

자바 프로그래밍

제4장 배열

이번 장에서 학습할 내용

학습목차

01 배열의 필요성

02 배열의 선언과 사용

LAB 성적 평균 계산하기

LAB 문자열 배열

LAB 최대값과 최소값 구하기

LAB 특정한 값 찾기

LAB 주사위 던지기

LAB 극장 예약 시스템

03 고급 배열

04 배열 정렬

05 2차원 배열

LAB TIC-TAC-TOE 게임

LAB 지뢰찾기 게임

LAB 랜덤 워크

06 ArrayList

07 래그드 배열

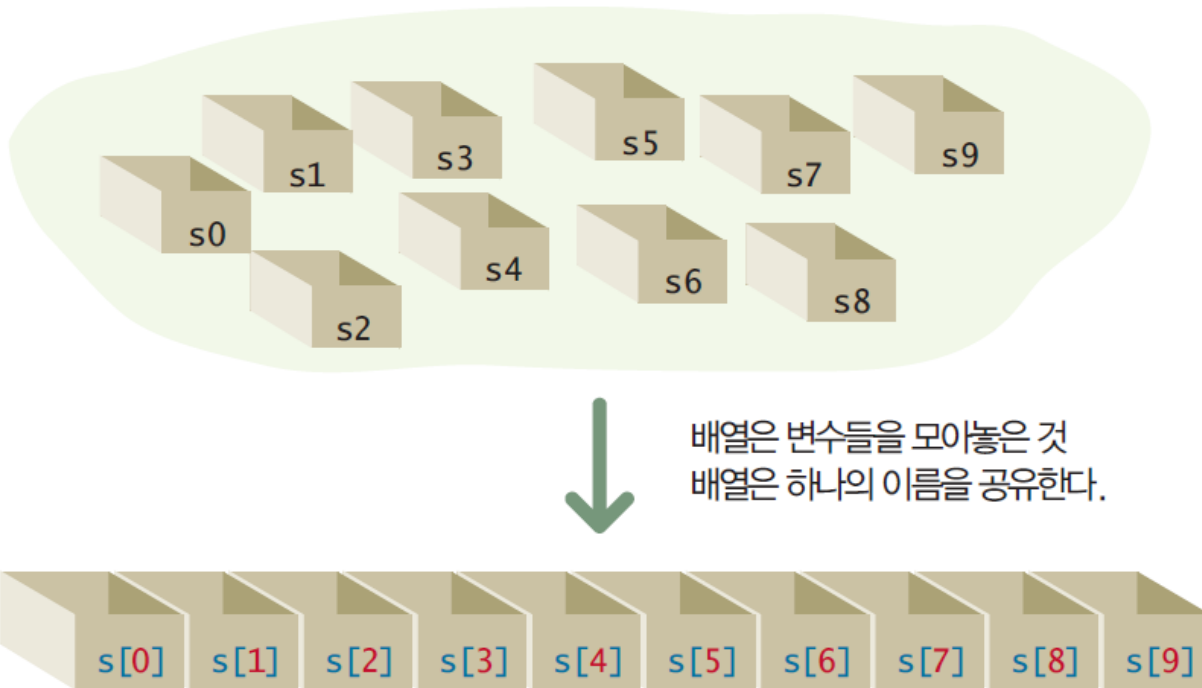
변수 1000개가 필요한데
어떻게 해야 하나요?

배열을 이용하면 한 번에
전부 만들 수 있습니다. 메모리만
충분하다면 더 큰 배열도
얼마든지 가능합니다.



