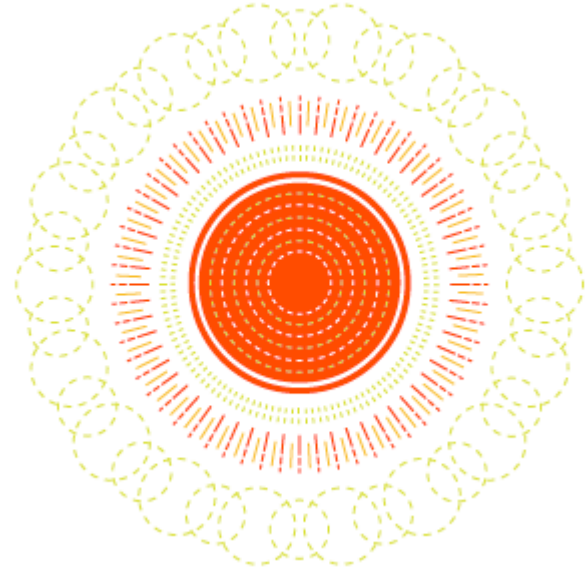


# JUST JAVA



객체지향에 충실한 자바다운 자바

## Chapter 03. 자바의 기초 문법

# 목차

01 변수와 자료형

02 연산자

03 분기문

04 반복문

05 배열

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 01. 변수와 자료형

### ■ 변수의 개념과 선언 방법

- 프로그래밍 언어에서 변수는 다양한 정보를 저장할 수 있는 메모리를 참조하는 이름.
- 자바 컴파일러가 명시적으로 자료형을 선언한 변수만 처리할 수 있음. (여기서 '명시적'이란 말은 선언하는 변수가 정수인지 실수인지 자료형을 정확히 지정해야 한다는 의미)

```
x = 1.5  
y = 3
```

X

```
double x = 1.5;  
-----  
실수를 의미하는 자료형  
int y = 3;  
-----  
정수를 의미하는 자료형
```

O

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 01. 변수와 자료형

### ■ 변수의 개념과 선언 방법

- 클래스를 변수로 선언하는 방법

```
Product product;  
-----  
Product 클래스 타입의 참조 변수  
Product product = null;
```

- 다양한 형태의 선언 방법

```
• int a; // 변수만 선언  
• int a = 1; // 변수를 선언하고 동시에 초깃값 대입  
• int a, b, c; // 동일 자료형 변수를 한 번에 선언  
• int a, b, c = 1; // c 변수만 1로 초기화하고 a, b는 초깃값 대입 안 됨  
• int a = 1, b = 2, c = 3; // 동일 자료형 변수를 한 번에 선언하면서 서로 다른 값으로 초깃값 대입
```

# 01. 변수와 자료형

## ■ 변수 이름 규칙

- 변수 이름의 길이에 제한이 없다.
- 반드시 문자나 언더바(\_), 달러 기호(\$)로 시작해야 한다.
- 자바의 연산자(+, -, \*, /)는 변수 이름에 넣을 수 없다.
- 대소문자를 구분한다.
  - ✓ `int result`와 `int Result`는 다른 변수이다.
- 첫 글자에 숫자가 올 수 없고, 이름 사이에 빈칸을 넣어서도 안 된다. 빈칸을 넣고 싶다면 언더바(\_)를 사용한다.
  - ✓ `int 10Seconds`; (×) → `int TenSeconds`; (○)
  - ✓ `int Time Interval`; (×) → `int Time_Interval`; (○) 또는 `int TimeInterval`;
- 자바의 키워드는 변수 이름으로 사용할 수 없다.
  - ✓ `int class`; (×), `int public`; (×)

# 01. 변수와 자료형

## ■ 변수 이름 규칙

예제 3-1 변수를 선언하고 저장된 값 출력하기

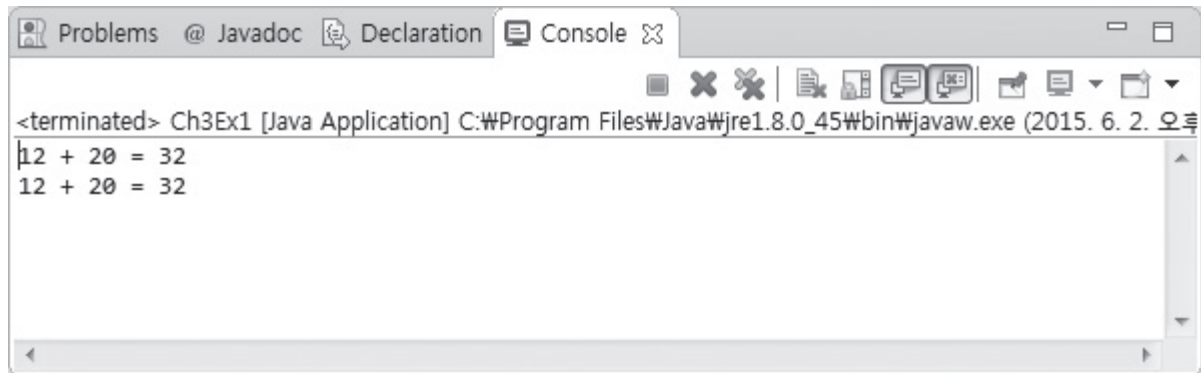
Ch3Ex1.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex1 {
04     public static void main(String[] args) {
05         int num1 = 12;
06
07         int num2 = 20;
08         int result = num1 + num2;
09
10         System.out.println(num1 + " + " + num2 + " = " + result);
11         System.out.printf("%d + %d = %d", num1, num2, result);
12     }
13 }
```

# 01. 변수와 자료형

## ■ 변수 이름 규칙

### [실행결과]



The screenshot shows a Java IDE's console window. The title bar includes tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The console output displays the following text:

```
<terminated> Ch3Ex1 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. 2. 오후 12 + 20 = 32  
12 + 20 = 32
```

# 01. 변수와 자료형

## ■ 변수 이름 규칙

### [코드설명]

표 3-1 printf()의 형식 문자와 자료형

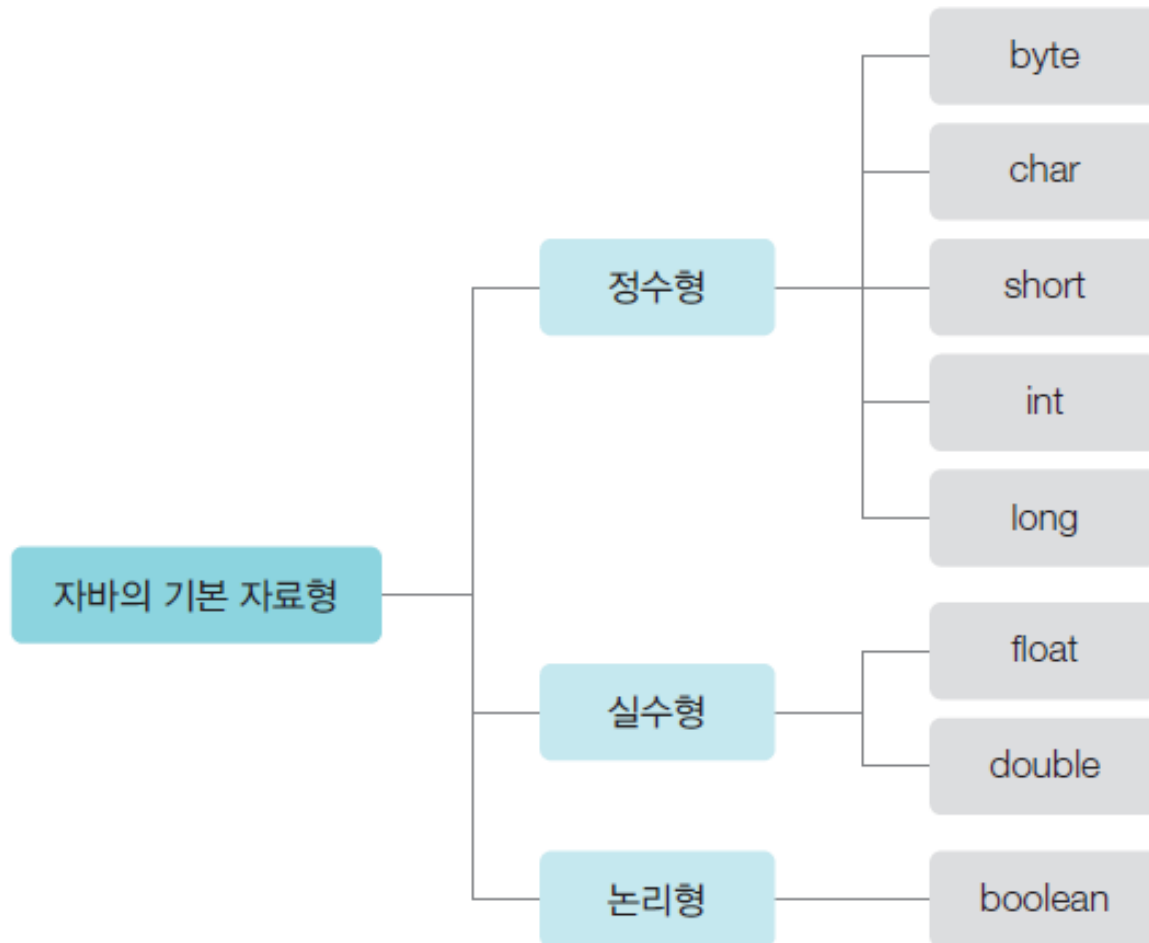
형식 문자	자료형	형식 문자	자료형
%c	문자형	%o	8진수
%d	정수형	%s	문자열
%e	지수형	%u	부호 없는 정수형
%f	실수형	%x	16진수
%i	정수형	%%, \w%	% 문자 출력



# 01. 변수와 자료형

## ■ 기본 자료형

그림 3-1 자바의 기본 자료형



# 01. 변수와 자료형

## ■ 기본 자료형 : 정수형

표 3-2 정수형에 해당하는 자료형과 변수 영역

자료형	크기	입출력 범위	설명
byte	1바이트	$-2^7 \sim 2^7 - 1$	작은 범위의 값을 저장하기에 유용하다. 컴퓨터 데이터 통신 프로그램에서 많이 사용한다.
char	2바이트	$0 \sim 2^{16} - 1$	음수를 표현하지 않는 unsigned 자료형으로, 문자를 저장하거나 출력하는 용도로 사용한다.
short	2바이트	$-2^{15} \sim 2^{15} - 1$	메모리에서 차지하는 크기가 작다는 것이 장점이지만 잘 사용하지 않는다.
int	4바이트	$-2^{31} \sim 2^{31} - 1$	정수 타입의 연산에 기본이 되는 자료형이다.
long	8바이트	$-2^{63} \sim 2^{63} - 1$	정수 표현 범위가 큰 데이터를 저장하기에 유용한 자료형이다.

# 01. 변수와 자료형

## ■ 기본 자료형 : 정수형 (뒷장 계속)

예제 3-2 정수형 자료형에 맞게 숫자 출력하기

Ch3Ex2.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex2 {
04     public static void main(String[] args) {
05         byte num1 = 'A';
06         int result;
07
08         short char1 = '}';
09         char char2 = 66;
10         long num2 = 9876543210L;
11     }
```

# 01. 변수와 자료형

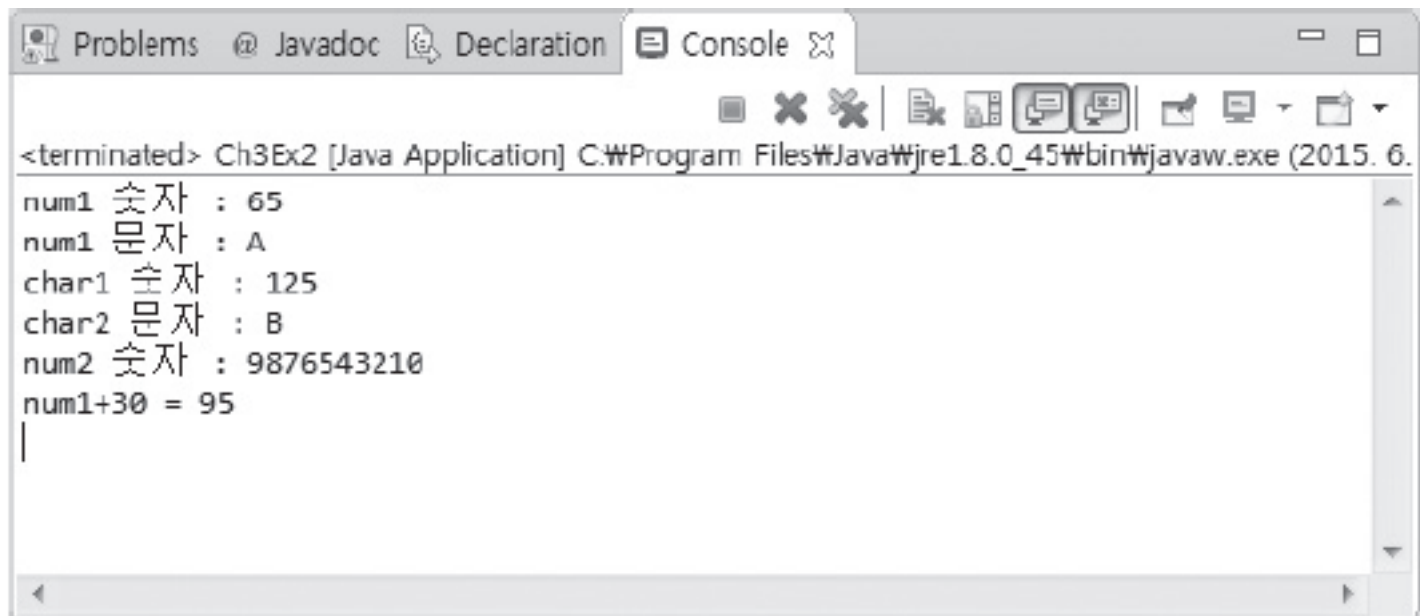
## ■ 기본 자료형 : 정수형

```
12      System.out.printf("num1 숫자 : %d \n", num1);
13      System.out.printf("num1 문자 : %c \n", num1);
14      System.out.printf("char1 숫자 : %d \n", char1);
15      System.out.printf("char2 문자 : %c \n", char2);
16      System.out.printf("num2 숫자 : %d \n", num2);
17
18      result = num1 + 30;
19      System.out.println("num1+30 = " + result);
20  }
21 }
```

# 01. 변수와 자료형

## ■ 기본 자료형 : 정수형

### [실행결과]



The screenshot shows a Java IDE's console window. The title bar includes tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The console output is as follows:

```
<terminated> Ch3Ex2 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6.  
num1 숫자 : 65  
num1 문자 : A  
char1 숫자 : 125  
char2 문자 : B  
num2 숫자 : 9876543210  
num1+30 = 95  
|
```

# 01. 변수와 자료형

## ■ 기본 자료형 : 논리형

- 논리형(boolean)은 1바이트 크기로 true값 또는 false값만 가짐.

```
• boolean b1 = true;  
• boolean b2 = FALSE;    // 틀림, 소문자만 사용
```

# 01. 변수와 자료형

## ■ 기본 자료형 : 실수형

- 실수형은 정수가 아닌 값을 부동소수점을 사용하여 저장.

