

자바 기초

# 목차

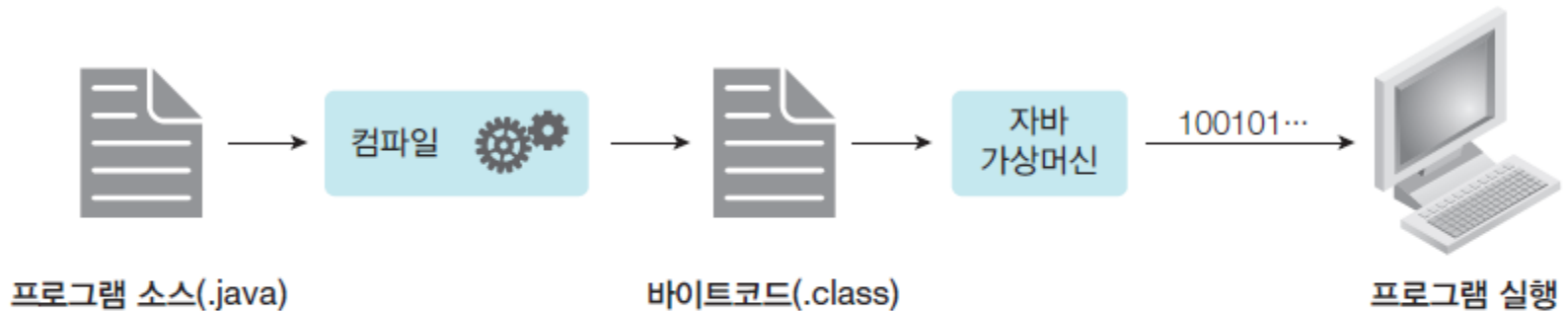
- 01 자바 프로그램 개발 절차
- 02 이클립스로 자바 프로젝트 생성
- 03 HelloWorld 프로그램 만들기

# 자바기초

## 01. 자바 프로그램 개발 절차

### ■ 자바 프로그램 개발단계

그림 2-1 자바 프로그램 개발 단계



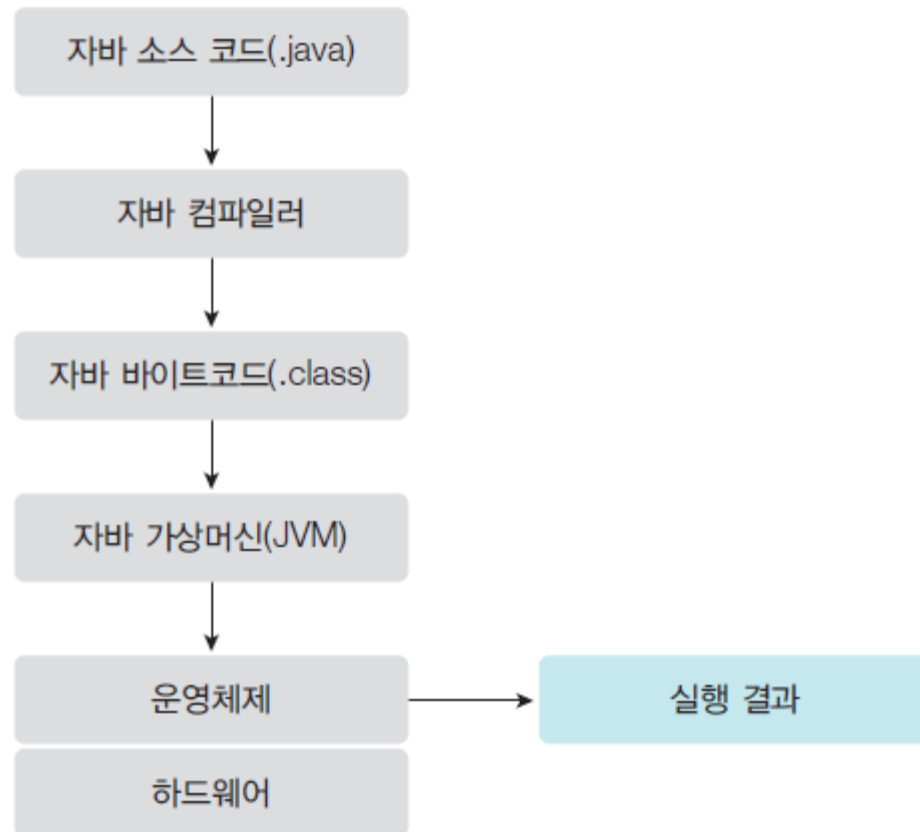
- ① 이클립스나 메모장 등 편집기를 이용하여 코드를 작성한다.
- ② 작성한 코드를 컴파일하여 중간 코드인 바이트코드로 변환한다.
- ③ 컴파일한 바이트코드를 실행한다.

# 자바 기초

## 01. 자바 프로그램 개발 절차

### ■ 자바 프로그램 실행구조

그림 2-2 자바 프로그램의 실행 구조

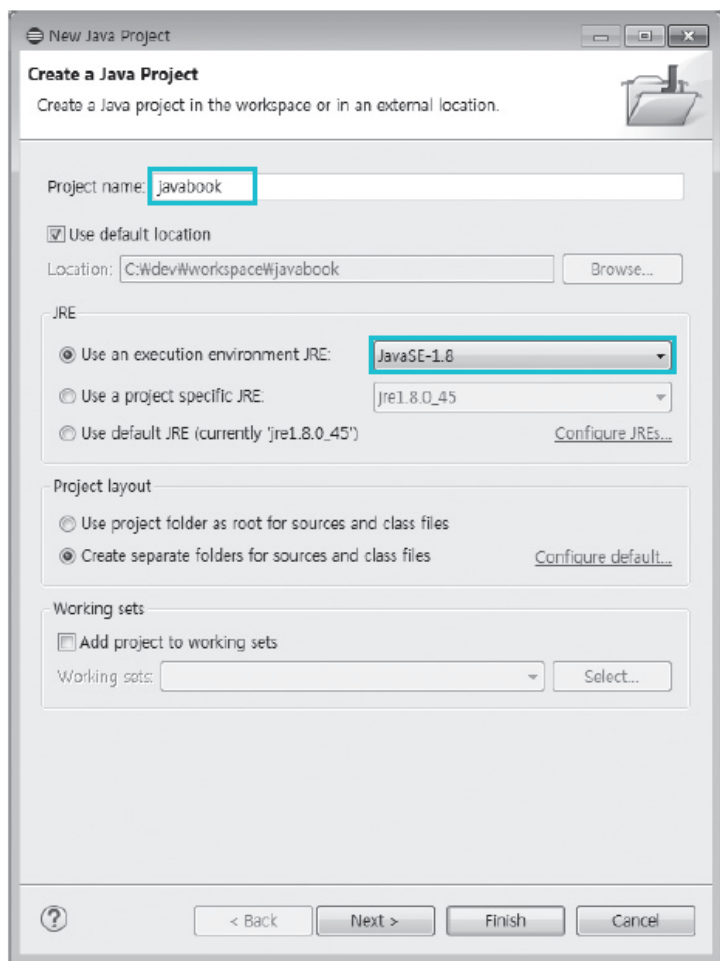


# 자바기초

## 02. 이클립스로 자바 프로젝트 생성

### ■ 프로젝트 생성

그림 2-3 이클립스 프로젝트 생성



- Project name : p0924
- JRE : JavaSE-1.8
- 나머지 설정 : 기본값 사용

# 자바기초

## 02. 이클립스로 자바 프로젝트 생성

### ■ 프로젝트 생성

#### Project name

프로젝트 이름을 지정. 프로젝트 생성 위치는 Use default location이 기본값으로 선택. 워크스페이스 폴더를 변경하려면 체크를 해제하고 [Browse] 버튼을 눌러 원하는 폴더를 지정.

#### JRE

JRE는 자바 런타임, 즉 자바의 실행 환경. 여기서는 자바 프로젝트의 산출물(프로그램)을 어떤 버전에서 실행할지 설정.

#### Project layout

프로젝트에서 생성하는 자바 소스와 컴파일된 클래스 파일(바이트코드)의 위치 설정과 관련. 가급적이면 기본값을 그대로 사용한. 오른쪽에 있는 [Configure default...] 링크를 클릭하면 모든 프로젝트에 적용하는 이클립스의 기본 설정값을 변경 가능.