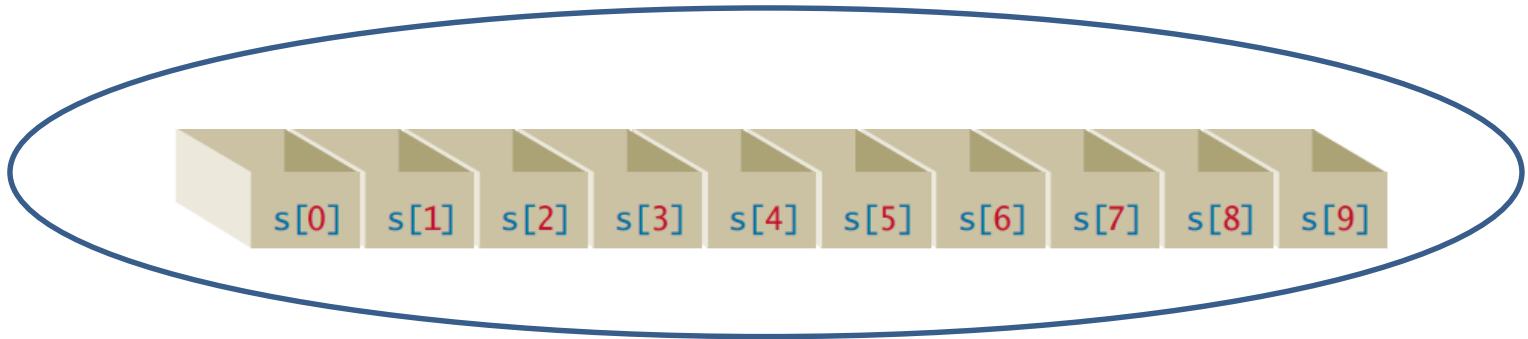
배열의 개념

- 배열(array): 같은 타입의 변수들의 모임



배열의 개념

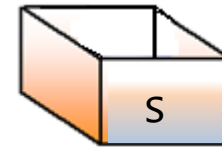
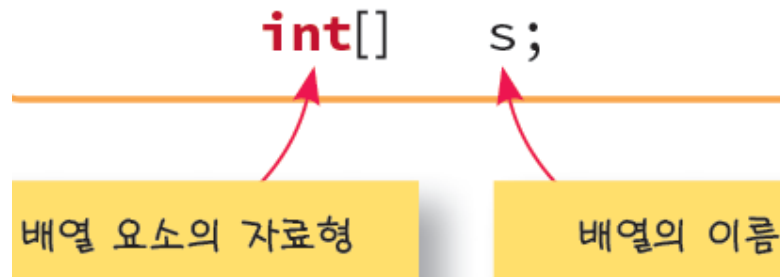
- 자바에서는 배열을 **객체**로 취급한다.



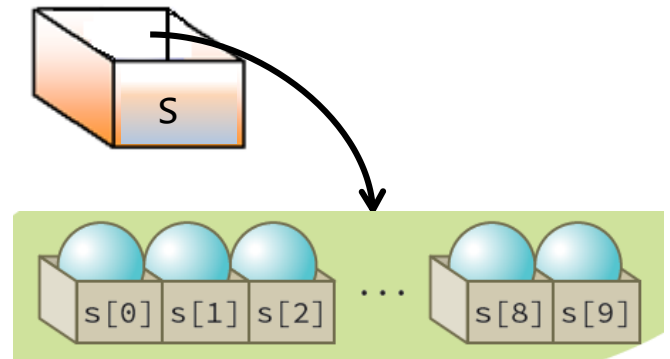
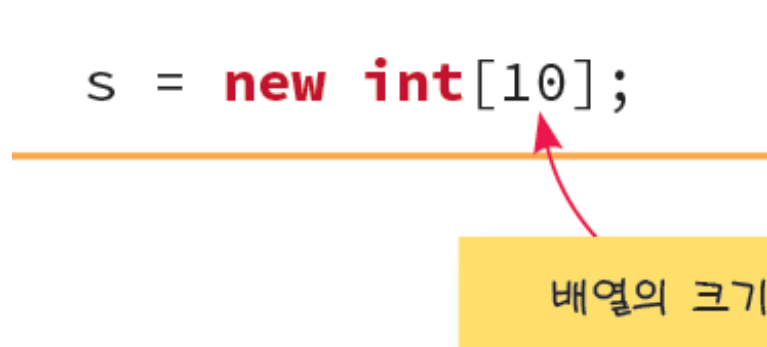
하나의 배열 객체에
10개의 int형 변수가 속함

배열을 만드는 절차

(1) 먼저 배열 참조 변수 s 를 선언



(2) `new` 연산자를 사용하여 배열 객체를 생성하고 s 에 지정



배열을 만드는 절차

- 배열 참조 변수 선언과 동시에 생성하는 것도 가능하다.

```
int[] s = new int[10];
```

- 어떤 자료형의 배열도 생성 가능하다.

```
double[] distances = new double[100];
```

// 실수 배열

```
char[] letters = new char[5];
```

// 문자 배열

```
String[] words = new String[10];
```

// 문자열 배열

배열 인덱스

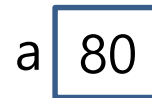
- 다음과 같은 int형 변수 a와 int형 배열 s가 있다고 하자.

int a;

int[] s = new int[5];