표 3-3 실수형 자료형

자료형	크기	입출력 범위	설명
float	4바이트	$1.4E^{-45} \sim 3.402823E^{38}$	표현 범위가 작다. 값을 지정할 때 숫자 뒤에 f나 F를 붙여서 구분한다.
double	8바이트	$4.9E^{324} \sim 3.402823E^{308}$	실수형에서 사용하는 기본 데이터형이다.

# • 01. 변수와 자료형

## ■ 기본 자료형 : 실수형

- 정실수형의 기본 자료형이 double이므로 float를 사용할 때는 숫자 뒤에 f나 F를 붙여야 한다.
- 지수형 표현법은 숫자En 또는 숫자E+n 형식으로 사용하고, 의미는 숫자\*En이 된다.

```
• float f1 = 21.34;      // 오류 발생
• float f2 = 21.34F;
• double d1 = 21.34;
• double d2 = 21.34E5;   // 지수 표현
```



# 01. 변수와 자료형

## ■ 기본 자료형 : 논리형, 실수형을 사용한 예제

예제 3-3 논리형과 실수형을 사용한 예제

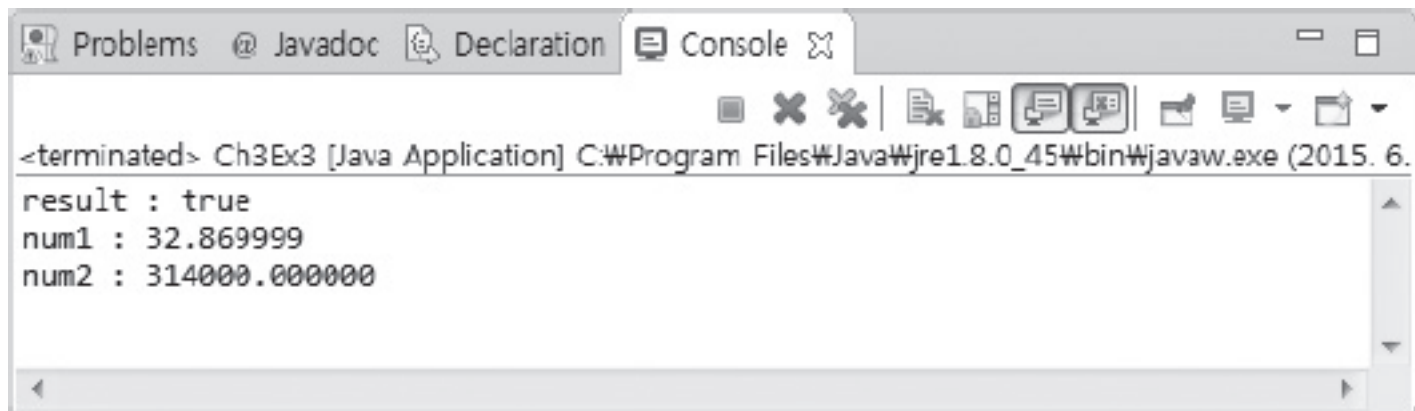
Ch3Ex3.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex3 {
04     public static void main(String[] args) {
05         boolean result = true;
06         float num1 = 32.87f;
07         double num2 = 3.14E5;
08
09         System.out.printf("result : %s \n", result);
10         System.out.printf("num1 : %f \n", num1);
11         System.out.printf("num2 : %f \n", num2);
12     }
13 }
```

# 01. 변수와 자료형

- 기본 자료형 : 논리형, 실수형을 사용한 예제

[실행결과]



The screenshot shows an IDE's console window with the following content:

```
<terminated> Ch3Ex3 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6.  
result : true  
num1 : 32.869999  
num2 : 314000.000000
```

# 01. 변수와 자료형

## ■ 자바 변수 유형

- **인스턴스 변수** : 자바 객체의 개별적인 값을 저장하는 데 사용하는 변수
- **클래스 변수** : static 한정자Modifier를 가진 멤버 변수로 '클래스 이름.변수 이름'의 형태로 접근
- **지역 변수** : 메서드 안에 선언하는 변수로, 해당 메서드 안에서만 접근할 수 있는 변수
- **파라미터** : 메서드의 인자로 전달하는 변수를 말하며, 기본적으로는 지역 변수에 준해서 사용

# 01. 변수와 자료형

## ■ 자바 변수 유형의 예

```
class TestClass {  
    int num1;  
    -----  
    인스턴스 변수  
    static num2;  
    -----  
    클래스 변수  
  
    void int sum(int num1) {  
        -----  
        파라미터  
        int result = this.num1 + num1;  
        -----  
        지역 변수  
        return result;  
    }  
}
```

## 02. 연산자

표 3-4 기본 연산자의 종류

분류	연산자	설명
형변환 연산자	(자료형)	하나의 항을 형변환하는 연산자이다. 괄호 안의 자료형으로 강제 형변환을 한다.
산술 연산자	+, -, *, /, %, +=, -=, *=, /=, %=	두 개의 항이나 왼쪽/오른쪽 항을 더하거나 빼는 등 가장 기본적인 연산을 하는 연산자이다.
관계 연산자	>, <, >=, <=, !=	대소 관계를 나타내는 연산자이다.
비트 연산자	&, ^,  , <<, >>, <<=, >>=, ^=, &=,  =	비트 단위로 연산한다. 주로 비트를 0과 1로 끄고 켜는 연산이나 비트 단위로 앞뒤로 이동하여 결과를 내는 연산자이다.
논리 연산자	&&,   , ! [조건식] ? [true] : [false]	두 항을 비교하여 참(true)과 거짓(false)의 결과를 연산하거나 조건에 따라 처리한다.

## 02. 연산자

### ■ 산술 연산자

표 3-5 산술 연산자의 종류

분류	연산자	설명
이항 연산자	+	두 수의 합을 구한다.
	-	두 수의 차를 구한다.
	*	두 수의 곱을 구한다.
	/	두 수를 나눈 몫을 구한다.
	%	두 수를 나눈 나머지를 구한다.
	+=	좌변과 우변을 더한 결과를 좌변에 대입한다.
	-=	좌변에서 우변을 뺀 결과를 좌변에 대입한다.
	%=	좌변에서 우변을 나눈 나머지를 좌변에 대입한다.
단항 연산자	++	변수값을 1증가시킨다.
	--	변수값을 1감소시킨다.

## 02. 연산자

### ■ 산술 연산자

자바 프로그램에서 산술 연산자를 표현하는 구문

- 산술 연산자에서 결과를 저장하는 변수는 항상 왼쪽에 위치

- `C` = A {산술 연산자} B;
- `C` = {단항 연산자} A 또는 B {단항 연산자};
- `C` {단항 연산자} = 상수;

## 02. 연산자

### ■ 산술 연산자 : 이항 연산과 단항 연산

- 이항 연산은 일반적인 연산의 형태로, 변수나 상수의 조합으로 계산

```
• int c2 = a * b;  
• int c3 = c2 + 1;
```

- 단항 연산은 연산을 수행하는 오른쪽이 하나의 변수나 상수로 구성

```
• int c3 = 3;  
• ++c3;           // c3 = c3 + 1과 같다.
```



## 02. 연산자

### ■ 산술 연산자의 기본 사용 예제

예제 3-4 다양한 산술 연산자를 사용하여 값 출력하기

Ch3Ex4.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex4 {
04     public static void main(String[] args) {
05         int num1 = 30;
06         int num2 = 14;
07
08         int result1 = num1 * num2;
09         int result2 = num1 % num2;
10
11         System.out.printf("result1 : %d \n", result1);
12         System.out.printf("result2 : %d \n", result2);
13         System.out.println("-----");
```

## 02. 연산자

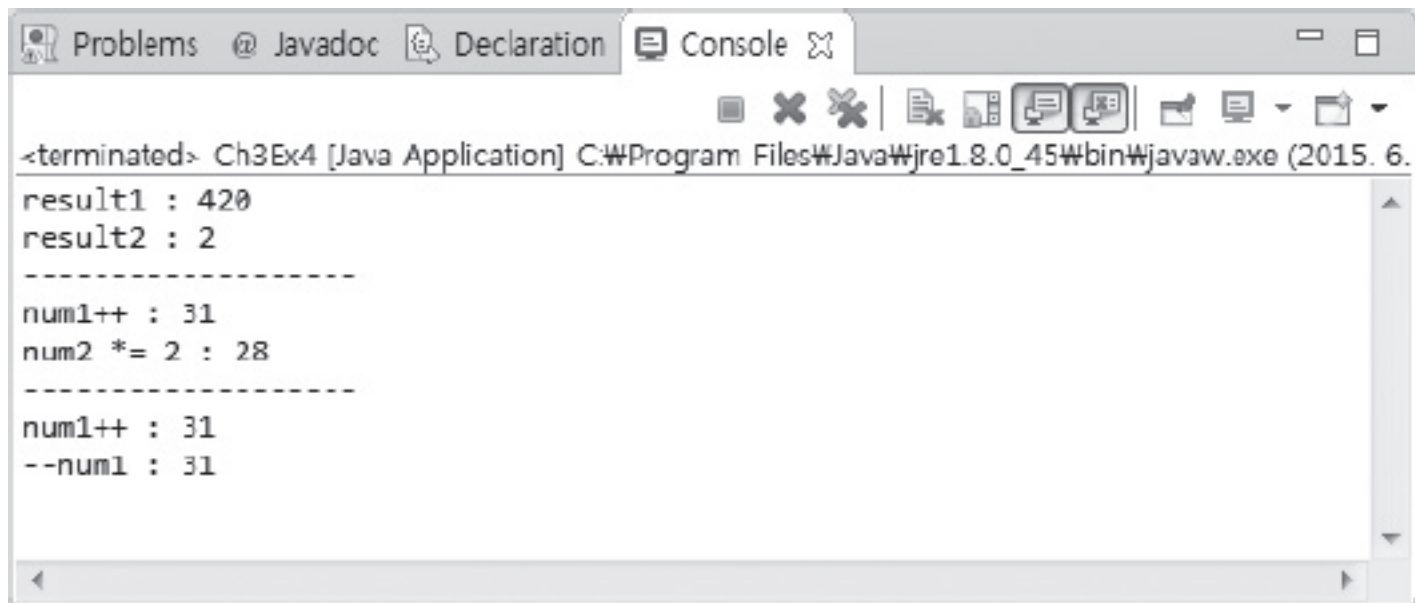
### ■ 산술 연산자의 기본 사용 예제

```
14
15     num1++;
16     num2 *= 2;
17     System.out.printf("num1++ : %d \n", num1);
18     System.out.printf("num2 *= 2 : %d \n", num2);
19     System.out.println("-----");
20
21     System.out.printf("num1++ : %d \n", num1++);
22     System.out.printf("--num1 : %d \n", --num1);
23 }
24 }
```

## 02. 연산자

### ■ 산술 연산자의 기본 사용 예제

#### [실행결과]



```
<terminated> Ch3Ex4 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6.  
result1 : 420  
result2 : 2  
-----  
num1++ : 31  
num2 *= 2 : 28  
-----  
num1++ : 31  
--num1 : 31
```

## 02. 연산자

### ■ 산술 연산자 : 산술 연산과 자료형

- 산술 연산을 수행할 때는 적절한 자료형을 사용하는 것이 매우 중요.

```
• int result = 202983738 + 903847563;    // 결과값 int 범위 초과
• int result = 5.0 / 2;                    // 실수와 연산한 결과는 실수이므로 오류
• int result = -100000000 * 100000000     // 음의 정수 영역 초과
```

## 02. 연산자

### ■ 산술 연산자 : 자동 형변환과 오버플로/언더플로

- 산술 연산을 수행할 때 기본적으로 피연산자 중에서 자료형이 큰 쪽을 따라 결과를 출력.

```
int num1 = 100;
double num2 = 100;
double result = num1 * num2;
System.out.println(num1 * num2);
```

