

제3장 반복문과 배열

2018년1학기

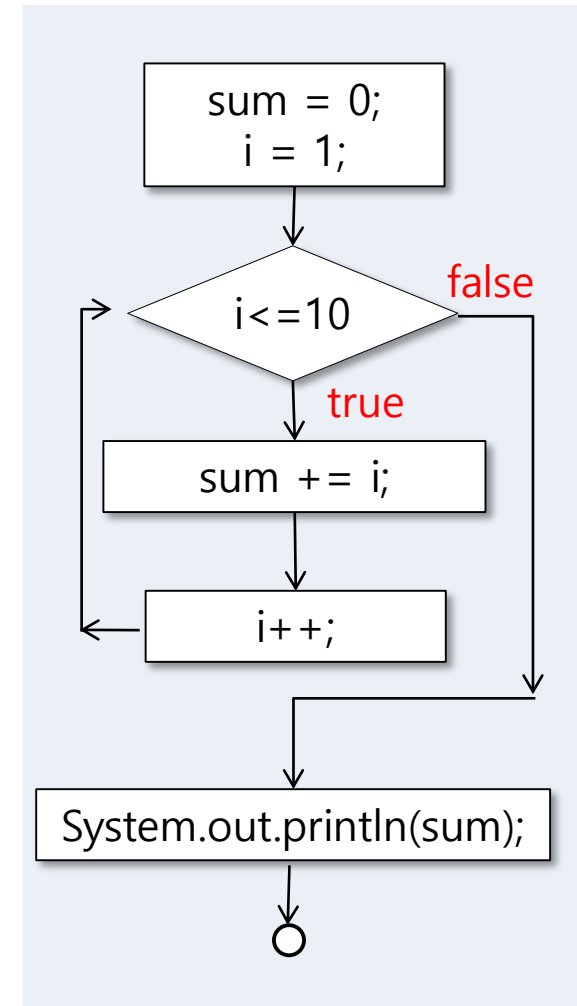
컴퓨터공학과 김 명 교수

- 반복문 (for, while, do-while)
- 배열
 - 1차원 배열
 - 2차원 배열
 - 비정방향 배열
 - 메소드에서 리턴하는 배열
- main() 의 command line arguments

for 문 : 기본적인 형태

3

```
/* 1~10의 정수를 더하여 출력하는 프로그램 */  
public class ForTest {  
    public static void main(String[] args) {  
        int i, sum;  
        sum = 0;  
        for (i = 1; i <= 10; i++)  
            sum += i;  
        System.out.println("sum = " + sum);  
    }  
}
```



for 문 : 변수 선언 포함

4

- for 문 속에서 변수를 선언할 수 있다. 이 변수는 for 문 안에서만 유효하다.

```
public class ForTest {  
    public static void main(String[] args) {  
        int sum;  
        sum = 0;  
        for (int i = 1; i <= 10; i++)  
            sum += i;  
        System.out.println("sum = " + sum);  
    }  
}
```

for 문 : 조건식이 true 인 경우

5

- 조건식이 true 이거나 비어 있는 경우는 무한 반복 (infinite loop)이 된다. 루프를 벗어나기 위해 break문을 쓸 수 있다.

```
public class ForTest {  
    public static void main(String[] args) {  
        int sum = 0;  
        for (int i = 1; true; i++) {  
            if (i <= 10)  
                sum += i;  
            else break;  
        }  
        System.out.println("sum = " + sum);  
    }  
}
```

for 문 : '반복 후 작업'할 문이 여럿 있는 경우

6

- '반복 후 작업'할 일이 여러 개 있는 경우는 콤마 (,) 로 분리하여 쓸 수 있다.

```
public class ForTest {  
    public static void main(String[] args) {  
        int sum = 0;  
        for (int i = 1; i <= 10; sum += i, System.out.println(sum), i++);  
    }  
}
```

while 문 : 기본적인 형태

7

/* while 문을 사용하여 1~10을 더하는 프로그램 */

```
public class WhileTest {
```

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

```
        int sum = 0, i = 1;
```

```
        while ( i <= 10 ){
```

```
            sum += i;
```

```
            i++;
```

← 변수 i의 값을 while 문 속에서 증가시켜 주어야 한다.

```
        }
```

```
        System.out.println("sum = " + sum);
```

```
    }
```

```
}
```

while 문 : 괄호 속이 true 인 경우

8

- 괄호 속이 true 인 경우는 무한 반복을 뜻한다. while 문을 벗어나려할 때는 break 문을 사용하면 된다.

