

# **COMP217**

# **Java Programming**

## **Spring 2018**

*Week 10*

*Arrays*

# Goals

- 이번 주에 배우게 될 내용 :
  - 배열의 선언
  - 배열의 사용
  - 배열과 메소드
  - 객체들의 배열
  - 2차원 배열

## 5명의 학생들의 성적을 입력받아 평균계산

```
int a, b, c, d, e;
Scanner s = new Scanner(System.in);
System.out.print("학생 1의 성적 : ");
a = s.nextInt();
System.out.print("학생 2의 성적 : ");
b = s.nextInt();
System.out.print("학생 3의 성적 : ");
c = s.nextInt();
System.out.print("학생 4의 성적 : ");
d = s.nextInt();
System.out.print("학생 5의 성적 : ");
e = s.nextInt();
double sum = a+b+c+d+e;
double average = sum/5;
System.out.println("평균은 "+average+"입니다.");
```

학생이 100명이면??

```
성적을 입력하시오:20
성적을 입력하시오:30
성적을 입력하시오:40
성적을 입력하시오:50
평균 성적은30입니다
```

# 배열의 선언과 사용

- 배열(array): 같은 타입의 변수들의 모임

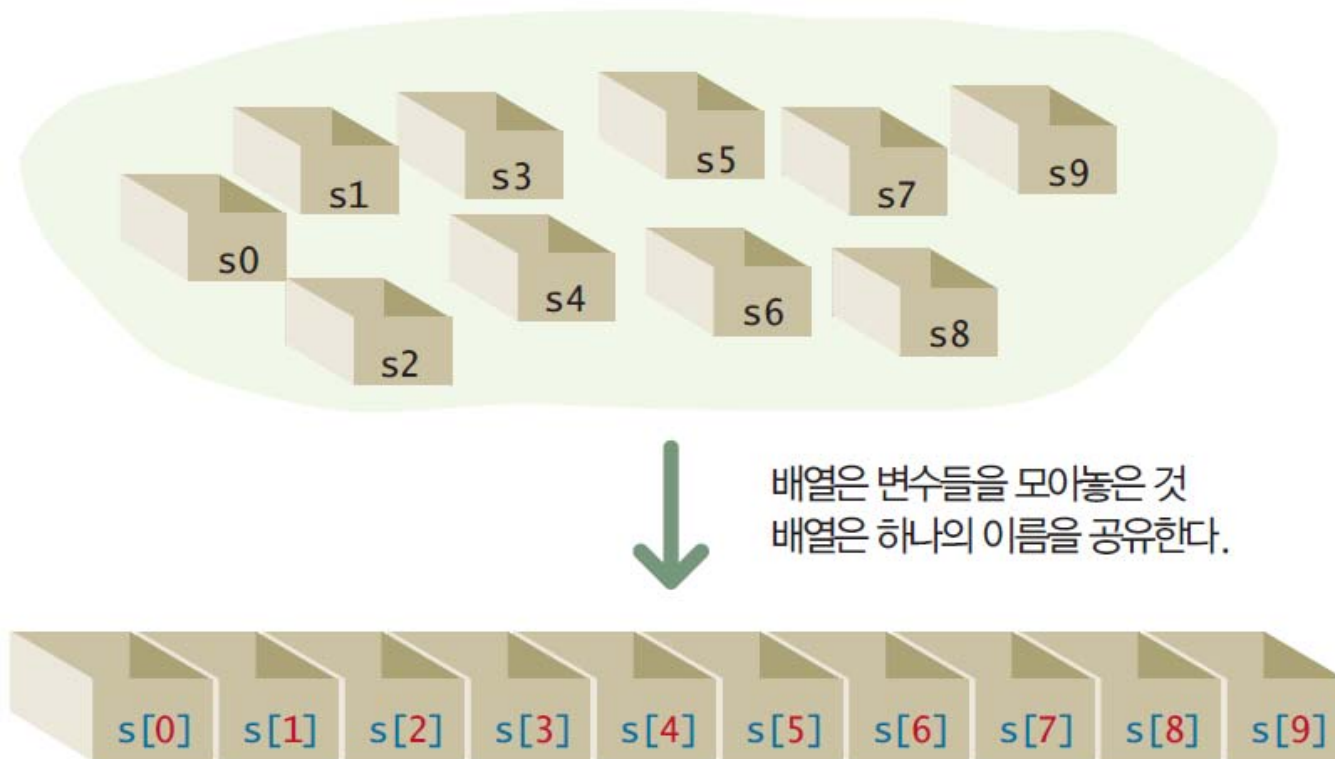


그림10-1. 배열은 변수들의 모임이다.