- 연산에서 오버플로(Overflow)는 주어진 자료형의 범위를 넘을 때 발생하는 현상.
- 언더플로(Underflow)는 해당 자료형에서 처리할 수 있는 가장 작은 수보다 더 작은 결과를 처리할 때 발생.

```
int result = 10000000 * 10000000;
System.out.println(result);
```



## 02. 연산자

### ■ 산술 연산자 : 자동 형변환과 오버플로/언더플로

예제 3-5 자동 형변환과 오버플로 예제

Ch3Ex5.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex5 {
04     public static void main(String[] args) {
05         int num1 = 313;
06         long num2 = 15L;
07
08         long result1 = num1 * num2;
09         System.out.println("== 자동 형변환 ==");
10         System.out.printf("result1 : %d \n", result1);
11
12         long result2 = num1 / num2;
13         double result3 = num1 / num2;
14         double result4 = num1 / 15.0;
```

## 02. 연산자

### ■ 산술 연산자 : 자동 형변환과 오버플로/언더플로

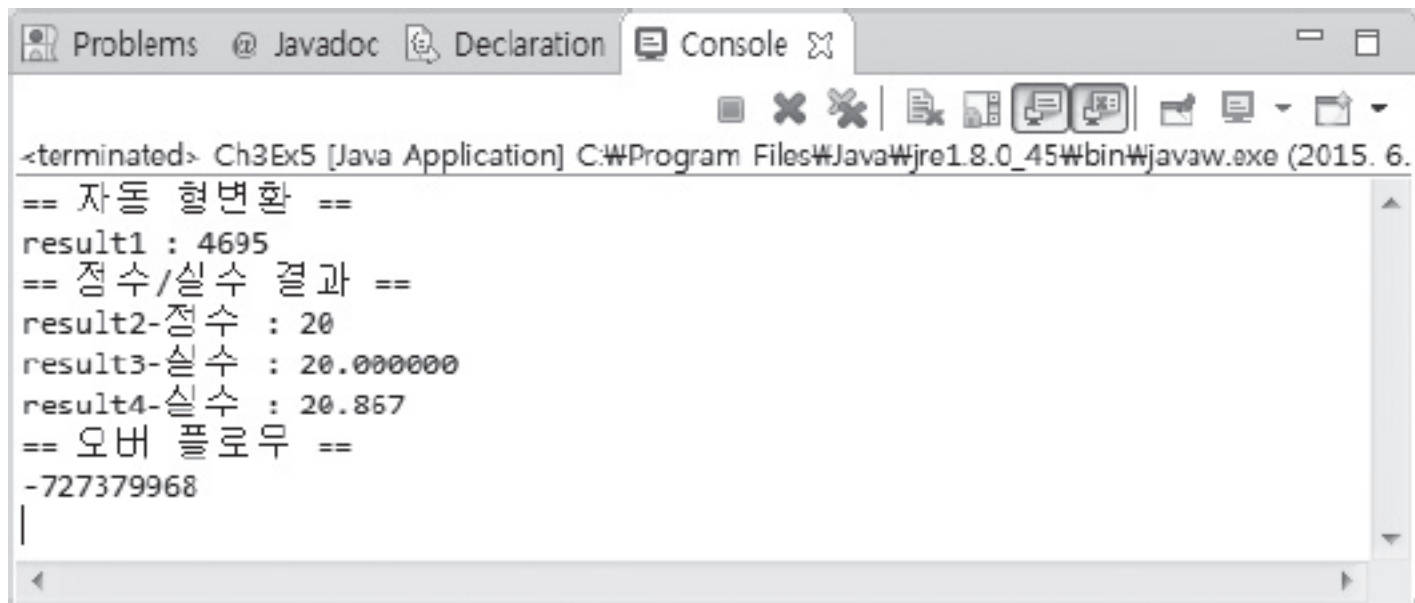
```
15
16     System.out.println("== 정수/실수 결과 ==");
17     System.out.printf("result2-정수 : %d \n", result2);
18     System.out.printf("result3-실수 : %f \n", result3);
19     System.out.printf("result4-실수 : %.3f \n", result4);
20
21     int result5 = 1000000 * 1000000;
22
23     System.out.println("== 오버 플로우 ==");
24     System.out.println(result5);
25 }
26 }
```



## 02. 연산자

- 산술 연산자 : 자동 형변환과 오버플로/언더플로

### [실행결과]



```
<terminated> Ch3Ex5 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. ...).
== 자동 형변환 ==
result1 : 4695
== 정수/실수 결과 ==
result2-정수 : 20
result3-실수 : 20.000000
result4-실수 : 20.867
== 오버 플로우 ==
-727379968
```

## 02. 연산자

### ■ 산술 연산자 : 연산자 우선순위

- 괄호가 있다면 괄호 안의 연산을 먼저 처리한다.
- 더하기와 빼기는 연산자 우선순위가 동일하므로 어떤 것을 먼저 계산하든지 결과가 같다. 다만 괄호가 없을 때는 왼쪽에서 오른쪽 방향으로 계산한다.
- 곱하기를 포함할 때는 계산 순서가 결과에 많은 영향을 미친다. 앞의 예에서 (a)와 (b)는 결과가 완전히 다르다. 이때는 괄호가 없어도 자동으로 곱하기와 나누기를 먼저 수행한 후 더하기와 빼기를 계산한다.
- 결과를 변수에 저장하는 대입 연산  $=$ 은 가장 나중에 처리한다.

## 02. 연산자

### ■ 관계 연산자

표 3-6 관계 연산자의 종류

연산자	설명
>	왼쪽 항이 크면 참(true), 아니면 거짓(false)을 내준다.
<	오른쪽 항이 크면 참(true), 아니면 거짓(false)을 내준다.
>=	왼쪽 항이 크거나 같으면 참(true), 아니면 거짓(false)을 내준다.
<=	오른쪽 항이 크거나 같으면 참(true), 아니면 거짓(false)을 내준다.
=	왼쪽과 오른쪽 항이 같으면 참(true), 아니면 거짓(false)을 내준다.
!=	왼쪽과 오른쪽 항이 다르면 참(true), 아니면 거짓(false)을 내준다.

## 02. 연산자

### ■ 관계 연산자

- 관계 연산자는 프로그램 안에서 다양한 논리(로직)를 풀어 나가는 데 유용.
- 주로 for 문, if문, while 문과 같은 제어문에서 실행이나 반복 조건에 사용.

```
if(max >= 100000000)
    이체 한도 초과 메시지 출력;
else
    이체 처리;
```

## 02. 연산자

### ■ 논리 연산자

표 3-7 논리 연산자의 종류

연산자	설명
&&	두 항(왼쪽과 오른쪽)의 논리값(true/false)이 참(true)이면 참(true)을, 아니면 거짓(false)을 내준다.
	두 항의 논리값(true/false) 중 하나 이상의 항이 참(true)이면 참(true)을, 아니면 거짓(false)을 내준다.
!	단항 연산을 하며, 연산되는 항이 참(true)이면 거짓(false)을, 거짓(false)이면 참(true)을 내준다.
[조건식] ? [true] : [false]	조건식의 결과가 참(true)일 때 [true] 항을 수행하고, 아니면 [false] 항을 수행한다.

## 02. 연산자

### ■ 논리 연산자

표 3-8 논리 연산 테이블

A	B	A && B	A    B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

## 02. 연산자

### ■ 논리 연산자 : AND 연산자

- AND 연산자는 &&로 표기하며, 각 항의 논리값이 모두 참(true)일 때만 참(true)을 반환.
- 프로그램에서는 제시된 여러 조건을 모두 만족하는지 검사하는 데 사용.

```
•boolean result = (3 < 2) && (4 > 1);    // true
•boolean result = (3 < 2) && (4 < 1);    // false
```

```
boolean memCheck = 제휴사 회원 && 신규;
if(memCheck)
    제휴 카드 발급;
else
    발급 거절;
```

## 02. 연산자

### ■ 논리 연산자 : OR 연산자

- ||로 표기하며, 각 항의 논리값 중 하나 이상이 참(true)이면 참(true)을 반환.
- 프로그램에서는 여러 조건 중 하나만 충족해도 가능한지 검사하는 데 사용.

```
•boolean result = (3 < 2) || (4 > 1);    // true
•boolean result = (3 > 2) || (4 < 1);    // false
```

```
boolean memCheck = 소득 5000만 원 이상 || 기존 고객 || 신용 등급 B+ 이상;
if(memCheck)
    제휴 카드 발급;
else
    발급 거절;
```



## 02. 연산자

### ■ 논리 연산자 : NOT 연산자

- NOT 연산자는 !로 표기하며, 결과값을 반대로 바꿔 주는 연산을 수행.
- 프로그램에서는 특정 조건이 아닌 경우에만 처리하는 목적으로 사용.

```
Boolean result = !(3 < 2);    // false
```

```
if(!기존 회원)
```

```
    카드 발급
```

```
search(거주지 = !서울);
```

## 02. 연산자

### ■ 논리 연산자 : 삼항 연산자

- if ~ then ~ else를 간단한 방법으로 대체할 수 있는 유용한 연산자.
- 삼항 연산자의 문법은 '[조건식] ? [명령어 1] : [명령어 2]'로 구성된다. 조건식이 참(true)이면 명령어 1을 수행하고, 조건식이 거짓(false)이면 명령어 2를 수행하라는 의미.

```
int i = 3;  
int result = (i > 2) ? i + 2 : i + 10; // (i > 2)가 참(true)이면 i + 2를, 거짓(false)  
                                         이면 i + 10을 수행
```

## 02. 연산자

### ■ 비트 연산자

표 3-9 비트 연산자의 종류

분류	연산자	설명
비트 논리	&	두 항의 비트가 모두 1이면 1, 아니면 0으로 연산한다.
		두 항의 비트가 모두 0이면 0, 아니면 1로 연산한다.
	^	두 항의 비트가 서로 다르면 1, 같으면 0으로 연산한다.
비트 시프트	>>	왼쪽 피연산자를 오른쪽 값만큼 부호 비트로 채우면서 오른쪽으로 이동시킨다.
	<<	왼쪽 피연산자를 오른쪽 값만큼 비트를 왼쪽으로 이동시킨다.
	>>>	왼쪽 피연산자를 오른쪽 값만큼 부호 비트를 무시한 채 0으로 채우면서 오른쪽으로 이동시킨다.

## 02. 연산자

### ■ 비트 연산자

비트 대입	<code>&amp;=</code>	두 항의 비트가 모두 1이면 1, 아니면 0으로 연산하여 왼쪽 피연산자에 대입한다.
	<code> =</code>	두 항의 비트가 모두 0이면 0, 아니면 1로 연산하여 왼쪽 피연산자에 대입한다.
	<code>^=</code>	두 항의 비트가 서로 다르면 1, 같으면 0으로 연산하여 왼쪽 피연산자에 대입한다.
	<code>&gt;&gt;=</code>	왼쪽 피연산자를 오른쪽 값만큼 부호 비트로 채우면서 오른쪽으로 이동한 후 대입한다.
	<code>&lt;&lt;=</code>	왼쪽 피연산자를 오른쪽 값만큼 비트를 왼쪽으로 이동한 후 대입한다.

## 02. 연산자

### ■ 비트 연산자 : 비트 논리 연산자

- 비트의 논리 연산자인 `&`, `|`은 각각 산술 논리 연산자 `&&`, `||`, 즉 AND 연산에 대응.
- 논리 연산 과정에서 두 항의 데이터를 비트 단위로 수행.

그림 3-3 비트 연산 수행 과정

`int a = 10;`

0	0	...	0	0	0	1	0	1	0
---	---	-----	---	---	---	---	---	---	---

`&`

`int b = 15;`

0	0	...	0	0	0	1	1	1	1
---	---	-----	---	---	---	---	---	---	---

`=`

결과 : 10

0	0	...	0	0	0	1	0	1	0
---	---	-----	---	---	---	---	---	---	---

그림 3-4 XOR 연산 수행 과정

`int a = 10;`

0	0	...	0	0	0	1	0	1	0
---	---	-----	---	---	---	---	---	---	---

`^`

`int b = 15;`

0	0	...	0	0	0	1	1	1	1
---	---	-----	---	---	---	---	---	---	---

`=`

결과 : 10

0	0	...	0	0	0	0	1	0	1
---	---	-----	---	---	---	---	---	---	---

## 02. 연산자

### ■ 비트 연산자 : 비트 논리 연산자 (뒷장 계속)

#### 예제 3-6 XOR을 이용하여 비밀번호 암호화하기

Ch3Ex6.java

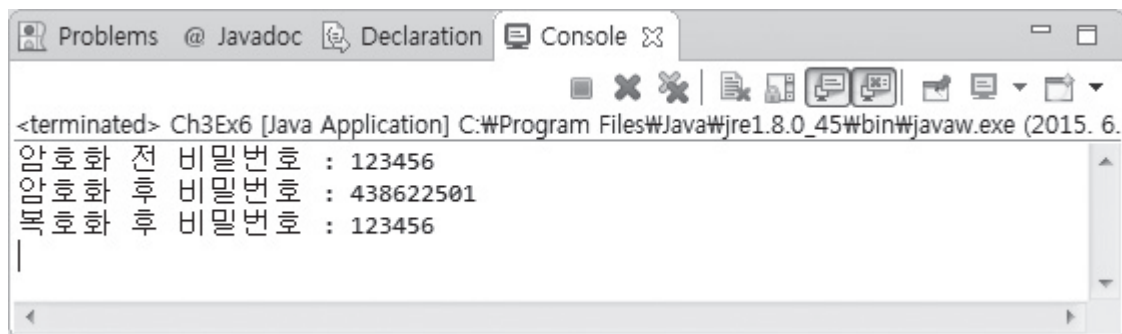
```
01 package javabook.ch3;
02
03 public class Ch3Ex6 {
04     public static void main(String[] args) {
05         int pwd = 123456;
06         int encPwd, decPwd;
07         int key = 0x1A253B65;
08
09         System.out.println("암호화 전 비밀번호 : " + pwd);
10
11         encPwd = pwd ^ key;
12
13         System.out.println("암호화 후 비밀번호 : " + encPwd);
14     }
```

## 02. 연산자

### ■ 비트 연산자 : 비트 논리 연산자

