

04 반복과 배열

Sangwook Lee

04 반복과 배열

1. 제어문 개요

2. 반복문

3. 배열

1. 제어문 개요

- 제어문(control flow statement)

- 조건에 따라 문장을 선택 / 반복 / 분기

순차

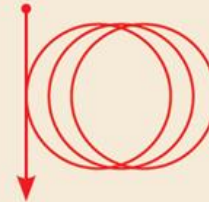
순차적 실행 구문



반복(순환)

반복조건에 따라 일정영역의 반복 구문

- for
- while
- do while



조건선택

조건에 대한 선택 구문

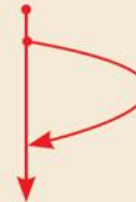
- if
- if else
- if else if
- nested if
- switch



분기처리

지정된 영역으로 실행을 이동하는 구문

- break
- continue
- return



04 반복과 배열

1. 제어문 개요

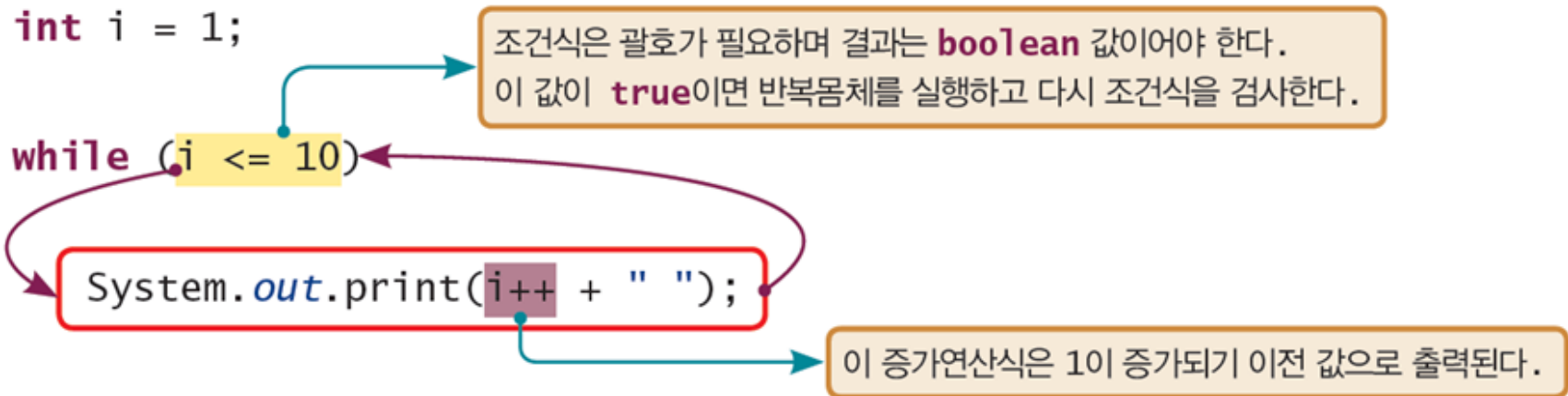
2. 반복문

3. 배열

2. 반복문

- **while**

- 조건식이 **true**이면 문장 반복, **false**이면 문장 반복 종료
- 조건식 평가 후 반복



2. 반복문

- **do while**

- 조건식이 **true**이면 문장 반복, **false**이면 문장 반복 종료
- 반복 후 조건식 평가

```
int i = 1;  
do  
    System.out.print(i++ + " ");  
while (i <= 10);
```

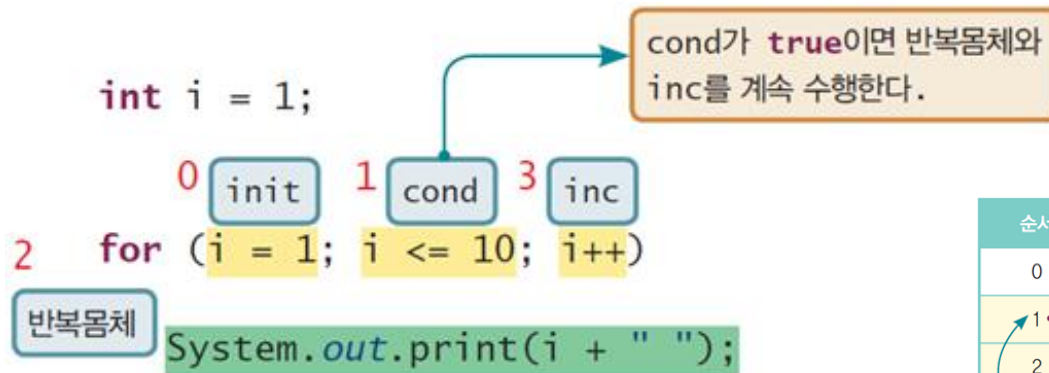


do while 문의 종료를 나타내는 세미콜론은 반드시 있어야 한다.