- 배열 원소에는 번호가 붙어 있는데 이것을 **인덱스(index)**라고 함
s[0], s[1], s[2], s[3], s[4]
- 각 배열 원소는 하나의 변수로 생각하면 됨

a = 80; // int형 변수



s[0] = 80; // int형 배열 원소



예: 성적 평균 계산

실행결과

성적 입력: 11
성적 입력: 20
성적 입력: 30
성적 입력: 40
성적 입력: 50
평균 성적은 30.2

```
import java.util.Scanner;


public class ArrayTest2 {
    public static void main(String[] args) {
        final int STUDENTS = 5;
        int total = 0;
        Scanner scan = new Scanner(System.in);

        int[] scores = new int[STUDENTS];

        for (int i = 0; i < scores.length; i++) {
            System.out.print("성적 입력: ");
            scores[i] = scan.nextInt();
        }
        for (int i = 0; i < scores.length; i++) {
            total += scores[i];
        }
        System.out.println("평균 성적은 " + (double) total / STUDENTS );
    }
}
```

scores.length는 scores 배열의
길이(원소 수)를 말함

배열 참조 변수 선언 방식

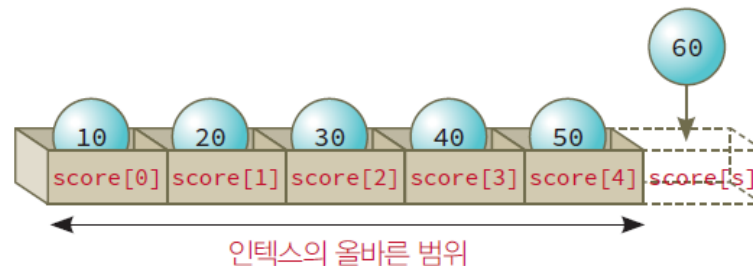
- **int**[] values; // ① 자바 방식 
- **int** values[]; // ② C언어 유사 방식

배열 인덱스가 잘못된 경우

- 인덱스가 범위를 벗어나지 않았는지 프로그래머가 확인하고 책임져야 한다.

```
int[] scores = new int[5];  
scores[0] = 10;  
scores[1] = 20;  
scores[2] = 30;  
scores[3] = 40;  
scores[4] = 50;  
scores[5] = 60;
```

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
at ArrayTest4.main(ArrayTest4.java:16)



배열을 생성하지 않은 경우

- 배열 변수(참조변수)를 선언했다고 해서 배열이 만들어지는 것이 아니다.

```
int[] scores;  
scores[0] = 100;    // 에러
```



```
int[] scores;  
score = new int[5];    // 배열 객체를 생성해야 함  
scores[0] = 100;    // ok
```

배열 선언과 초기화

```
public class ArrayTest2 {  
    public static void main(String[] args) {  
  
        double[] numbers = {1.5, 2.5, 3.5};  
  
        for (int i = 0; i < numbers.length; i++)  
            System.out.println(numbers[i]);  
    }  
}
```

초기화 리스트를 사용하여 배열 선언과 동시에 초기화할 수 있다. 이 경우 new 연산자를 이용하지 않아도 배열 객체가 생성된다.

각 배열의 length 필드는 배열의 길이(원소 개수)를 나타낸다.

실행결과

1.5
2.5
3.5

문자열 배열

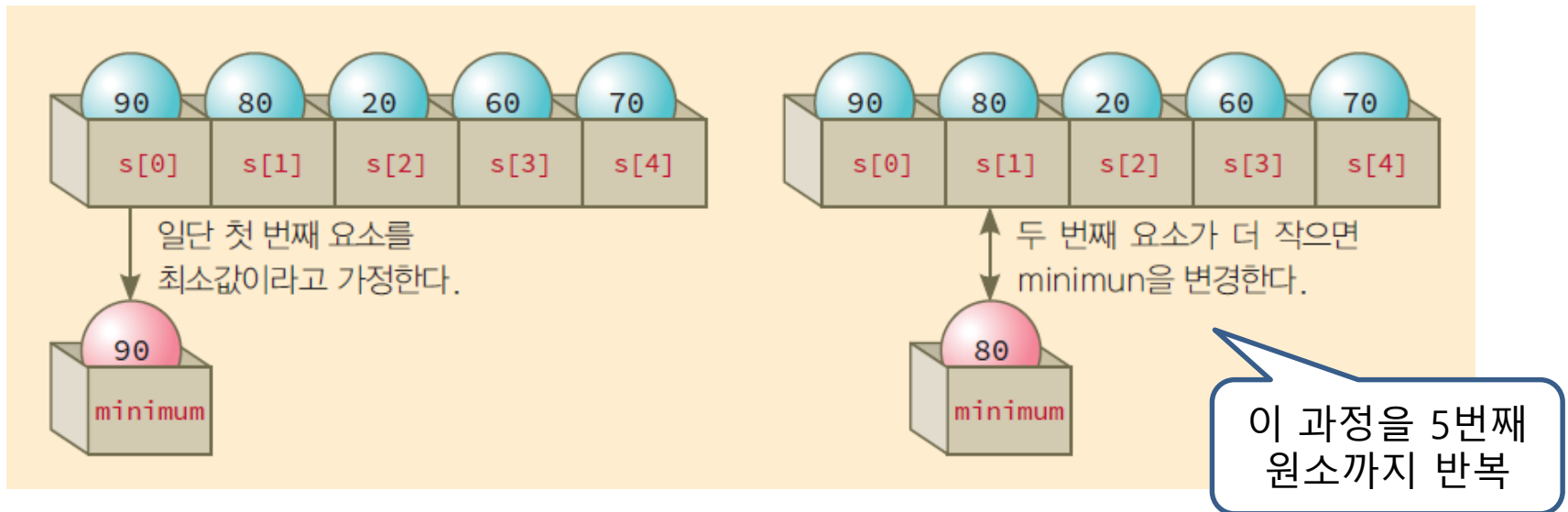
```
public class PizzaTopping {  
    public static void main(String[] args) {  
        String[] toppings = { "Pepperoni", "Mushrooms",  
                                "Onions", "Sausage", "Bacon" };  
  
        for (int i = 0; i < toppings.length; i++) {  
            System.out.print(toppings[i] + " ");  
        }  
    }  
}
```