#### Working sets

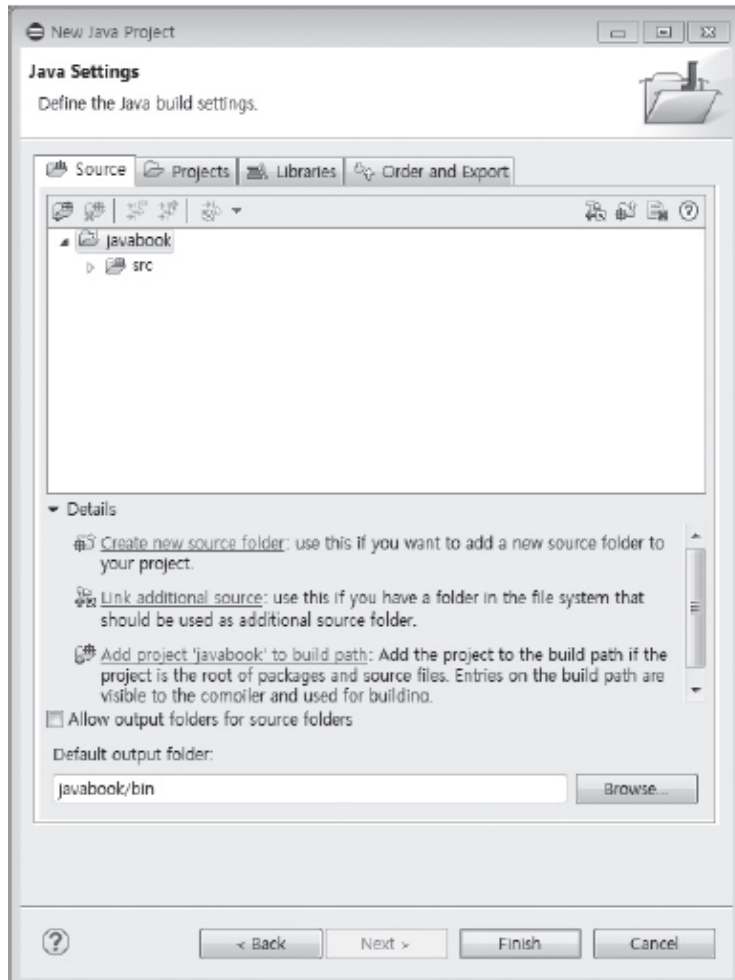
일종의 프로젝트 그룹. 이클립스에 프로젝트가 많을 때 패키지나 프로젝트 탐색기 뷰가 복잡해 보이지 않게 하고, 성격이 동일한 프로젝트나 참조되는 프로젝트를 하나로 묶어서 관리하는 용도로 사용. 새로운 Working set을 만들거나 기존 Working set 추가 가능.

# 자바기초

## 02. 이클립스로 자바 프로젝트 생성

### ■ 프로젝트 기본 설정

그림 2-4 프로젝트 기본 설정



- 각 설정 : 기본값 사용

# 자바기초

## 02. 이클립스로 자바 프로젝트 생성

### ■ 프로젝트 기본 설정

#### [Source] 탭

프로젝트에 사용할 소스 파일의 위치를 설정하는 부분. 기본적으로 [src] 폴더를 사용하며, 다른 폴더를 추가하고 싶다면 이곳에서 추가한 후 사용. 화면 아래에 있는 Default output folder 항목은 컴파일 된 클래스 파일을 저장하는 폴더. 기본적으로 프로젝트 폴더 안에 b[in] 폴더를 생성하여 저장.

#### [Projects] 탭

현재 프로젝트의 소스에서 다른 프로젝트에 있는 클래스를 참조해야 할 때 해당 프로젝트를 추가하는 곳.

#### [Libraries] 탭

오픈소스 라이브러리나 다른 팀원이 개발한 클래스 라이브러리를 사용할 때는 이곳에 파일을 추가. 동일한 기능을 구현한 코드를 중복해서 개발하지 않아도 됨. 또 상속 등의 기능을 이용하면 다른 사람이 만든 클래스를 확장할 수도 있어 개발 생산성 향상에 큰 도움이 됨.

#### [Order and Export] 탭

프로젝트 안에서 라이브러리 참조 우선순위를 지정하고, 프로젝트 패키징을 할 때 포함할 대상과 포함하지 않을 대상을 지정. 옵션을 다 설정하고 [Finish] 버튼을 누르면 왼쪽 패키지 탐색기(Package Explorer) 뷰에 프로젝트가 생성.



# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ HelloWorld 클래스 생성

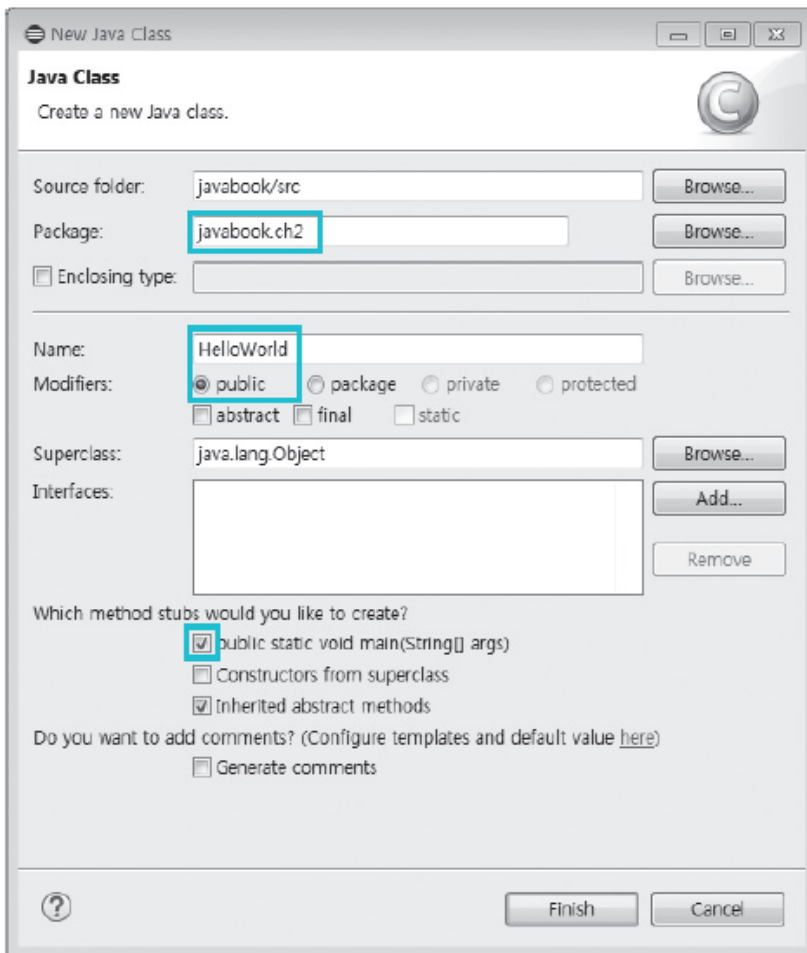
프로젝트를 생성 후 왼쪽의 패키지 탐색기 뷰에서 javabook 프로젝트를 선택한 후 마우스 오른쪽 버튼을 눌러 [New]-[Class] 메뉴를 선택. 이클립스에서 클래스 생성은 곧 자바 소스 파일, 자바 프로그램을 만드는 것.

# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ HelloWorld 클래스 생성

그림 2-5 클래스 생성



- Package : javabook.ch2
- Name : HelloWorld
- Modifiers : public
- public static void main(String[ ] args)에 체크
- 그 외 나머지 옵션 : 기본값 사용

# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ HelloWorld 클래스 생성

#### Source folder

자바 소스를 저장할 폴더를 지정. 기본적으로 [src] 폴더를 사용.

#### Package

자바 소스 파일을 비슷한 성격끼리 모아 두는 일종의 폴더 개념. 패키지가 없어도 프로그램을 실행할 수 있지만, 소스가 많아지면 비슷한 성격이나 목적에 따라 클래스들을 구분 해야 함. 또 객체지향의 특징을 반영하려면 클래스의 계층 구조를 만들어야 하므로, 패키지 관리는 반드시 필요.

#### Name

클래스 이름을 지정하는 부분. 클래스 이름은 영문으로 입력하고 첫 글자는 대문자로 입력. 공백은 허용하지 않고, 두 단어 이상의 이름을 사용할 때는 두 번째 단어의 첫 글자 역시도 대문자로 입력하는 것이 기본 원칙.

# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ HelloWorld 클래스 생성

#### Modifiers

클래스에 외부 접근 범위 등을 제한하는 자바 문법 요소.

#### Superclass

다른 클래스의 속성과 기능을 물려받아 (상속) 현재 클래스를 개발할 때 지정. [Browse] 버튼을 눌러 현재 프로젝트에서 참조 가능한 클래스 중 원하는 클래스를 선택. 상속이 필요 없다면 기본값을 그대로 사용. 특정 클래스를 상속받지 않아도 모든 자바 클래스는 `java.lang.Object` 클래스를 상속함.