2. 반복문

- for

– 변수 초기화, 조건식 평가, 변수 증감을 통해 반복을 결정

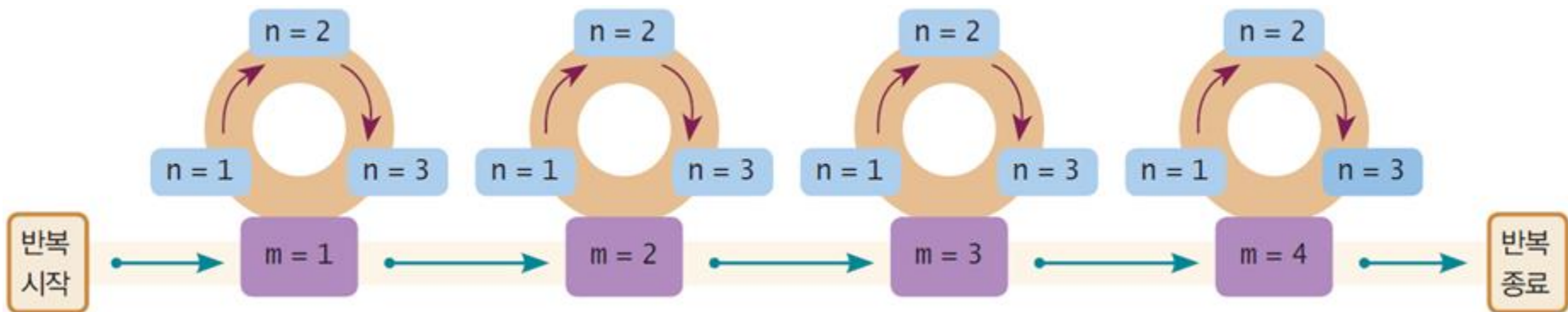
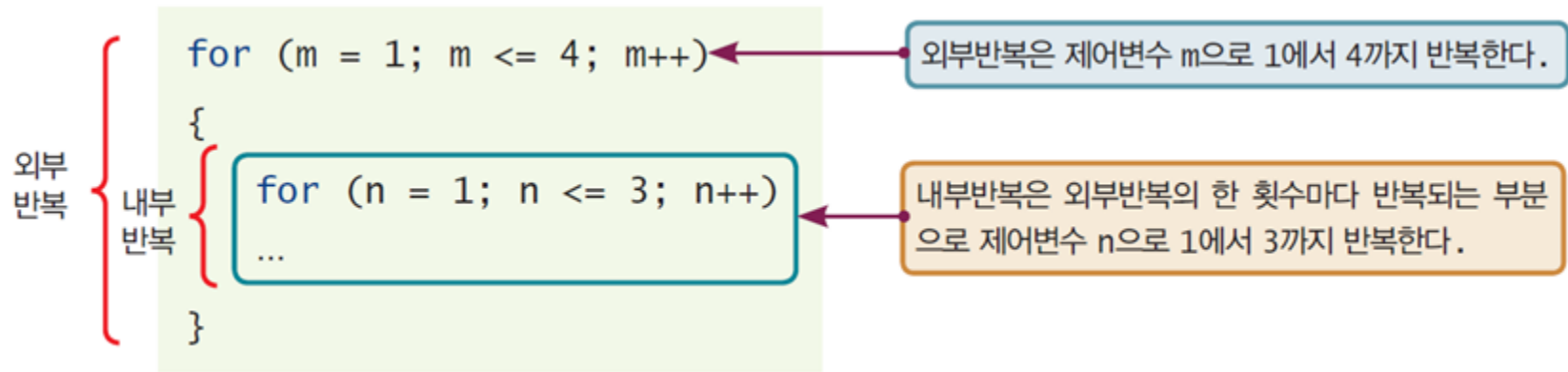


순서	의미	문장
0	init (한 번만 실행됨)	<code>i = 1</code>
1	cond (결과가 boolean)	<code>i <= 10</code>
2	반복몸체 (여러 개면 블록으로)	<code>...print(i + " ");</code>
3	inc (주로 증감 연산 문장)	<code>i++</code>

1인 cond가 true이면 2인 반복몸체와 3인 inc를 계속 수행한다.

2. 반복문

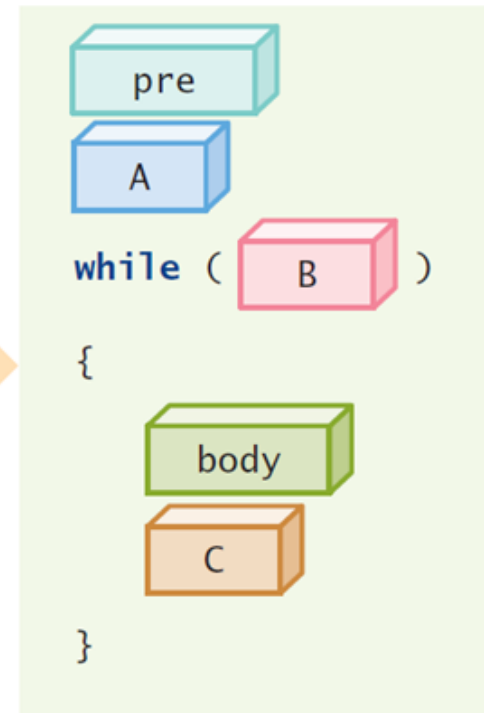
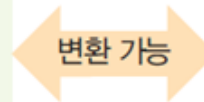
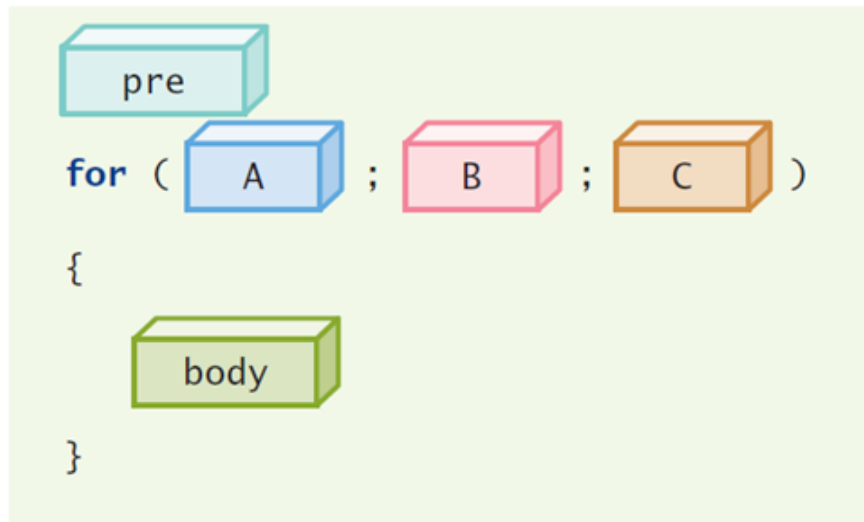
- Nested loop
 - 반복문 안에 반복문이 있는 구조