실행결과

Pepperoni Mushrooms Onions Sausage Bacon

예: 최소값 알고리즘

- 배열 s 에 저장된 값 중 최소값을 찾아 출력해보자.
 - 최소값을 구할 때는 일단 배열의 첫 번째 원소를 최소값으로 가정하고 시작한다.

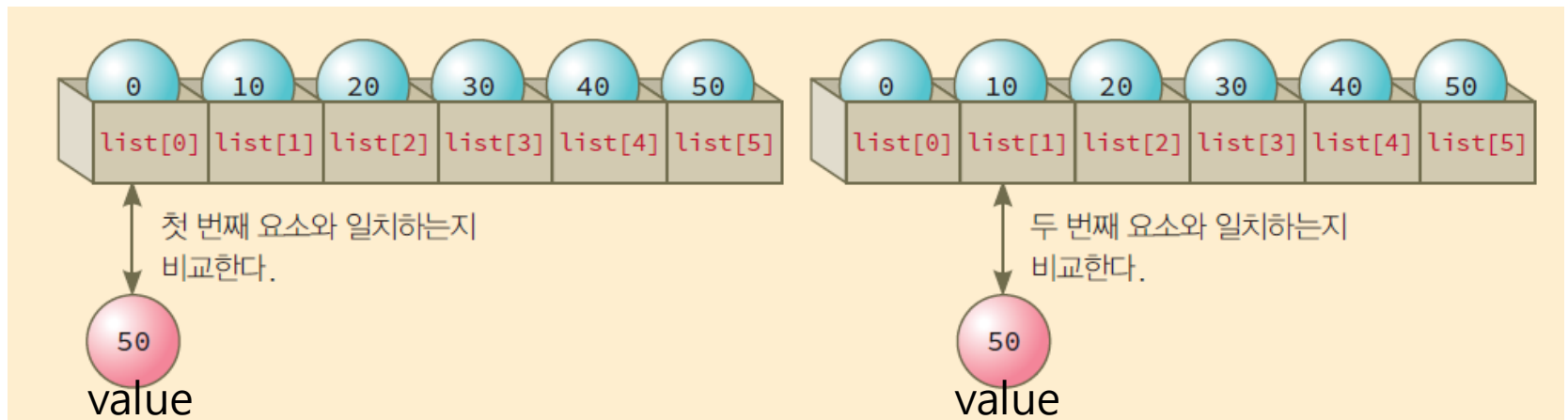


```
public class GetMin {  
    public static void main(String[] args) {  
  
        int[] s = { 12, 3, 19, 6, 18, 8, 12, 4, 1, 19 };  
        int minimum;  
  
        minimum = s[0];  
  
        for (int i = 1; i < s.length; i++) {  
            if (s[i] < minimum)  
                minimum = s[i];  
        }  
  
        System.out.println("최소값은 " + minimum);  
    }  
}
```

두번째 원소부터
마지막 원소까지
반복

예: 순차탐색 알고리즘

- 배열 list의 원소를 순서대로 하나씩 꺼내어 탐색할 값 (value)과 비교하여 이 값이 저장된 위치(index)를 알아낸다.




```
import java.util.Scanner;

public class SeqSearch {
    public static void main(String[] args) {
        int[] list = { 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; index == -1 && i < list.length; i++) {
            if (list[i] == value)
                index = i;
        }

        if (index >= 0)
            System.out.println(value + "값의 위치: " + index);
        else
            System.out.println(value + "값 탐색 실패");
    }
}
```

예: 주사위 던지기

- 6면 주사위를 10000번 던져 각 면이 나온 횟수를 구한다.