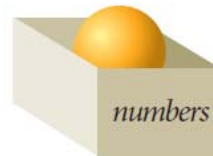
- 10개의 변수
- 인덱스 주의 : 0 부터 시작

# 배열을 만드는 절차

## ① 먼저 배열 참조 변수부터 선언

```
int[] numbers;
```

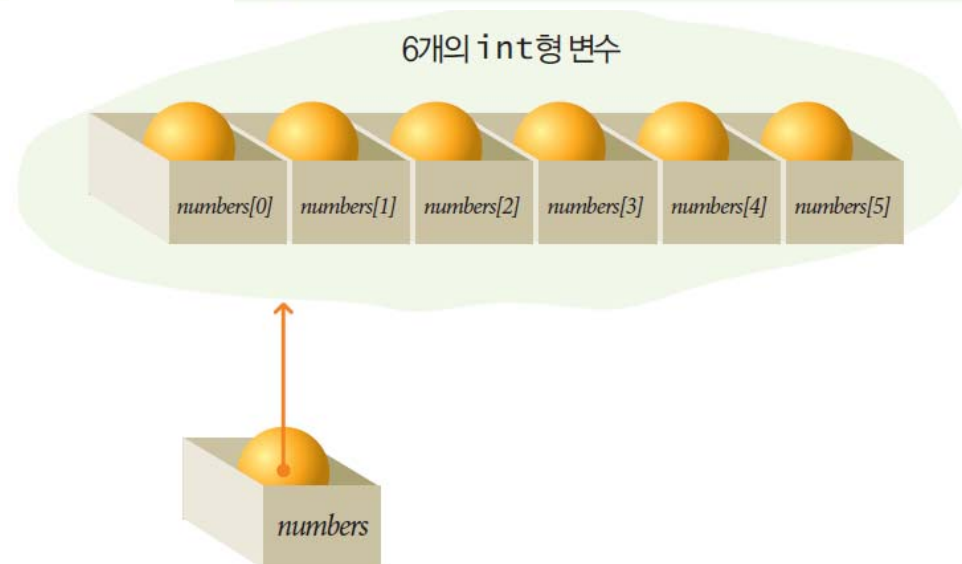
```
// 배열 참조 변수 선언
```



## ② 배열을 new 연산자를 사용하여 생성

```
numbers = new int[6];
```

```
// 배열 객체 생성
```



# 배열을 만드는 절차

- 배열을 선언과 동시에 생성하는 것도 가능하다.

```
int[]    numbers = new int[6];           // 선언과 동시에 생성
```

- 어떤 자료형의 배열도 생성이 가능하다.

```
float[]  distances = new float[20];      // 실수 배열  
char[]   letters = new char[50];        // 문자 배열
```

```
int [] scores = new int[5];
Scanner s = new Scanner(System.in);
int sum = 0;
for(int i=0; i<5; i++){
    System.out.printf("학생 %d의 성적 : ", i+1);
    scores[i] = s.nextInt();
    sum += scores[i];
}
double average = (double)sum/5;
System.out.println("평균은 "+average+"입니다.");
```

**실행결과**

성적을 입력하시오: 10  
성적을 입력하시오: 20  
성적을 입력하시오: 30  
성적을 입력하시오: 40  
성적을 입력하시오: 50  
평균 성적은 30입니다

# The Length of an Array & index

Once an array is created, its size is fixed. It cannot be changed. You can find its size using

참조변수.**length**

The array elements are accessed through the index. The array indices are *0-based*, i.e., it starts from 0 to `arrayRefVar.length-1`.

For example,

`scores.length` returns 5



# 배열의 초기화

ArrayTest2.java

```
int[] numbers = new int[]{10, 20, 30};
```

```
01 public class ArrayTest2 {  
02     public static void main(String[] args) {  
03         int[] numbers = { 10, 20, 30 };  
04         for (int i = 0; i < numbers.length; i++)  
05             System.out.println(numbers[i]);  
06     }  
07 }
```

이 문장은 상당히 많은 작업을 하는데 먼저 배열 참조 변수를 선언하고 배열을 생성하며 주어진 초기값을 배열 원소에 저장한다. 이 경우 new를 사용하여 생성하지 않아도 주어진 초기값 개수만큼의 배열이 자동적으로 생성됨을 유의하라.