2. 반복문

- while vs for

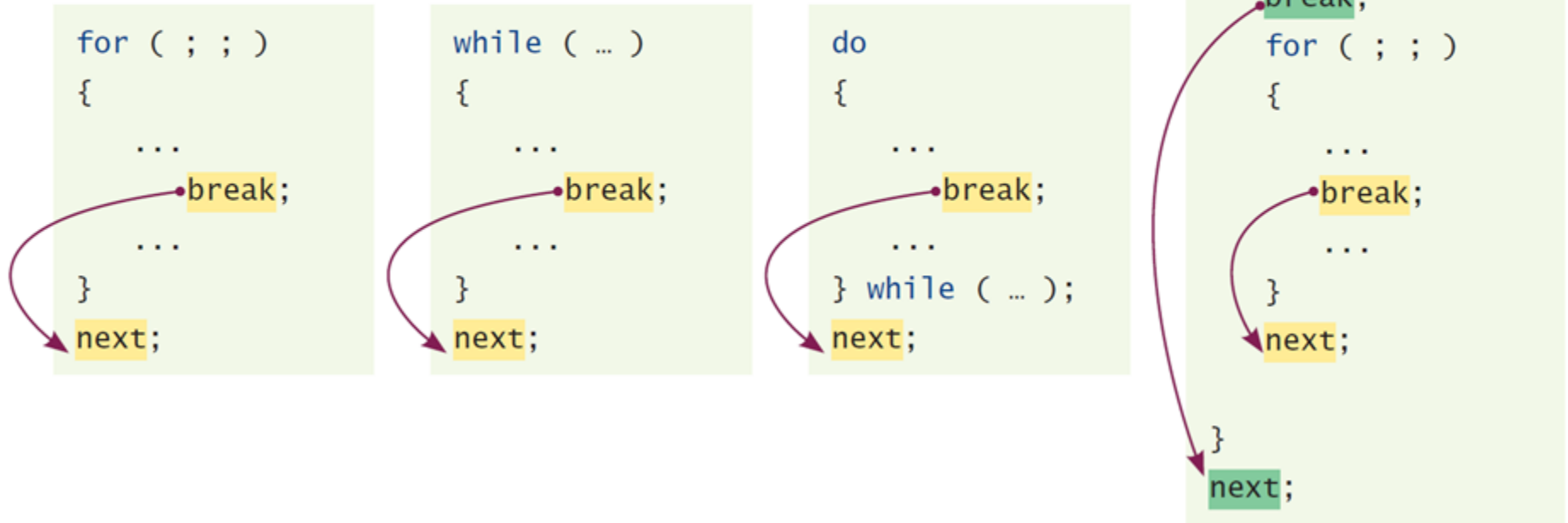
- for는 반복 횟수가 미리 정해져 있는 경우에 적합하고, while은 특정 조건에 따라 반복을 결정해야 될 경우 적합하다고는 하지만...
- for와 while 변환이 가능하다



2. 반복문

- 분기문

- break: 반복 종료



2. 반복문

- 분기문

- continue: 이후 문장 실행하지 않고 증감식이나 조건식으로 이동

```
for (init ; cond ; inc)
{
    ...
    continue;
    stmt1;
    stmt2;
}
```

```
while (cond)
{
    ...
    continue;
    stmt1;
    stmt2;
}
```

```
do
{
    ...
    continue;
    stmt1;
    stmt2;
} while (cond);
```

continue를 만나면 실행되지 않는 부분

```
while ( cond1 )
{
    ...
    continue;
    for (init ;cond2 ;inc )
    {
        ...
        continue;
        stmt1;
        stmt2;
    }
}
```

2. 반복문

[예제 4-6] 1에서 10까지 수 중 3의 배수가 아닌 것만 출력

```
public class Continue {  
    public static void main(String[] args) {  
        int i = 1;  
  
        for (i = 1; i <= 10; i++) {  
            if (i%3 == 0) continue;  
            System.out.print(i + " ");  
        }  
  
        System.out.println();  
    }  
}
```

04 반복과 배열

1. 제어문 개요
2. 반복문
3. 배열

3. 배열

- 배열이란

- 동일한 형태의 자료를 정해진 수만큼 저장 처리하는 객체

- 관련 용어

- 원소(element)

- 배열 구성 항목(개별 저장 공간)

- 배열 크기(array length)

- 원소 수

- 첨자(index)