#### Interfaces

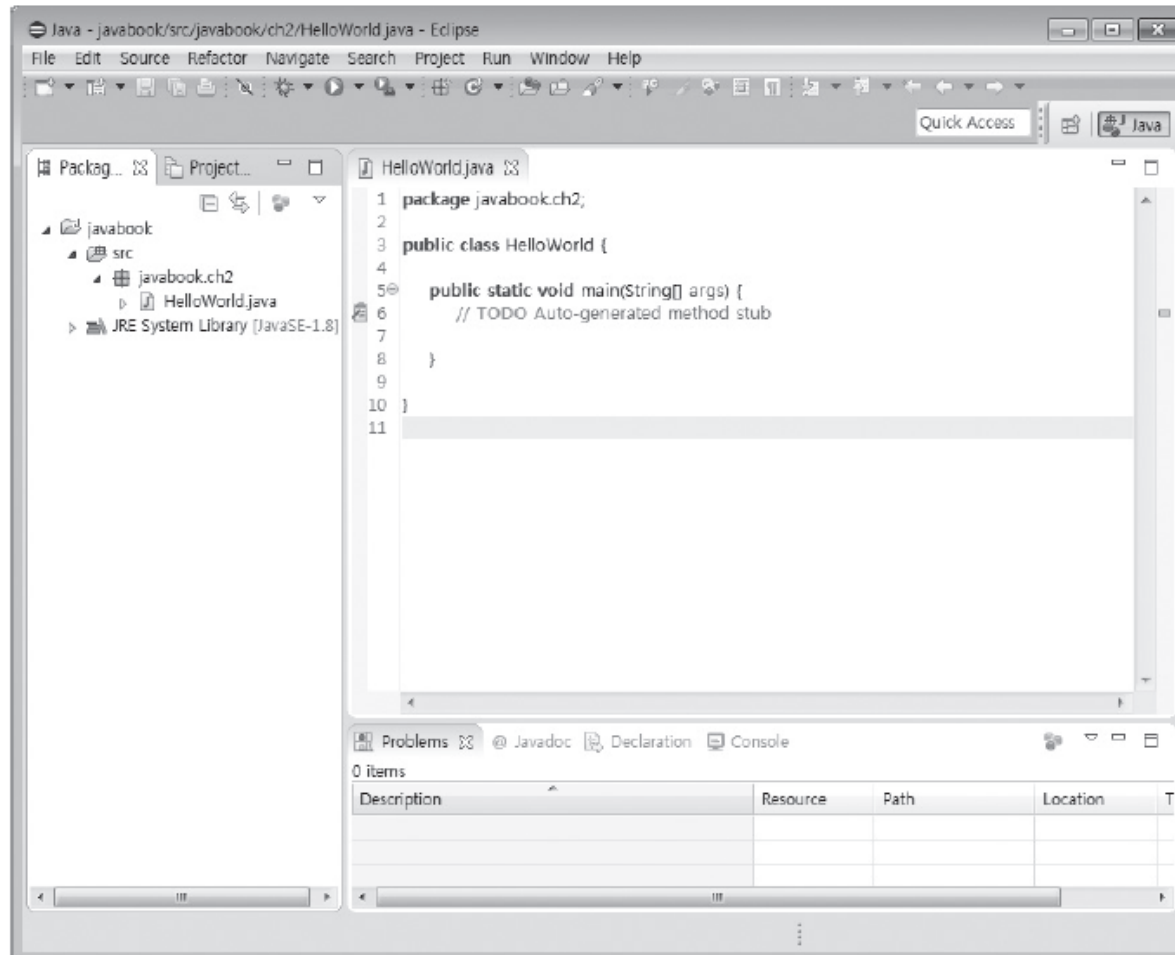
메서드 정의만 포함하는 클래스. 단독으로는 사용할 수 없고, 반드시 인터페이스를 구현하는 (Implements) 클래스를 새로 만들어 정의된 메서드의 바디(실행 코드)를 작성해야 함. 보통 프로그램을 동일한 규격으로 개발할 수 있도록 가이드 역할.

# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ 프로그램 작성 및 기본 코드 분석

그림 2-6 클래스가 생성된 화면



# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ 프로그램 작성 및 기본 코드 분석

기본 생성된 코드를 분석한 후 간단한 출력문을 추가해 보자.

```
package javabook.ch2; —①

public class HelloWorld { —②
    public static void main(String[] args) { —
        // TODO Auto-generated method stub —③
    } —
}
```

- ① 패키지 선언부
- ② 클래스 선언부
- ③ 자바 프로그램을 실행하는 main( ) 메서드

## 03. HelloWorld 프로그램 만들기

### ■ 프로그램 작성 및 기본 코드 분석

main( ) 메서드에 간단한 다음 코드를 추가해 보자.

```
System.out.println("Hello World!!");
```

HelloWorld 프로그램의 최종 소스

```
package javabook.ch2;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!!");
    }
}
```

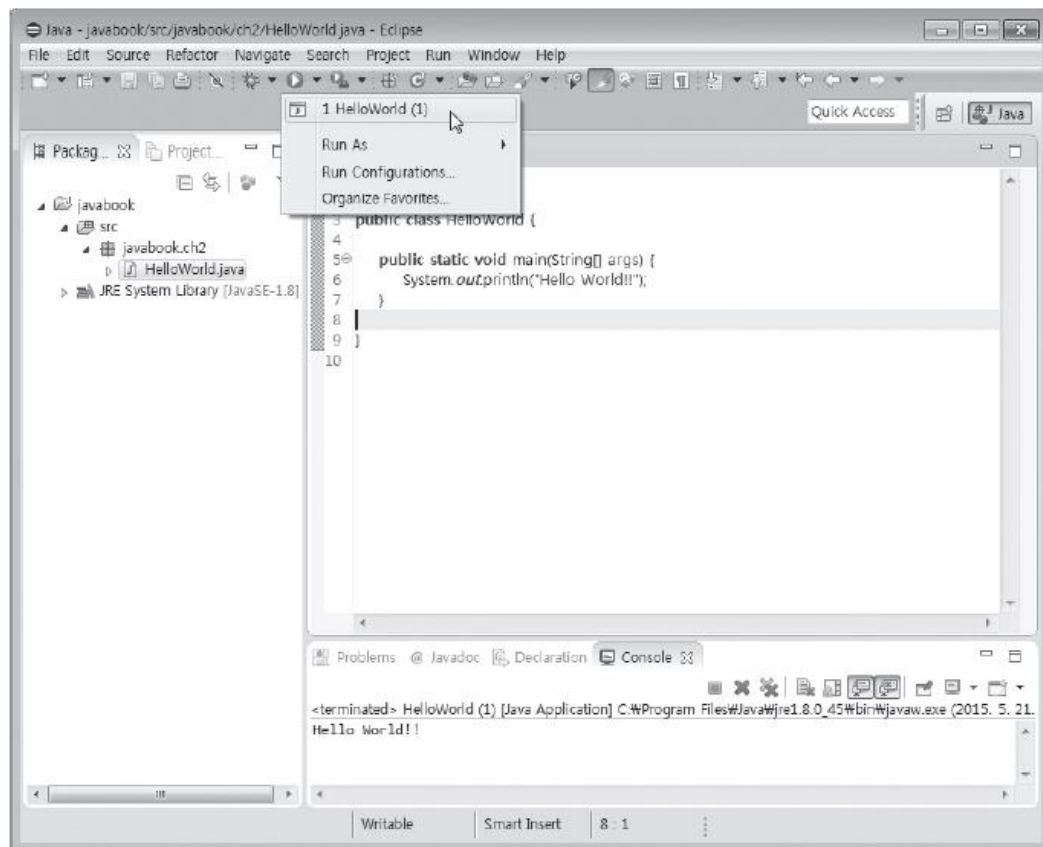
# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ 프로그램 실행 방법

- 패키지 탐색기 뷰에서 HelloWorld.java를 마우스 오른쪽 버튼으로 눌러 [Run As]-[Java Application] 메뉴를 선택
- 편집기 뷰의 코드가 활성화된 상태에서 이클립스 메뉴 바 아래쪽에 있는 '실행' 아이콘을 클릭.

그림 2-7 프로그램 실행



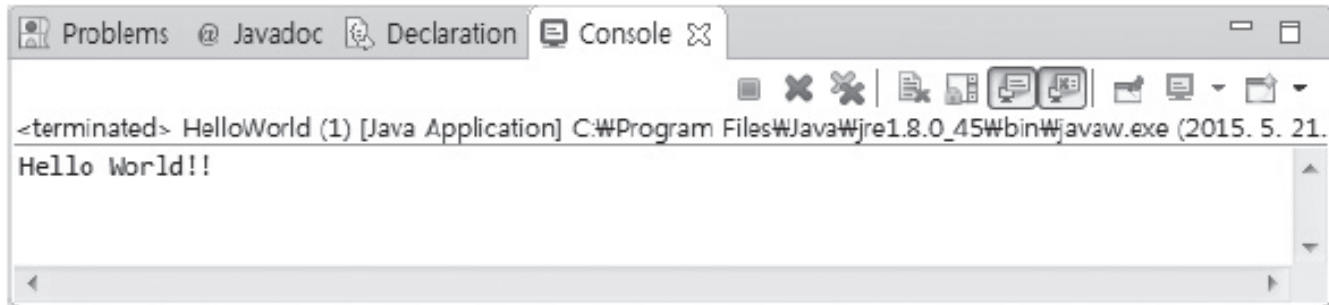


# 자바기초

## 03. HelloWorld 프로그램 만들기

### ■ 프로그램 실행 방법

그림 2-8 실행 결과



# 자바기초

## 1. System.out의 기본(1)

### ■ System.out.printf( ) 메소드의 기본적인 사용법

```
System.out.println("안녕하세요?");  
System.out.println("Java입니다.");
```

실행 결과 ►

```
안녕하세요?  
Java입니다.
```

```
System.out.print("안녕하세요?");  
System.out.print("Java입니다.");
```

실행 결과 ►

```
안녕하세요?Java입니다.
```

# 자바기초

## 1. System.out의 기본(1)

```
System.out.printf("안녕하세요?");
```

실행 결과 ►

안녕하세요?