```
15         decPwd = encPwd ^ key;
16
17         System.out.println("복호화 후 비밀번호 : " + decPwd);
18     }
19 }
```

#### [실행결과]



```
<terminated> Ch3Ex6 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6.
암호화 전 비밀번호 : 123456
암호화 후 비밀번호 : 438622501
복호화 후 비밀번호 : 123456
```

## 02. 연산자

### ■ 비트 연산자 : 비트 시프트 연산자

- 모든 비트를 오른쪽이나 왼쪽으로 이동(시프트)시키는 것
- 정수의 모든 비트를 오른쪽으로 한 칸 이동시키는 것은 2로 나누는 것과 같고, 왼쪽으로 한 칸 이동시키는 것은 2를 곱하는 것과 같다.

그림 3-5 20  $\gg$  2의 연산 과정

▶ 20의 2진수 비트 구조

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

➡ 오른쪽으로 두 번 시프트

▶ 20의 2진수 비트 구조, 결과적으로  $20 / 2^2 = 5$ 와 같다.

0	0	0	0	0	1	0	1
---	---	---	---	---	---	---	---

그림 3-6 20  $\ll$  2의 연산 과정

▶ 20의 2진수 비트 구조

0	0	0	1	0	1	0	0
---	---	---	---	---	---	---	---

➡ 왼쪽으로 두 번 시프트

▶ 80의 2진수 비트 구조, 결과적으로  $20 * 2^2 = 80$ 과 같다.

0	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---



## 02. 연산자

### ■ 비트 연산자 : 비트 시프트 연산자

예제 3-7 비트 시프트를 사용하여 연산하기

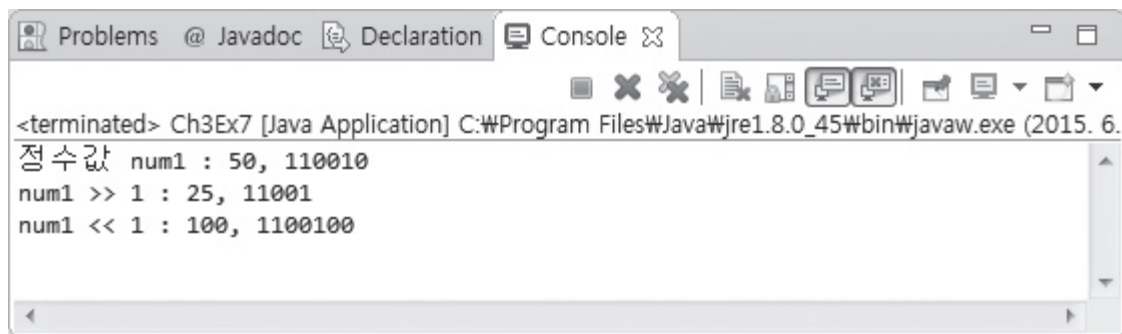
Ch3Ex7.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex7 {
04     public static void main(String[] args) {
05         int num1 = 50;
06         System.out.printf("정수값 num1 : %d, %s\n", num1, Long.toBinaryString(num1));
07
08         int result1 = num1 >> 1;
09         System.out.printf("num1 >> 1 : %d, %s\n", result1, Long.toBinaryString(result1));
10
11         int result2 = num1 << 1;
12         System.out.printf("num1 << 1 : %d, %s", result2, Long.toBinaryString(result2));
13     }
14 }
```

## 02. 연산자

### ■ 비트 연산자 : 비트 시프트 연산자

#### [실행결과]



```
<terminated> Ch3Ex7 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. 10.)
정수값 num1 : 50, 110010
num1 >> 1 : 25, 11001
num1 << 1 : 100, 1100100
```

## 02. 연산자

### ■ 연산자 우선순위

표 3-10 연산자 우선순위

우선순위	연산자
1	., [], ()
2	!, ~, +/-, ++/--, (cast)
3	+, -, *, /, %
4	<<, >>, >>>
5	<, >, <=, >=, ==, !=
6	&, ^,
7	&&,
8	[조건식] ? [true] : [false]
9	=, +=, -=, *=, /=, %=, <=, >=, ^=, &=, !=
10	++/-- (후위형 증감 연산자)

## 03. 분기문

### ■ if 문

- if ~ else 문, if ~ else if 문 등의 형태로 사용

### ■ if 문 : if 문 단독 사용 (뒷장 계속)

**사용 분야** 특정 조건만 처리하는 로직이 필요할 때 사용한다.

**사용 예**

- 프로그램에서 관리자로 로그인하면 일반인에게는 보이지 않는 추가 메뉴 출력
- 우수 회원에게는 추가 할인 적용

---

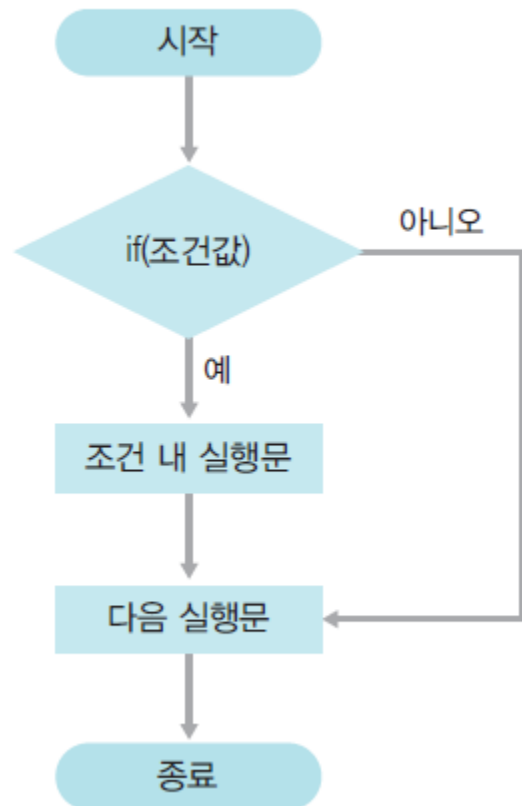
#### 기본 구조

```
if(조건값) {  
    명령문;  
}
```

## 03. 분기문

### ■ if 문 : if 문 단독 사용

순서도



## 03. 분기문

### ■ if 문 : if 문 단독 사용 예제 (뒷장 계속)

예제 3-8 if 문을 사용하여 메뉴 선택 처리하기

Ch3ExX1.java

```
01 package javabook.ch3;
02
03 import java.util.Scanner;
04
05 public class Ch3ExX1 {
06     public static void main(String[] args) {
07         Scanner scan = new Scanner(System.in);
```

## 03. 분기문

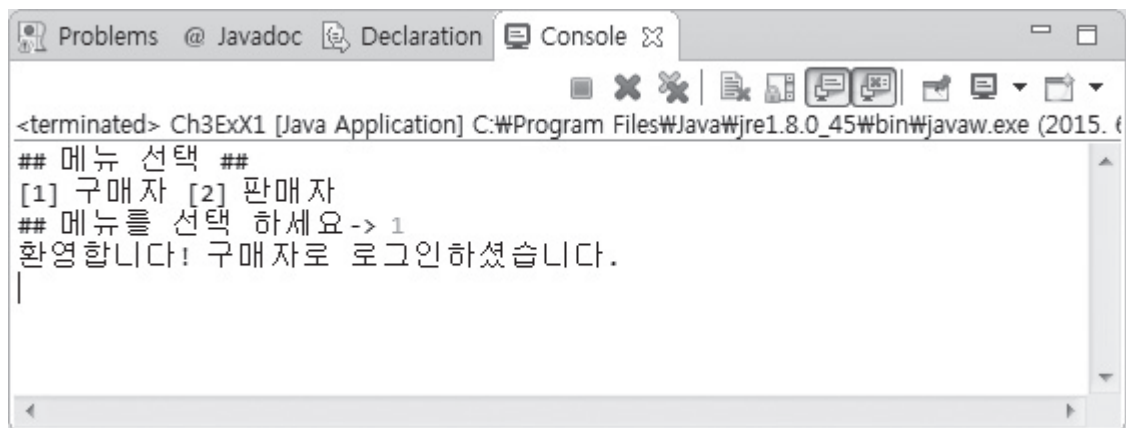
### ■ if 문 : if 문 단독 사용 예제

```
08
09     System.out.println("## 메뉴 선택 ##");
10     System.out.println("[1] 구매자 [2] 판매자");
11     System.out.print("## 메뉴를 선택 하세요-> ");
12
13     String sel = scan.next();
14
15     if(sel.equals("1")) {
16         System.out.println("환영합니다! 구매자로 로그인하셨습니다.");
17     }
18     if(sel.equals("2")) {
19         System.out.println("환영합니다! 판매자로 로그인하셨습니다.");
20     }
21 }
22 }
```

## 03. 분기문

### ■ if 문 : if 문 단독 사용 예제

#### [실행결과]



The screenshot shows a Java IDE's console window. The title bar includes tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The console text is as follows:

```
<terminated> Ch3ExX1 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. 10. 14:54:12)  
## 메뉴 선택 ##  
[1] 구매자 [2] 판매자  
## 메뉴를 선택 하세요 -> 1  
환영합니다! 구매자로 로그인하셨습니다.  
|
```



## 03. 분기문

### ■ if 문 : if ~ else 문

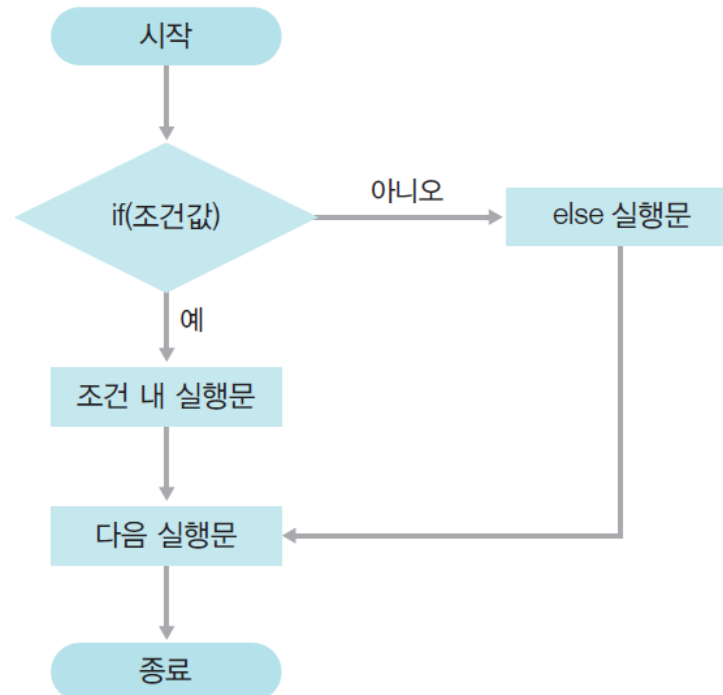
**사용 분야** 특정 조건이 참(true)일 때 처리해야 할 일과 거짓(false)일 때 처리해야 할 일이 다르다면 사용한다.

