6장. 배열

Section 1 배열의 개요
Section 2 배열의 선언과 생성
Section 3 배열의 초기화 및 확장 for문
Section 4 1차원 배열
Section 5 다차원 배열
Section 6 Arrays 클래스와 System 클래스를 이용한 배열의 사용

처음시작하는 JAVA프로그래밍 Essential Course

학습목표

- 배열의 개념을 학습합니다.
- 배열의 선언과 생성을 학습하고, 메모리 구조를 학습합니다.
- 배열을 초기화하는 다양한 방법과 배열을 효율적으로 사용하는 for문을 학습합니다.
- 1차원 배열과 다차원 배열을 예제를 통하여 학습합니다.
- 라이브러리 클래스인 Arrays 클래스와 System 클래스를 이용한 배열 사용 방법을 학습합니다.

- 같은 형의 데이터를 여러 개 사용할 때 많은 변수를 사용하기 보다는 배열을 사용하는 것이 효율적

- 배열 : 같은 형의 데이터를 하나의 자료구조에 저장할 수 있게 만든 것이 배열

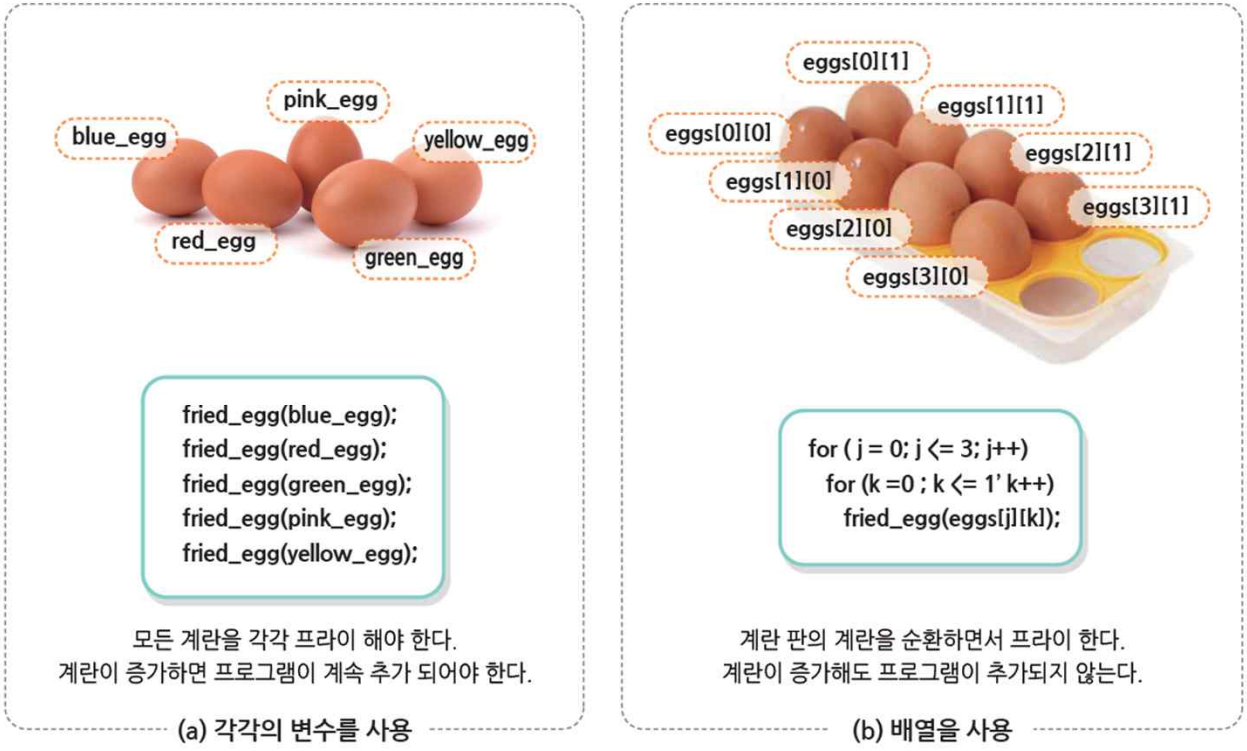


그림 6-1 변수와 배열의 차이

● 배열은 기본 자료형이 아니라 참조 자료형이다

- 배열 각각의 요소는 기본 자료형, 참조 자료형 모두 가능하다.