- TIP : println( )에서 ln은 line feed의 약자로 행을 넘긴다는 의미  
System.out.printf( )에서 f는 format의 약자로 서식을 지정한다는 의미

```
System.out.printf("100");
```

실행 결과 ►

100

```
System.out.printf("%d", 100);
```

실행 결과 ►

100

- 위 첫 번째 System.out.printf("100")의 결과 100은 숫자 100이 아닌 글자 100(일영영)임
- 두 번째 System.out.printf("%d", 100)의 결과 100은 숫자 100을 의미. 서식(%d)이 지정된 '숫자'는 그대로 숫자의 의미임

# 자바기초

## 1. System.out의 기본(1)

### 실습 3-1 System.out.printf() 메소드 사용 예 1

```
01 public class Ex03_01 {
02     public static void main(String[] args) {
03         System.out.printf("100+100"); ----- 모두 글자로 취급한다.
04         System.out.printf("\n"); ----- System.out.printf()는 행이 넘어가지 않으므로 강제로
                                           행이 넘어가게 한다.
05         System.out.printf("%d", 100 + 100); ----- 숫자와 계산해서 결과를 출력한다.
06         System.out.printf("\n"); ----- System.out.printf()는 행이 넘어가지 않으므로 강제로
                                           행이 넘어가게 한다.
07     }
08 }
```

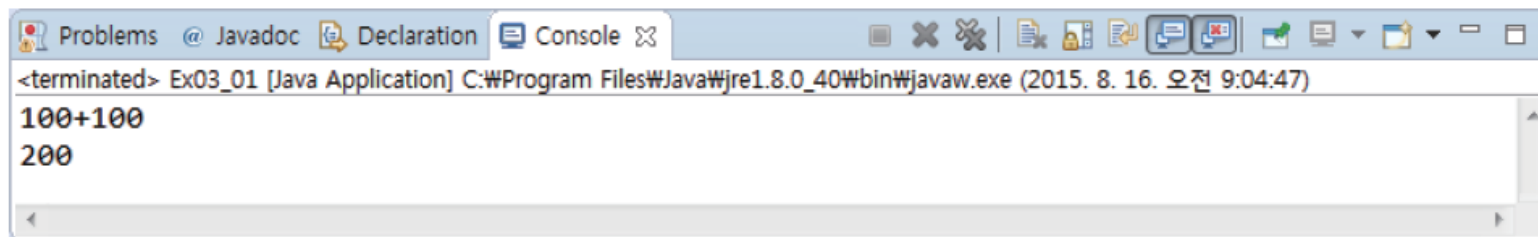


그림 3-1 실행 결과

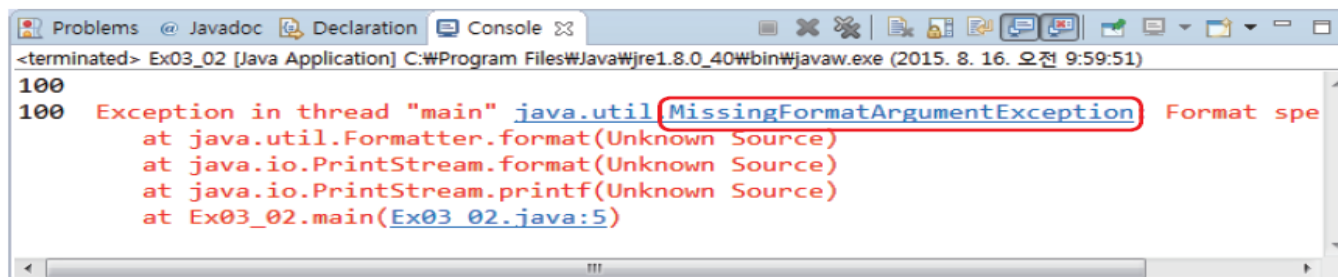
- 5행의 "%d"는 정수(decimal)를 의미, 서식의 개수와 큰따옴표 뒤에 나오는 숫자(또는 문자)의 개수가 같아야 함

# 자바기초

## 1. System.out의 기본(1)

### 실습 3-2 System.out.printf() 메소드 사용 예 2

```
01 public class Ex03_02 {  
02     public static void main(String[] args) {  
03         System.out.printf("%d", 100, 200); ----- %d는 1개, 숫자는 2개이다.  
04         System.out.printf("%n");  
05         System.out.printf("%d %d", 100); ----- %d는 2개, 숫자는 1개이다.  
06         System.out.printf("%n");  
07     }  
08 }
```



System.out.printf("%d %d", 100, 200);

그림 3-3 서식과 숫자의 대응

# 자바기초

## 직접 풀어보기 3-1

- 100과 200을 더한 결과가 나올 수 있도록 %d를 3개 사용하여 `System.out.printf( )` 문을 만들어보자. 또한 나눗셈 결과도 나오게 해보자. 즉 다음과 같이 출력되게 한다.

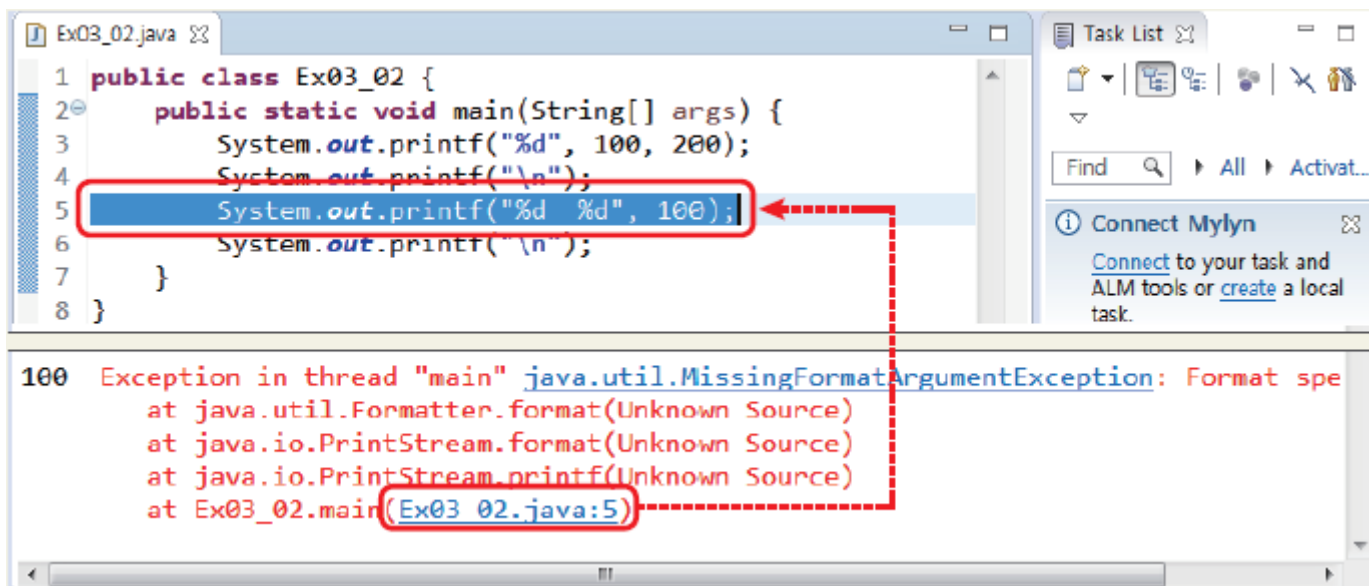
`100+200=300`

`100/200=0.5`

# 자바기초

## ■ 오류의 확인과 수정

- JAVA는 오류가 발생하면 대부분 그 이유와 오류가 발생한 행까지 알려준다. [실습 3-2]에서 발생한 오류를 보면 원인과 행 번호인 5행까지 표시되어 있는데, 오류 행 번호를 클릭하면 바로 소스코드의 행으로 이동한다. 앞으로 오류를 이렇게 확인하면 좀 더 수월하게 문제를 해결할 수 있을 것이다.