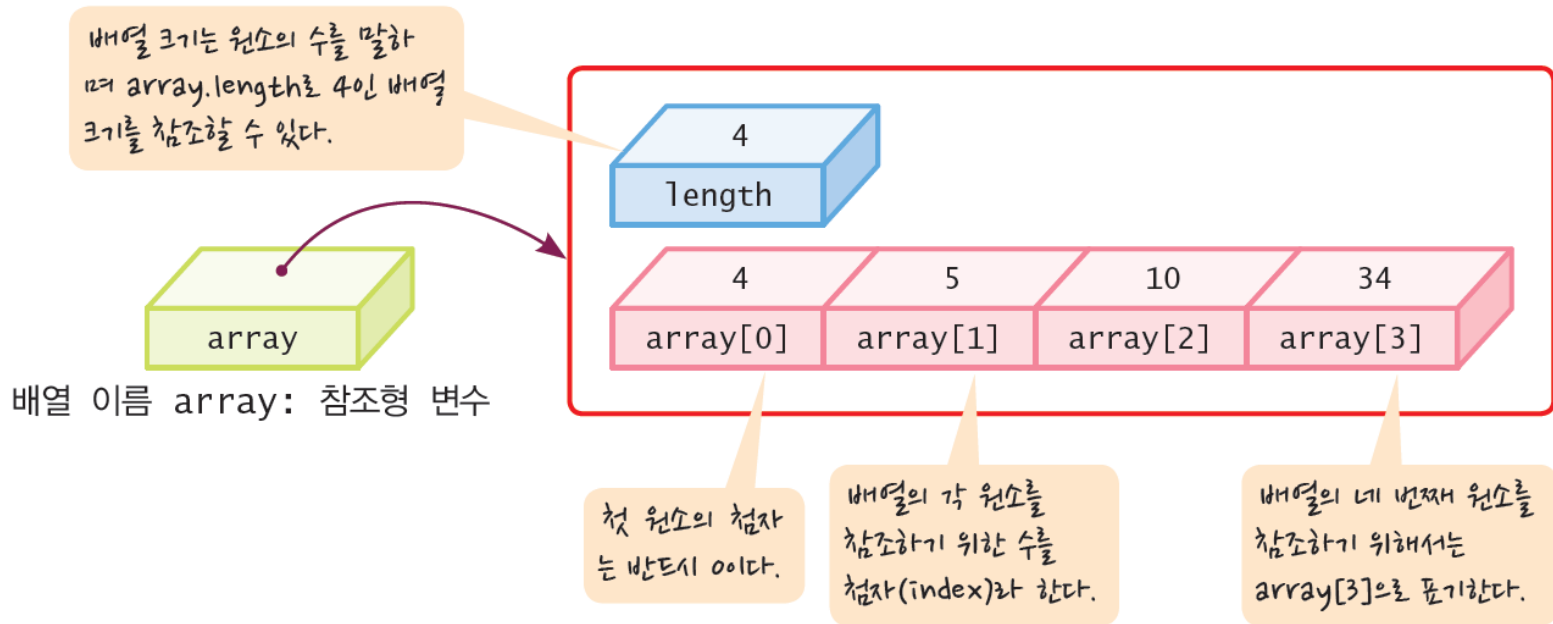
- 원소 참조(접근) 번호

- 원소의 수가 n 일 경우 첨자 범위는 $0 \sim n-1$

3. 배열

자바에서의 배열은...

참조형 변수로서 배열 원소를 위한 공간과 함께
배열 크기(`length`)가 저장되는 공간을 가진 객체



3. 배열

- 배열 선언

- 배열 (객체) 참조 변수 선언
- 선언 시 배열 크기를 표시할 수 없음
- 선언 방법

```
자료형 배열명[];
```

OR

```
자료형 []배열명;
```

OR

```
자료형[] 배열명;
```

```
배열명  
=  
배열 변수명  
=  
배열 참조 변수명
```

3. 배열

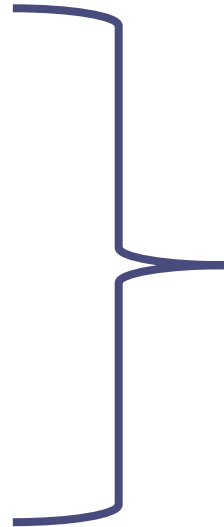
- 배열 선언

— 배열 선언 예

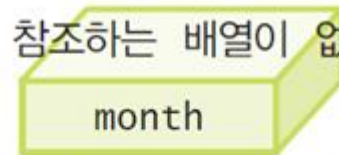
```
int month[];  
double values[];
```

```
int []month;  
double []values;
```

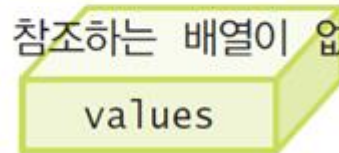
```
int[] ary1, ary2;
```



아직 참조하는 배열이 없음.



아직 참조하는 배열이 없음.



