열 2열 3열 4열

배열의 인덱스

- 다음과 같은 1차원 배열을 생각해 보자.

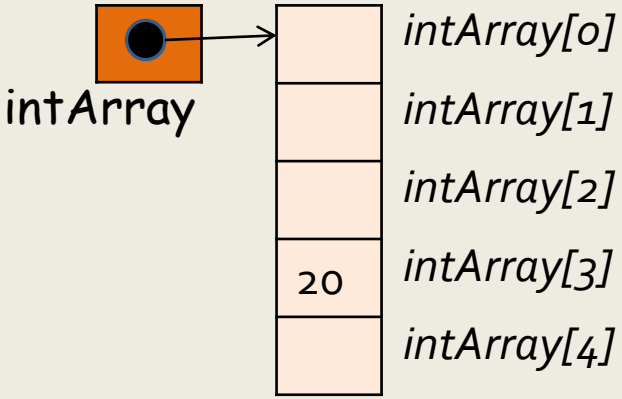


- 배열 엘리먼트에는 번호가 붙어 있는데 이것을 인덱스**(index)**라고 하며, index는 항상 0부터 시작하고 마지막 index는 (크기-1)이다
  - 첫 번째 엘리먼트의 index는 0이고, 마지막 엘리먼트의 index는 11
  - 배열의 5번째 요소에 80을 저장하려면 a[4] = 80 한다

배열의 선언과 생성

- 배열은 객체이므로 기본형 변수와는 다르게 선언과 생성, 2단계를 거쳐 사용가능하다
- 1) **배열의 참조 변수 선언**
  - **변수type[ ] 변수명 ;** → **int[ ] intArray;** //일반적으로 사용
  - 변수type 변수명[ ]; → int intArray[];

- 2) **키워드 new를 사용하여 배열 객체 생성**
  - **intArray = new int[5];**



- 3) 인덱스 기반으로 배열 객체 참조
  - intArray[3] = 20;

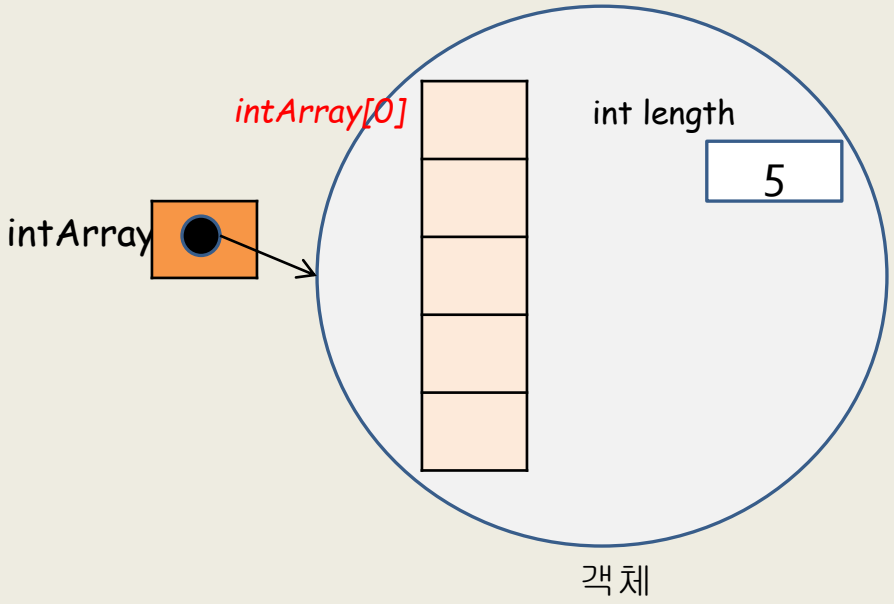
- **선언과 생성을 동시에 할 수 있다**
  - **int[] intArray = new int[5] ;**

```
byte[]   arrayOfBytes;     //byte 타입의 정수를 수용하는 배열
short[]  arrayOfShorts;    //short 타입의 정수를 수용하는 배열
char[]   arrayOfChars;     // char 타입의 문자를 수용하는 배열
String[] arrayOfStrings;   //String 타입의 객체를 수용하는 배열
```

배열의 크기

- 배열의 크기는 배열 참조 변수를 선언할 때 결정되지 않음
- 배열의 크기는 배열 생성 시에 결정되며, 나중에 바꿀 수 없다.
- 배열의 크기는 배열 객체의 **length**라는 필드에 저장되어 있다.