**사용 예**     사용자로 로그인할 때 아이디와 비밀번호가 맞으면 메인 화면을 표시하고, 그렇지 않으면 오류 메시지를 출력한 후 다시 로그인 화면 표시

기본 구조

```
if(조건값) {  
    명령문;  
}  
else {  
    명령문;  
}
```

순서도



## 03. 분기문

### ■ if 문 : if ~ else 문 예제 (뒷장 계속)

예제 3-9 if ~ else 문을 사용하여 메뉴 처리하기

Ch3ExX2.java

```
01 package javabook.ch3;
02
03 import java.util.Scanner;
04
05 public class Ch3ExX2 {
06     public static void main(String[] args) {
07         Scanner scan = new Scanner(System.in);
08
09         System.out.println("## 메뉴 선택 ##");
10         System.out.println("[1] 구매자 [2] 판매자");
11         System.out.print("## 메뉴를 선택 하세요-> ");
12
13         String sel = scan.next();
14     }
```

## 03. 분기문

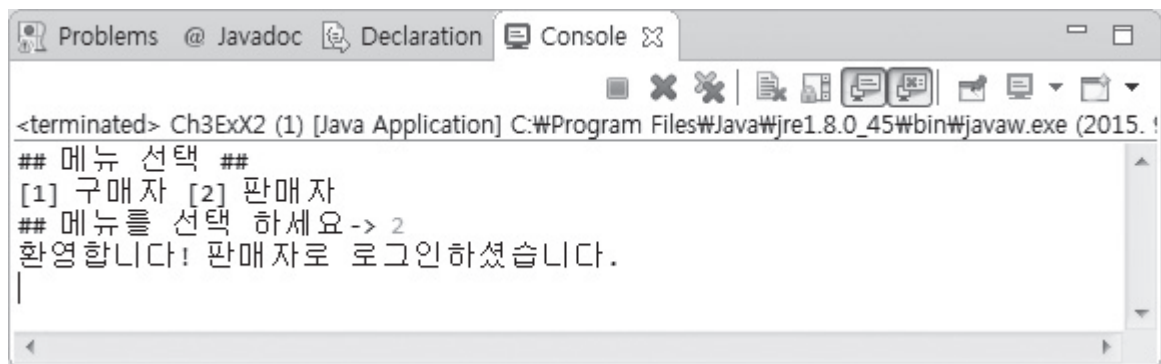
### ■ if 문 : if ~ else 문 예제

```
15         if(sel.equals("1")) {  
16             System.out.println("환영합니다! 구매자로 로그인하셨습니다.");  
17         }  
18         else {  
19             System.out.println("환영합니다! 판매자로 로그인하셨습니다.");  
20         }  
21     }  
22 }
```

## 03. 분기문

### ■ if 문 : if ~ else 문 예제

#### [실행결과]



```
<terminated> Ch3ExX2 (1) [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. !
## 메뉴 선택 ##
[1] 구매자 [2] 판매자
## 메뉴를 선택 하세요 -> 2
환영합니다! 판매자로 로그인하셨습니다.
|
```

## 03. 분기문

### ■ if 문 : if ~ else if ~ else 문

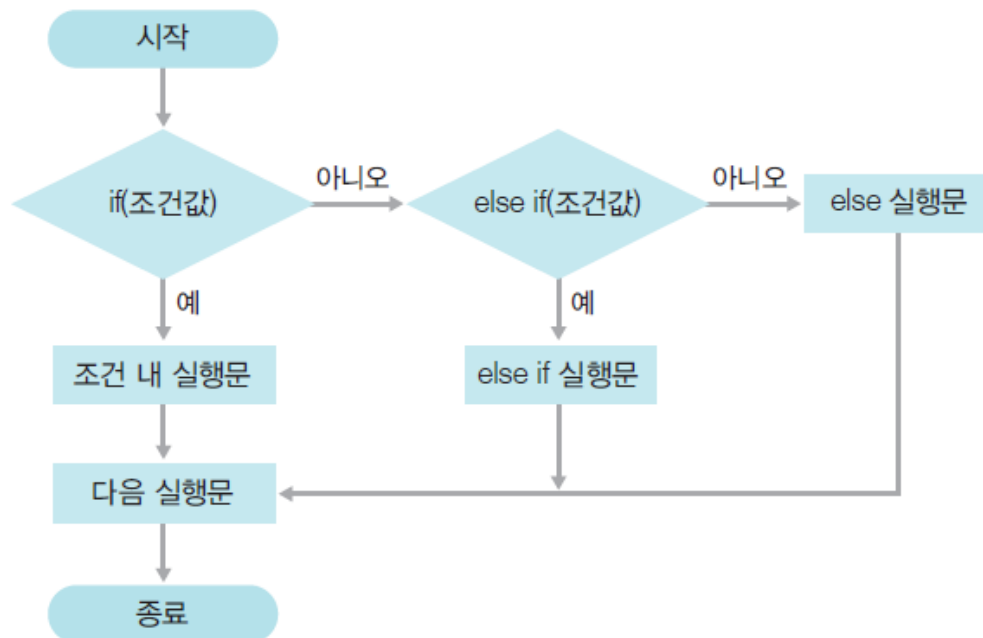
**사용 분야** 여러 조건이 있고, 각 조건마다 처리해야 하는 내용이 다를 때 사용한다. 조건이 세분화되어 있으면 else if 문을 여러 개 사용할 수 있다.

**사용 예** 쇼핑몰 회원을 네 등급으로 나누고, 등급별로 할인율이나 쿠폰을 다르게 적용

#### 기본 구조

```
if(조건값 1) {  
    명령문;  
}  
else if(조건값 2) {  
    명령문;  
}  
else if(조건값 3) {  
    명령문;  
}  
.....  
else {  
    명령문;  
}
```

#### 순서도



## 03. 분기문

### ■ if 문 : if ~ else if ~ else 문 예제 (뒷장 계속)

예제 3-10 여러 if 문을 응용한 로그인과 메뉴 선택하기

Ch3Ex8.java

```
01 package javabook.ch3;
02
03 import java.util.Scanner;
04
05 public class Ch3Ex8 {
06     public static void main(String[] args) {
07         Scanner scan = new Scanner(System.in);
08         System.out.println("## 로그인(admin 혹은 임의 아이디) ##");
09         System.out.print("# 로그인 아이디 : ");
10
11         String user = scan.next();
12
```

## 03. 분기문

### ■ if 문 : if ~ else if ~ else 문 예제 (뒷장 계속)

```
13         if(user.equals("admin")) {
14             System.out.println("관리자 로그인!!");
15         }
16         else {
17             System.out.println(user + " 로그인!!");
18         }
19
20         System.out.println("## 메뉴를 선택 하세요(1~2) ##");
21         System.out.print("# 메뉴 선택 : ");
22
23         String sel = scan.next();
24
25         if(sel.equals("1") && user.equals("admin")) {
26             System.out.println("관리자 1번 선택함!!");
27         }
```

## 03. 분기문

### ■ if 문 : if ~ else if ~ else 문 예제

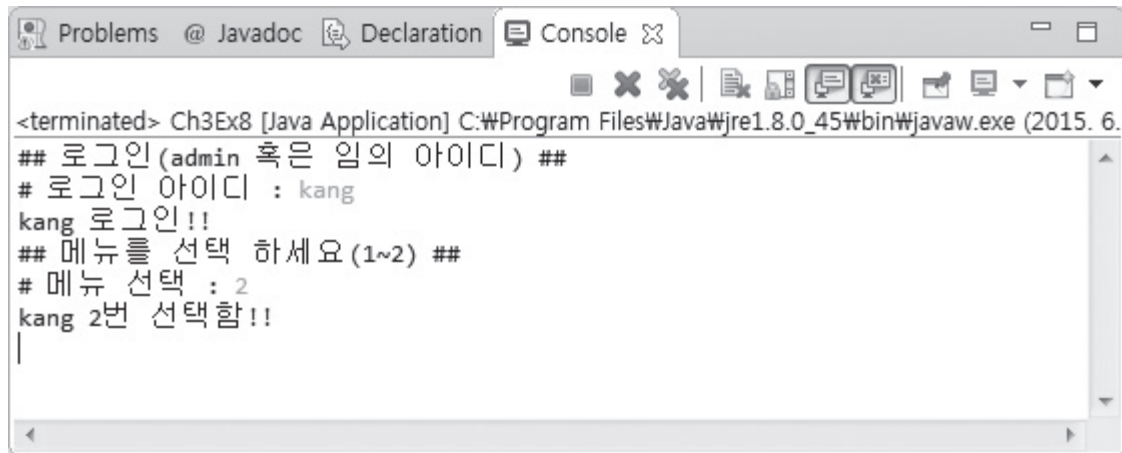
```
28         else if(sel.equals("2") && user.equals("admin")) {
29             System.out.println("관리자 2번 선택함!!");
30         }
31         else if(sel.equals("1") && !user.equals("admin")) {
32             System.out.println(user + " 1번 선택함!!");
33         }
34         else if(sel.equals("2") && !user.equals("admin")) {
35             System.out.println(user + " 2번 선택함!!");
36         }
37     }
38 }
```



## 03. 분기문

### ■ if 문 : if ~ else if ~ else 문 예제

#### [실행결과]



```
<terminated> Ch3Ex8 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6.
## 로그인 (admin 혹은 임의 아이디) ##
# 로그인 아이디 : kang
kang 로그인!!
## 메뉴를 선택 하세요 (1~2) ##
# 메뉴 선택 : 2
kang 2번 선택함!!
|
```

## 03. 분기문

### ■ if 문 : 중첩 if 문

**사용 분야** 특정 조건이 성립되어 또 다른 조건들을 연속적으로 체크해야 할 때 사용한다.

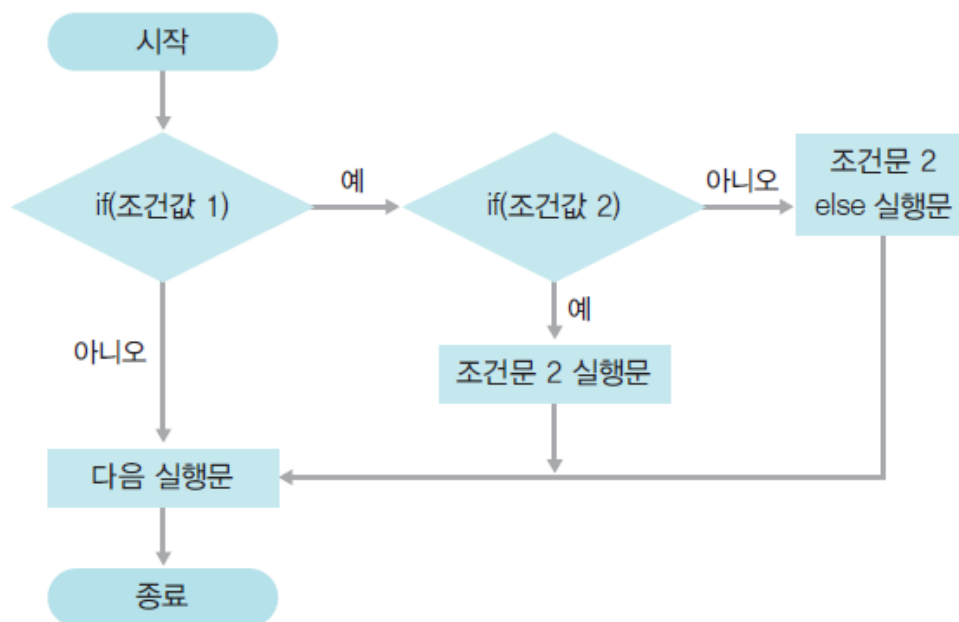
**사용 예**

- 회원 로그인할 때 아이디가 맞는지 먼저 비교한 후 맞으면 비밀번호 비교
- 신용카드를 재발급할 때 신원 확인을 완료한 후 신용도를 비롯한 재발급 조건 체크

#### 기본 구조

```
if(조건값 1) {  
    명령문;  
    if(조건값 2) {  
        명령문;  
    }  
}  
else {  
    명령문;  
    if(조건값 3) {  
        명령문;  
    }  
}
```

#### 순서도



## 03. 분기문

### ■ if 문 : 중첩 if 문 예제 (뒷장 계속)

예제 3-11 중첩 if 문을 이용하여 아이디와 비밀번호 확인하기

Ch3Ex9.java

```
01 package javabook.ch3;
02
03 import java.util.Scanner;
04
05 public class Ch3Ex9 {
06     public static void main(String[] args) {
07         Scanner scan = new Scanner(System.in);
08         System.out.print("## 아이디를 입력하세요: ");
09
10         String uname = scan.next();
11
12         if(uname.equals("hong")) {
13             System.out.print("# 비밀번호를 입력하세요: ");
14
15             String pwd = scan.next();
```

## 03. 분기문

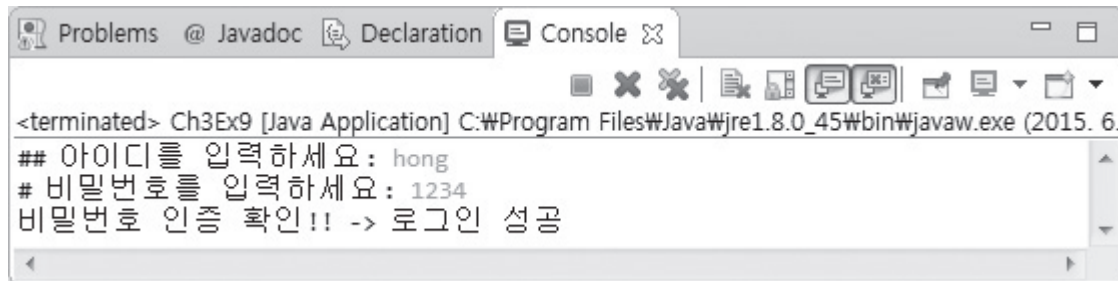
### ■ if 문 : 중첩 if 문 예제 (뒷장 계속)

```
16
17         if(pwd.equals("1234")) {
18             System.out.println("비밀번호 인증 확인!! -> 로그인 성공");
19         }
20         else
21             System.out.println("비밀번호가 틀렸습니다.!!");
22     }
23     else
24         System.out.println("아이디가 틀렸습니다.!!");
25 }
26 }
```

## 03. 분기문

### ■ if 문 : 중첩 if 문 예제 (뒷장 계속)

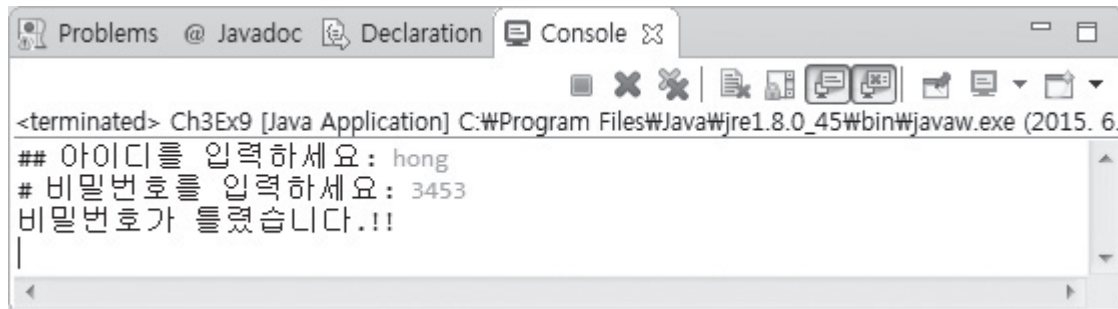
#### [실행결과]