3. 배열

- 배열 생성

- 배열 객체 생성

- 배열 요소들과 크기(length) 필드 생성

- 생성 방법

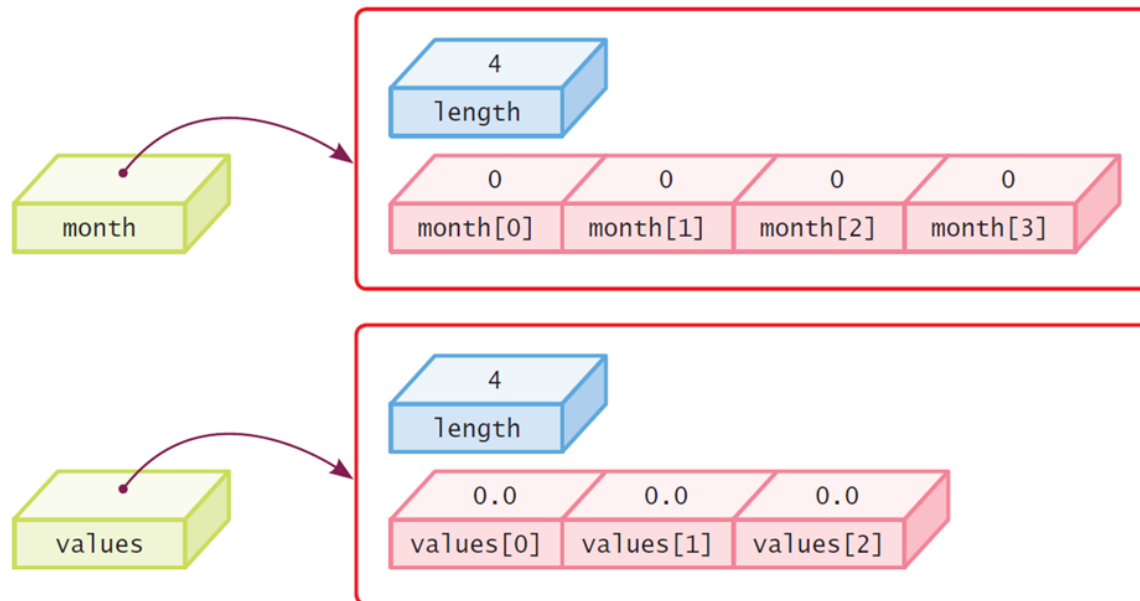
```
배열명 = new 자료형[크기];
```

3. 배열

- 배열 생성

- 배열 객체 생성 예

```
//배열 생성  
month = new int[4];  
values = new double[3];
```



<배열 객체 생성>

3. 배열

- 배열 (참조 변수) 선언 + 배열 (객체) 생성

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

```
double[] values = new double[3];
```

3. 배열

[예제 4-7] 배열 선언과 생성, 배열 크기 출력

```
public class ArrayDeclaration {  
    public static void main(String[] args) {  
        int[] month;  
        double[] values;  
  
        month = new int[4];  
        values = new double[3];  
  
        System.out.println(month.length);  
        System.out.println(values.length);  
  
        char[] ch = new char[6];  
        float[] data = new float[5];  
  
        data = new float[7];  
  
        System.out.println(ch.length);  
        System.out.println(data.length);  
    }  
}
```

실행 결과는?

3. 배열

- 배열 원소 참조

- 방법

```
배열명[첨자];
```

- 배열 첨자가 유효 범위를 벗어나면 실행 시 오류

☞ 첨자 범위: [0] ~ [크기-1]

```
double[] points = new double[3];
```

```
points[0] = 28;    //첨자가 0에서 3까지 유효
```

```
points[3] = 31;    //실행 시 오류 발생
```

<원소 참조와 첨자 범위 초과 예>

3. 배열

- 배열 초기화

- 선언과 동시에 초기화

- 대입 연산자와 중괄호 이용...

```
자료형 배열명[] = { 값, 값, 값, ... };
```

*new 키워드를
사용하지 않아도
객체 생성!*

배열 크기는 기술할 수 없다. 초기 값을 기술한 수인 3이 자동으로 배열 크기가 된다.

```
double dScore[] = {2.78, 4.28, 3.18};
```

```
String sbjt[] = {"국어", "영어", "수학"};
```

```
int[] data = {3, 4, 6}, values = {12, 82, 65};
```

```
double ary[] = {3.23, 5.24, 9.67}, d = 3.2678; //d는 double 기본형
```

<배열 선언 초기화>

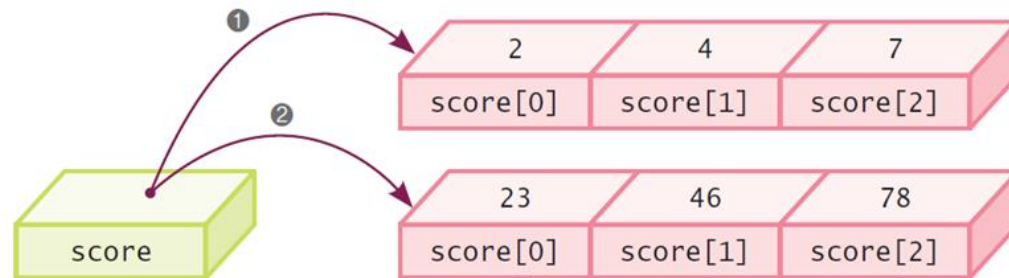
3. 배열

- 배열 초기화

- 선언 이후 초기화

- 초기 값을 위한 중괄호 앞에 **new 자료형[]** 기술

```
int score[];  
score = {2, 4, 7};           //오류발생  
① score = new int[] {2, 4, 7}; //배열 선언 이후 초기화 방법  
...  
② score = new int[] {23, 46, 78}; //다른 배열을 생성하여 대입
```