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

int size = intArray.length;
// size는 5
```



예제: 반복문과 배열

```
class ArrayTest {
    public static void main(String[] args) {
        int[] anArray;  // 정수 배열의 선언
        anArray = new int[10]; // 배열의 생성 및 요소의 개수 명시

        for(int i=0; i<10; i++) {
            anArray[i] = (i+1)*10;
        }
        for(int i=0; i<10; i++) {
            System.out.println("Element at index " + i + ": " + anArray[i]);
        }
    }
}
```

```
Element at index 0: 10
Element at index 1: 20
Element at index 2: 30
Element at index 3: 40
Element at index 4: 50
Element at index 5: 60
Element at index 6: 70
Element at index 7: 80
Element at index 8: 90
Element at index 9: 100
```

배열의 length 필드를 이용하여 배열 크기만큼 키보드에서 정수를 입력 받고 평균을 구하는 프로그램을 작성하시오.

```
import java.util.Scanner;
public class ArrayLength {
   public static void main (String[] args) {
      Scanner in = new Scanner(System.in);
      int intArray[] = new int[5];
      double sum = 0;

      for (int i = 0; i < intArray.length; i++)
         intArray[i] = in.nextInt();
      for (int i = 0; i < intArray.length; i++) {
         sum += intArray[i];
      }
      System.out.print("배열 원소의 평균은 " + sum/intArray.length + "입니다.");
   }
}
```

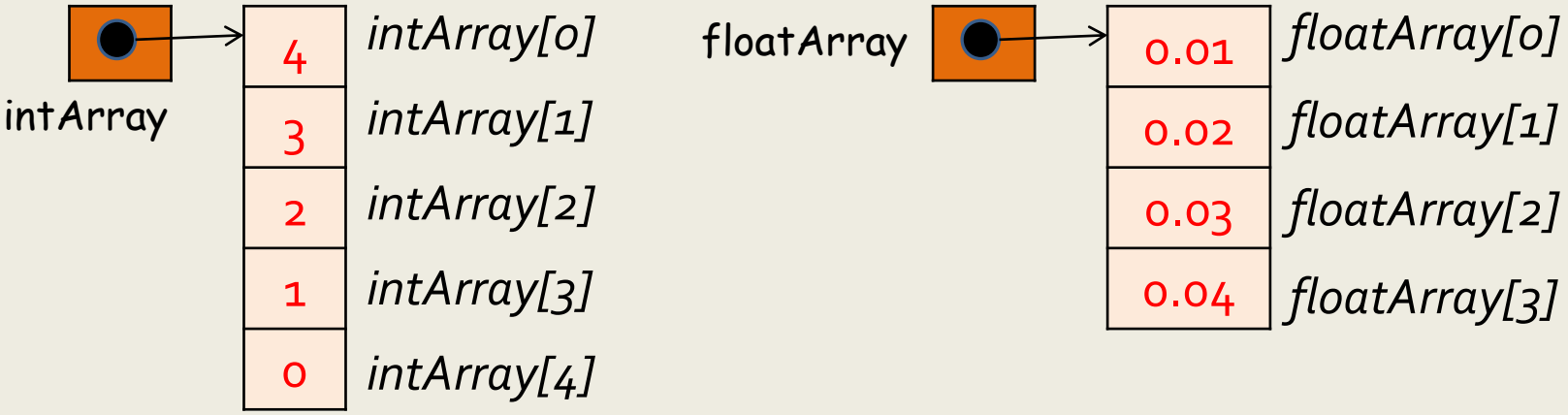
```
10
20
30
40
50
배열 원소의 평균은 30.0입니다.
```

배열의 초기화

- **배열의 초기화**

1. 배열은 선언하면서 초기화 될 수 있으며, 초기화는 생성을 포함한다