# 자바기초

## ■ 정수 외에 자주 사용되는 서식

### 실습 3-3 서식을 사용한 출력의 예 1

```
01 public class Ex03_03 {  
02     public static void main(String[] args) {  
03         System.out.printf("%d / %d = %d", 100, 200, 0.5); ----- %d가 3개, 숫자도 3개이다.  
04         System.out.printf("\n");  
05     }  
06 }
```

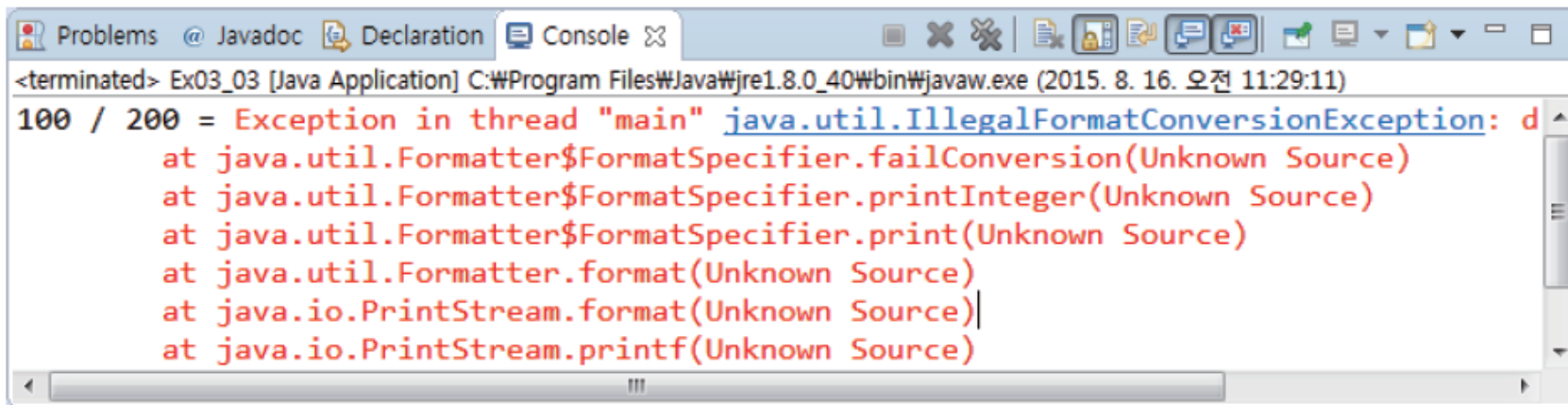


그림 3-4 실행 결과



# 자바기초

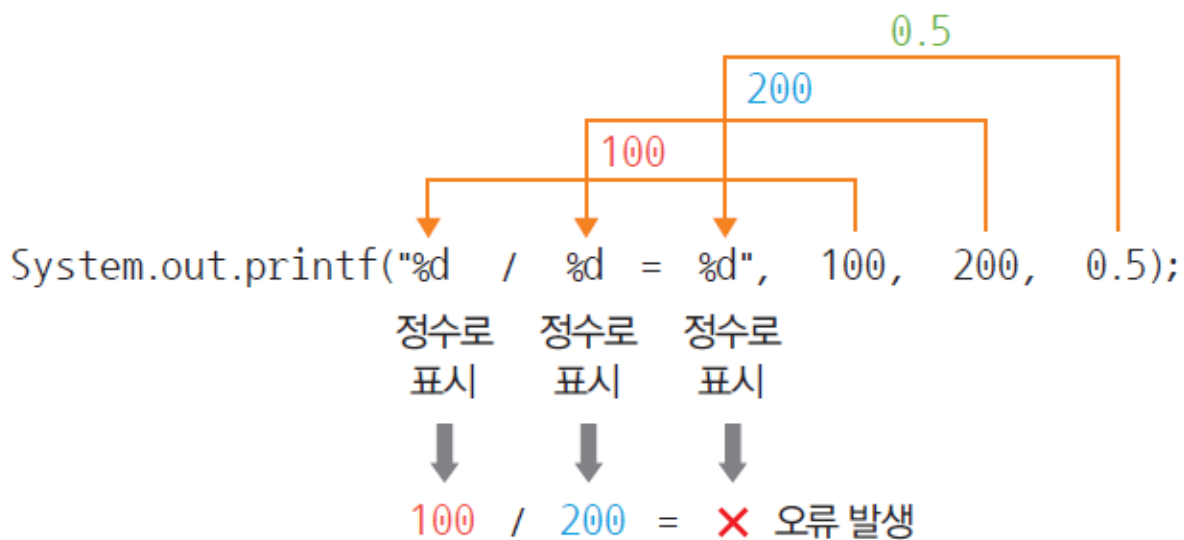


그림 3-5 서식과 숫자의 불일치 상황

표 3-1 System.out.printf()의 대표적 서식

서식	설명	값의 예
%d, %x, %o	정수(10진수, 16진수, 8진수)	10, 100, 1234
%f	실수(소수점이 있는 수)	0.5, 1.0, 3.14
%c	문자. 반드시 한 글자이고 작은따옴표(')로 묶여 있어야 함	'a', 'b', 'F'
%s	문자열. 한 글자 이상이고 큰따옴표(" ")로 묶여 있어야 함	"안녕", "abcdefg", "a"

# 자바기초

## ■ 자릿수를 맞춘 출력

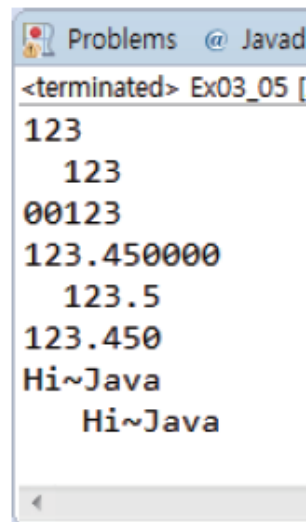
### 실습 3-5 다양한 서식 활용 예 1

```
01 public class Ex03_05 {  
02     public static void main(String[] args) {  
03         System.out.printf("%d\n", 123);  
04         System.out.printf("%5d\n", 123);  
05         System.out.printf("%05d\n", 123);  
06  
07         System.out.printf("%f\n", 123.45);  
08         System.out.printf("%7.1f\n", 123.45);  
09         System.out.printf("%7.3f\n", 123.45);  
10  
11         System.out.printf("%s\n", "Hi~Java");  
12         System.out.printf("%10s\n", "Hi~Java");  
13     }  
14 }
```

정수형 서식을 활용했다.

실수형 서식을 활용했다.

문자열형 서식을 활용했다.



```
Problems @ Javad  
<terminated> Ex03_05 [  
123  
123  
00123  
123.450000  
123.5  
123.450  
Hi~Java  
Hi~Java
```