=====	
면	빈도
=====	
1	1690
2	1729
3	1634
4	1649
5	1614
6	1684

```

public class RollDice {
    public static void main(String[] args) {
        final int SIZE = 6;
        int[] freq = new int[SIZE];

        for (int i = 0; i < 10000; i++)
            ++freq[(int) (Math.random() * SIZE)];

        System.out.println("=====");
        System.out.println("면빈도");
        System.out.println("=====");

        for (int i = 0; i < SIZE; i++)
            System.out.println((i + 1) + "면" + freq[i]);
    }
}

```

[0, 1) 범위의 랜덤넘버
를 얻음

배열 매개변수

```
void main(String[] args) {  
    int[] s = {1, 2, 3, 4};  
    method3(s);  
    System.out.println(array[0]);  
}
```

s



값이 복사된다

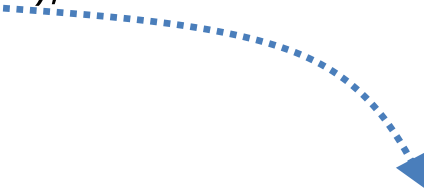
array



```
void method3(int[] array) {  
    System.out.println(array[0]);  
    array[0] = 9;  
}
```

예: 배열 원소 출력1

```
public class ArrayTest {  
  
    public static void main(String[] args) {  
        int[] scores = {10, 20, 30, 40, 50};  
        printArray(scores);  
    }  
  
    private static void printArray(int[] array) {  
        for (int i = 0; i < array.length; i++)  
            System.out.println(array[i]);  
    }  
}
```



ArrayTest 클래스
내에 두 개의
메소드를 정의:
main()
printArray()

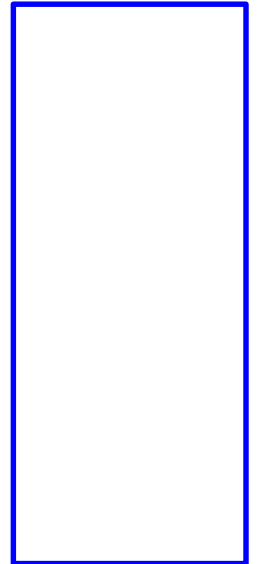
실행결과

10
20
30
40
50

예: 배열 원소 출력2

```
public class ArrayTest {  
    public static void main(String[] args) {  
        int[] scores = {10, 20, 30, 40, 50};  
        printArray(scores);  
  
        int[] numbers = {7, 8, 9};  
        printArray(numbers);  
    }  
  
    private static void printArray(int[] array) {  
        for (int i = 0; i < array.length; i++)  
            System.out.println(array[i]);  
    }  
}
```

실행결과



무명 배열

- 배열의 이름을 지정하지 않고 초기값만으로 배열을 생성할 수 있다.
- 무명 배열(**anonymous arrays**)은 즉시 배열을 만들어서 함수의 인수로 전달하고자 할 때 많이 사용된다.

형식

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

배열의 이름이 없다. 주어진 초기값을 가지는 배열이 생성된다.

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.

sum()은 매개변수로 받은
배열의 원소를 모두 더해
합을 리턴

숫자들의 합 : 10

for-each 루프

- 형식

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

- 예

```
int[] numbers = {10, 20, 30};  
for (int value : numbers) {  
    System.out.println(value);  
}
```

numbers 배열의 원소를 하나
씩 int형 변수 value에 담아
반복 문장을 실행

실행결과

```
10  
20  
30
```

배열 크기를 변수로 지정

```
import java.util.Scanner;

public class ArrayTest {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int size;
        System.out.print("배열의 크기 입력: ");
        size = scan.nextInt();

        int[] scores = new int[size];

        for (int i = 0; i < scores.length; i++) {
            System.out.print("성적 입력: ");
            scores[i] = scan.nextInt();
        }
        ...
    }
}
```

배열의 크기를 변수로 두면,
실행 시간에 사용자가 배열
크기를 결정할 수 있다.

단, 배열의 크기가 결정된
후에는 변경할 수 없다.