```
int intArray[] = {4, 3, 2, 1, 0};
float floatArray[] = {0.01, 0.02, 0.03, 0.04};
```

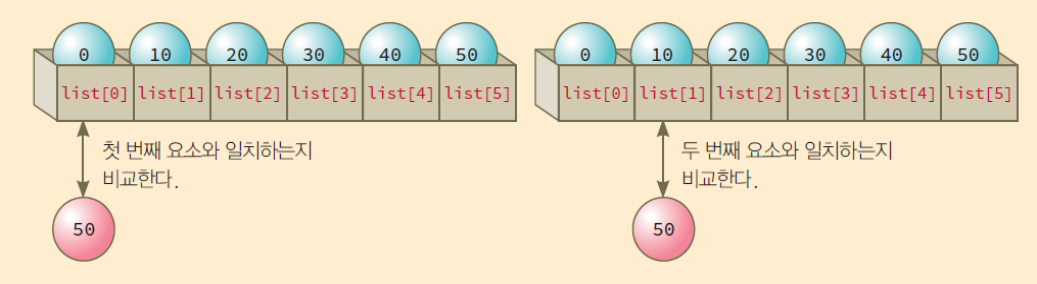


문자열 배열

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

                String[] toppings = { "Pepperoni", "Mushrooms", "Onions", "Sausage",
"Bacon" };

                for (int i = 0; i < toppings.length; i++) {
                        System.out.print(toppings[i] + " ");
                }
        }
}
```

순차탐색(sequential search) 알고리즘

- 배열의 원소를 순서대로 하나씩 꺼내서 탐색키와 비교하여 원하는 값을 찾아가는 방법




```
import java.util.Scanner;
public class SeqSearch {
public static void main(String[] args) {
        int s[] = { 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };
        int value, index = -1;
        Scanner scan = new Scanner(System.in);
        System.out.print("탐색할 값을 입력하시오: ");
        value = scan.nextInt();
        for (int i = 0; i < s.length; i++) {
                if (s[i] == value)
                        index = i;
        }
        if (index < s.length && index >= 0)
                System.out.println("" + value + "값은 " + index + "위치에 있습니다.");
}}
```

무명 배열(Anonymous Array)

- 자바에서는 배열의 이름을 지정하지 않고 단순히 초기값만으로 배열을 생성시킬 수 있다.
- 무명 배열(**anonymous arrays**)은 즉시 배열을 만들어서 함수의 인수로 전달하고자 할 때 많이 사용된다
- 형식은 다음과 같다

```
new int[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

- 참조 변수(이름)가 없으므로 생성한 메소드에서는 참조할 수가 없으며, 그를 파라메터로 넘겨받은 함수에서 참조하여 사용할 수 있다

AnonymousArray.java

```
01 public class AnonymousArray {
02
03    public static void main(String[] args) {
04       System.out.println("숫자들의 합 : " +
05                      sum(new int[] { 1, 2, 3, 4 }));
06    }
07
08    public static int sum(int[] numbers) {
09       int total = 0;
10       for (int i = 0; i < numbers.length; i++) {
11          total = total + numbers[i];
12       }
13       return total;
14    }
15 }
```

무명 배열이 생성되어 sum()으로 전달된다.

```
숫자들의 합 : 10
```

배열과 for-each 루프

- 배열(array)이나 나열(enumeration)의 각 원소를 순차적으로 접근하는 데 유용한 for 문

```
int[] num = { 1,2,3,4,5 };
int sum = 0;
for (int k : num) // 반복될 때마다 k는 num[0], num[1], ..., num[4] 값으로 설정
   sum += k;
System.out.println("합은 " + sum);
```

```
합은 15
```

```
String names[] = { "사과", "배", "바나나", "체리", "딸기", "포도" } ;
for (String s : names) // 반복할 때마다 s는 names[0], names[1], ..., names[5] 로 설정
   System.out.print(s + " ");
```

```
사과 배 바나나 체리 딸기 포도
```

```
enum Week { 월, 화, 수, 목, 금, 토, 일 }
for (Week day : Week.values()) // 반복될 때마다 day는 월, 화, 수, 목, 금, 토, 일로 설정
   System.out.print(day + "요일 ");
```

```
월요일 화요일 수요일 목요일 금요일 토요일 일요일
```

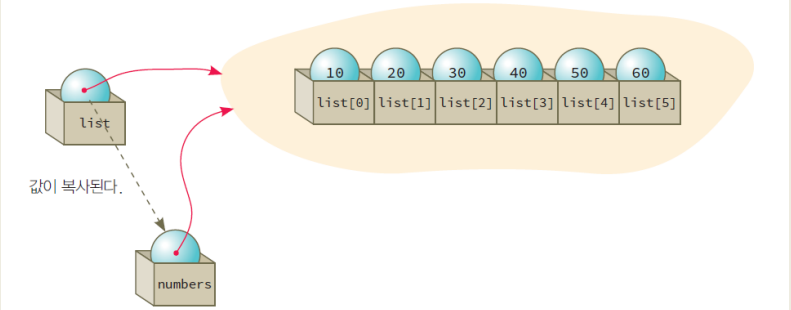
```
public class foreachEx {
    enum Week { 월, 화, 수, 목, 금, 토, 일 }

    public static void main(String[] args) {
        int[] num = { 1,2,3,4,5 };
        String names[] = { "사과", "배", "바나나", "체리", "딸기", "포도" } ;
        int sum = 0;