/* while 문을 사용하여 1~10을 더하는 프로그램 */

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

```
        int sum = 0, i = 1;
```

```
        while (true){ ← 괄호 속을 비워 두면 안됨
```

```
            if ( i <= 10 ){
```

```
                sum += i;
```

```
                i++;
```

```
            }
```

```
            else break;
```

```
        }
```

```
        System.out.println("sum = " + sum);
```

```
    }
```

```
}
```


do-while 문 : 기본적인 형태

9

- do-while 문은 적어도 한 번 실행된다. 한 번 실행한 후에야 조건이 점검되기 때문이다.

```
/* do-while 문을 사용하여 1~10을 더하는 프로그램 */  
public class WhileTest {  
    public static void main(String args[]){  
        int sum = 0, i = 1;  
        do {  
            sum += i;  
            i++;  
        } while ( i <= 10 );  
        System.out.println("sum = " + sum);  
    }  
}
```

do-while 문 : 조건에 true 를 쓰는 경우

10

- 조건이 true 이므로 무한 반복이 된다. do-while 을 벗어나기 위해서는 do-while 문 안에서 break 문을 사용하면 된다.

```
public class WhileTest {  
    public static void main(String args[]){  
        int sum = 0, i = 1;  
        do {  
            if ( i <= 10) {  
                sum += i;  
                i++;  
            }  
            else break;  
        } while (true);  
        System.out.println("sum = " + sum);  
    }  
}
```

입력된 수의 평균 구하기 (while 의 조건)

11

```
import java.util.Scanner;
public class WhileSample {
    public static void main (String[] args) {
        Scanner rd = new Scanner(System.in);
        int n = 0;
        double sum = 0;
        int i=0;
        while ((i = rd.nextInt()) != 0) {
            sum += i;
            n++;
        }
        System.out.println("입력된 수의 개수는 " + n + "개이며 평균은 " +
                           sum / n + "입니다.");
    }
}
```

실행 결과

```
10
20
30
40
0 // 입력의 끝을 의미함
입력된 수의 개수는 4개이며
평균은 25.0입니다.
```

입력된 숫자 개수 세기 (while, break)

12

```
import java.util.Scanner;
public class BreakSample {
    public static void main (String[] args) {
        Scanner in = new Scanner(System.in);
        int num = 0;

        while (true) {
            if (in.nextInt() == -1)
                break;
            num++;
        }
        System.out.println("입력된 숫자 개수는 " + num);
    }
}
```

실행 결과

10
8
9
5
-1
입력된 숫자 개수는 4

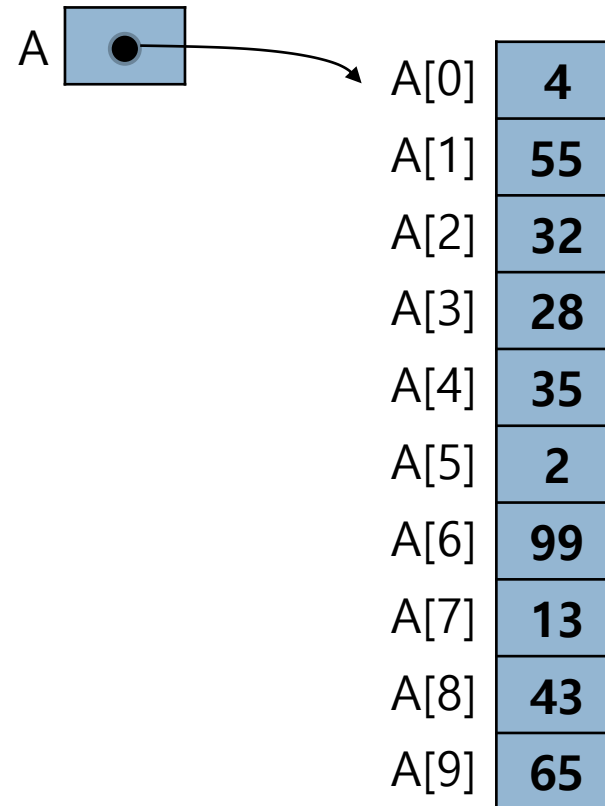
배열 (Array)

13

- 동일한 종류의 데이터를 순차적으로 저장하는 공간
- 배열 원소의 인덱스는 배열의 시작 위치에서부터 데이터가 저장되어 있는 곳의 상대적인 위치를 나타낸다. 인덱스는 0부터 시작한다.

(예제) 10개의 정수로 구성된 배열

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



1차원 배열의 선언

14

- 배열 선언과 배열 생성의 두 단계가 필요하다.

- 배열 선언**

```
int    intArray[ ];  
char   charArray[ ];  
float  floatArray[ ];
```

- 배열 생성**

```
intArray = new int[10];  
charArray = new char[20];  
floatArray = new float[5];
```

- 배열 선언과 동시에 생성**

```
int    intArray[ ] = new int[10];  
char   charArray[ ] = new char[20];  
float  floatArray[ ] = new float[5];
```