각 배열은 length라는 필드를 가지고 있다. length 필드는 배열의 크기를 나타낸다. 따라서 이것을 이용하면 배열의 크기만큼 반복을 시킬 수 있다.

## 실행결과

```
10  
20  
30
```

# Declaring, creating, initializing Using the Shorthand Notation

```
double[] myList = {1.9, 2.9, 3.4, 3.5};
```

```
// declare, create, initialize at the same time
```

This shorthand notation is equivalent to the following statements:

```
double[] myList = new double[4];
```

```
myList[0] = 1.9;
```

```
myList[1] = 2.9;
```

```
myList[2] = 3.4;
```

```
myList[3] = 3.5;
```

\*This is an error

```
double[] myList;
```

```
myList = {1.9, 2.9, 3.4, 3.5};
```

# for-each 루프

```
for (자료형 변수 : 배열이름)
{
    // 반복 문장들
}
```

```
double[] numbers = {10.0, 21.2, 300.2};
for(double x: numbers){
    System.out.println(x);
}
```

```
for(int value: numbers)
    System.out.println(value);
```

```
int[] numbers = {10, 20, 30};
for(int i=0; i<numbers.length; i++){
    System.out.println(numbers[i]);
}
```

ArrayTest3.java

```
01 public class ArrayTest3 {
02     public static void main(String[] args) {
03         int[] numbers = { 10, 20, 30 };
04         for (int value : numbers)
05             System.out.println(value);
06     }
07 }
```

이 문장은 상당히 많은 작업을 하는데 먼저 배열 참조 변수를 선언하고 배열을 생성하며 주어진 초기값을 배열 원소에 저장한다. 이 경우 new를 사용하여 생성하지 않아도 주어진 초기값 개수만큼의 배열이 자동적으로 생성됨을 유의하라.

변수 value에는 첫 번째 원소부터 마지막 배열 원소까지 차례대로 대입된다.

# 사용자가 배열의 크기를 지정

ArrayTest4.java

5명의 학생들의 성적을 입력받아 평균계산

```
01 import java.util.Scanner;
02 public class ArrayTest4 {
03     public static void main(String[] args) {
04         int total = 0;
05         int size;
06         Scanner scan = new Scanner(System.in);
07         System.out.print("배열의 크기를 입력하시오:");
08         size = scan.nextInt();
09         int[] scores = new int[size];
10
11         for (int i = 0; i < scores.length; i++) {
12             System.out.print("성적을 입력하시오:");
13             scores[i] = scan.nextInt();
14         }
15         for (int i = 0; i < scores.length; i++)
16             total += scores[i];
17         System.out.println("평균 성적은" + total / scores.length + "입니다");
18     }
19 }
```

-----변수로 배열의 크기를 지정할 수 있다.

# 메소드의 매개변수로 배열 전달

```
int[] values = new int[6];  
for(int i=0; i<values.length; i++){  
    values[i] = 10*(i+1);  
}
```