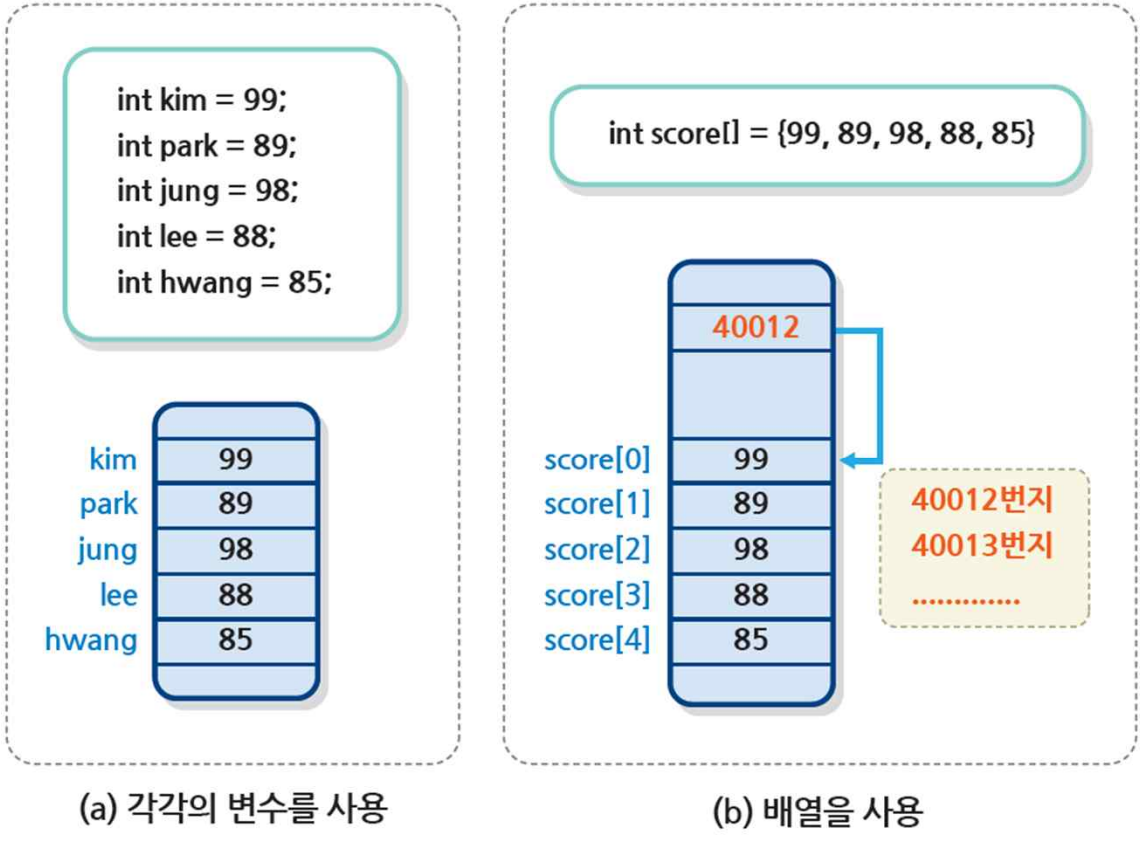


그림 6-2 변수를 사용하는 경우와 배열을 사용하는 경우의 메모리 구조

2 배열의 선언과 생성

- 자바에서 배열을 사용하기 위해서는 배열을 선언하고, 생성하는 과정이 필요

배열의 선언

```
type name[];
type[] name;
```
← 1차원 배열 name 선언. []를 형이나 이름에 붙인다.

```
type[][] name;
type name[][];
type[] name[];
```
← 2차원 배열 name 선언. []를 하나씩 나누어 붙일 수 있다.

배열의 생성

```
name = new type[size];
```
← size 크기의 1차원 배열 생성

```
name = new type[size][size];
```
← size 크기의 2차원 배열 생성

배열의 선언과 생성 : 한 문장으로 선언과 생성 가능

```
type[] name = new type[size];
type name[] = new type[size];
type[][] name = new type[size][size];
type name[][] = new type[size][size];
type[] name[] = new type[size][size];
```

2 배열의 선언과 생성

배열의 선언과 생성의 예

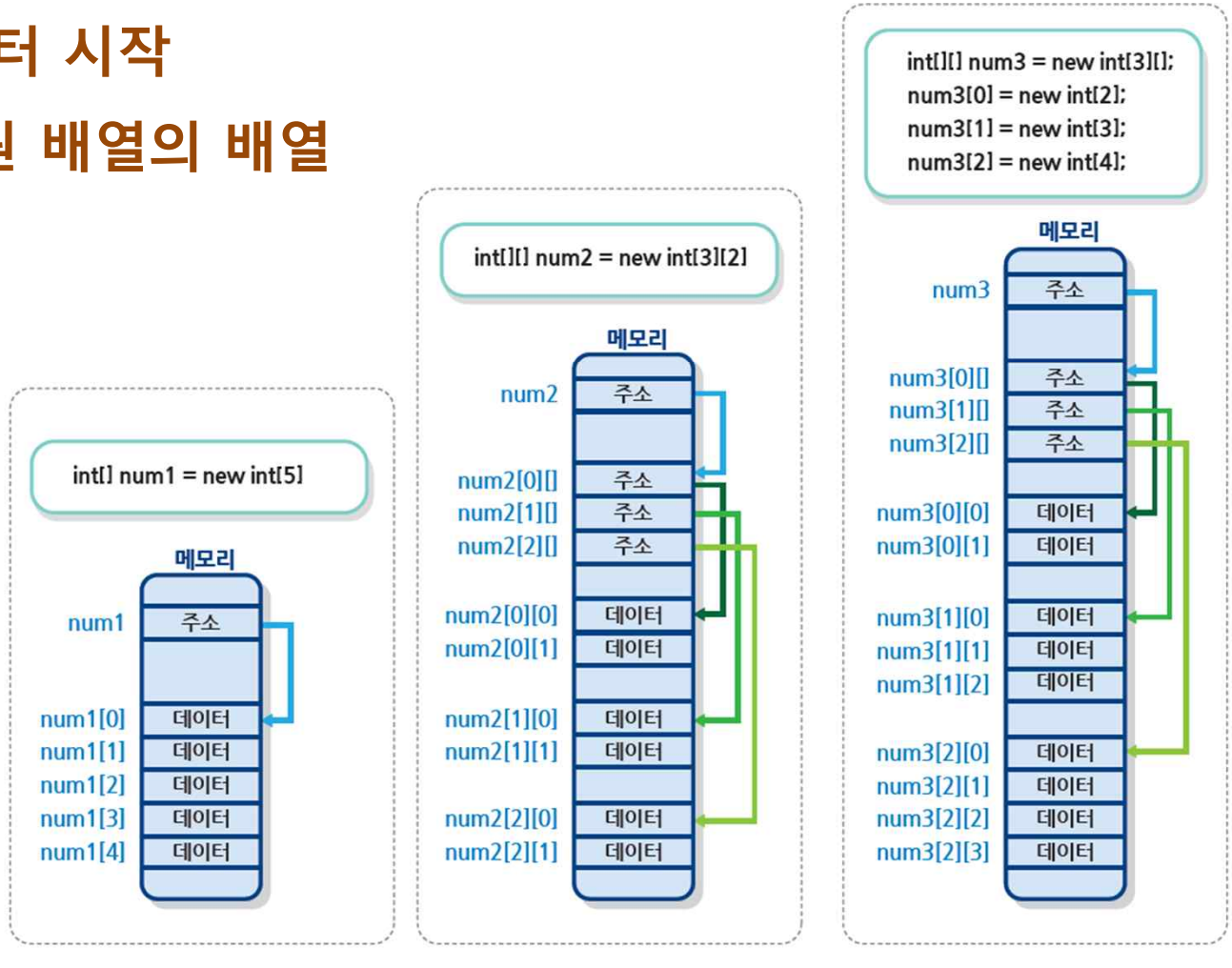
```
int[] id;
id = new int[3];          ← 3개의 int 요소를 가진 배열 생성
또는
int[] id = new int[3];
String[] student_name ;
student_name = new String[3];   ← 3개의 문자열 요소를 가진 배열 생성
또는
String[] student_name = new String[3];
int[][] id_and_score = new int[5][2];   ← 5행과 2열을 가진 정수 2차원 배열 생성
String add_and_name[][] = new String[10][10];   ← 10행과 10열을 가진 문자열 배열 생성
```