- 배열 선언과 초기화**

- 배열이 생성되면서 원소의 값이 초기화됨

```
int intArray[ ] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

배열 선언과 생성의 차이


15

(1) 배열에 대한 레퍼런스 변수 intArray 선언

int intArray [] ;

↑ ↑ ↑

배열 배열에 배열
타입 대한 선언
레퍼런스 변수


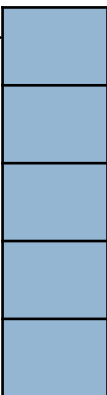
intArray 

(2) 배열 생성

intArray = new int [5];

↑ ↑ ↑ ↑

배열에 배열 원소
대한 생성 개수
레퍼런스 타입
변수

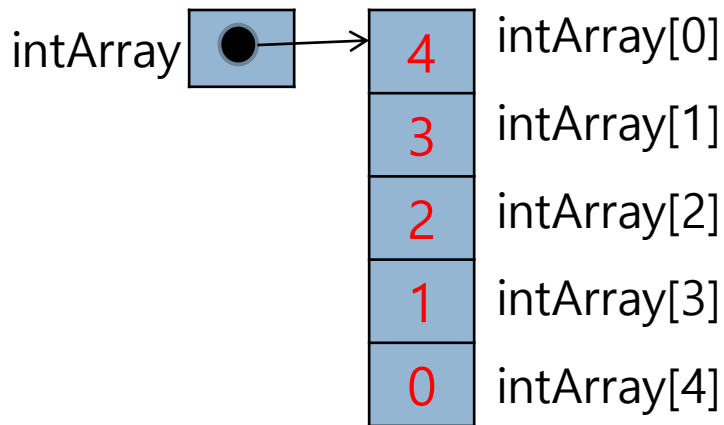
intArray  → 

intArray[0]
intArray[1]
intArray[2]
intArray[3]
intArray[4]

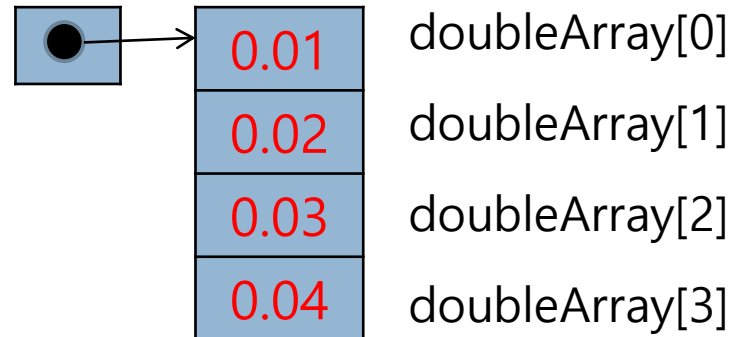
배열을 초기화하면서 생성한 결과

16

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



doubleArray

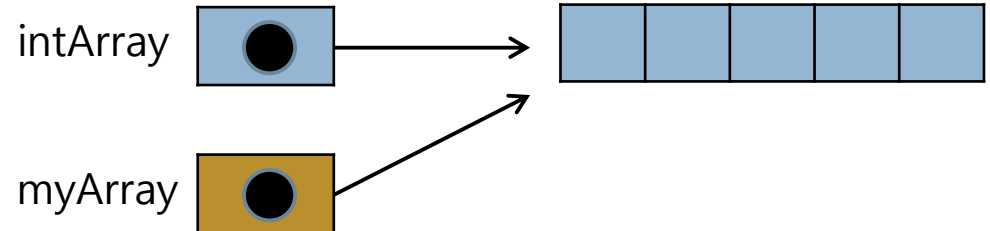


배열 참조

17

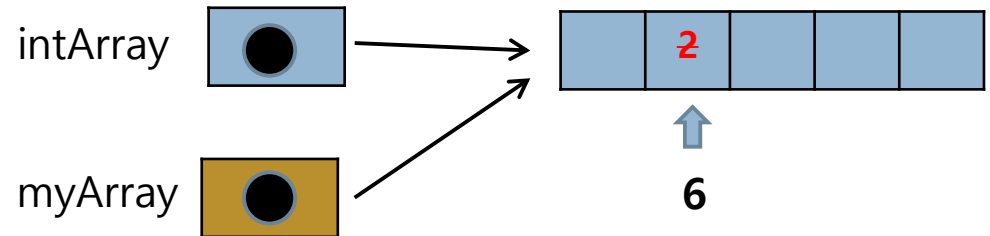
배열 생성

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



배열 원소에 대입

```
intArray[1] = 2;  
myArray[1] = 6;
```



* 여러 레퍼런스가 하나의 배열을 참조할 수 있다.

배열에 입력받은 5개의 수 중에서 가장 큰 수 찾기

18

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

        for (int i = 0; i < 5; i++) {
            intArray[i] = in.nextInt();
            if (intArray[i] > max)
                max = intArray[i];
        }
        System.out.print("입력된 수에서 가장 큰 수는 " + max + "입니다.");
    }
}
```

실행 결과

1
39
78
100
99
입력된 수에서 가장 큰
수는 100입니다.