The screenshot shows an IDE console window with the following text:

```
<terminated> Ch3Ex9 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6.  
## 아이디를 입력하세요 : hong  
# 비밀번호를 입력하세요 : 1234  
비밀번호 인증 확인!! -> 로그인 성공
```

비밀번호가 맞지 않을 때



The screenshot shows an IDE console window with the following text:

```
<terminated> Ch3Ex9 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6.  
## 아이디를 입력하세요 : hong  
# 비밀번호를 입력하세요 : 3453  
비밀번호가 틀렸습니다.!!  
|
```

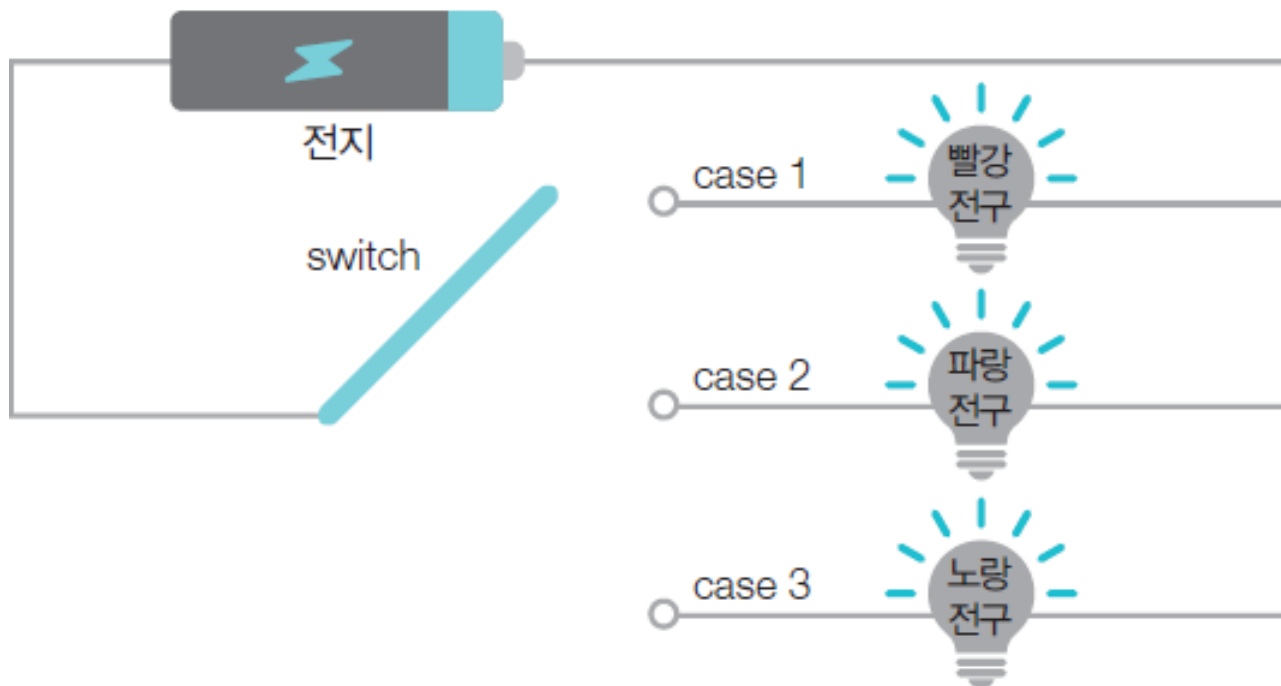
# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 03. 분기문

### ■ switch 문

- switch 문은 if ~ else if 문과 유사한 구조로, 여러 조건 중 하나를 선택할 수 있게 하는 분기문.

그림 3-7 switch 문의 동작 구조



# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 03. 분기문

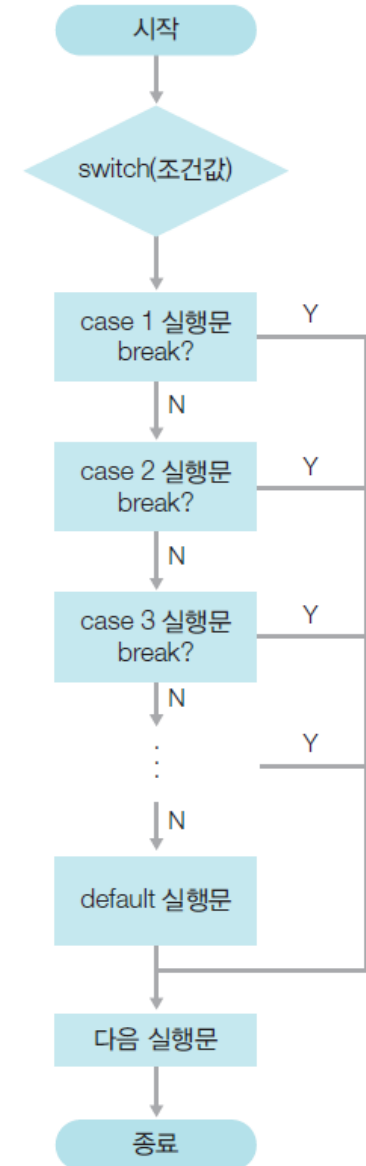
### ■ switch 문

- **사용 분야**  
여러 조건별로 처리하는 방법이 다를 때 사용
- **사용 예**  
쇼핑몰 회원을 네 등급으로 나누고, 등급별로 할인율이나 쿠폰을 다르게 적용할 때 사용  
(if ~ else 문으로도 구현 가능)

기본 구조

```
switch(조건값) {  
    case 조건 1:  
        명령문;  
        .....  
        break;  
    case 조건 2:  
        명령문;  
        .....  
        break;  
    case 조건 3:  
        명령문;  
        .....  
        break;  
    default:  
        명령문;  
        break;  
}
```

순서도



# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 03. 분기문

### ■ switch 문

예제 3-12 switch 문을 이용하여 회원 등급별 혜택 적용하기

Ch3Ex10.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex10 {
04     public static void main(String[] args) {
05         String memberLevel = "YOUNG";    // VIP, NEW, YOUNG 중 선택
06         String msg;
07
08         switch(memberLevel) {
09             case "VIP":
10                 msg = "VIP 고객 혜택 적용";break;
11             case "NEW":
12                 msg = "신규 고객 혜택 적용";break;
13             case "YOUNG":
14                 msg = "청소년 고객 혜택 적용";break;
```



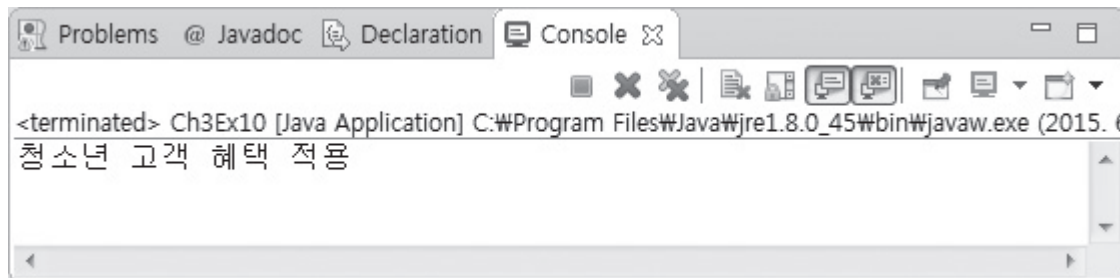
# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 03. 분기문

### ■ switch 문

```
15         default:
16             msg = "등급없음";
17         }
18
19         System.out.println(msg);
20     }
21 }
```

#### [실행결과]



# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 03. 분기문

여기서 잠깐 switch 문에 break 문이 없을 때

프로그램을 수행할 때 상황에 따라 switch 문에서 break 문을 생략할 수 있다. 그러나 실수로 break 문을 누락했다면 그 다음에 오는 case 문의 명령어들이 연속적으로 수행되어 의도하지 않은 결과를 초래할 수 있으므로 주의한다.

▷ break 문을 생략한 예제

```
class BreakExam {  
    public static void main(String[] args) {  
        int iValue = 2;  
  
        switch(iValue) {  
            case 1:  
                System.out.println("case 1입니다.");  
            case 2:  
                System.out.println("case 2입니다.");  
            case 3:  
                System.out.println("case 3입니다.");  
            default:  
                System.out.println("default입니다.");  
        }  
    }  
}
```

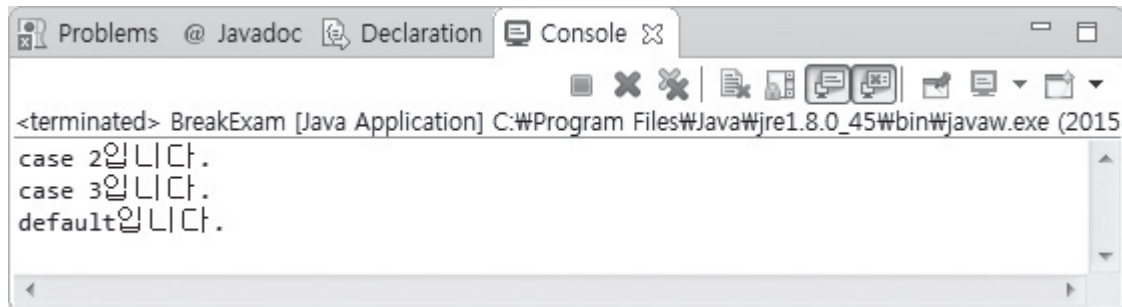
[뒷장 계속]

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 03. 분기문

여기서 잠깐 switch 문에 break 문이 없을 때

[실행결과]



```
<terminated> BreakExam [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015
case 2입니다.
case 3입니다.
default입니다.
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문

- for 문은 시작과 끝의 조건이 정해져 있을 때 사용.
- while 문은 시작과 종료 시점이 명확하지 않고 가변적일 때나 특정 조건을 수행하는 동안 계속 반복할 때 사용.

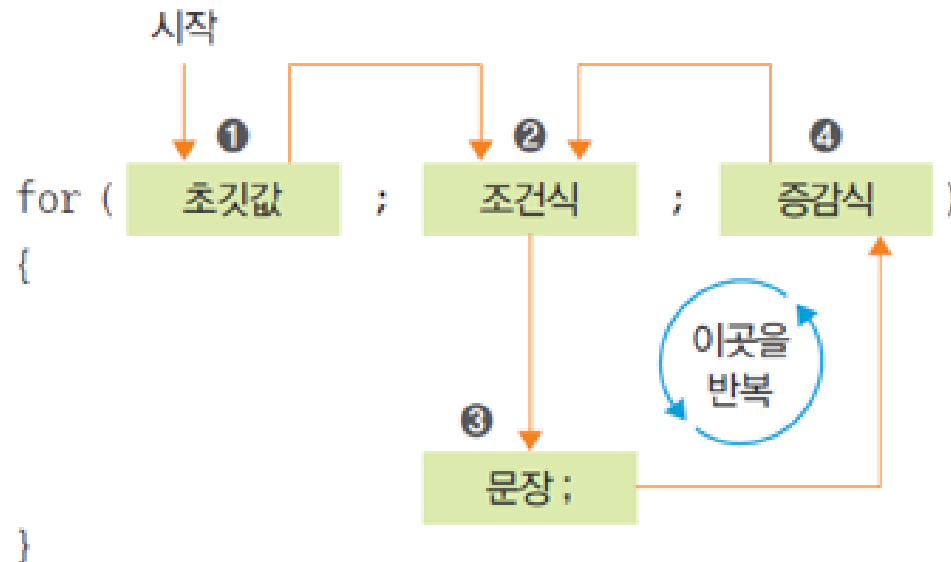


그림 for 문의 실행 순서

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

**사용 분야** 시작과 끝의 조건이 정해져 있고, 일정하게 변하는 값에 따라 반복적으로 처리할 때 사용한다.

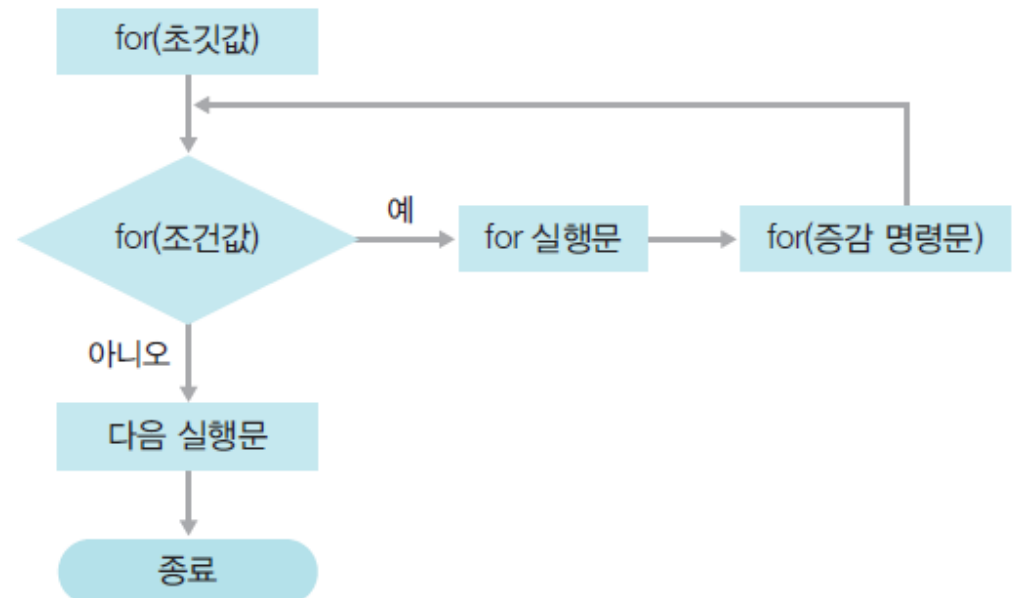
**사용 예**

- 인터넷 쇼핑몰에서 장바구니에 들어 있는 상품을 결제할 때 선택된 모든 상품의 수량과 가격을 합산한 최종 결제 금액 계산
- 카드사에서 전체 회원에게 홍보 E-mail 발송

#### 기본 구조

```
for(초깃값; 조건값; 증감식) {  
    명령문;  
    .....  
}
```

#### 순서도



# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

- for 문의 실행 순서
- 반복문 : for 문

초깃값 → 조건값 → 명령문(for 블록) → 증감식 반영 → 조건값 → 명령문.....