        // 아래 for-each에서 k는 num[0], num[1], ..., num[4]로 반복됨
        for (int k : num)
            sum += k;
        System.out.println("합은 " + sum);

        // 아래 for-each에서 s는 names[0], names[1], ..., names[5]로 반복됨
        for (String s : names)
            System.out.print(s + " ");
        System.out.println();

        // 아래 for-each에서 day는 월, 화, 수, 목, 금, 토, 일 값으로 반복됨
        for (Week day : Week.values())
            System.out.print(day + "요일 ");
         System.out.println();
    }
}
```

```
합은 15
사과 배 바나나 체리 딸기 포도
월요일 화요일 수요일 목요일 금요일 토요일 일요일
```

배열의 복사

- 배열 참조 변수의 복사

```
int [] list = { 10, 20, 30, 40, 50 };
int [] numbers = list;
```



- 한 배열의 모든 값을 다른 배열로 복사하고 싶다면 **Arrays** 클래스의 **copyOf()** 메소드를 사용
- (예) **int** [] list_copy = **Arrays**.copyOf(list, list.length);
- copyOf()는 배열의 크기를 변경하고자 할 때 많이 사용한다

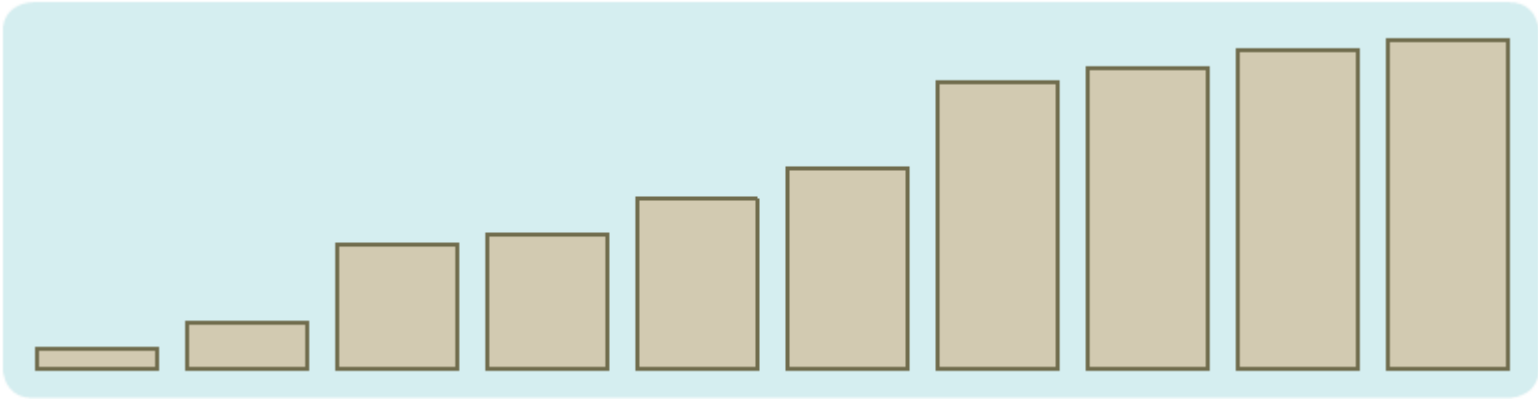
```
int[]   list = {1, 2, 3, 4, 5};
list = Arrays.copyOf(list, 2*list.length);
```

배열 list 의 크기는 5에서 10으로 늘어난다

배열의 정렬

- 배열에 저장된 숫자를 크기 순으로 정렬하려면 **Arrays.sort()** 사용

```
int[] a = new int[100];
a[0] = 32;
a[1] = 21;
...
Arrays.sort(a);
```



배열 정렬 예

SortExample.java