배열의 크기와 인덱스

19

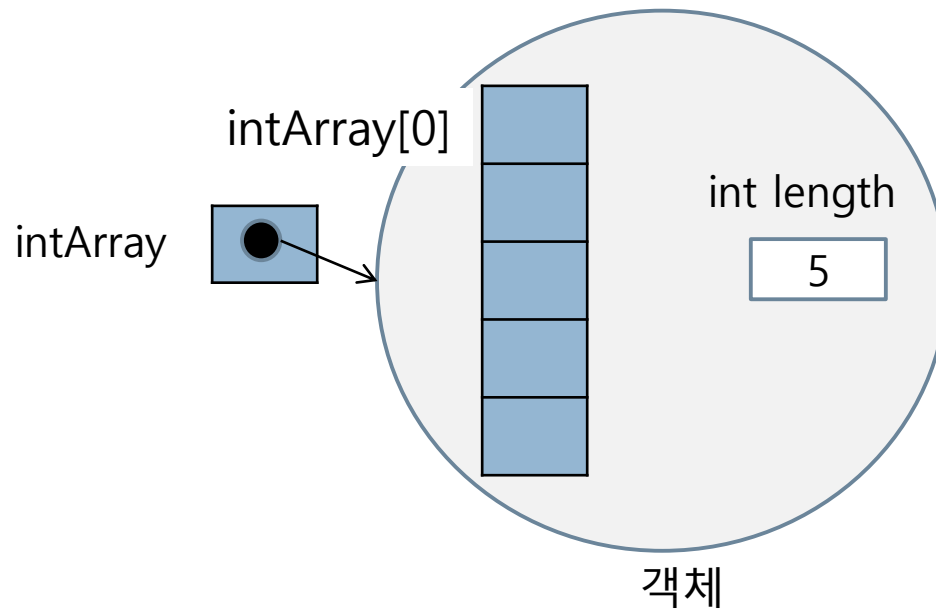
- 배열 인덱스
 - ▣ 인덱스는 0부터 시작하며 마지막 인덱스는 (배열 크기 -1)임
 - ▣ 인덱스는 정수 타입만 가능
- 배열의 크기
 - ▣ 배열의 크기는 배열 레퍼런스 변수를 선언할 때 결정되지 않음
 - ▣ 배열의 크기는 배열 생성 시에 결정되며, 나중에 바꿀 수 없음
 - ▣ 배열의 크기는 배열의 length라는 필드에 저장

```
int size = intArray.length;
```

배열 : 객체로 관리한다.

20

```
int intArray [];  
intArray = new int[5];  
int size = intArray.length; // size는 5
```



배열 원소의 평균 구하기

21

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.");
    }
}
```

실행 결과

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

배열과 for-each 문 (1/2)

22

- for-each 문 : 배열(array)이나 열거형 변수(enumeration)의 각 원소를 순차적으로 접근하는데 유용한 for 문

```
public class foreachEx2 {  
    public static void main(String[] args) {  
        int[] num = { 1, 2, 3, 4, 5 };  
        int sum = 0;  
        for (int k : num)  
            // 반복될 때마다 k는 num[0], num[1], ..., num[4] 값으로 설정된다.  
            sum += k;  
        System.out.println("합은 " + sum);  
    }  
}
```

합은 15

배열과 for-each 문 (2/2)

23

```
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

```
public class foreachEx2 {  
    enum Week { 월, 화, 수, 목, 금, 토, 일 }  
    public static void main(String[] args) {  
        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

for-each 문을 이용한 반복문 활용

24

```
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;
```

```
        for (int k : num)  
            sum += k;  
        System.out.println("합은 " + sum);
```

```
        for (String s : names)  
            System.out.print(s + " ");  
        System.out.println();
```

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

```
        for (Week day : Week.values())  
            System.out.print(day + "요일 ");  
        System.out.println();
```

```
    }  
}
```


2차원 배열

25

□ 2차원 배열 선언

```
int    A[ ][ ];  
char   B[ ][ ];  
float  C[ ][ ];
```

2차원 배열 생성

```
A = new int[2][5];  
B = new char[5][5];  
C = new float[5][2];
```

□ 2차원 배열 선언과 동시에 생성

```
int    A[ ][ ] = new int[2][5];  
char   B[ ][ ] = new char[5][5];  
float  C[ ][ ] = new float[5][2];
```

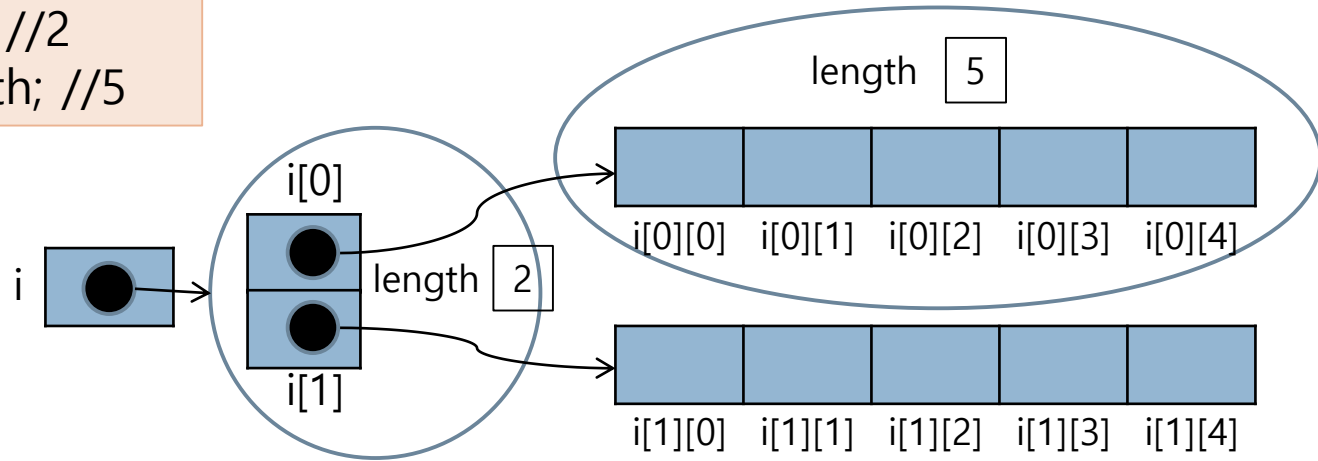
□ 2차원 배열 선언, 생성, 초기화