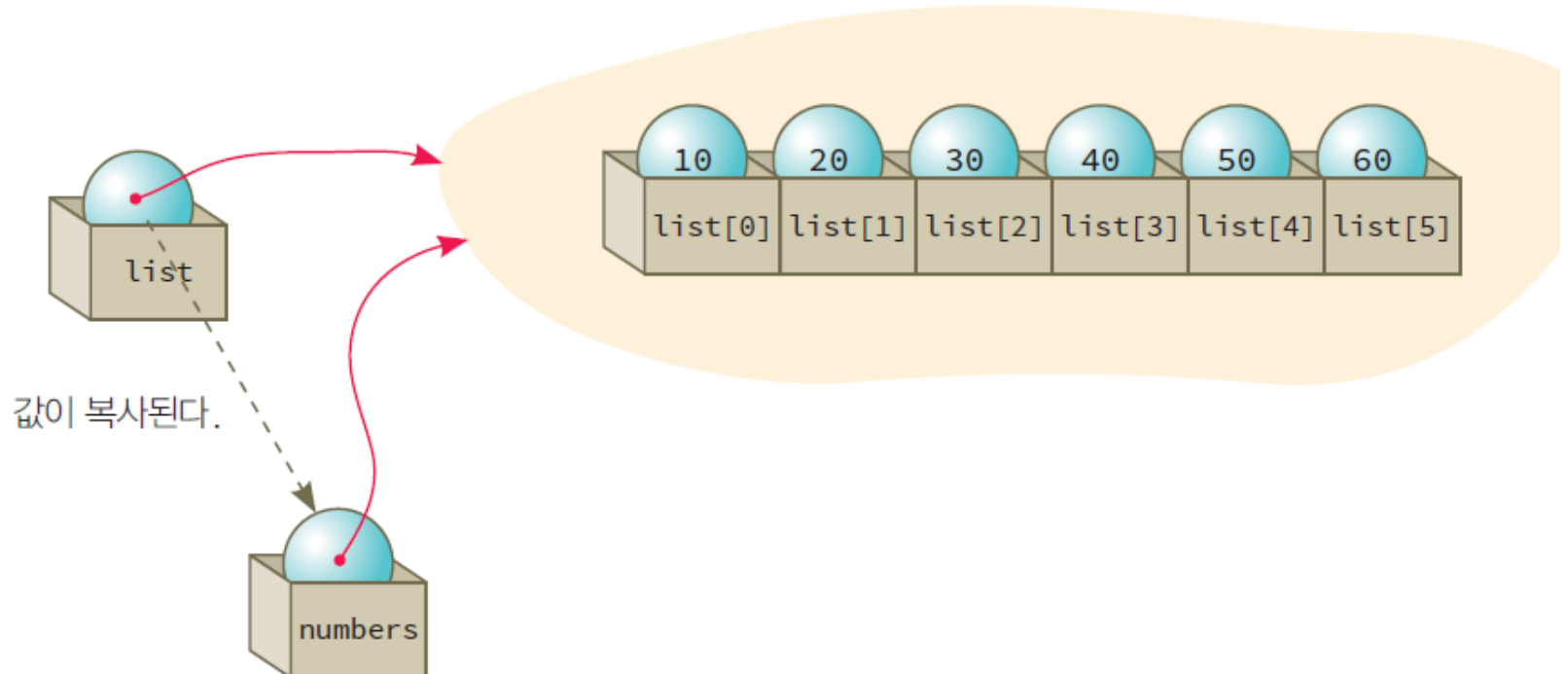
배열의 복사

- 배열 참조 변수의 복사

```
int[] list = {10, 20, 30, 40, 50, 60};
```

```
int[] numbers;
```

```
numbers = list; // 값이 복사되어, 동일한 배열 객체를 가리킨다.
```



배열의 복사

- 한 배열의 모든 원소를 다른 배열로 복사한 복사본을 만들고 싶으면 `Arrays` 클래스의 `copyOf()` 메소드를 사용

- 예

...

```
int[] list = {10, 20, 30, 40, 50, 60};
```

```
int[] numbers = Arrays.copyOf(list, list.length);
```