배열 요소의 사용 : 배열 이름과 첨자를 이용하여 접근

```
int[] id = new int[3];          ← 3개의 요소를 가진 int 배열
int sum = int[0] + int[1];      ← 첨자는 0부터 시작
int[][] stnum = new int[3][3];
stnum[0][2] = stnum[0][0]+stnum[0][1];   ← 2차원 배열 첨자 사용
```

2 배열의 선언과 생성

- 배열의 첨자는 0부터 시작
- 2차원 배열은 1차원 배열의 배열



(a) 1차원 배열과 메모리 (b) 2차원 배열과 메모리 (c) 배열의 요소 수가 다른 2차원 배열과 메모리

그림 6-3 배열과 메모리 구조

3-1 배열의 초기화와 요소의 수(length)

- **배열의 초기화 : 처음 생성된 배열에 데이터를 저장하는 과정**
- **변수와 달리 배열은 초기화 하지 않아도 사용이 가능(오류 발생 안함)**
  - 배열이 초기화 되지 않을 경우에는 묵시적인 값으로 자동 설정된다

3-1 배열의 초기화와 요소의 수(length)

배열의 생성과 초기화

```
int[] id = new int[5];
id[0] = 201195041;
id[1] = 201195042;
id[2] = 201195043;
id[3] = 201195044;
id[4] = 201195045;
```

1차원 배열 선언과 생성

초기화. 각 요소에 값을 하나씩 저장하여 초기화

또는

배열의 선언과 생성, 초기화 과정을 한 문장으로 작성

```
int id[] = {201195041, 201195042, 201195043, 201195044, 201195045}
String[][] name_addr = new String[3][2]
name_addr[0][0] = "kim";
name_addr[0][1] = "seoul";
name_addr[1][0] = "park";
name_addr[1][1] = "busan";
name_addr[2][0] = "lee";
name_addr[2][1] = "inchon";
```

2차원 배열 선언과 생성

초기화. 각 요소에 값을 하나씩 저장하여 초기화

또는

배열의 선언과 생성, 초기화 과정을 한 문장으로 작성

```
String[][] name_addr = {{"kim","seoul"},{"park","busan"},{"lee","inchon"}};
```

3-1 배열의 초기화와 요소의 수(length)

- **배열이 초기화 되지 않을 경우 가지는 묵시적인 값**

형	묵시적 값
byte, short, int, long	0
float, double	0.0
char	공백 문자
boolean	거짓(false)
참조 자료형	null

3-1 배열의 초기화와 요소의 수(length)

- **배열의 길이를 나타내는 length 속성**

```
int a[] = { 10, 20, 30, 40, 50 };
System.out.println(a.length); <-------- 5를 출력

int b[][] = {{10,20,30},{40,50,60,70}};
System.out.println(b.length); <-------- 배열 b의 행의 길이 2를 출력
System.out.println(b[0].length); <-------- 배열 b 첫 번째 행의 길이 3을 출력
System.out.println(b[1].length); <-------- 배열 b 두 번째 행의 길이 4를 출력
```

3-1 배열의 초기화와 요소의 수(length)

- 예제 6.1

예제 6.1 ArraysTest1.java

```
01: import java.util.Scanner;
02: public class ArraysTest1 {
03:     public static void main(String args[])
04:     {
05:         Scanner stdin = new Scanner(System.in);
06:         int i;
07:         double sum=0.0, avg;;
08:         double dnum[] = new double[5];
09:         System.out.println("dnum 배열의 길이 : " + dnum.length);
10:         System.out.print("초기화 하지 않은 dnum[]의 값: ");
11:         for (i=0; i < dnum.length ; i++)
12:             System.out.print(dnum[i]+" ");
13:         System.out.println();
14:
15:         for (i=0; i < dnum.length ; i++){
16:             System.out.print("dnum["+(i)+"] 번째 데이터 입력 : ");
17:             dnum[i] = stdin.nextDouble();
18:         }
19:         for (i=0; i < dnum.length ; i++)
20:             sum = sum + dnum[i];
21:         System.out.println("배열의 합은 " + sum + "입니다");
22:         avg = sum/dnum.length;
23:         System.out.println("배열 값의 평균은  " + avg + "입니다");
24:
```

1차원 double 배열 선언

length를 이용하여 배열의 길이 출력

for문의 조건으로 length 속성 사용. 초기화하지 않은 배열값 출력

반복문을 이용하여 배열 초기화

배열 요소의 합을 구한다.