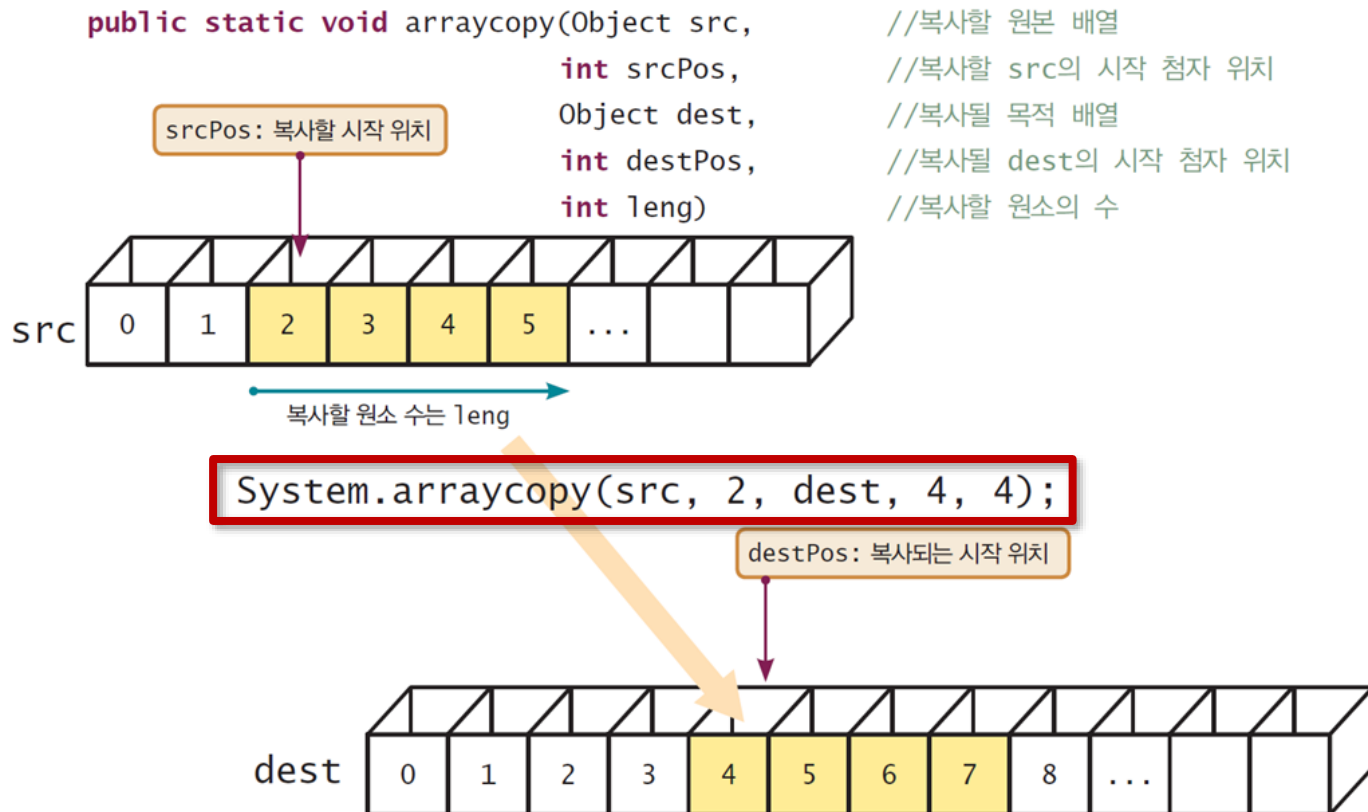


<배열 선언 이후 초기화>

3. 배열

- 배열 복사

- System 클래스의 arraycopy 메소드 사용



3. 배열

- 배열 원소 출력

- for 사용

- for 문의 조건식에 `배열명.length` 사용

- 예

```
int[] copyFrom = {1, 2, 3, 4, 5, 6, 7};
```

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

3. 배열

- 배열 원소 출력

- for each 사용

- for each 는 첨자를 사용하지 않아도 원소 참조 가능

- 참조 순서는 첫 원소부터 순차적으로...

- 문장 구조

```
for (int value : copyTo)  
    System.out.print(value + " ");
```

```
for (배열 원소자료형 배열 원소가저장되는변수이름 : 배열 이름) {  
    ... value ...  
}
```

3. 배열

[예제 4-11] 배열 복사 및 원소 출력

```
public class Arraycopy {  
    public static void main(String[] args) {  
        int[] copyFrom = {1, 2, 3, 4, 5, 6, 7};  
        int[] copyTo = {10, 20, 30, 40, 50, 60, 70, 80};  
        System.arraycopy(copyFrom, 4, copyTo, 1, 3);  
  
        for (int i=0; i < copyFrom.length; i++)  
            System.out.print(copyFrom[i] + " ");  
  
        System.out.println();  
  
        for (int value : copyTo)  
            System.out.print(value + " ");  
    }  
}
```

3. 배열

- 이차원 배열

- 행(row)과 열(column)로 이루어진 테이블 형태로 원소 생성
- 참조 변수 선언 방법
 - 빈 대괄호 2개 사용...

```
자료형 배열명[][];
```