그림 3-7 실행 결과

# 자바기초

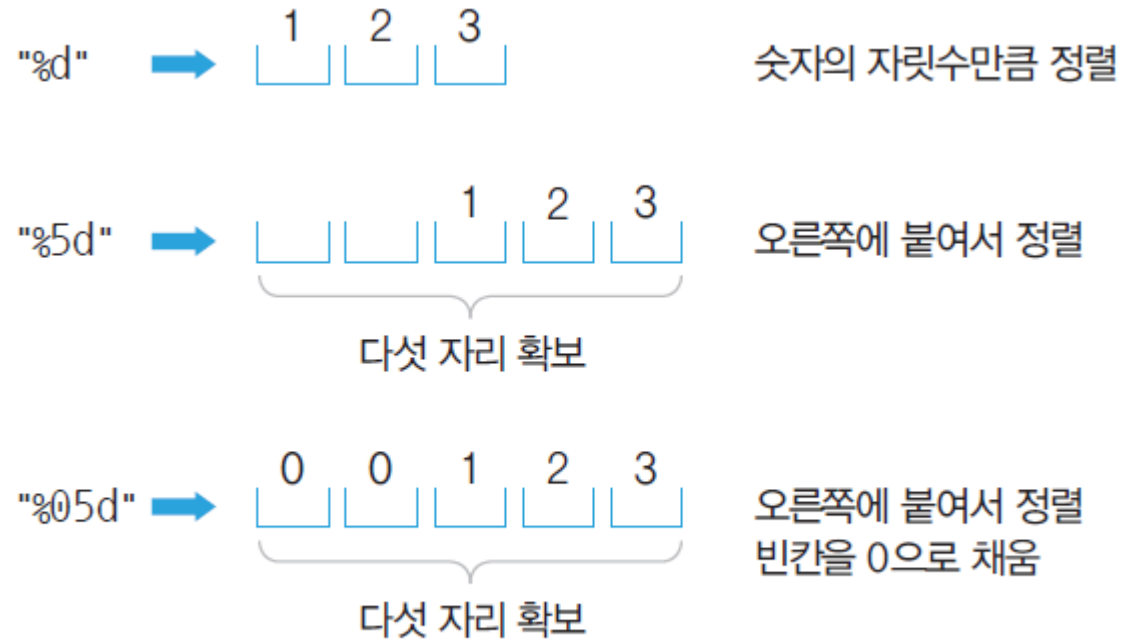


그림 3-8 정수형 데이터 서식 지정

# 자바기초

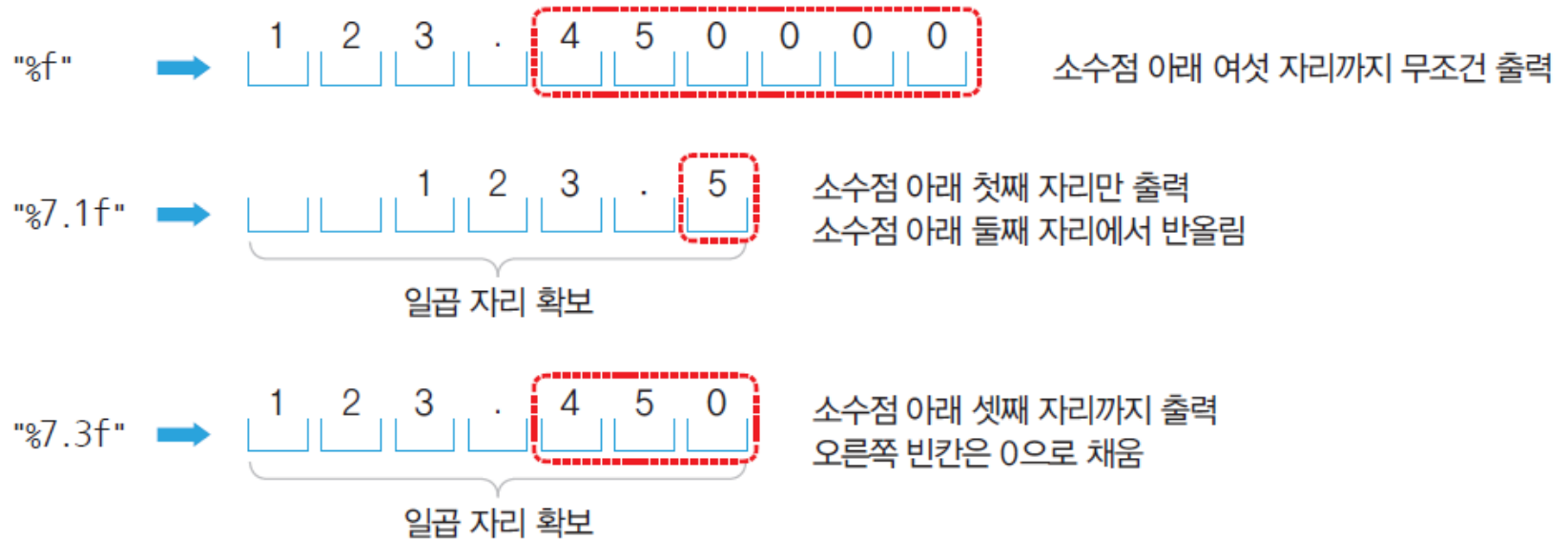


그림 3-9 실수형 데이터 서식 지정

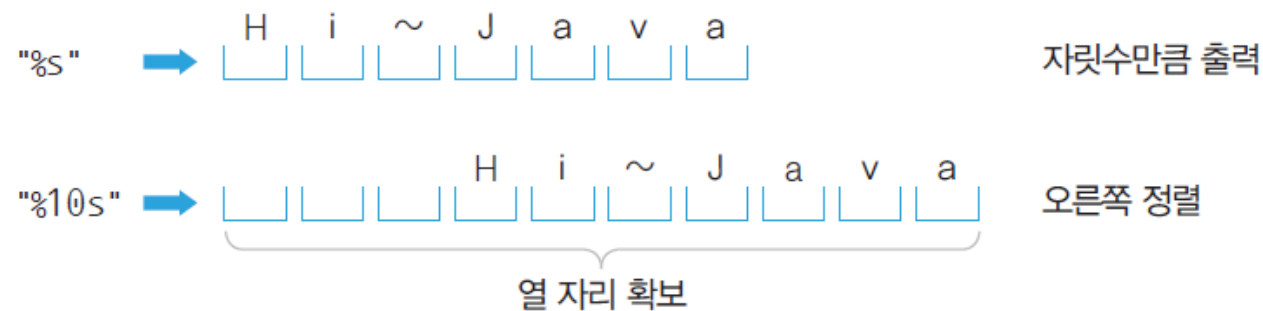


그림 3-10 문자열형 데이터 서식 지정

# 자바 기초

## 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

## 02. java.util 패키지의 주요 클래스

### ■ Scanner 클래스

- Scanner 클래스는 입력 스트림으로, 데이터를 입력받는 클래스이다.
- 고급 입출력은 자바 I/O에서 제공하는 클래스들을 사용해야 하며, Scanner 클래스는 상대적으로 간단한 입력을 처리하는 데 사용한다.
- 객체를 생성할 때 입력 스트림을 변경하면 키보드 외의 파일, 네트워크 등에서 데이터 입력을 받아들 수 있다.

```
Scanner scan = new Scanner(System.in);  
int num = scan.nextInt();  
String str = scan.next();
```

# 자바기초

## 02. java.util 패키지의 주요 클래스

### ■ Scanner 클래스의 주요 메서드

표 6-8 Scanner 클래스의 주요 메서드

메서드	설명
<code>boolean hasNext( )</code>	다음으로 읽어 올 데이터가 있는지 검사한다. <code>hasNextByte</code> , <code>hasNextBigInteger</code> , <code>hasNextDouble</code> 등 여러 데이터 타입을 지정하여 검사할 수 있는 메서드도 있다.
<code>String next( )</code>	가장 기본 메서드로, 다음 문자열을 가져온다. 이외에도 <code>nextInt</code> , <code>nextDouble</code> , <code>nextBigInteger</code> 등 여러 데이터 타입을 지정하여 가져올 수 있는 메서드도 있으니, 자세한 내용은 API 문서를 참고한다.
<code>Scanner useDelimiter (String pattern)</code>	파라미터의 패턴을 델리미터(구분자)로 사용하는 Scanner 객체를 리턴한다. 리턴된 Scanner는 해당 델리미터를 기준으로 데이터를 파싱한다.
<code>void reset( )</code>	현재 Scanner를 리셋한다.
<code>close( )</code>	현재 Scanner의 입력을 닫는다.

## 02. java.util 패키지의 주요 클래스

### ■ Scanner 클래스의 활용 예제

예제 6-6 키보드 입력을 처리하는 Scanner 클래스 사용하기

Ch6Ex6.java

```
01 package javabook.ch6;
02
03 import java.util.Scanner;
04
05 public class Ch6Ex6 {
06     public static void main(String[] args) {
07         Scanner scan = new Scanner(System.in);
08         String msg = scan.next();
09         System.out.println("입력: " + msg);
10         int num = scan.nextInt();
11         System.out.println("입력: " + num);
12     }
13 }
```

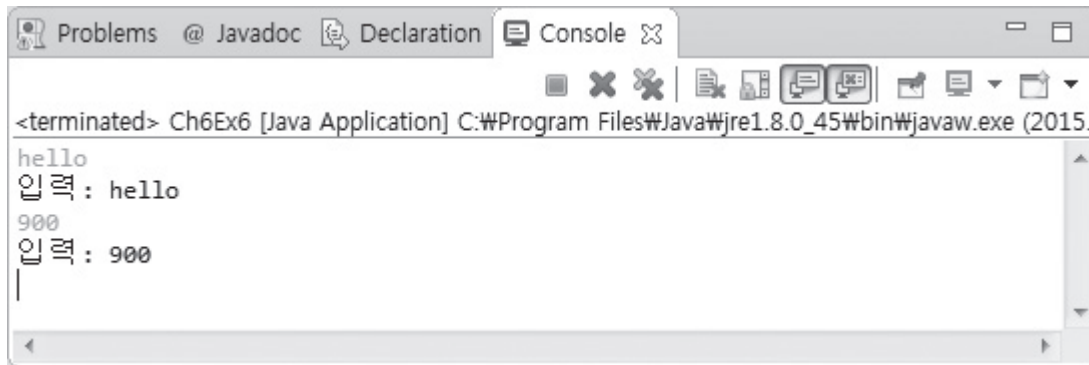


# 자바기초

## 02. java.util 패키지의 주요 클래스

### ■ Scanner 클래스의 활용 예제

[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 program has terminated and displays the input 'hello' and '900' with their corresponding prompts in Korean. The command prompt path is visible at the top of the console area.

```
<terminated> Ch6Ex6 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015.  
hello  
입력: hello  
900  
입력: 900  
|
```

# 자바기초

## ■ 표준 입력 : Scanner

```
Scanner s = new Scanner(System.in);  
변수 = s.메소드( )
```

제공 메소드	설명
nextByte()	정수를 입력받는다(범위: -128~+127).
nextShort()	정수를 입력받는다(범위: -32768~+32767).
nextInt()	정수를 입력받는다(범위: 약 -21억~+21억).
nextLong()	정수를 입력받는다(범위: 약 -900경~+900경).
nextFloat()	실수를 입력받는다(정밀도는 소수점 아래 약 일곱 자리).
nextDouble()	실수를 입력받는다(정밀도는 소수점 아래 약 열다섯 자리).
next()	한 단어를 입력받는다.
nextLine()	한 줄을 입력받는다.