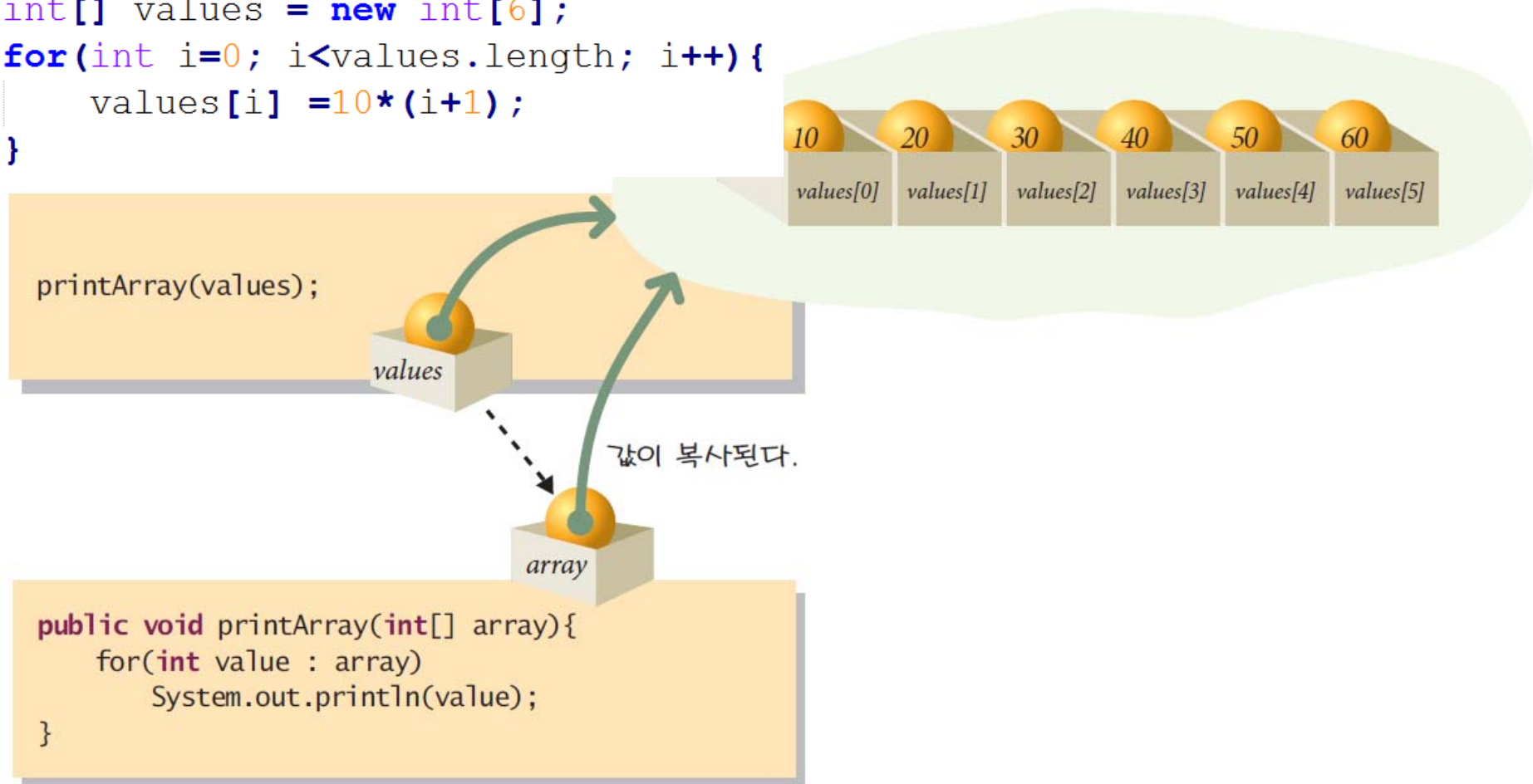
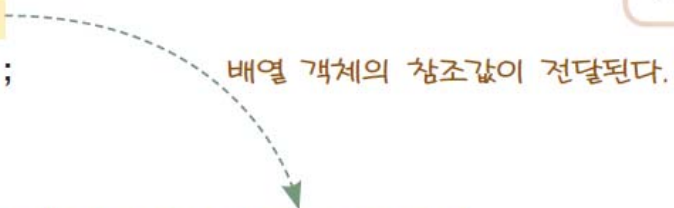


그림10-2. 메소드와 배열

## ArrayTest5.java

```
01 import java.util.Scanner;
02
03 public class ArrayTest5 {
04     final static int STUDENTS = 5;
05
06     public static void main(String[] args) {
07         int[] scores = new int[STUDENTS];
08         getValues(scores);
09         getAverage(scores);
10     }
11
12     private static void getValues(int[] array) {
13         Scanner scan = new Scanner(System.in);
14         for (int i = 0; i < array.length; i++) {
15             System.out.print("성적을 입력하시오:");
16             array[i] = scan.nextInt();
17         }
18     }
19
20     private static void getAverage(int[] array) {
21         int total = 0;
22         for (int i = 0; i < array.length; i++)
23             total += array[i];
24         System.out.println("평균 성적은 " + total / array.length + "입니다");
25     }
26 }
```



## 실행결과

성적을 입력하시오:10  
성적을 입력하시오:20  
성적을 입력하시오:30  
성적을 입력하시오:40  
성적을 입력하시오:50  
평균 성적은 30입니다



# 중간 점검

1. `int`형의 100개의 원소를 가지고 `array`로 참조되는 배열을 생성하는 문장을 써라. `int[] array = new int[100];`
2. 10개의 원소를 가지는 배열에서 올바른 인덱스 값의 범위는? `0~9`
3. 만약 배열의 인덱스가 올바르지 않으면 어떻게 되는가? `예외발생`
4. { 1.2, 3.1, 6.7 }의 값으로 초기화되는 `double`형 배열을 생성하는 문장을 작성하시오. `double[] arr = {1.2, 3.1, 6.7};`
5. `int`형 정수를 저장하고 있는 배열 `array`의 모든 원소의 값을 두배로 만드는 반복 루프를 작성하라.  

```
for(int i=0;i<array.length;i++){  
    array[i]*=2;  
}
```