```
int A[ ][ ] = {{0, 1, 2}, {3, 4, 5}, {6, 7, 8}};  
char B[ ][ ] = {{'a', 'b', 'c'}, {'d', 'e', 'f'}};  
float C[ ][ ] = {{0.01, 0.02}, {0.03, 0.04}};
```

2차원 배열의 length 필드

26

```
int i[ ][ ] = new int[2][5];  
int size1 = i.length; //2  
int size2 = i[0].length; //5
```



- 2차원 배열의 length
 - ▣ `i.length` → 2차원 배열의 행의 개수로서 2
 - ▣ `i[n].length`는 `n`번째 행의 열의 개수
 - `i[0].length` → 0번째 행의 열의 개수로서 5
 - `i[1].length` → 1번째 행의 열의 개수로서 역시 5

3년간 매출 총액과 평균 구하기

27

한 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.

실행 결과

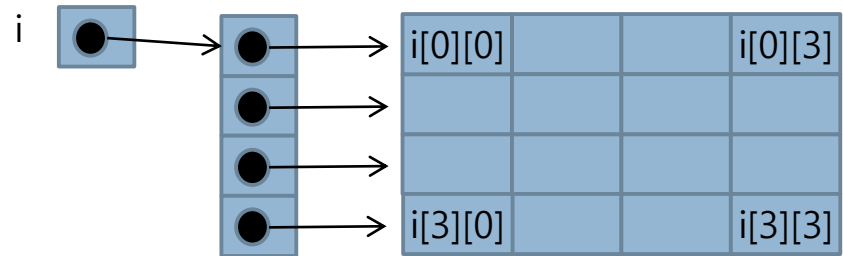
비정방형 배열

28

□ 정방형 배열

- 각 행의 열의 개수가 같은 배열

```
int i[][];  
i = new int[4][4];
```

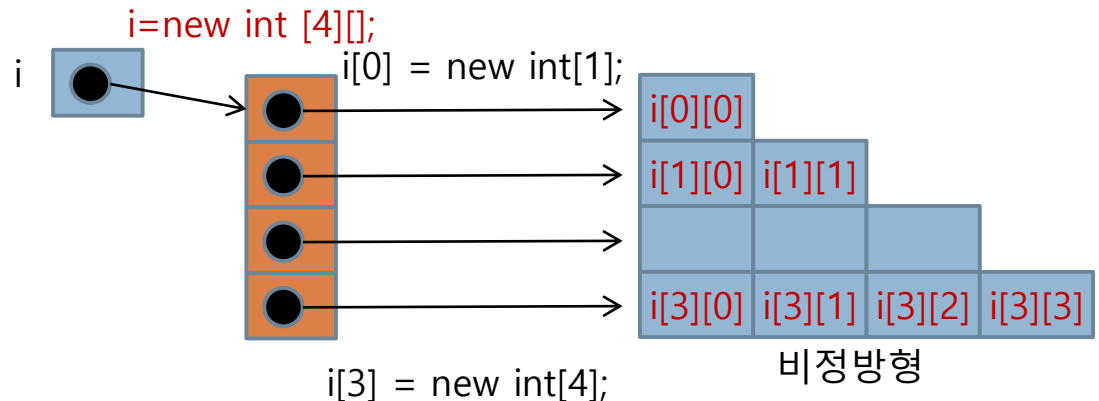


정방형

□ 비정방형 배열

- 각 행의 열의 개수가 다른 배열
- 비정방형 배열의 생성

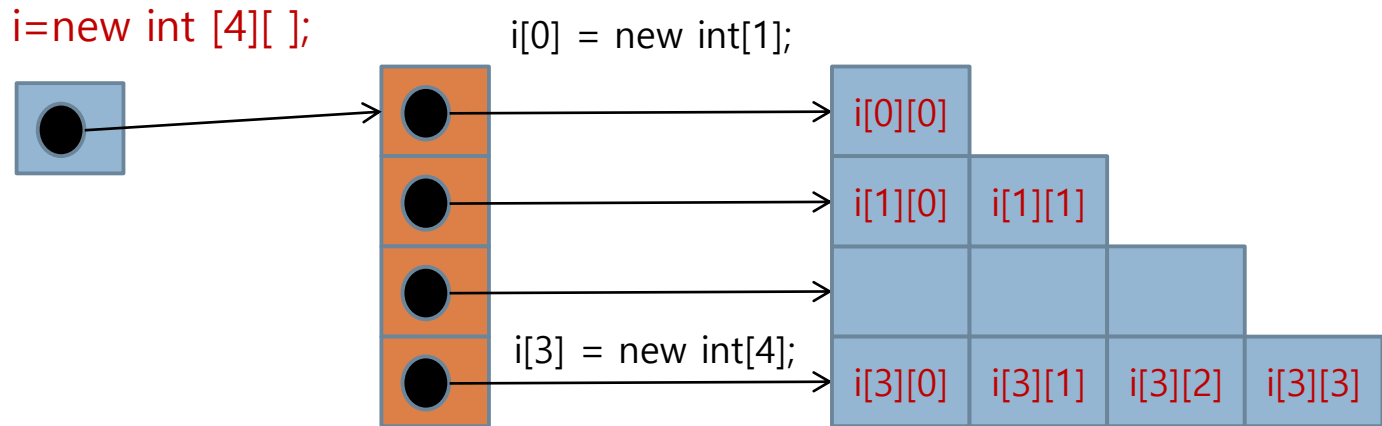
```
int i[][];  
i = new int [4][];  
i[0] = new int [1];  
i[1] = new int [2];  
i[2] = new int [3];  
i[3] = new int [4];
```



비정방형

비정방형 배열의 length

29



- 비정방형 배열의 length
 - `i.length` → 2차원 배열의 행의 개수 (4)
 - `i[n].length`는 `n`번째 행의 열의 개수
 - `i[0].length` → 0번째 행의 열의 개수 (1)
 - `i[1].length` → 1번째 행의 열의 개수 (2)
 - `i[2].length` → 2번째 행의 열의 개수 (3)
 - `i[3].length` → 3번째 행의 열의 개수 (4)

비 정방형 배열의 생성과 접근

30

다음 그림과 같은
비정방형 배열을
만들어 값을
초기화하고
출력하시오.

10	11	12
20	21	
30	31	32
40	41	