Arrays 클래스를 사용하려면
`import java.util.Arrays;`

main() 메소드의 매개 변수

- command-line에서 프로그램을 실행시킬 때 주는 인수들이 매개변수 args(문자열 배열)에 저장됨

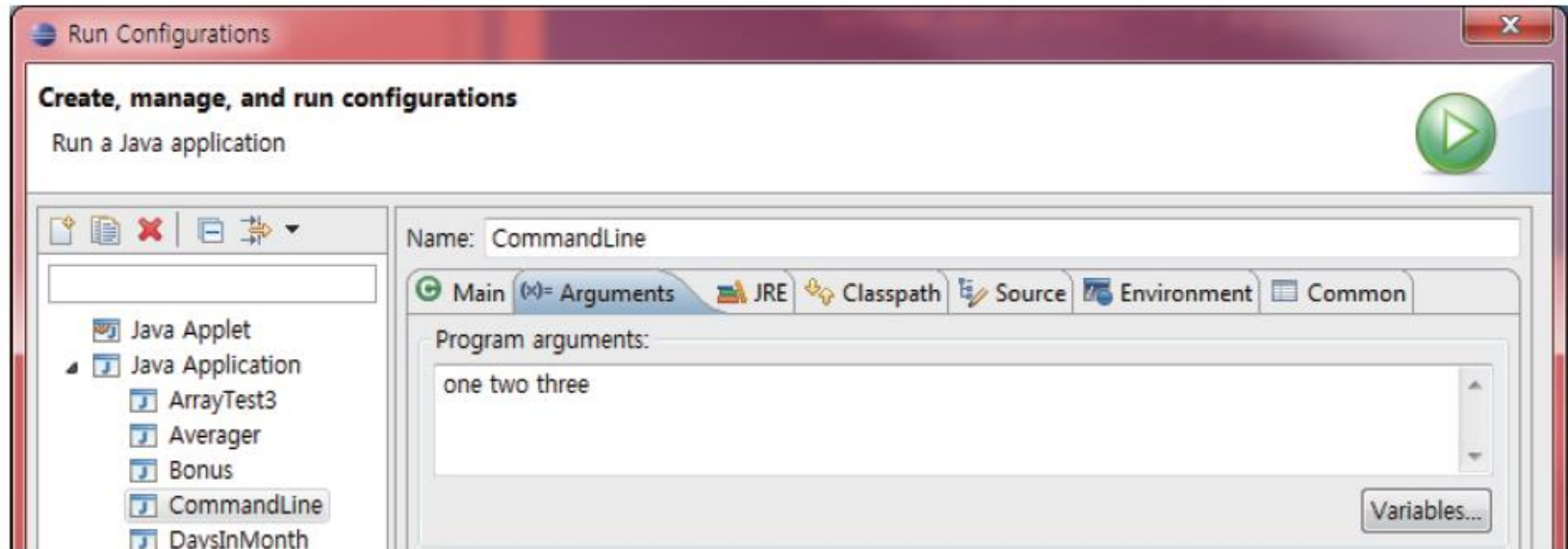
```
01 public class CommandLine {  
02     public static void main(String[] args) {  
03  
04         if (args.length > 0) {  
05             for (int i = 0; i < args.length; i++)  
06                 System.out.print(" " + args[i]);  
07  
08             if (args[0].equals("-h"))  
09                 System.out.print("HELP ");  
10         }  
11     }  
12 }
```

실행결과

```
> javac CommandLine.java  
> java CommandLine one two three  
one two three
```

command-line argument

- 이클립스에서 command-line 인수를 넣는 방법
 - Run > Run Configurations > Arguments 탭에서 Program arguments 에 적어주면 됨



배열 정렬

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

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

```

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         for (int r : numbers)
14             System.out.print(r + " ");
15         Arrays.sort(numbers);
16
17         for (int r : numbers)
18             System.out.print(r + " ");
19
20         }
21     }
22 }

```

0~99 범위의 랜덤넘버

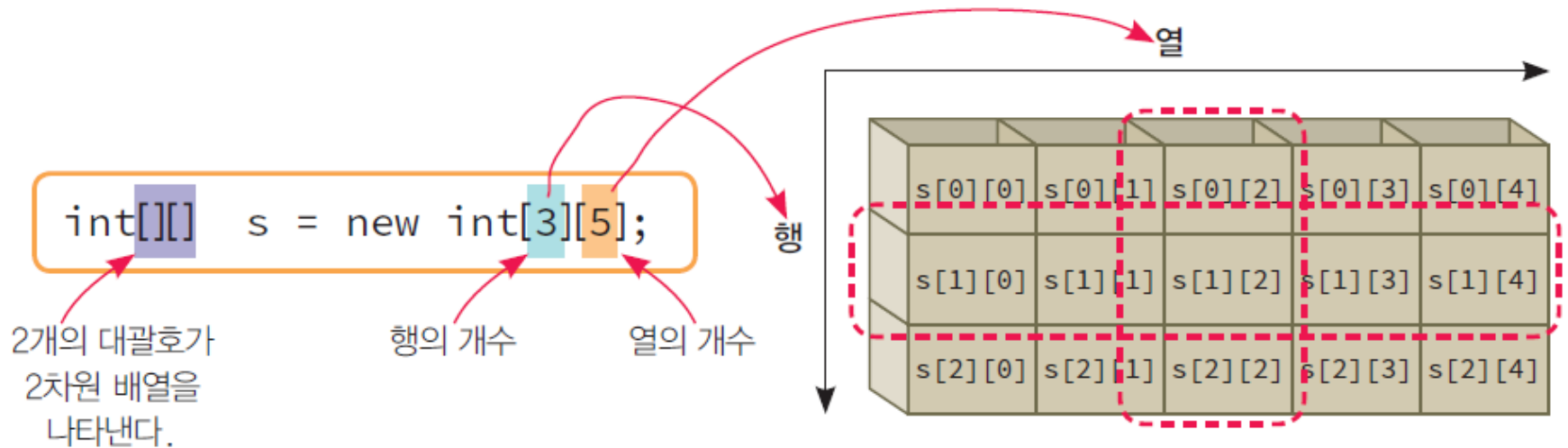
Arrays.sort()를 이용하여
배열 정렬

```

83 72 73 58 45 59 93 72 84 94
45 58 59 72 72 73 83 84 93 94