`System.out.print("배열 크기 : ");  
Scanner s = new Scanner(System.in);  
int size = s.nextInt();  
double [] arr = new double[size];`
6. 사용자에게 배열의 크기를 받아서 `double`형 배열을 생성하는 문장을 작성하라.
7. `for-each`와 전통적인 `for` 루프를 비교하라.  
`for-each` : 배열의 크기를 신경쓰지 않아도 됨.  
          인덱스 변수 생성할 필요 없이 첫번째 원소부터 마지막 원소까지 처리 가능  
`for` : 일부 원소만 처리하는 경우. 배열 원소 변경 가능.
8. 하나의 배열을 다른 배열로 복사하는 반복 루프를 작성하라.  

```
for(i=0;i<array1.length;i++){  
    array2[i] = array1[i];  
}
```
9. 배열이 메소드로 전달되면 어떤 값이 전달되는 것인가? `참조값`
10. 배열 원소가 전달되는 경우와 배열 전체가 전달되는 경우를 비교하여 차이점을 설명하라. `원소가 전달 되는 경우 값이 전달,  
배열전체가 전달되는 경우 참조값이 전달`

# Processing Arrays(type: double): 10

- 배열의 원소개수 사용자로부터 입력 받아 double타입의 배열 생성
  - 아래와 같은 작업을 하는 메소드 만들어 테스트
1. (Initializing arrays with random values,  $0 \leq r < 10$ )
  2. (Printing arrays)
  3. (Summing all elements) - return type: double
  4. (Finding the largest element) - return type: double
  5. (Finding the smallest index of the largest element) – return type: ?? Void는 아님
  6. (*Random shuffling*)
  7. (*Shifting elements*)
    - 방향 : 오른쪽, 왼쪽
    - shift 개수: i개