실행 결과

```
dnum 배열의 길이 : 5
초기화하지 않은 dnum[]의 값: 0.0 0.0 0.0 0.0 0.0
dnum[0] 번째 데이터 입력 : 101.2
dnum[1] 번째 데이터 입력 : 210.3
dnum[2] 번째 데이터 입력 : 330.4
dnum[3] 번째 데이터 입력 : 460.5
dnum[4] 번째 데이터 입력 : 600.6
배열의 합은 1703.0입니다
배열값의 평균은  340.6입니다
```

3-2 배열 처리를 위한 확장된 for문

- **자바는 배열의 처리를 편리하게 제공하기 위한 확장된 for문 제공**
  - 배열의 요소를 순차적으로 처리하는 간결한 구문 제공

확장된 for문의 형식

```
for ( type 변수명 : 배열 이름 )
```

- 처리할 배열의 이름
- 배열의 값을 저장할 변수명
- 배열값을 저장할 변수의 형

배열 이름으로 지정된 배열의 첫 번째 요소부터 마지막 요소까지를 변수명에 배정하여 반복 처리를 수행.

```
sum=0;
int[] inum = { 10, 20, 30, 40, 50 };
for ( int x : inum )
    sum = sum + x;
System.out.println(sum);
```

← inum 배열의 첫 번째 요소부터 마지막 요소까지를 차례로 변수 x에 배정하여 반복을 처리한다.

3-2 배열 처리를 위한 확장된 for문

● 예제 6.2

예제 6.2 ArraysTest2.java

```
01: public class ArraysTest2 {
02:     public static void main(String args[])
03:     {
04:         int score[] = {88,97,53,62,92,68,82};
05:         int max=score[0];
06:         for (int i : score)
07:         // 기존 for문 : for (int i=1; i < score.length ; i=i+1)
08:         {
09:             if ( i > max )
10:             // 기존의 for 문을 사용할 때의 if문 : if ( score[i] > max )
11:                 max = i;
12:                 // max = score[i];
13:         }
14:         System.out.println("배열 요소의 최대값은  " + max + "입니다");
15:     }
16: }
```

확장된 for문 사용. 변수 x에는 인덱스가 아닌 배열 요소의 값이 순차적으로 배정

기존의 for문 형태. 변수 i는 배열의 인덱스

i 값을 max와 비교

배열에서 인덱스 i번째의 값과 max 비교

실행 결과

```
배열 요소의 최대값은  97입니다
```

● 예제 6.3

예제 6.3 OneArraysTest1.java

```
01: import java.util.Scanner;
02: public class OneArraysTest1 {
03:     public static void main(String args[])
04:     {
05:         int inum[] = {8,7,3,6,9,6,8,7,0,4,1,2};
06:         Scanner stdin = new Scanner(System.in);
07:         System.out.print("찾고 싶은 숫자 입력 : ");
08:         int key = stdin.nextInt();
09:         int count = 0;
10:         for (int i = 0 ; i < inum.length ; i++)
11:         {
12:             if ( inum[i] == key ) {
13:                 count++;
14:                 System.out.println((i+1) + "번째 데이터와 일치");
15:             }
16:         }
17:         if (count == 0)
18:             System.out.println(key + "값은 배열에 없습니다");
19:         else
20:             System.out.println(key+ "값은 배열에 "+count+"개 있습니다");
21:     }
22: }
```