OR

```
자료형 [][]배열명;
```

OR

```
자료형 []배열명[];
```

3. 배열

- 이차원 배열

- 객체 생성 방법

```
배열명 = new 자료형[행크기][열크기];
```

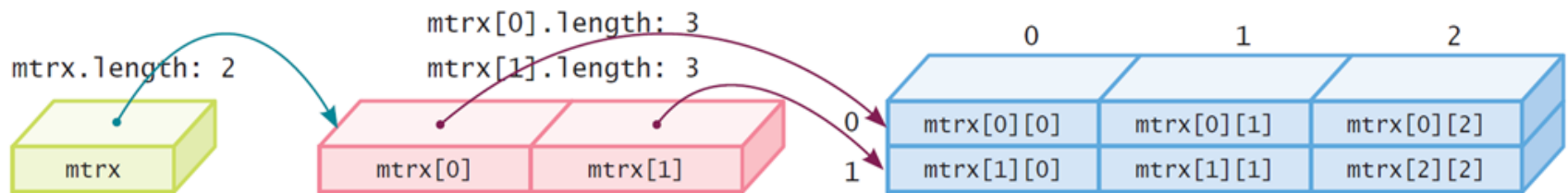
- 원소 참조 방법

```
배열명[행첨자][열첨자];
```

3. 배열

- 이차원 배열
 - 선언과 생성 예

```
int mtrx[] [];  
mtrx = new int[2][3];
```



3. 배열

[예제 4-12] 이차원 배열 원소 참조 및 출력

```
public class MultiDimArray {  
    public static void main(String[] args) {  
        int mtrx[][];  
        mtrx = new int[2][3];  
  
        mtrx[0][0] = 3; mtrx[0][1] = 5; mtrx[0][2] = 0;  
        mtrx[1][0] = 7; mtrx[1][1] = 2; mtrx[1][2] = 8;  
  
        for (int i=0; i<mtrx.length; i++) {  
            for (int j=0; j<mtrx[i].length; j++)  
                System.out.print(mtrx[i][j] + " ");  
            System.out.println();  
        }  
    }  
}
```

mtrx.length → 행의 개수: 2

mtrx[0].length → 첫 번째 행의 열의 개수: 3

mtrx[1].length → 두 번째 행의 열의 개수: 3

3. 배열

- 이차원 배열 초기화

- 선언과 동시에 초기화

- 중괄호를 행 단위로 중복 사용...

*new 키워드를
사용하지 않아도
객체 생성!*

```
자료형 배열명[][] = {{값, 값, ... }, ..., {값, 값, ... }};
```

- 선언 이후 초기화

- 초기 값을 위한 중괄호 앞에 **new** 자료형[][] 기술

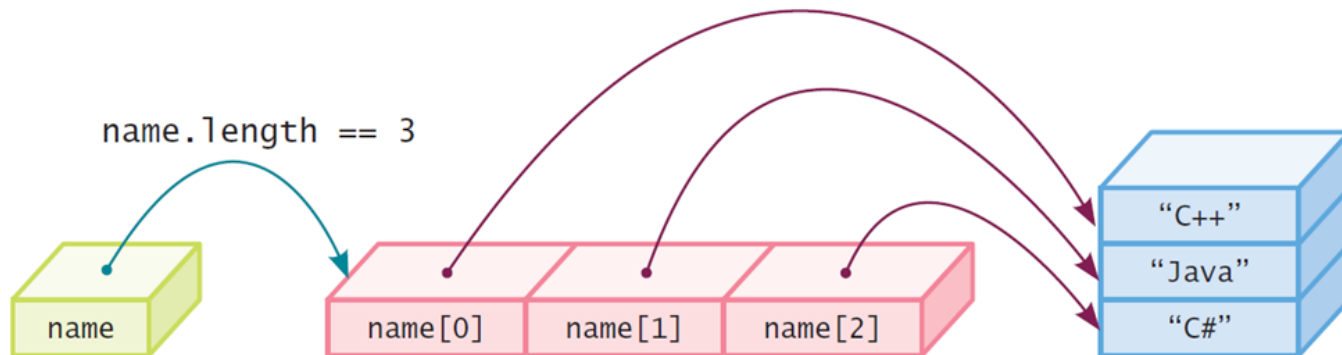
```
배열명[][] = new 자료형[][] {{값, 값, ... }, ..., {값, 값, ... }};
```

3. 배열

- 문자열 저장 배열

- 배열의 원소는 모든 자료형이 가능하므로...
- String형으로 배열 선언 및 생성
- 예

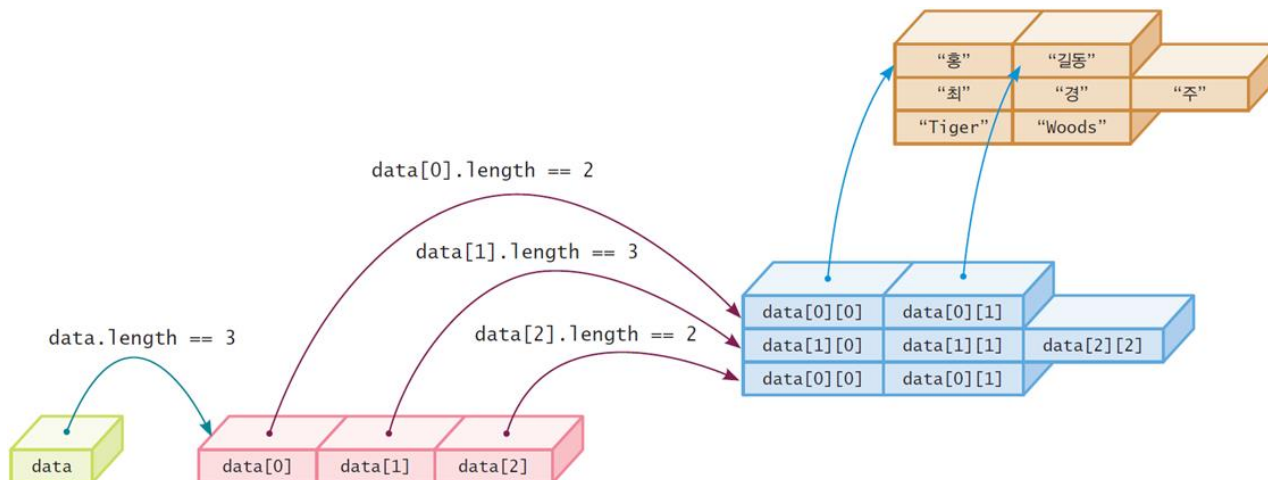
```
String name[] = {"C++", "Java", "C#"};
```