# 객체(class)들의 배열

- 객체들의 배열: 객체들이 저장된 배열
  - 객체에 대한 참조값만을 저장

```
Car[] cars = new Car[6];
```

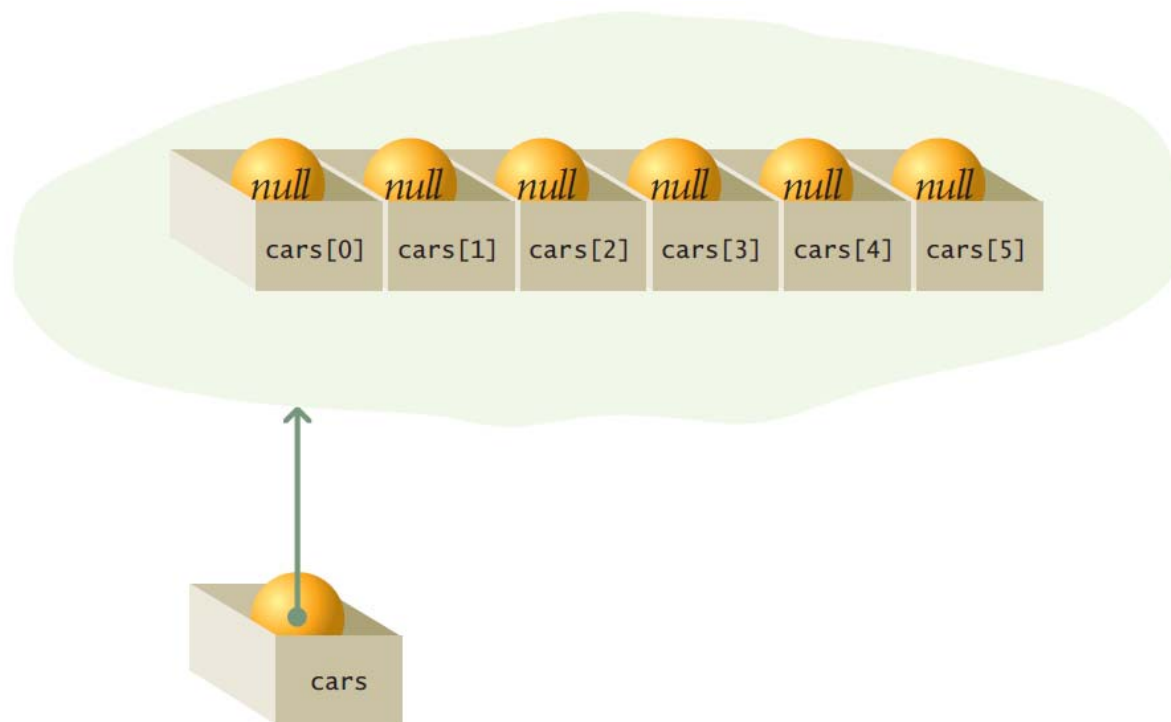


그림10-3. 객체들의 배열

# 객체 배열의 원소: 따로 생성해야 함

```
Car[] cars = new Car[5];
```

```
cars[0] = new Car();
```

```
cars[1] = new Car();
```

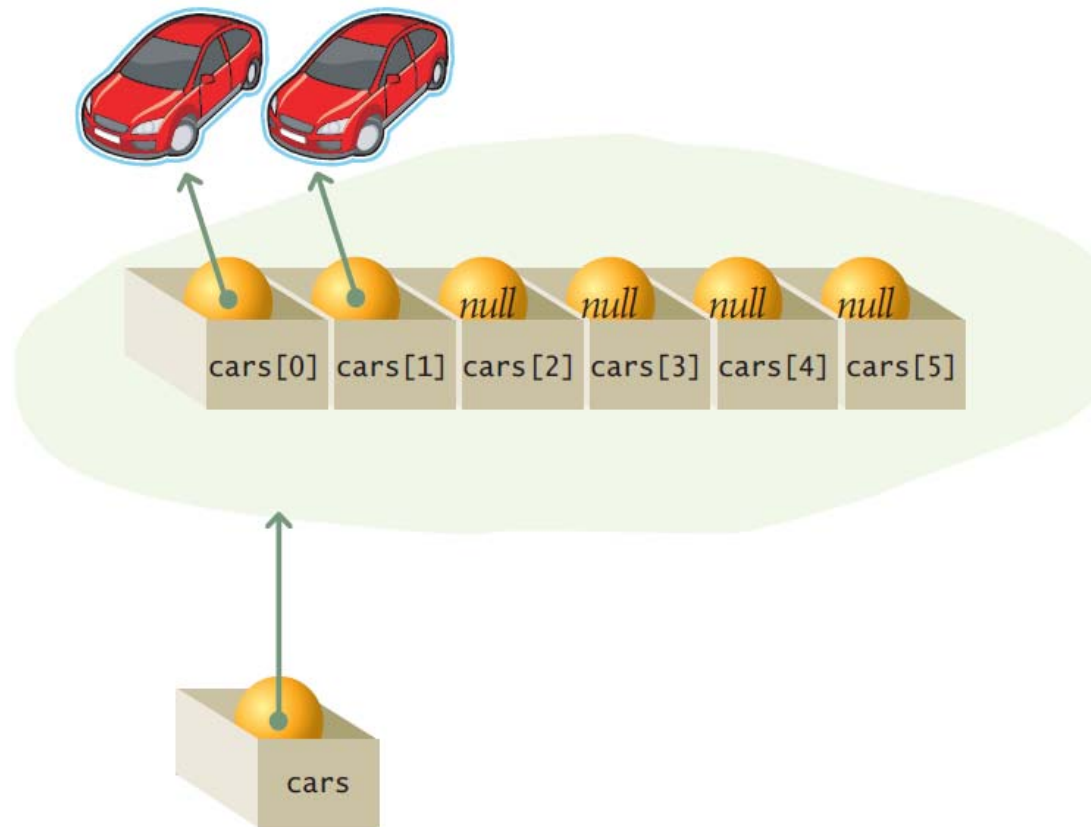


그림10-4. 객체들의 배열은 사실 참조값만을 저장한다.