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문



그림 for 문의 개념과 실제 사용

초깃값 → 조건식 → 반복할 문장 → 증감식 → 조건식 → 반복할 문장 → 증감식 → 조건식 → 반복할 문장  
→ 증감식 → 조건식 ...

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

① 초깃값 수행

```
int i;  
for ( i=0 ; i < 3 ; i++ )  
{  
    출력 ("안녕하세요? IT CookBook.\n");  
}
```

② 조건식 확인

```
int i;  
for( i=0 ; i < 3 ; i++ )  
{  
    출력 ("안녕하세요? IT CookBook.\n");  
}
```

조건이  
거짓이면

→ ⑤ 반복문 탈출

↓ 조건이 참이면



# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

③ 반복할 문장 실행

```
int i;  
for( i=0 ; i < 3 ; i++ )  
{  
    출력 ("안녕하세요? IT CookBook.Wn");  
}
```



④ 증감식 실행

```
int i;  
for( i=0 ; i < 3 ; i++ )  
{  
    출력 ("안녕하세요? IT CookBook.Wn");  
}
```

그림 for 문이 반복되는 순서

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

- 제1회 : ❶ 초깃값을 수행한다(현재  $i=0$ ).
- 제2회 : ❷ 조건식을 확인한다. 현재  $i$  값이 0이므로  $i<3$ 는 참이다.
- 제3회 : ❸ `System.out.printf` 문을 수행한다('안녕하세요? ...' 출력).
- 제4회 : ❹ 증감식  $i++$ 를 수행하여  $i$  값을 1 증가시킨다(현재  $i=1$ ).
- 제5회 : 다시 ❷ 조건식을 확인한다. 현재  $i$  값이 1이므로  $i<3$ 는 참이다.
- 제6회 : 다시 ❸ `System.out.printf` 문을 수행한다('안녕하세요? ...' 출력).
- 제7회 : 다시 ❹ 증감식  $i++$ 를 수행하여  $i$  값을 1 증가시킨다(현재  $i=2$ ).
- 제8회 : 다시 ❷ 조건식을 확인한다. 현재  $i$  값이 2이므로  $i<3$ 는 참이다.
- 제9회 : 다시 ❸ `System.out.printf` 문을 수행한다('안녕하세요? ...' 출력).
- 제10회 : 다시 ❹ 증감식  $i++$ 를 수행하여  $i$  값을 1 증가시킨다(현재  $i=3$ ).
- 제11회 : 다시 ❷ 조건식을 확인한다. 현재  $i$  값이 3이므로 드디어  $i<3$ 가 거짓이다.
- 제12회 : 조건이 거짓이므로 ❺ 반복문을 탈출하고 반복문 블록(`{ }`) 밖의 내용을 수행한다.

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

예제 3-13 for 문으로 구구단 출력하기

Ch3Ex11.java

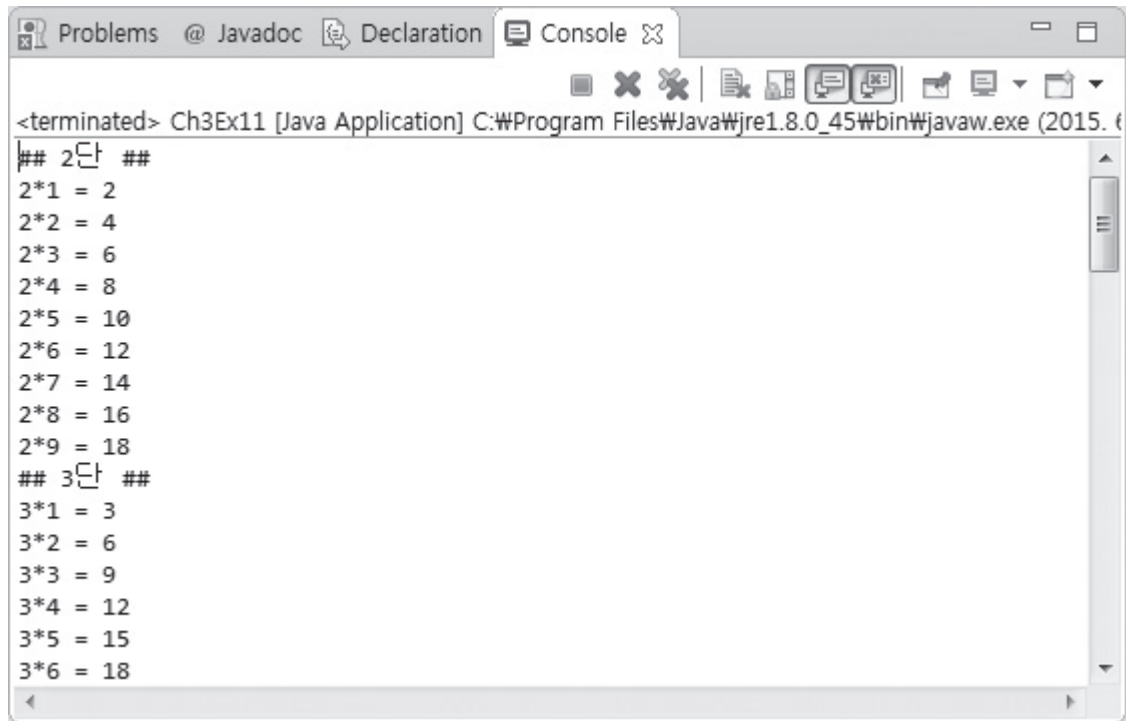
```
01 package javabook.ch3;
02
03 public class Ch3Ex11 {
04     public static void main(String[] args) {
05         for(int i = 2; i < 10; i++) {
06             System.out.println("## " + i + "단 ##");
07             for(int j = 1; j < 10; j++) {
08                 System.out.printf("%d * %d = %d\n", i, j, i * j);
09             }
10         }
11     }
12 }
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

[실행결과]



```
<terminated> Ch3Ex11 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. 10. 11:11:11)
## 2단 ##
2*1 = 2
2*2 = 4
2*3 = 6
2*4 = 8
2*5 = 10
2*6 = 12
2*7 = 14
2*8 = 16
2*9 = 18
## 3단 ##
3*1 = 3
3*2 = 6
3*3 = 9
3*4 = 12
3*5 = 15
3*6 = 18
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

- ✓ for 문을 이용하여 1부터 100까지 수 중 짝수의 합과 홀수의 합을 출력하라.

```
<terminated> Hello [Java Application] C:\#Program
```

```
1부터 100까지의 수중 홀수 합 = 2500
```

```
1부터 100까지의 수중 짝수 합 = 2550
```


# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

[실습하기] 다음과 같은 결과가 나오도록 반복문을 작성하시오.

시작값, 끝값, 증가값을 입력받아 반복문을 작성(for문 사용)



```
<terminated> Ex06_11 [Java Application] C:\#Program
시작값 입력 : 2
끝값 입력 : 300
증가값 입력 : 3
2에서 300까지 3씩 증가한 값의 합: 15050
```

그림 실행 결과

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : for 문

[실습하기] 앞의 구구단을 참고하여 다음과 같이 출력되도록 구구단을 작성하십시오.(for문 사용)

1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7	1*8=8	1*9=9
2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	8*9=72
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : while 문

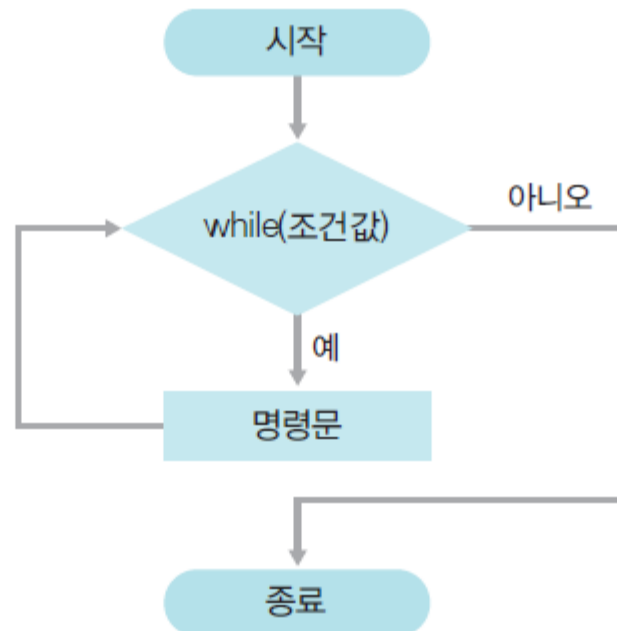
**사용 분야** 특정 조건을 만족하는 동안 반복할 때 사용한다.

- 사용 예**
- 프로그램이 실행되어 사용자 요청을 계속 처리하다가 종료 버튼을 누르면 프로그램 종료
  - 비행기 게임에서 에너지가 30% 이상일 때 정상적으로 비행하도록 처리

#### 기본 구조

```
while(조건값) {  
    명령문;  
    .....  
}  
  
do {  
    명령문;  
    .....  
} while(조건값)
```

#### 순서도





# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : while 문

예제 3-14 while 문 종료 조건 변경하기

Ch3Ex12.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex12 {
04     public static void main(String[] args) {
05         int num = 20;
06         while(num > 10) {
07             System.out.println(num--);
08         }
09
10         boolean flag = true;
11         while(flag) {
12             num--;
13             if(num == 3) {
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : while 문

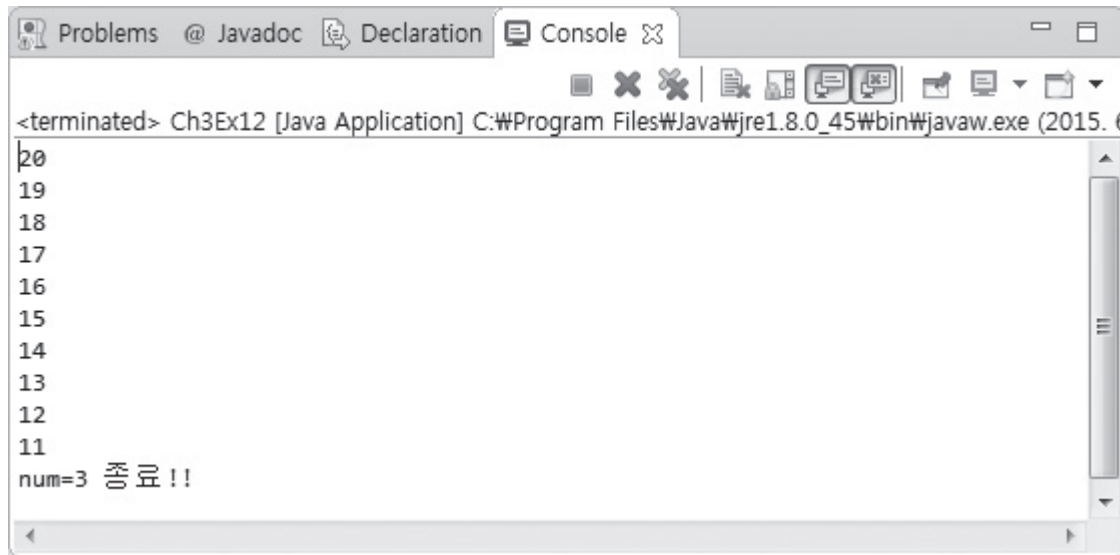
```
14         flag = false;
15         System.out.println("num=3 종료!!");
16     }
17 }
18 }
19 }
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : while 문

[실행결과]



The screenshot shows a Java IDE's console window. The title bar includes tabs for 'Problems', '@ Javadoc', 'Declaration', and 'Console'. The console output shows the execution of a Java application named 'Ch3Ex12'. The output displays line numbers 10 through 11, followed by the text 'num=3 종료!!' (num=3 End!!). The console window has a standard toolbar with icons for running, debugging, and other IDE functions.