```

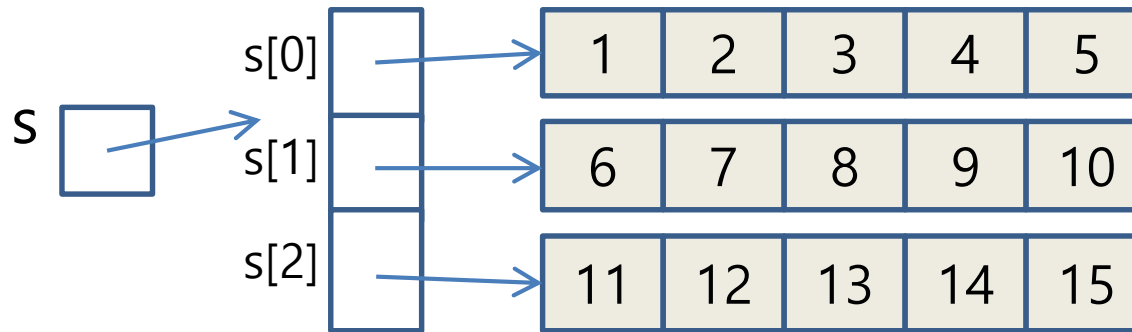

2차원 배열



```
int[][] s = new int[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

```
public class ArrayTest {  
    public static void main(String[] args) {  
        int[][] s = { {1, 2, 3, 4, 5}, {6, 7, 8, 9, 10}, {11, 12, 13, 14, 15} };
```



```
        for (int r = 0; r < s.length; r++)  
            for (int c = 0; c < s[r].length; c++)  
                System.out.println(s[r][c] + " ");
```

배열 `s`의 길이(행의 개수) = 3

행 `s[r]`의 길이 = 5

```
    }  
}
```

ArrayList

15장 제네릭과 컬렉션
에서 다시 다룸

- 전통적인 배열은 크기가 한 번 결정되면 변경할 수 없다.
- ArrayList 라는 클래스를 사용하면 배열의 크기를 자동으로 변경하면서 사용할 수 있다.

- 예) 문자열 배열

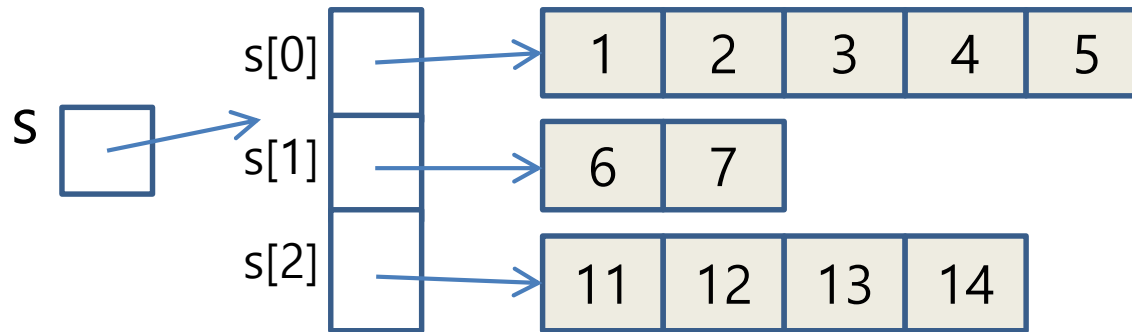
```
ArrayList<String> list = new ArrayList<String>();  
list.add("철수");  
list.add("영희");  
list.remove(1);  
for(int i = 0; i < list.size(); i++)  
    System.out.println(list.get(i));
```

- 예) 정수 배열

```
ArrayList<Integer> list2 = new ArrayList<Integer>();
```

래그드 배열(ragged array)

```
public class ArrayTest {  
    public static void main(String[] args) {  
        int[][] s = { {1, 2, 3, 4, 5}, {6, 7}, {11, 12, 13, 14} };
```



```
        for (int r = 0; r < s.length; r++)  
            for (int c = 0; c < s[r].length; c++)  
                System.out.println(s[r][c] + " ");
```

배열 `s`의 길이(행의 개수) = 3

행 `s[r]`의 길이 = ???

```
    }  
}
```

다차원 배열

- 자바에서도 얼마든지 다차원 배열을 생성할 수 있다.
- 예) double형 3차원 배열 생성

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