3. 배열

- 문자열 저장 배열
 - 문자열 이차원 배열
 - 예

```
String data[][] = {{ "홍", "길동"}, { "최", "경", "주"}, { "Tiger", "Woods" }};
```



행마다 열의 개수가 다른 배열은
래기드 배열(ragged array)이라고 함

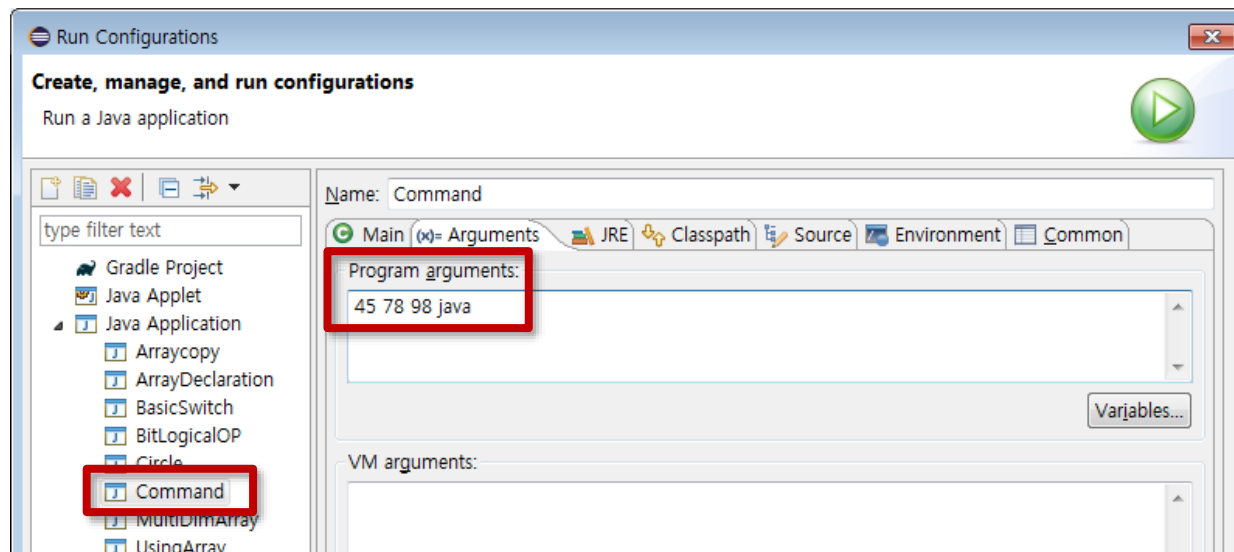
3. 배열

- [예제 4-14] 일차원 및 이차원 문자열 저장 배열 출력

```
public class UsingArray {  
    public static void main(String[] args) {  
        String name[] = {"C++", "Java", "C#"};  
  
        for (String str : name)  
            System.out.print(str + " ");  
        System.out.println();  
  
        String data[][] = {{ "홍", "길동"}, {"최", "경", "주"}, {"Tiger", "Woods"}};  
  
        for (String ary[] : data) {  
            for (String value : ary)  
                System.out.print(value + " ");  
            System.out.println();  
        }  
    }  
}
```

3. 배열

- 명령형 인자(command line arguments)
 - 프로그램 실행 시 입력한 인자는 main 메소드의 args에 저장
 - 이클립스에서 명령형 인자 입력
 1. Run > Run Configurations
 2. Java Application > Command
 3. Arguments > Program arguments



3. 배열

- [예제 4-16] 명령행 인자 출력

```
public class Command {  
    public static void main(String[] args) {  
        System.out.println("명령행 인자 개수 > " + args.length);  
  
        for (String value: args)  
            System.out.print(value + " ");  
    }  
}
```

Exercise

[4-1] 다음 프로그램의 출력 결과는?

```
public class Ex {  
    public static void main(String[] args) {  
        int n = 10;  
  
        for (int i = 1; i <= n; i++) {  
            int mult = 1;  
            for (int j = 1; j <= i; j++) {  
                System.out.printf("%d", j);  
                String str = j == i ? " = " : " * ";  
                System.out.printf("%s", str);  
                mult *= j;  
            }  
            System.out.printf("%d\n", mult);  
        }  
    }  
}
```

Exercise

[4-1] 정답

$$1 = 1$$

$$1 * 2 = 2$$

$$1 * 2 * 3 = 6$$

$$1 * 2 * 3 * 4 = 24$$

$$1 * 2 * 3 * 4 * 5 = 120$$

$$1 * 2 * 3 * 4 * 5 * 6 = 720$$

$$1 * 2 * 3 * 4 * 5 * 6 * 7 = 5040$$

$$1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 = 40320$$

$$1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 = 362880$$

$$1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10 = 3628800$$

Exercise

[4-2] 다음을 출력하는 프로그램을 중첩된 for 문을 이용하여 작성하시오.

```
0
101
21012
3210123
432101234
54321012345
6543210123456
765432101234567
```

Exercise

[4-2] 코드

```
public class Ex {  
    public static void main(String[] args) {  
        int i, j;  
        final int MAX = 7;  
  
        for (i = 0; i <= MAX; i++) {  
            for (j = MAX; j >= -MAX; j--) {  
                if (Math.abs(j) > i) System.out.print(' ');  
                else System.out.printf("%d", Math.abs(j));  
            }  
            System.out.println();  
        }  
    }  
}
```