```
<terminated> Ch3Ex12 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. 10.)
10
19
18
17
16
15
14
13
12
11
num=3 종료!!
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : while 문

- ✓ while 문을 이용하여 1부터 100까지 수 중 짝수의 합과 홀수의 합을 출력하라.

```
<terminated> Hello [Java Application] C:\#Program
```

```
1부터 100까지의 수중 홀수 합 = 2500
```

```
1부터 100까지의 수중 짝수 합 = 2550
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

■ 반복문 : while 문

[실습하기] 다음과 같은 결과가 나오도록 반복문을 작성하시오.

시작값, 끝값, 증가값을 입력받아 반복문을 작성(while문 사용)



```
<terminated> Ex06_11 [Java Application] C:\Program
시작값 입력 : 2
끝값 입력 : 300
증가값 입력 : 3
2에서 300까지 3씩 증가한 값의 합: 15050
```

그림 실행 결과

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 반복문 : while 문

[실습하기] 앞의 구구단을 참고하여 다음과 같이 출력되도록 구구단을 작성하시오.(while문 사용)

1*1=1	1*2=2	1*3=3	1*4=4	1*5=5	1*6=6	1*7=7	1*8=8	1*9=9
2*1=2	2*2=4	2*3=6	2*4=8	2*5=10	2*6=12	2*7=14	2*8=16	2*9=18
3*1=3	3*2=6	3*3=9	3*4=12	3*5=15	3*6=18	3*7=21	3*8=24	3*9=27
4*1=4	4*2=8	4*3=12	4*4=16	4*5=20	4*6=24	4*7=28	4*8=32	4*9=36
5*1=5	5*2=10	5*3=15	5*4=20	5*5=25	5*6=30	5*7=35	5*8=40	5*9=45
6*1=6	6*2=12	6*3=18	6*4=24	6*5=30	6*6=36	6*7=42	6*8=48	6*9=54
7*1=7	7*2=14	7*3=21	7*4=28	7*5=35	7*6=42	7*7=49	7*8=56	7*9=63
8*1=8	8*2=16	8*3=24	8*4=32	8*5=40	8*6=48	8*7=56	8*8=64	8*9=72
9*1=9	9*2=18	9*3=27	9*4=36	9*5=45	9*6=54	9*7=63	9*8=72	9*9=81

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ do~while 문

- while 문이나 for 문은 조건식이 처음부터 거짓이면 한 번도 수행하지 않고 종료. 하지만 do~while 문은 어떠한 경우라도 한 번은 수행함

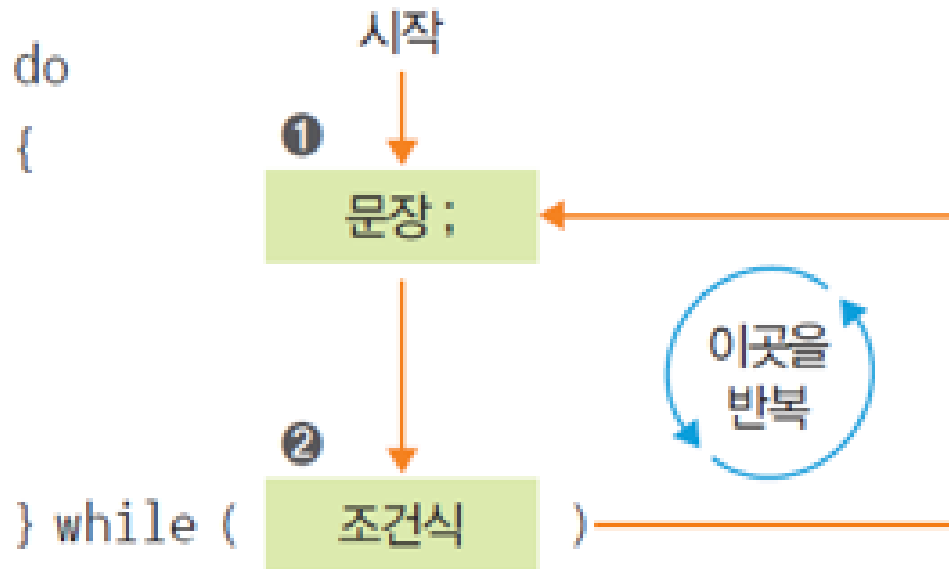


그림 do~while 문의 형식과 실행 순서

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ do~while 문

#### 실습 7-5 do~while 문 사용 예 1

```
01 public class Ex07_05 {  
02     public static void main(String[] args) {  
03         int a = 100;  
04  
05         while (a == 200) {  
06             System.out.printf("while 문 내부에 들어 왔습니다.\n");  
07         }  
08  
09         do {  
10             System.out.printf("do ~ while 문 내부에 들어 왔습니다.\n");  
11         } while (a == 200);  
12     }  
13 }
```

while 문 실행 : 먼저  
조건식을 판단한다.

do~while 문 실행 :  
먼저 내용을 실행한  
다음 조건식을  
판단한다.



# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 04. 반복문

### ■ 실습

#### ■ do~while 문

- do~while 문은 사용하여 다음과 같은 결과가 나오도록 작성하여라.



```
<terminated> Ex07_06 [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe (2015. 8. 19. 오후 2:32:46)

손님 주문하시겠습니까 ?
<1>카페라떼 <2>카푸치노 <3>아메리카노 <4>그만시킵니다 ==> 3
#아메리카노 주문하셨습니다.

손님 주문하시겠습니까 ?
<1>카페라떼 <2>카푸치노 <3>아메리카노 <4>그만시킵니다 ==> 1
#카페라떼 주문하셨습니다.

손님 주문하시겠습니까 ?
<1>카페라떼 <2>카푸치노 <3>아메리카노 <4>그만시킵니다 ==> 4
주문하신 커피 준비하겠습니다.
```

그림 실행 결과

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 배열의 개념

- 인덱스(순차 번호)와 데이터로 구성된 일종의 자료구조.
- 데이터는 인덱스를 사용하여 값을 넣거나 가져온다(참조). 모든 프로그램 언어에서 배열을 지원하며, 자바 역시 다른 고급 자료구조와 함께 기본적으로 배열을 지원한다.

**사용 분야** 집합형의 데이터를 관리할 필요가 있을 때 사용한다.

**사용 예** 근퇴 관리 프로그램에서 출근한 순서대로 사번을 입력하고, 필요한 시점에 출근한 사람들의 사번을 모두 출력

---

### 기본 구조

```
int[] num = new int[4];  
int num[] = new int[4];  
int[] num = {10, 20, 40, 99}  
int[][] num = {{10, 20}, {30, 40}, {50, 60}};
```

크기가 4인 배열 선언

배열 선언과 함께 초깃값 할당

2차원 배열 선언과 함께 초깃값 할당

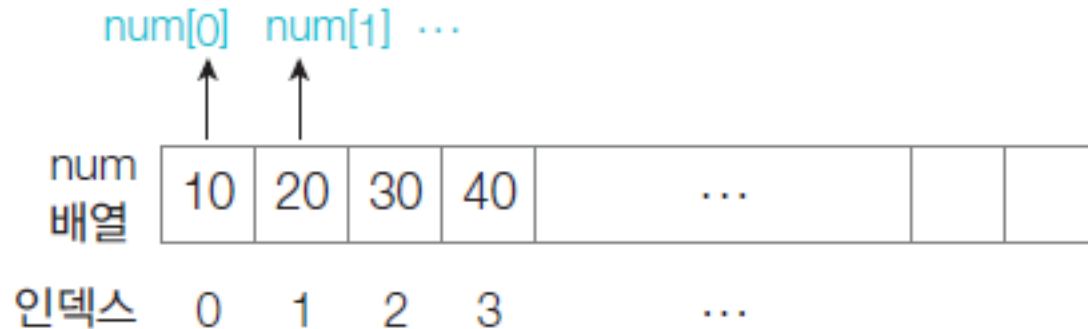
# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 배열의 개념

- 같은 자료형으로만 구성한다.
- 배열 안 데이터는 0부터 시작하는 인덱스를 참조한다.
- 배열을 선언할 때 크기를 지정해야 하며, 나중에 그 크기를 변경할 수 없다.
- 특정한 값으로 초기화하지 않은 배열 안 데이터를 참조하면 Null Pointer Exception이 발생한다.

그림 3-8 배열 구조



# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 자바 배열의 특징

- 자바는 객체지향 언어이고 객체 타입의 참조 변수를 지원하기 때문에 배열에서도 원시 자료형 외의 객체 타입을 사용할 수 있다

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 자바 배열의 특징

```
class Member {  
    int id;  
    String name;  
    Date birth;  
    String tel;  
    String dept;  
}
```

```
Member[] mlist = new Member[100]; — ①
```

앞에서 선언한 클래스 이름 Member를 원소로 하는 100개짜리 배열을 mlist 이름으로 생성

```
mlist[0] = new Member(101, "홍길동", "1990.08.16", "010-456-1234", "한국주식회사"); — ②
```

배열의 첫 번째 원소

Member 클래스의 인스턴스를 생성하여 배열 원소에 할당

.....

```
for(int i = 0; i < mlist.length; i++) { —
```

배열의 크기를 구함

```
    System.out.println(mlist[i].getName());  
    System.out.println(mlist[i].getCompany());  
    .....
```

```
}
```

③

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 자바 배열의 특징

예제 3-15 배열을 사용하여 제품 목록 출력하기

Ch3ExX3.java

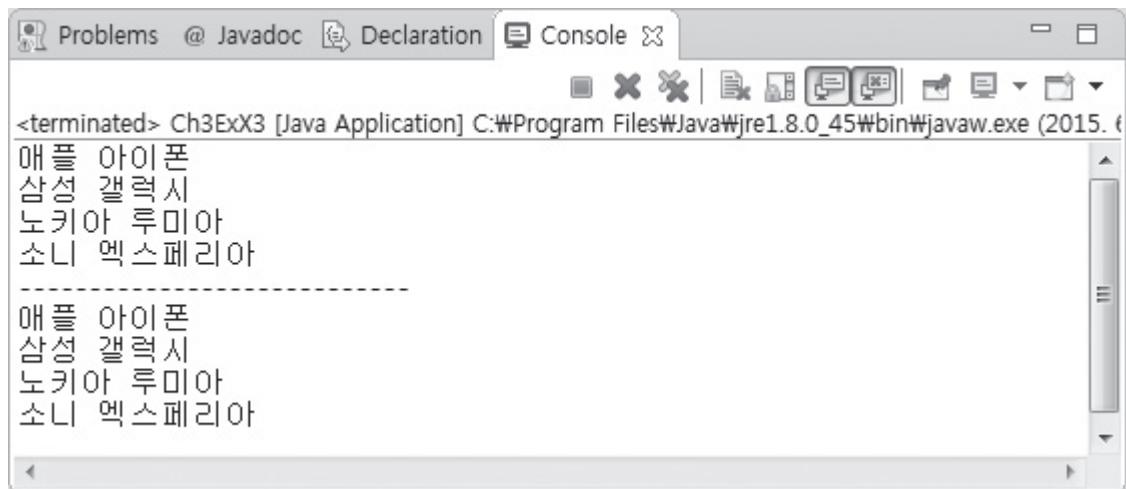
```
01 package javabook.ch3;
02
03 public class Ch3ExX3 {
04     public static void main(String[] args) {
05         String[] products = {"애플 아이폰", "삼성 갤럭시", "노키아 루미아", "소니 엑스페리아"};
06
07         for(int i = 0; i < products.length; i++) {
08             System.out.println(products[i]);
09         }
10
11         System.out.println("-----");
12
13         // 새로운 for 문 사용
14         for(String s : products) {
15             System.out.println(s);
16         }
17     }
18 }
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 자바 배열의 특징

#### [실행결과]



```
<terminated> Ch3ExX3 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015.6.1)
애플 아이폰
삼성 갤럭시
노키아 루미아
소니 엑스페리아
-----
애플 아이폰
삼성 갤럭시
노키아 루미아
소니 엑스페리아
```

# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 자바 배열의 특징

예제 3-16 2차원 배열을 사용한 자료구조 처리하기

Ch3Ex13.java

```
01 package javabook.ch3;
02
03 public class Ch3Ex13 {
04     public static void main(String[] args) {
05         String[][] members = {"101", "홍길동"}, {"102", "김사랑"}, {"103", "이신용"};
06
07         for(int i = 0; i < members.length; i++) {
08             System.out.println(members[i][0] + ":" + members[i][1]);
09         }
10     }
11 }
```

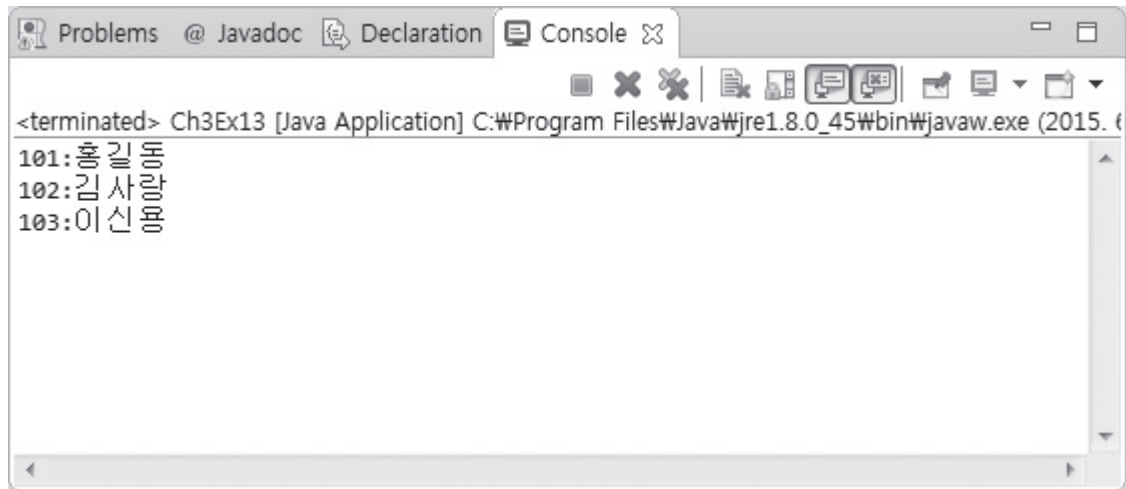


# 1. 개발환경 구축하기(2001020203\_14v2.1)

## 05. 배열

### ■ 자바 배열의 특징

#### [실행결과]



```
<terminated> Ch3Ex13 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015.6.12)
101:홍길동
102:김사랑
103:이신용
```