횟수를 저장하기 위한 변수 선언

위치를 알아내야 하기 때문에 확장된 for문 사용 불가

값이 지정된 값과 일치하면 count 값을 증가하고 출력

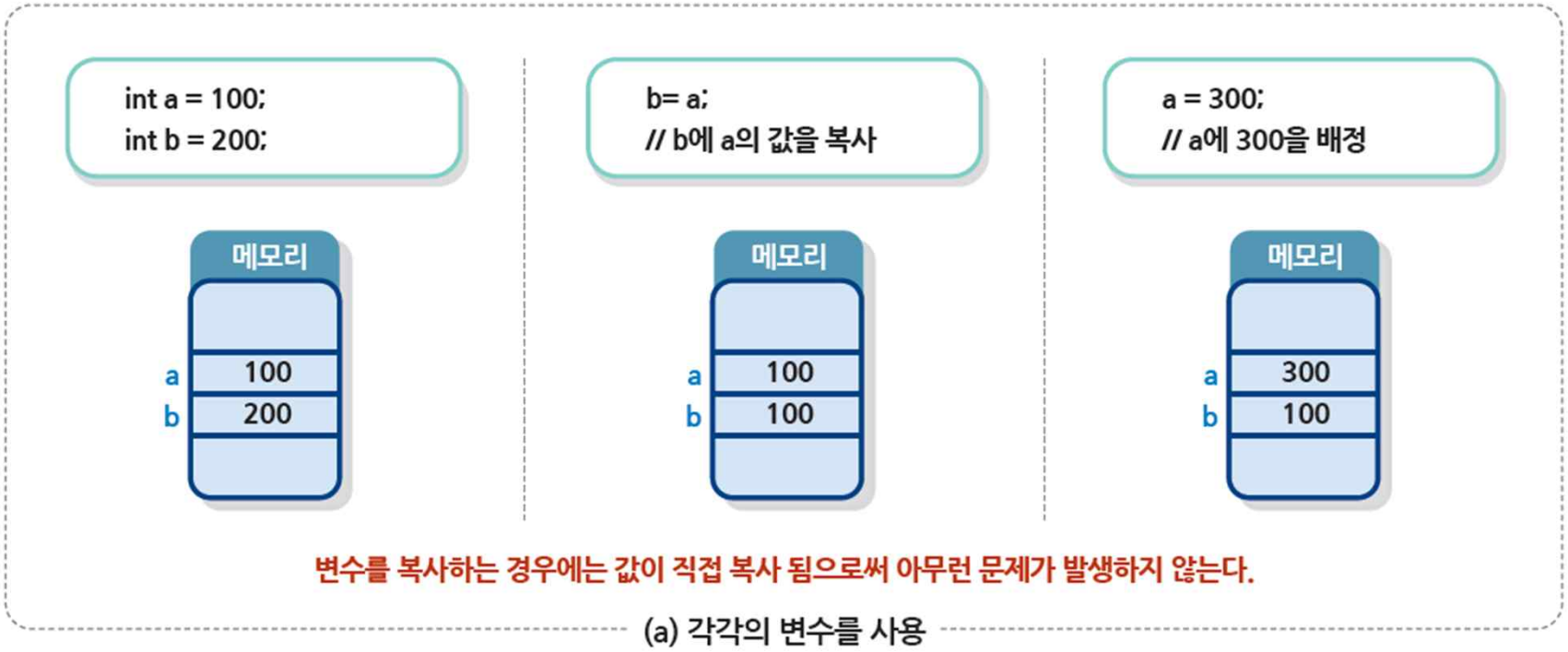
값이 없을 경우 값이 없음을 출력

값이 있는 경우 횟수를 출력

실행 결과 3번 실행

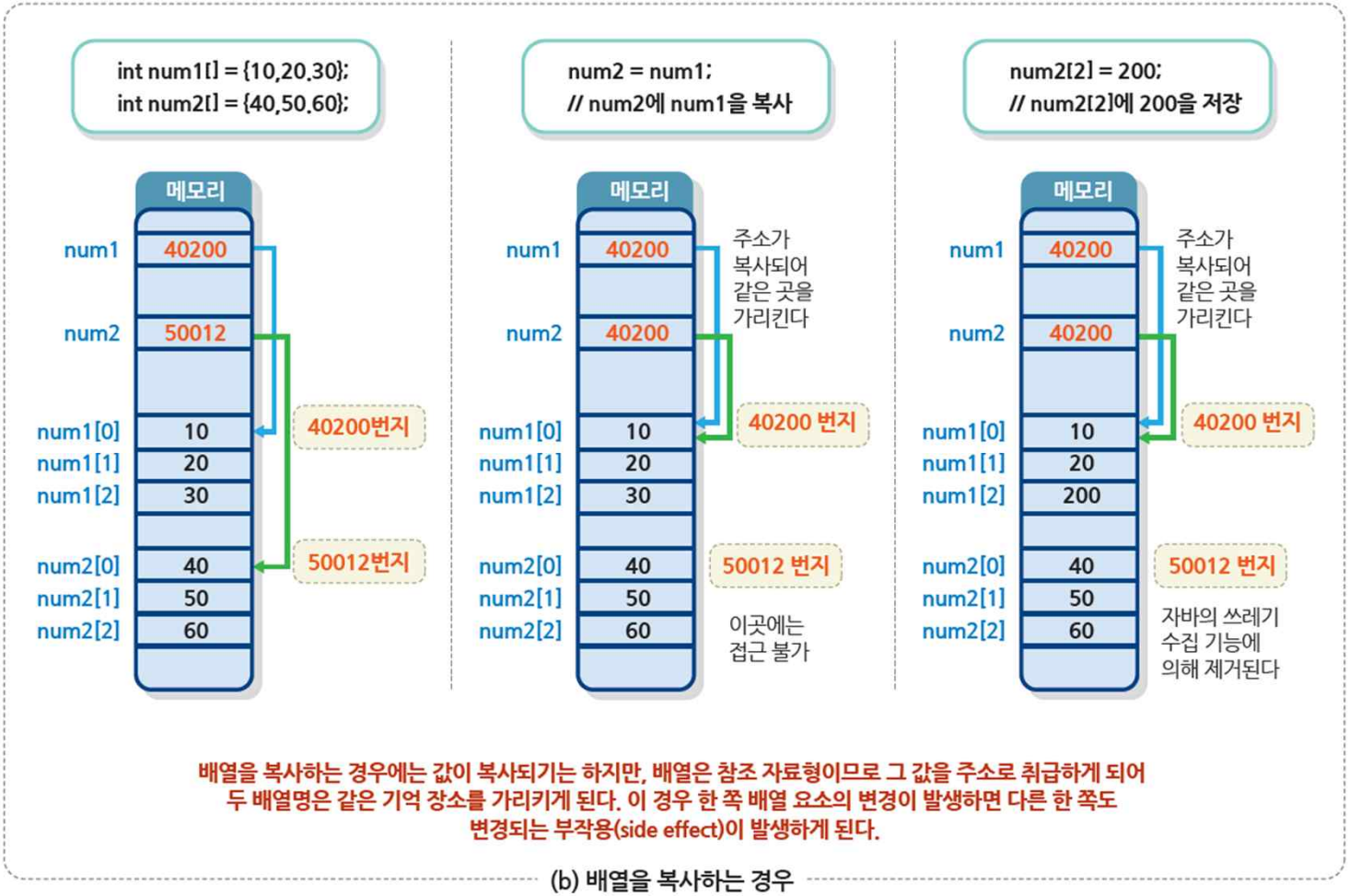
```
찾고 싶은 숫자 입력 : 0
9번째 데이터와 일치
0값은 배열에 1개 있습니다
찾고 싶은 숫자 입력 : 8
1번째 데이터와 일치
7번째 데이터와 일치
8값은 배열에 2개 있습니다
찾고 싶은 숫자 입력 : 5
5값은 배열에 없습니다
```