```
public class IrregularArray {  
    public static void main (String[] args) {  
        int a = 0;  
        int intArray[][] = new int[4][];  
        intArray[0] = new int[3];  
        intArray[1] = new int[2];  
        intArray[2] = new int[3];  
        intArray[3] = new int[2];  
  
        for (int i = 0; i < intArray.length; i++)  
            for (int j = 0; j < intArray[i].length; j++)  
                intArray[i][j] = (i+1)*10 + j;  
  
        for (int i = 0; i < intArray.length; i++) {  
            for (int j = 0; j < intArray[i].length; j++)  
                System.out.print(intArray[i][j] + " ");  
            System.out.println();  
        }  
    }  
}
```

실행 결과

```
10 11 12  
20 21  
30 31 32  
40 41
```

메소드에서 배열 리턴

31

- 메소드가 리턴하는 배열
 - ▣ 메소드가 리턴하는 배열의 타입과 차원은 리턴받는 배열 레퍼런스의 타입과 차원에 일치해야 함
 - ▣ 리턴 타입에 배열의 크기를 지정하지 않음

```
int [] makeArray() {  
    int temp [] = new int [4];  
    return temp;  
}
```

배열 리턴 예제

32

배열을 생성하고 각 원소 값을 출력하는 프로그램을 작성하시오. 배열 생성은 배열을 생성하여 각 원소의 인덱스 값으로 초기화하여 반환하는 메소드를 이용한다.