# 자바기초

## Section 02 표준 입출력(5)

### 실습 10-6 표준 입력 사용 예

```
01 import java.util.Scanner;
02
03 public class Ex10_06 {
04     public static void main(String[] args) {
05         Scanner s = new Scanner(System.in); ----- Scanner형의 변수 s를 선언한다.
06         byte a;
07         short b;
08         int c;
09         long d;
10         float e;
11         double f;
12         String str1, str2; ----- 다양한 변수를 선언한다.
```

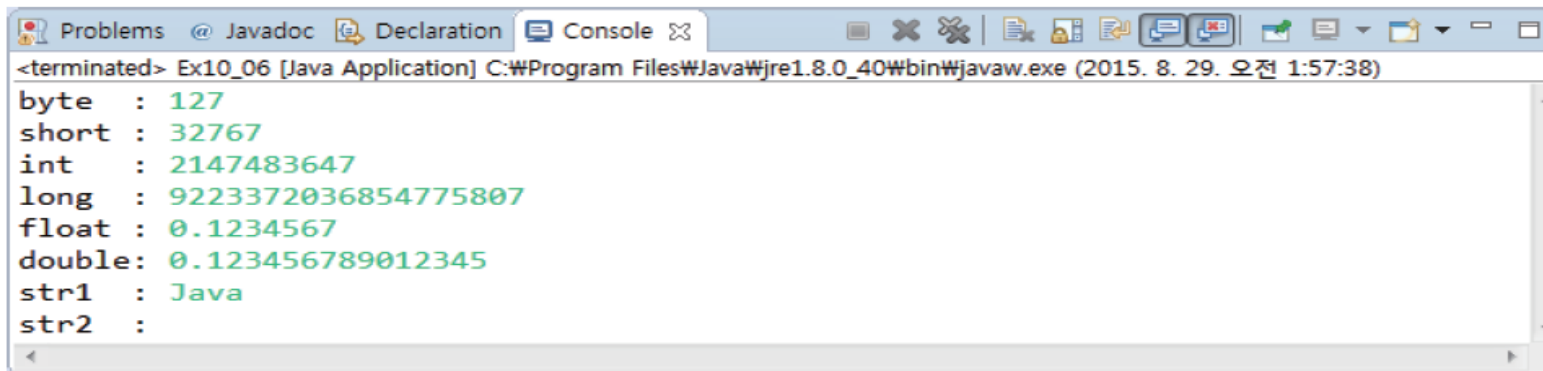
# 자바기초

## Section 02 표준 입출력(6)

13

```
14 System.out.print("byte : ");    a = s.nextByte();
15 System.out.print("short : ");   b = s.nextShort();
16 System.out.print("int  : ");    c = s.nextInt();
17 System.out.print("long  : ");   d = s.nextLong();
18 System.out.print("float : ");   e = s.nextFloat();
19 System.out.print("double: ");   f = s.nextDouble();
20 System.out.print("str1  : ");   str1 = s.next();
21 System.out.print("str2  : ");   str2 = s.nextLine();
22 }
23 }
```

각 변수에 값을 입력한다.



```
<terminated> Ex10_06 [Java Application] C:\Program Files\Java\jre1.8.0_40\bin\javaw.exe (2015. 8. 29. 오전 1:57:38)
byte   : 127
short  : 32767
int    : 2147483647
long   : 9223372036854775807
float  : 0.1234567
double: 0.123456789012345
str1   : Java
str2   :
```

그림 10-9 실행 결과

# 자바기초

## Section 02 표준 입출력(7)

▪ 20행

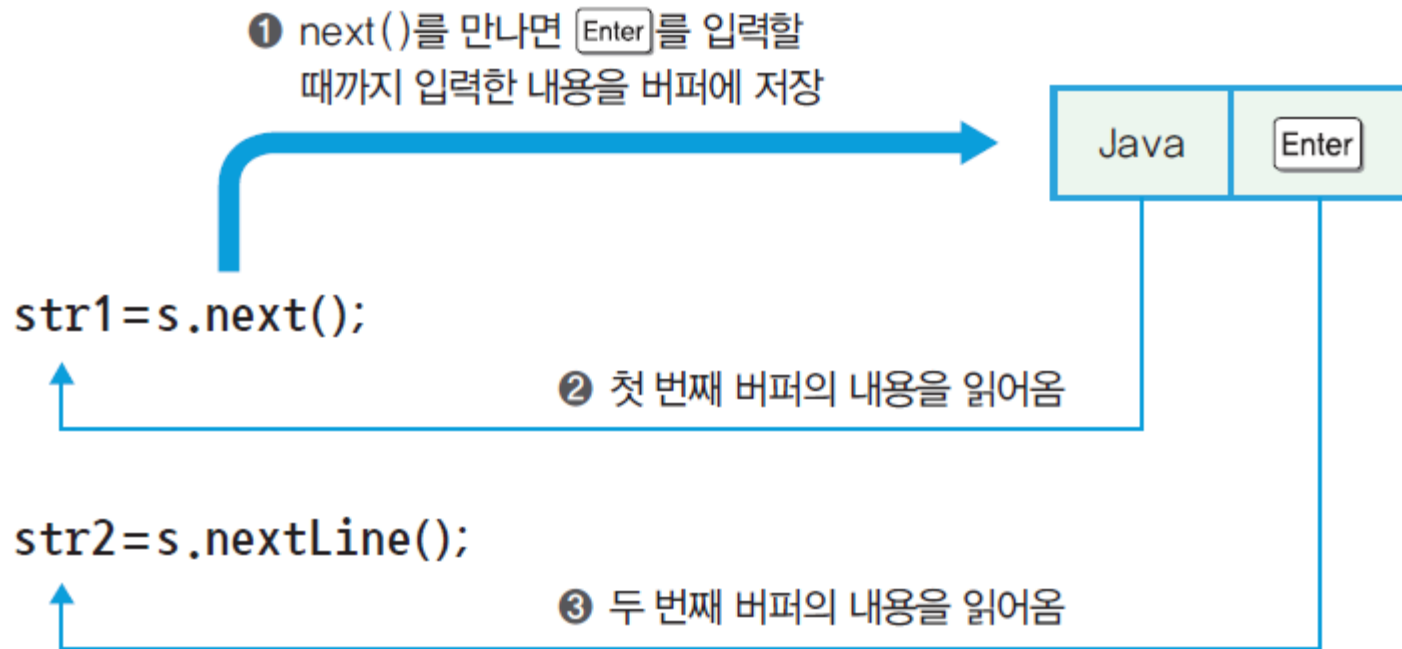


그림 10-10 next()의 작동

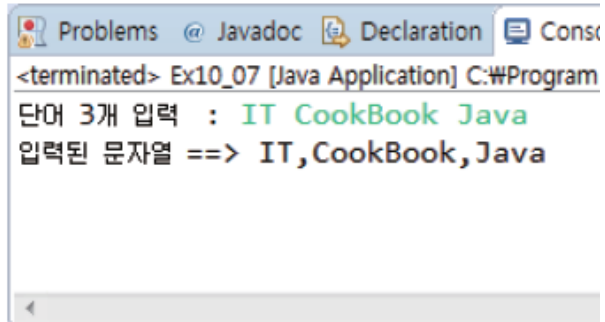
# 자바기초

## Section 02 표준 입출력(8)

### 실습 10-7 next()의 작동 예

```
01 import java.util.Scanner;
02
03 public class Ex10_07 {
04     public static void main(String[] args) {
05         Scanner s = new Scanner(System.in);
06         String str1, str2, str3;
07
08         System.out.print("단어 3개 입력 : ");
09         str1 = s.next();
10         str2 = s.next();
11         str3 = s.next();
12
13         System.out.print("입력된 문자열 ==> ");
14         System.out.print(str1 + "," + str2 + "," + str3);
15     }
16 }
```

버퍼에서 3개의 단어를 읽어들이다.