● 예제 6.4 : 변수의 복사와 배열의 복사



(a) 각각의 변수를 사용

● 예제 6.4 : 변수의 복사와 배열의 복사



(b) 배열을 복사하는 경우

● 예제 6.4

예제 6.4 CopyArraysTest1.java

```
01: public class CopyArraysTest1 {
02:     public static void main(String args[])
03:     {
04:         int num1[] = {10,20,30};
05:         int num2[] = {40,50,60};
06:         num2 = num1; <---------- 배열명을 이용하여 복사
07:         num2[2] = 200; <---------- num2[2] 번째 요소의 값을 200으로 변경
```

● 예제 6.4

```
08:         System.out.print("num1 배열의 값 : ");
09:         for (int i : num1)
10:             System.out.print(i+" ");
11:         System.out.print("\nnum2 배열의 값 : ");
12:         for (int i : num2)
13:             System.out.print(i+" ");
14:         int num3[] = {100,200,300};
15:         int num4[] = {400,500,600};
16:         for (int i = 0 ; i < num3.length ; i ++)
17:             num4[i] = num3[i];
18:         num4[2]=999;
19:         System.out.print("\nnum3 배열의 값 : ");
20:         for (int i : num3)
21:             System.out.print(i+" ");
22:         System.out.print("\nnum4 배열의 값 : ");
23:         for (int i : num4)
24:             System.out.print(i+" ");
25:     }
26: }
```

num1의 값을 출력. num1[2] 값이 200으로 출력

num2의 값을 출력

num3, num4를 생성

배열 요소의 값들을 각각 복사

num4[2] 요소를 999로 변경

num3의 값을 출력. num3[2]의 값은 그대로 300 출력

num4의 값 출력. num4[2]는 999 출력

실행 결과 3번 실행

```
num1 배열의 값 : 10 20 200  <— num2[2] 값의 변경에 따라 같이 변함
num2 배열의 값 : 10 20 200
num3 배열의 값 : 100 200 300 <— num4[2] 값이 999로 변경되어도 변하지 않음
num4 배열의 값 : 100 200 999
```

● 예제 6.5

예제 6.5 OneArraysTest2.java

```
01: public class OneArraysTest2 {
02:     public static void main(String args[])
03:     {
04:         String slist[] = {"seoul", "daejeon", "daegu", "kwangju", "inchon", "jeju", "busan"};
05:         System.out.print("원래의 배열 : ");
06:         for (String s : slist)
07:             System.out.print(s + " ");
08:         System.out.println();
09:         for (int i = 0 ; i < slist.length / 2 ; i++)
10:         {
11:             String temp = slist[i];
12:             slist[i] = slist[slist.length - i - 1];
13:             slist[slist.length - i - 1] = temp;
14:         }
15:         System.out.print("역순으로 재배치된 배열 : ");
16:         for (String s : slist)
17:             System.out.print(s + " ");
18:     }
19: }
```

원래의 배열을 출력

배열 길이의 반에 해당되는 횟수만큼 교환. 개수가 홀수인 경우는 가운데는 바꾸지 않는다.

역순으로 재배치된 배열 출력

실행 결과

```
원래의 배열 : seoul daejeon daegu kwangju inchon jeju busan
역순으로 재배치된 배열 : busan jeju inchon kwangju daegu daejeon seoul
```

● 예제 6.6

예제 6.6 MultiArraysTest1.java

```
01: import java.util.Random;
02: public class MultiArraysTest1 {
03:     public static void main(String args[])
04:     {
05:         Random ran = new Random();
06:         int[][] score = new int[4][4];
07:         for(int i=0 ; i< 4 ; i++)
08:             for(int j=0; j<4 ; j++)
09:                 score[i][j] = ran.nextInt(10);
10:
11:         for (int k=0 ; k < 4 ; k++) {
12:             int sum=0;
13:             for (int value : score[k]) {
14:                 System.out.print(value + " ");
15:                 sum = sum + value;
16:             }
17:            System.out.println("의 합계는 "+sum);
18:         }
19:     }
20: }
```

Random 클래스를 import 시킨다.

Random 객체를 생성

2차원 배열 score 생성

score 배열을 0-9 사이의 정수 난수로 초기화

각 행의 합계를 구하기 위해 sum 변수 0으로 설정

각 행을 출력

각 행의 합계를 출력

실행 결과

```
2 2 7 6 의 합계는 17
2 5 0 6 의 합계는 13
4 6 8 6 의 합계는 24
9 5 9 0 의 합계는 23
```

● 예제 6.7

예제 6.7 MultiArraysTest2.java

```
01: public class MultiArraysTest2 {
02:     public static void main(String args[])
03:     {
04:         int twoD[][] = new int[4][];   <-- 2차원 배열을 선언
05:         twoD[0] = new int[1];          <-- 각 행에 해당하는 1차원 배열을 생성
06:         twoD[1] = new int[2];
07:         twoD[2] = new int[3];
08:         twoD[3] = new int[4];
```

● 예제 6.7

```
09:         System.out.println("2차원 배열에서 행의 길이는 : "+twoD.length);
10:         System.out.println("첫 번째 행의 요소 수는 : "+twoD[0].length);
11:         System.out.println("두 번째 행의 요소 수는 : "+twoD[1].length);
12:         System.out.println("세 번째 행의 요소 수는 : "+twoD[2].length);
13:         System.out.println("네 번째 행의 요소 수는 : "+twoD[3].length);
14:         int i, j, k = 0;
15:         for(i=0 ; i < twoD.length ; i++)
16:             for(j=0 ; j< twoD[i].length ; j++)
17:             {
18:                 twoD[i][j] = k;
19:                 k++;
20:             }
21:         for(i=0 ; i < twoD.length ; i++)
22:         {
23:             for(int val : twoD[i])
24:                 System.out.print(val + " ");
25:             System.out.println();
26:         }
27:     }
28: }
```