```
public class ReturnArray {  
    static int[] makeArray() {  
        int temp[] = new int[4];  
        for (int i=0;i<temp.length;i++)  
            temp[i] = i;  
        return temp;  
    }  
    public static void main (String[] args) {  
        int intArray [];  
        intArray = makeArray();  
        for (int i = 0; i < intArray.length; i++)  
            System.out.println(intArray[i]);  
    }  
}
```

실행 결과

0
1
2
3

main() 메소드

33

객체 생성 전부터
호출 가능

다른 클래스에서
메소드 접근 허용

리턴 값
없음

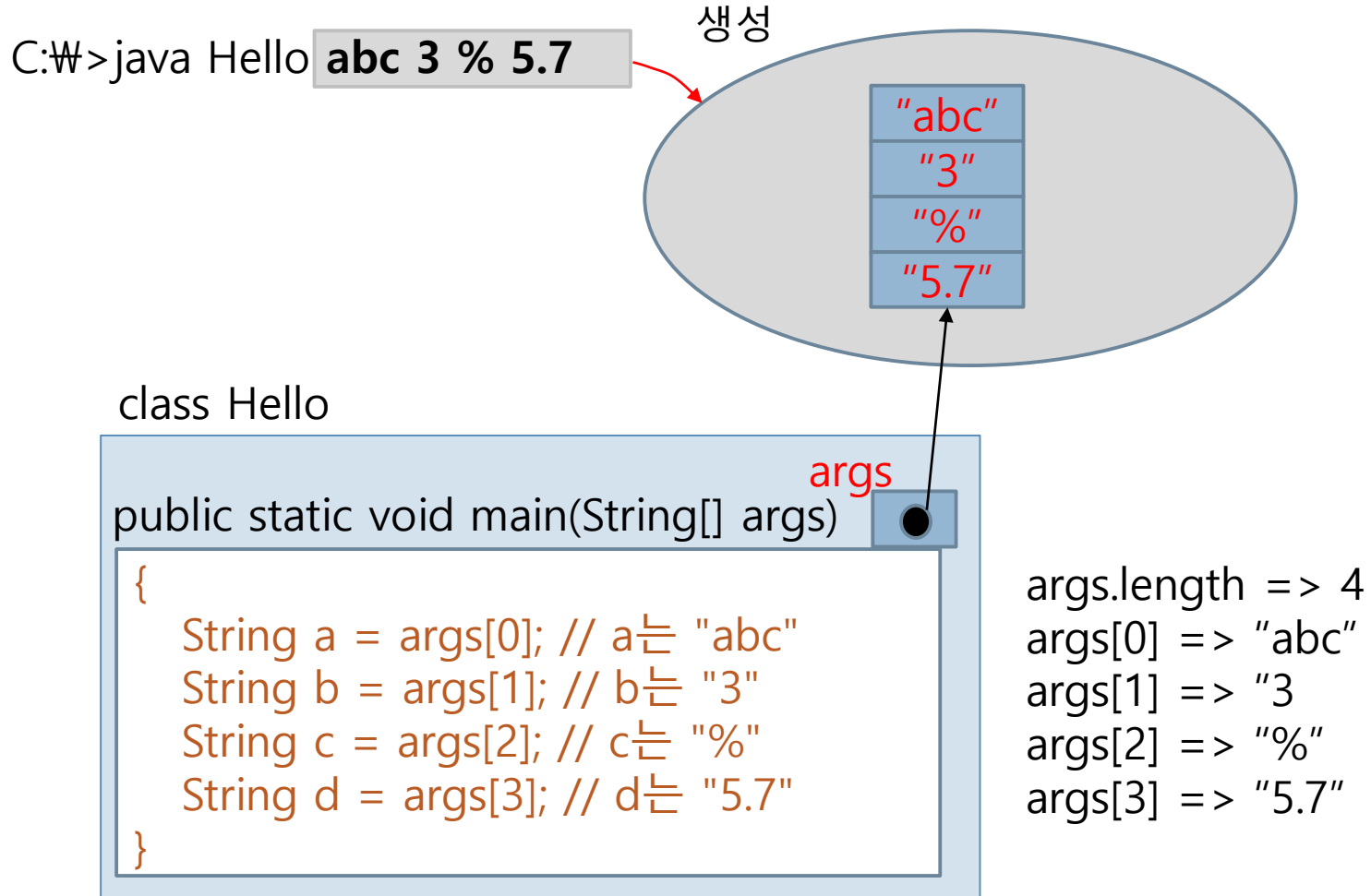
문자열
배열

인자

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

main(string [] args) 메소드의 인자 전달

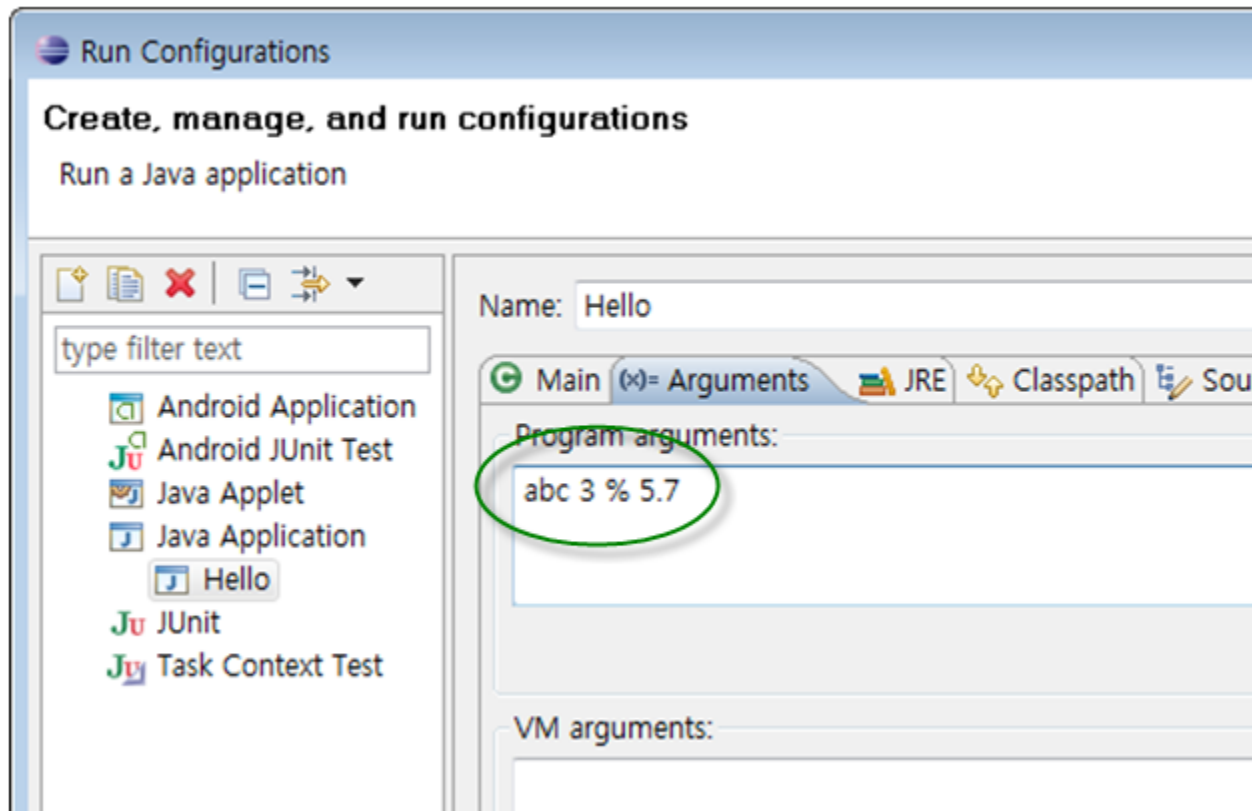
34



이클립스에서 main() 메소드의 인자전달

35

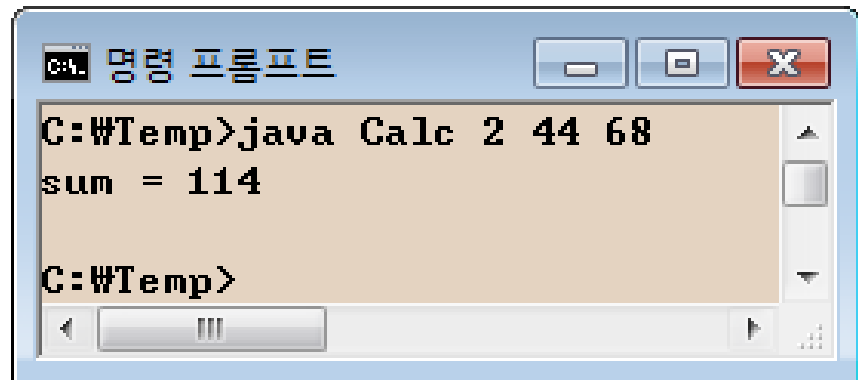
- Run 메뉴의 Run Configurations 항목에서 main() 메소드의 인자를 나열



main()의 인자 이용 예

36