# 예제

```
for(int i; i < cars.length; i++)  
    cars[i] = new Cars();
```

## CarArrayTest.java

```
01  class Car {  
02      public int speed;           // 속도  
03      public int gear;           // 기어  
04      public String color;       // 색상  
05  
06      public Car() {  
07          speed = 0;  
08          gear = 1;  
09          color = "red";  
10      }  
11      public void speedUp() {     // 속도 증가 메소드  
12          speed += 10;  
13      }  
14      public String toString() {  // 객체의 상태를 문자열로 반환하는 메소드  
15          return "속도: " + speed + " 기어: " + gear + " 색상: " + color;  
16      }  
17  }  
18
```

# 예제

```
19 public class CarArrayTest {
20     public static void main(String[] args) {
21         final int NUM_CARS = 5;
22         Car[] cars = new Car[NUM_CARS];
23         for (int i = 0; i < cars.length; i++)
24             cars[i] = new Car();
25         for (int i = 0; i < cars.length; i++)
26             cars[i].speedUp();
27         for (int i = 0; i < cars.length; i++)
28             System.out.println(cars[i]);
29     }
30 }
```

배열 객체 생성

배열의 각 원소에 Car 객체를 생성하여 대입한다.

## 실행결과

속도: 10 기어: 1 색상: red  
속도: 10 기어: 1 색상: red  
속도: 10 기어: 1 색상: red  
속도: 10 기어: 1 색상: red  
속도: 10 기어: 1 색상: red

# 중간 점검

1. 은행 계좌를 나타내는 클래스 `BankAccount`를 가정하자. `BankAccount`는 잔액만을 가지고 있다. 3개의 원소를 가지는 `BankAccount`의 배열을 생성하고 각 배열 원소를 `BankAccount` 객체로 초기화하라.

```
BankAccount[] ba = new BankAccount[3];  
for(int i=0;i<ba.length;i++) ba[i] = new BankAccount();
```

2. 객체들의 배열을 메소드로 전달하면 무엇이 전달되는가?

참조값

## 2차원 배열

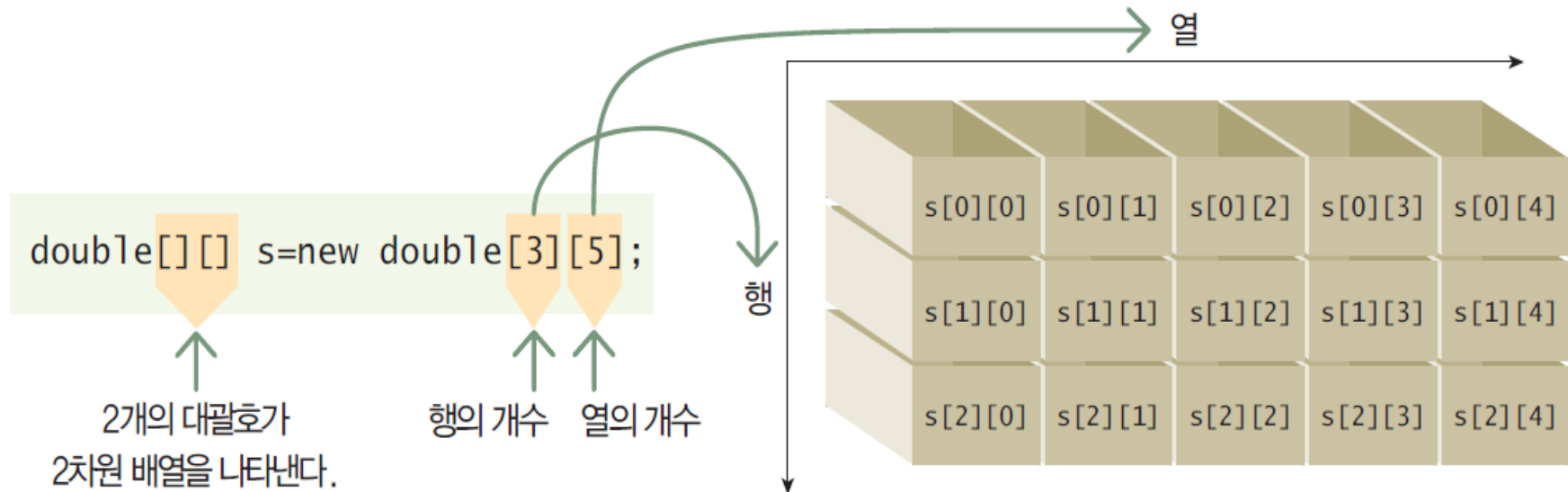


그림10-5. 2차원 배열

예. 학생 3명의 5과목 점수

```
double[][] s = new double[3][5]
```

```
for(int i = 0; i < 3; i++)  
    for(int j = 0; j < 5; j++)  
        System.out.println(s[i][j]);
```