```
01 import java.util.Arrays;
02
03 public class SortExample {
04    public static void main(String[] args) {
05       final int SIZE = 10;
06       int[] numbers = new int[SIZE];
07
08       for (int i = 0; i < SIZE; i++) {
09          int r = (int) (Math.random() * 100);
10          numbers[i] = r;
11       }
12
13       System.out.print("최초의 리스트: ");
14       for (int r : numbers)
15          System.out.print(r + " ");
16       Arrays.sort(numbers);
17
18       System.out.print("\n정렬된 리스트: ");
19       for (int r : numbers)
20          System.out.print(r + " ");
21    }
22 }
```


```
최초의 리스트: 83 72 73 58 45 59 93 72 84 94
정렬된 리스트: 45 58 59 72 72 73 83 84 93 94
```

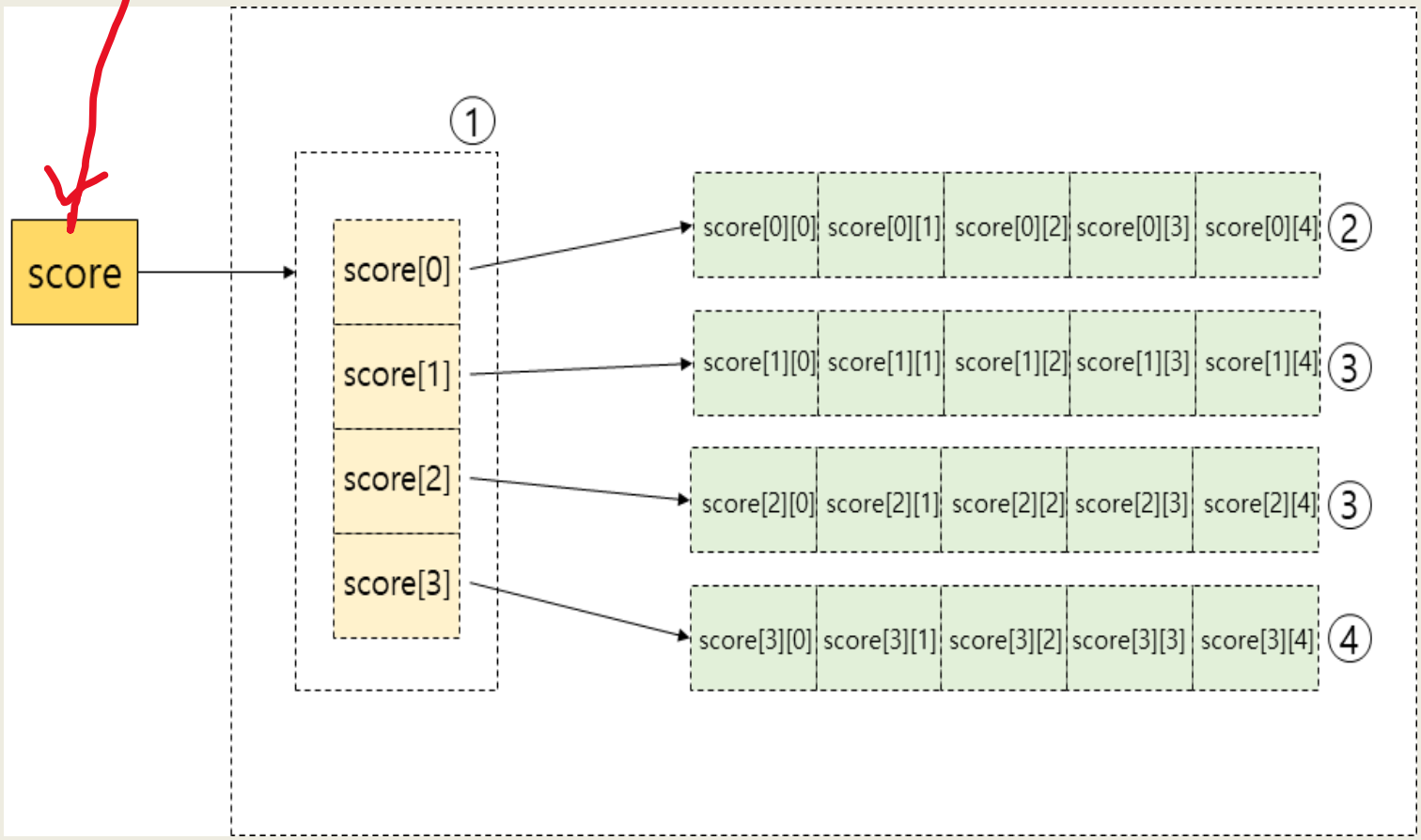
다차원배열

- 다차원 배열의 선언
  - 1차원 배열과 마찬가지로
  - 1) 배열 참조 변수의 선언,
  - 2) 해당 배열의 생성의 2단계로 선언된다
  - 다만 다른 것은 2차원 배열의 경우 대괄호를 2개, 3차원 배열의 경우 3개를 사용한다