```
public class Calc {  
    public static void main(String[] args) {  
        int sum = 0;  
        for(int i=0; i<args.length; i++) { // 명령행 인자의 개수만큼 반복  
            int n = Integer.parseInt(args[i]); // 명령행 인자인 문자열을 정수로 변환  
            sum += n; // 숫자를 합한다.  
        }  
        System.out.println("sum = " + sum);  
    }  
}
```



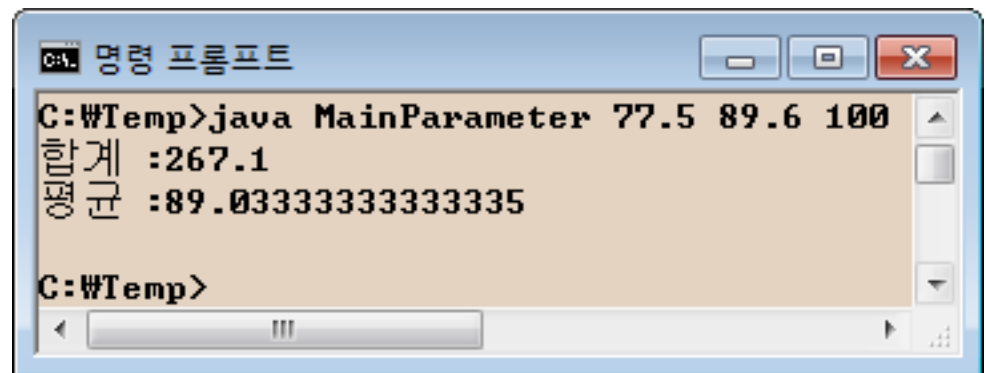
```
명령 프롬프트  
C:\WTemp>java Calc 2 44 68  
sum = 114  
C:\WTemp>
```

main()의 인자들을 받아서 평균값을 계산하는 예제

37

실수를 main() 메소드 인자로 전달받아 평균값을 구하는 프로그램을 작성하시오.

```
public class MainParameter {  
    public static void main (String[] args) {  
        double sum = 0.0;  
  
        for (int i=0; i<args.length; i++)  
            sum += Double.parseDouble(args[i]);  
        System.out.println("합계 :" + sum);  
        System.out.println("평균 :" + sum/args.length);  
    }  
}
```



```
C:\Temp>java MainParameter 77.5 89.6 100  
합계 :267.1  
평균 :89.03333333333335  
C:\Temp>
```

학습 정리

38

- 반복문은 C 언어의 반복문과 사용법이 같다.
단, 반복문 안에서 정의되는 변수들은 반복문 안에서만 쓸 수 있다.
- 배열 선언
 - ▣ 배열 레퍼런스 변수의 선언과 실제 배열의 생성으로 이루어진다.
- 배열이 생성될 때 배열의 크기는 **배열이름.length** 에 **자동으로 저장된다**.
- 비정방형 배열은 행마다 원소 개수를 달리 할 수 있는 배열이다.
- foreach 문: 배열이나 열거형 변수의 각 원소를 순차적으로 접근하는데 유용한 for 문
- 함수 (메소드) 에서 생성된 배열을 그 함수가 호출된 곳으로 보낼 수 있다.
- main()의 command line argument 는 문자열이며, 프로그램 안에서는 이를 적절한 데이터형으로 변환하여 사용할 수 있다.