# 2차원 배열의 초기화

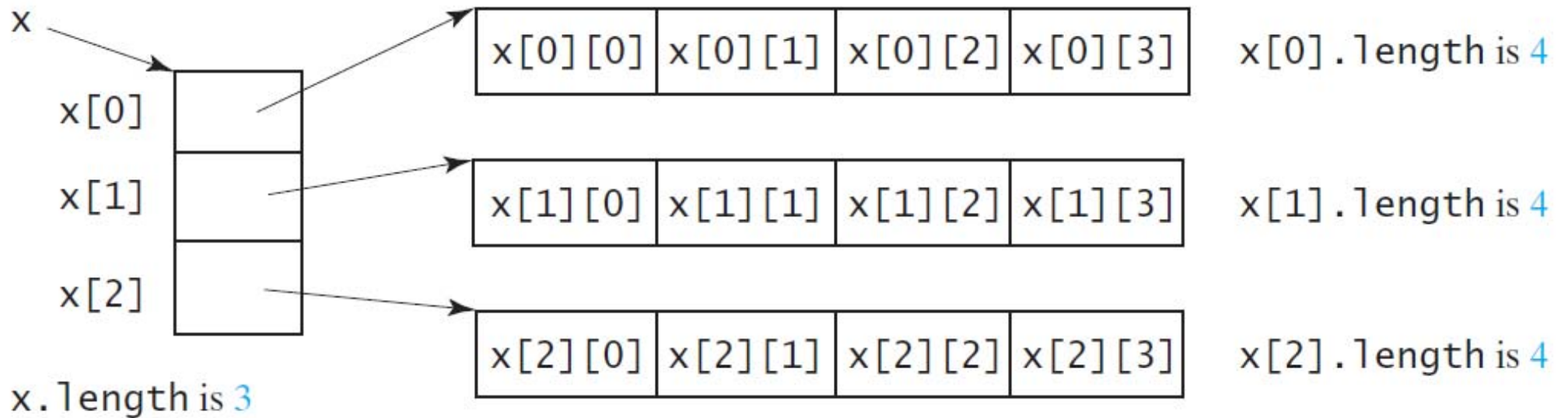
- 2차원 배열의 초기화도 중괄호를 이용한다.
  - 같은 행의 원소를 중괄호로 묶는다.

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

- 1차원 배열의 경우와 마찬가지로 초기화 리스트가 존재하는 경우에는 new 연산자를 사용할 필요가 없다.
- 위의 예제에서 첫 번째 행의 원소는 {10, 20, 30}이고 두 번째 행은 {40, 50, 60}, 세 번째 행은 {70, 80, 90}이다.

# Lengths of Two-dimensional Arrays

```
int[][] x = new int[3][4];
```





# 2차원 배열에서의 length 필드

ArrayTest6.java

```
01  import java.util.Scanner;
02
03  public class ArrayTest6 {
04      public static void main(String[] args) {
05          int[][] array = { { 10, 20, 30, 40 }, { 50, 60, 70, 80 },
06                          { 90, 100, 110, 120 } };
07
08          for (int r = 0; r < array.length; r++) {
09              for (int c = 0; c < array[r].length; c++) {
10                  System.out.println(r + "행" + c + "열:" + array[r][c]);
11              }
12          }
13      }
14  }
```

실행결과

```
0행0열:10
0행1열:20
...
2행2열:110
2행3열:120
```

# Ragged Arrays

The rows may have **different** lengths, called a *ragged array*. For example,

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

```
matrix.length is 5  
matrix[0].length is 5  
matrix[1].length is 4  
matrix[2].length is 3  
matrix[3].length is 2  
matrix[4].length is 1
```

# Processing Two-Dimensional Arrays: 3\*2

See the examples in the text.

1. (Initializing arrays with input values)
2. (Printing arrays)
3. (Summing all elements)
4. (Summing all elements by column)
5. (Which row has the largest sum)
6. (Finding the smallest index of the largest element)
7. (*Random shuffling*)

# 다차원 배열

- 다른 언어와 마찬가지로 자바에서도 얼마든지 다차원 배열을 생성할 수 있다. 예를 들어서 다음 문장은 3차원 배열을 생성한다.

```
double[][][] sales = new double[3][2][12]
```

- 책이 8개의 서가에 저장되어 있는 서점을 가정하자. 각 서가는 최대 100권의 책을 가지고 있다. 이 서점의 책들을 저장할 수 있는 2차원 배열을 생성하라.

```
String [][] books = new String [8][100];
```

- 2차원 배열이 메소드로 전달되면 무엇이 전달되는가?

역시 참조값