배열의 길이를 출력

배열의 요소를 초기화(0부터 1씩 증가시켜 가며 저장)

배열을 출력

실행 결과

```
2차원 배열에서 행의 길이는 : 4
첫 번째 행의 요소 수는 : 1
두 번째 행의 요소 수는 : 2
세 번째 행의 요소 수는 : 3
네 번째 행의 요소 수는 : 4
0
1 2
3 4 5
6 7 8 9
```

예제 6.8

예제 6.8 MultiArraysTest3.java

```
01: public class MultiArraysTest3 {
02:     public static void main(String args[])
03:     {
04:         int[][][] threeD = new int[3][4][5];
05:         int i, j, k, count=11;
06:         for( i=0 ; i < threeD.length ; i++)
07:             for( j=0; j < threeD[i].length ; j++)
08:                 for( k=0; k < threeD[i][j].length ; k++)
09:                 {
10:                     threeD[i][j][k] = count;
11:                     count++;
12:                 }
```

3차원 배열을 선언

3차원 배열의 초기화

● 예제 6.8

```
13:         for(i=0; i<threeD.length; i++)
14:         {
15:             System.out.println((i+1) + "번째 2 차원 배열 ");
16:             for(j=0; j<threeD[i].length; j++)
17:             {
18:                 for(int val : threeD[i][j])
19:                     System.out.print(val + " ");
20:                 System.out.println();
21:             }
22:             System.out.println();
23:         }
24:     }
25: }
```

3차원 배열의

실행 결과

```
1번째 2차원 배열
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
26 27 28 29 30

2번째 2차원 배열
31 32 33 34 35
36 37 38 39 40
41 42 43 44 45
46 47 48 49 50

3번째 2차원 배열
51 52 53 54 55
56 57 58 59 60
61 62 63 64 65
66 67 68 69 70
```

- 자바는 라이브러리 클래스로 Arrays 클래스와 배열을 복사하기 위한 메소드(arraycopy())를 제공하는 System 클래스를 제공한다
  - Arrays 클래스의 주요 메소드

메소드	설명
static int binarySearch(int[] a, int key)	배열 a에서 key로 지정된 값을 찾아 반환. boolean을 제외한 7가지 기본 자료형과 참조 자료형 사용 가능. 이 메소드는 배열 요소들이 정렬된 상태에서 사용되어야 한다. 배열에서 key로 지정된 값의 위치를 반환한다.
static boolean equals(int[] a, int[] b)	배열 a와 b가 같은지를 비교하여 결과를 반환. 8개의 기본 자료형과 참조 자료형에서도 사용 가능
static void fill(int[] a, int value)	배열 a의 모든 요소를 value값으로 설정한다.
static void fill (int[] a, int from, int to, int value)	배열 a의 from부터 to-1까지를 value값으로 설정한다(인덱스 값 기준). 8개의 기본 자료형과 참조 자료형에서도 사용 가능
static void sort(int[] a)	배열 a의 요소들을 정렬. boolean을 제외한 7가지 기본 자료형과 참조 자료형에서도 사용 가능.
static void sort(int[] a, int from, int to)	배열 a의 from부터 to까지를 정렬. boolean을 제외한 7가지 기본 자료형과 참조 자료형에서도 사용 가능.
static String toString(int[] a)	배열 a의 요소들을 문자열로 반환한다. 이 메소드는 모든 자료형에 적용 가능.

Arrays 클래스의 사용 예

```
int[] a = new int[10];
Arrays.fill(a,1);                          ← 배열 a의 모든 요소를 1로 채운다.
Arrays.fill(a,1,5,10);                     ← a[1]부터 a[4]까지의 값을 10으로 채운다.
System.out.println(Arrays.toString(a));    ← 배열 a를 문자열로 출력

int[] b = { 3, 7, 1, 0, 8, 9 };
Arrays.sort(b);                            ← 배열 b의 요소들을 오름차순으로 정렬
System.out.println(Arrays.toString(b));    ← 정렬된 결과 출력
System.out.println(Arrays.binarySearch(b, 7));  ← 7의 인덱스 값 3 출력

int[] c = {1, 2, 3};
int[] d = {1, 2, 3};
int[] e = {4, 5, 6};
System.out.println(Arrays.equals(c,d));    ← true 출력
System.out.println(Arrays.equals(c,e));    ← false 출력
```

System 클래스의 arrayscopy() 메소드와 사용 예

메소드	설명
static void arraycopy(int[] a, int s_index, int[] b, int t_index, int num)	배열 a에서 s_index로 지정된 요소부터 배열 b의 t_index로 지정된 위치로 num 개의 요소를 복사한다.

```
int[] a = {1, 2, 3};
int[] b = {4, 5, 6};
int[] c = new int[3];
System.arraycopy(a,0,b,0,3); <-------- 배열 a의 처음부터 3개의 요소를 배열 b에 복사
System.out.println(Arrays.toString(b)); <-------- [1, 2, 3] 출력
System.arraycopy(a,0,c,0,3); <-------- 배열 a의 처음부터 3개의 요소를 배열 c에 복사
System.out.println(Arrays.toString(c)); <-------- [1, 2, 3] 출력
```