```
type[ ][ ] nameOfArray2;  // 2차원 배열의 선언

nameOfArray2 = new type[M][N];      // 배열의 생성


type[ ][ ][ ] nameOfArray3;   // 3차원 배열의 선언

nameOfArray3 = new type[K][M][N];   // 배열의 생성
```

2차원 배열

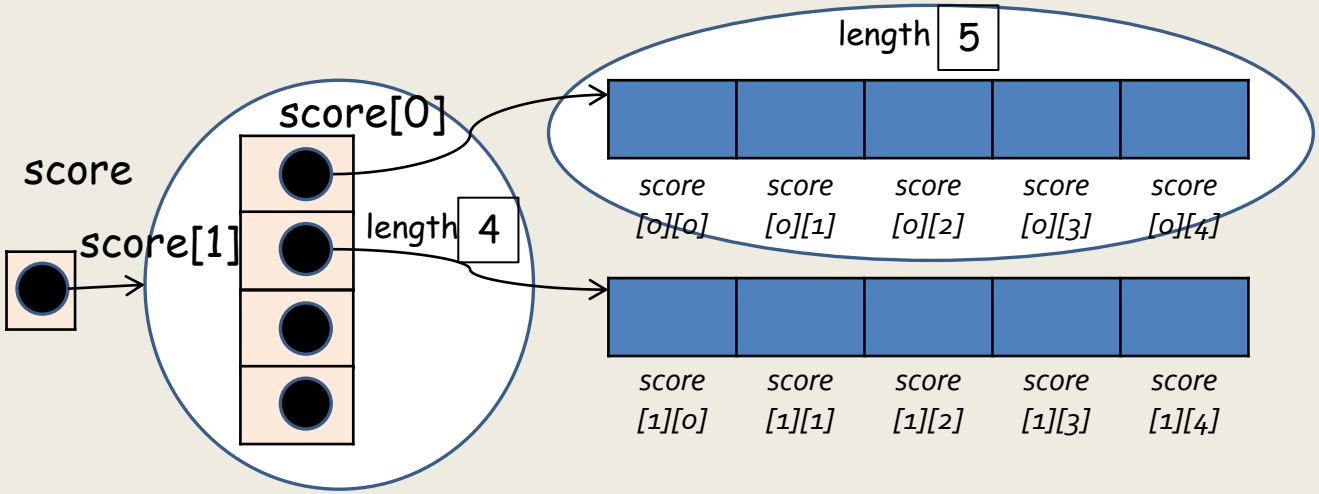
```
int[ ][ ] score;
score = new int[4][5];
```



2차원 배열의 length 필드

- 2차원 배열의 length
  - score.length -> 2차원 배열의 행의 개수로서 4
  - score[n].length는 n번째 행의 열의 개수
    - score[0].length -> 0번째 행의 열의 개수로서 5
    - score[1].length -> 1번째 행의 열의 개수로서 역시 5

```
int[ ][ ] score;
score = new int[4][5];
```



2차원 배열 예제

어떤 회사의 지난 3년간 분기별 매출의 총액과 연평균 매출을 구하는 프로그램을 작성하시오.

```
public class SalesRevenue {
   public static void main (String[] args) {
      int intArray[][] = {{90, 90, 110, 110}, {120, 110, 100, 110}, {120, 140, 130, 150}};
      double sum = 0;

      for (int i = 0; i < intArray.length; i++)
         for (int j = 0; j < intArray[i].length; j++)
            sum += intArray[i][j];

      System.out.println("지난 3년간 매출 총액은 " + sum + "이며 연평균  매출은 " +
                           sum/intArray.length + "입니다.");
   }
}
```

지난 3년간 매출 총액은 1380.0이며 연평균 매출은 460.0입니다.

3차원 배열

int[ ][ ][ ] = subject;

subject = int[4][2][3];

score[K][M][N] 배열의 경우 1차원 배열의 총 개수는 (1 + K + K x M)이 된다.
- 길이 K인 1차원 배열 1개,
- 길이 M인 1차원 배열 K개,
- 길이 N인 1차원 배열 (K x M)개가 생성되는 것이다.