6. Arrays 클래스와 System 클래스를 이용한 배열의 사용

● 예제 6.9

예제 6.9 ArraysCMethodTest1.java

```
01: import java.util.Arrays;
02: public class ArraysCMethodTest1 {
03:     public static void main(String[] args) {
04:         int[] int1 = {9,1,7,3,5,4,6,2,8,0};
05:         System.out.println("초기배열 : " + Arrays.toString(int1));
06:         Arrays.fill(int1, 3, 5, 33);
07:             System.out.println("fill() 수행 후 : " + Arrays.toString(int1));
08:             Arrays.sort(int1);
09:             System.out.println("sort() 수행 후 : " + Arrays.toString(int1));
10:             System.out.println("33은 배열의 " + Arrays.binarySearch(int1,33) +
    "번째 요소");
11:             int[] int2 = {5,4,3,2,1};
12:             System.out.println("두 번째 배열 : " + Arrays.toString(int2));
13:             System.out.println("두 개의 배열이 같은가? " + Arrays.equals(int1,
    int2));
14:           int[] int3 = new int[5];
15:             System.arraycopy(int2, 0, int3, 0, 5);
16:             System.out.println("복사된 배열 : " + Arrays.toString(int3));
17:     }
18: }
```

배열의 요소를 출력

int1의 int[3]과 int[4]를 33으로 바꾼다.

int1 배열을 오름차순으로 정렬

정렬된 int1 배열에서 33의 위치를 찾는다. 2진 탐색.

int1과 int2가 같은가? false 출력

5개의 요소를 가진 빈 배열을 선언

배열을 복사

실행 결과

```
초기 배열 : [9, 1, 7, 3, 5, 4, 6, 2, 8, 0]
fill() 수행 후 : [9, 1, 7, 33, 33, 4, 6, 2, 8, 0]
sort() 수행 후 : [0, 1, 2, 4, 6, 7, 8, 9, 33, 33]
33은 배열의 8번째 요소
두 번째 배열 : [5, 4, 3, 2, 1]
두 개의 배열이 같은가? false
복사된 배열 : [5, 4, 3, 2, 1]
```

● 예제 6.10

예제 6.10 ArraysCMethodTest2.java

```
01: import java.util.Arrays;
02: public class ArraysCMethodTest2 {
03:     public static void main(String args[]) {
04:         String[] array1 = {"IMF", "제주도", "자바도사", "한글나라", "Computer",
    "모카", "JAVA", "인터넷탐색", "초롱초롱", "come", "바람", "스크립터",
    "군고구마", "도서", "their","country" }; <-- 16개의 문자열 요소를 가진 배열 생성
05:         System.out.println("═══ 정렬 전 데이터 ═══");
06:         System.out.println(Arrays.toString(array1)); <-- array1 배열 출력
07:         Arrays.sort(array1); <-- array1 배열을 오름차순으로 정렬
08:         System.out.println("═══ 정렬 후 데이터 ═══");
09:         System.out.println(Arrays.toString(array1)); <-- 정렬된 array1 배열 요소 출력
```

● 예제 6.10

```
10:         System.out.println("군고구마는 배열의 " + Arrays.binarySearch(array1,"
    군고구마") + "번째 요소");
11:         String[] array2 = array1;
12:         System.out.println("array1과 array2가 같은가? : "
    +Arrays.equals(array1,array2));
13:         String[] array3 = new String[20];
14:         System.arraycopy(array2, 0, array3, 0, array2.length);
15:         System.out.println("array3 배열 : " + Arrays.toString(array3));
16:     }
17: }
```

"군고구마"가 배열의 몇 번째 요소인지 출력

array2에 array1을 대입(같은 배열을 가리킨다)

두 배열이 같은가? true 출력

20개의 요소를 가진 문자열 배열 생성

array3 배열을 출력. 마지막 4개의 요소는 null

실행 결과

```
═══ 정렬 전 데이터 ═══
[IMF, 제주도, 자바도사, 한글나라, Computer, 모카, JAVA, 인터넷탐색, 초롱초롱, come, 바람, 스크립터, 군
고구마, 도서, their, country]
═══ 정렬 후 데이터 ═══
[Computer, IMF, JAVA, come, country, their, 군고구마, 도서, 모카, 바람, 스크립터, 인터넷탐색, 자바도
사, 제주도, 초롱초롱, 한글나라]
군고구마는 배열의 6번째 요소
array1과 array2가 같은가? : true
array3 배열 : [Computer, IMF, JAVA, come, country, their, 군고구마, 도서, 모카, 바람, 스크립터, 인터
넷탐색, 자바도사, 제주도, 초롱초롱, 한글나라, null, null, null, null]
```

▪ 학습 정리

● 배열의 개요

① 배열은 동일한 형의 다수 개의 데이터를 저장할 수 있는 자료 구조입니다.

② 배열은 참조 자료형으로서 배열명은 데이터가 저장된 메모리의 주소를 가집니다.

● 배열의 선언과 생성

① 배열은 사용하기 전에 반드시 선언되어야 합니다.

② 2차원 배열에서 배열의 크기를 행마다 다르게 설정할 수 있습니다.

③ 배열의 첨자는 0부터 시작됩니다.

▪ 학습 정리

- **배열의 초기화 및 확장 for문**

  ① 배열은 다양한 방법으로 초기화 될 수 있습니다. 배열의 선언과 생성, 초기화를 하나의 문장으로 지정할 수 있습니다.

  ② 배열은 초기화 되지 않으면 묵시적인 값으로 초기화가 자동으로 이루어집니다.

  ③ 확장된 for문을 사용하여 배열의 요소를 간결하게 처리할 수 있습니다.

- **Arrays 클래스와 System 클래스를 이용한 배열의 사용 1차원 배열, 다차원 배열**

  ① 라이브러리 클래스인 Arrays 클래스와 System 클래스의 클래스 메소드를 이용하여 배열을 편리하게 사용할 수 있습니다.