# 자바 기초

## 01. 변수와 자료형

### ■ 변수의 선언

- 변수 : 어떤 값을 저장하기 위한 메모리 공간

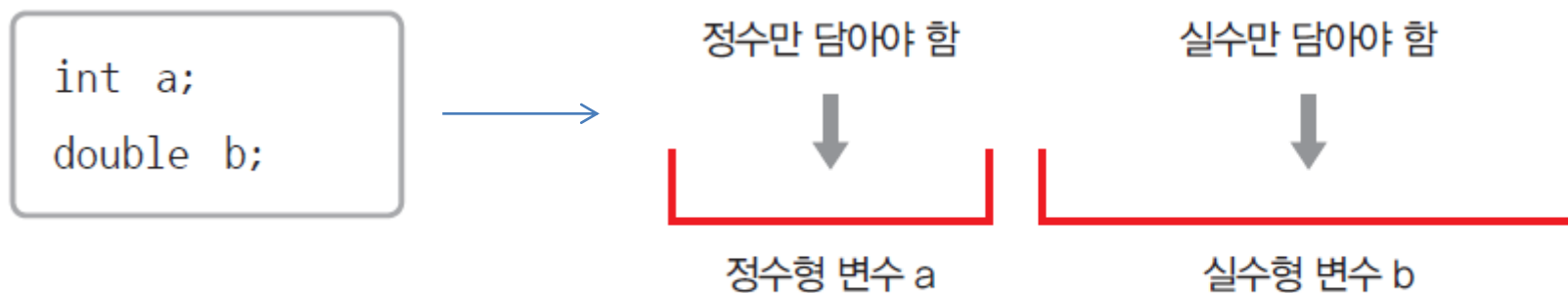


그림 3-12 정수형 변수와 실수형 변수의 간단한 개념

- TIP : 정수형 int는 4byte, 실수형 float는 4byte, 실수형 double은 8byte.  
[그림 3-12]는 실수형이 double이라고 가정한 상태임

# 자바기초

## 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
```

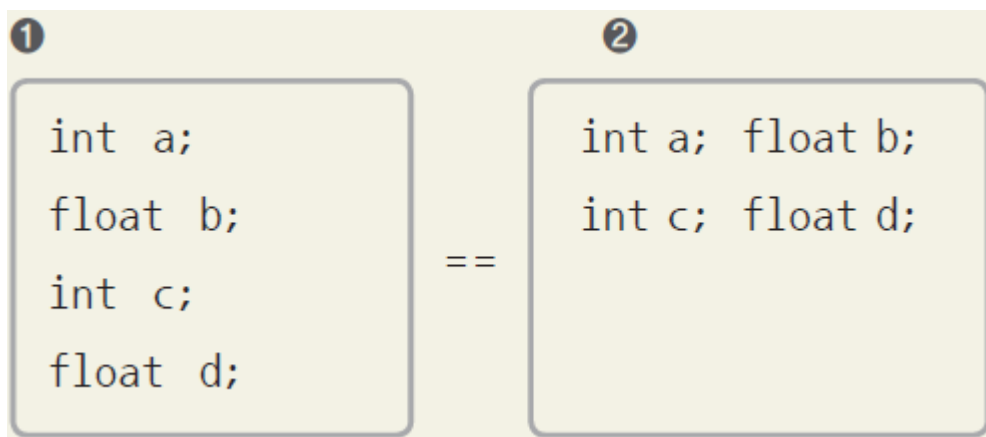


# 자바 기초

## 01. 변수와 자료형

### ■ 세미콜론(;)으로 문장 구분하기

- 한 줄에 하나의 데이터 형식만 선언할 수 있다고 했는데, 엄밀하게 말하면 '한 줄'이 아니라 '한 문장'이라고 해야 옳다. ❷은 올바른 형식이다. 세미콜론(;)으로 구분한 것은 완전히 분리된 문장으로 취급하기 때문에 아래 ❶과 ❷은 동일한 의미이다.



# 자바 기초

## 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 result와 int Result는 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; (×)

# 자바기초

## 01. 변수와 자료형

### ■ 변수 이름 규칙

예제 3-1 변수를 선언하고 저장된 값 출력하기

Ch3Ex1.java

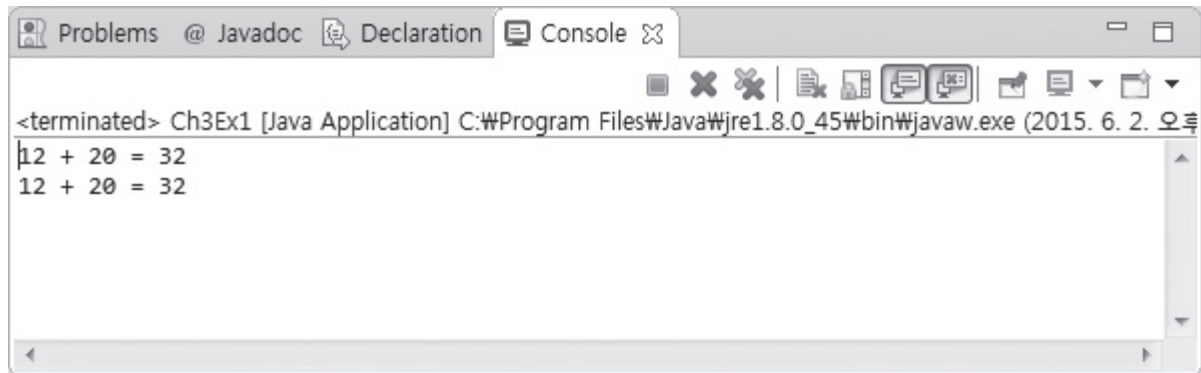
```
01 package javabook.ch3;
02
03 public class Ch3Ex1 {
04     public static void main(String[] args) {
05         int num1 = 12;
06
07         int num2 = 20;
08         int result = num1 + num2;
09
10         System.out.println(num1 + " + " + num2 + " = " + result);
11         System.out.printf("%d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

# 자바기초

## 01. 변수와 자료형

### ■ 변수 이름 규칙

[실행결과]



```
<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  
12 + 20 = 32
```

# 자바기초

## 01. 변수와 자료형

### ■ 변수 이름 규칙

#### [코드설명]

표 3-1 printf( )의 형식 문자와 자료형

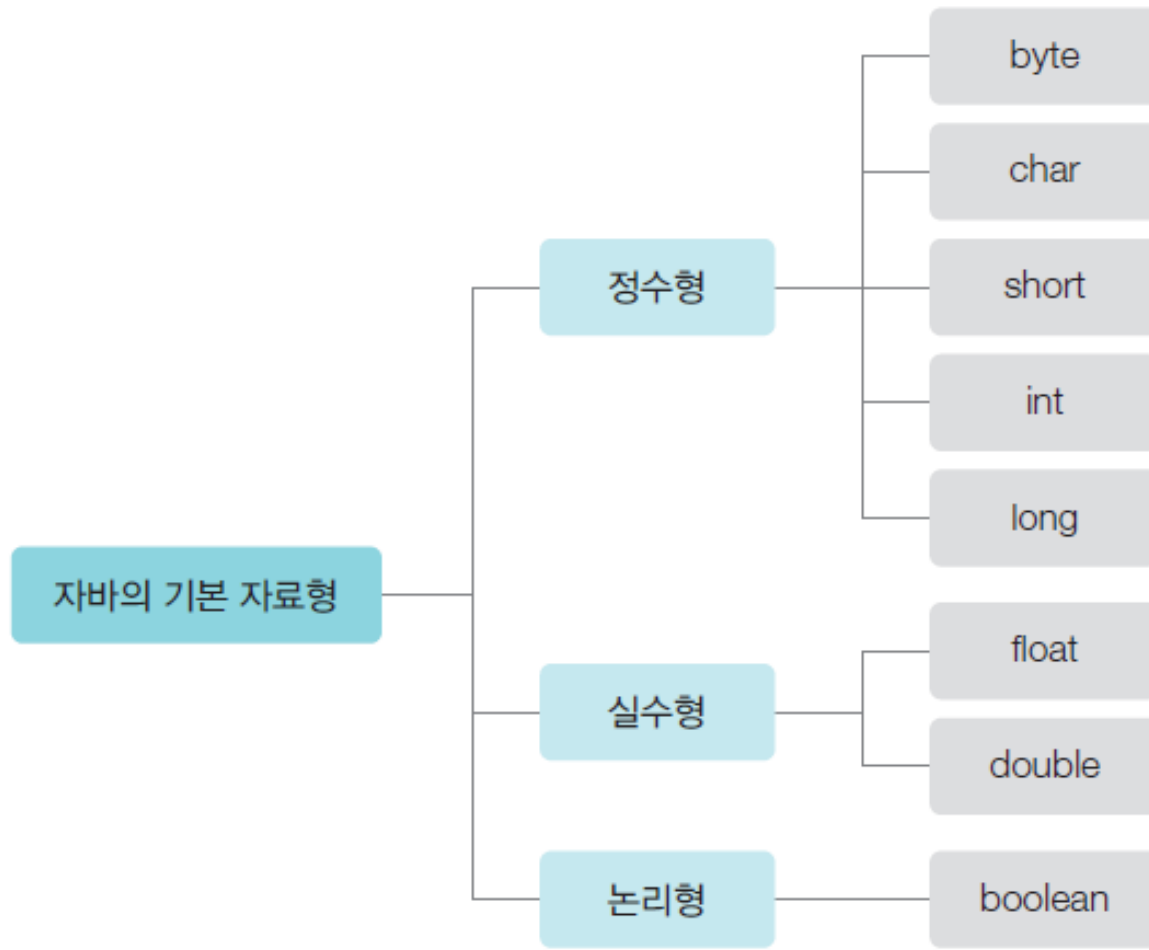
형식 문자	자료형	형식 문자	자료형
%c	문자형	%o	8진수
%d	정수형	%s	문자열
%e	지수형	%u	부호 없는 정수형
%f	실수형	%x	16진수
%i	정수형	%%, \w%	% 문자 출력

# 자바기초

## 01. 변수와 자료형

### ■ 기본 자료형

그림 3-1 자바의 기본 자료형



# 자바 기초

## 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.

# 자바기초

## 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     }
```



# 자바기초

## 01. 변수와 자료형

### ■ 기본 자료형 : 정수형

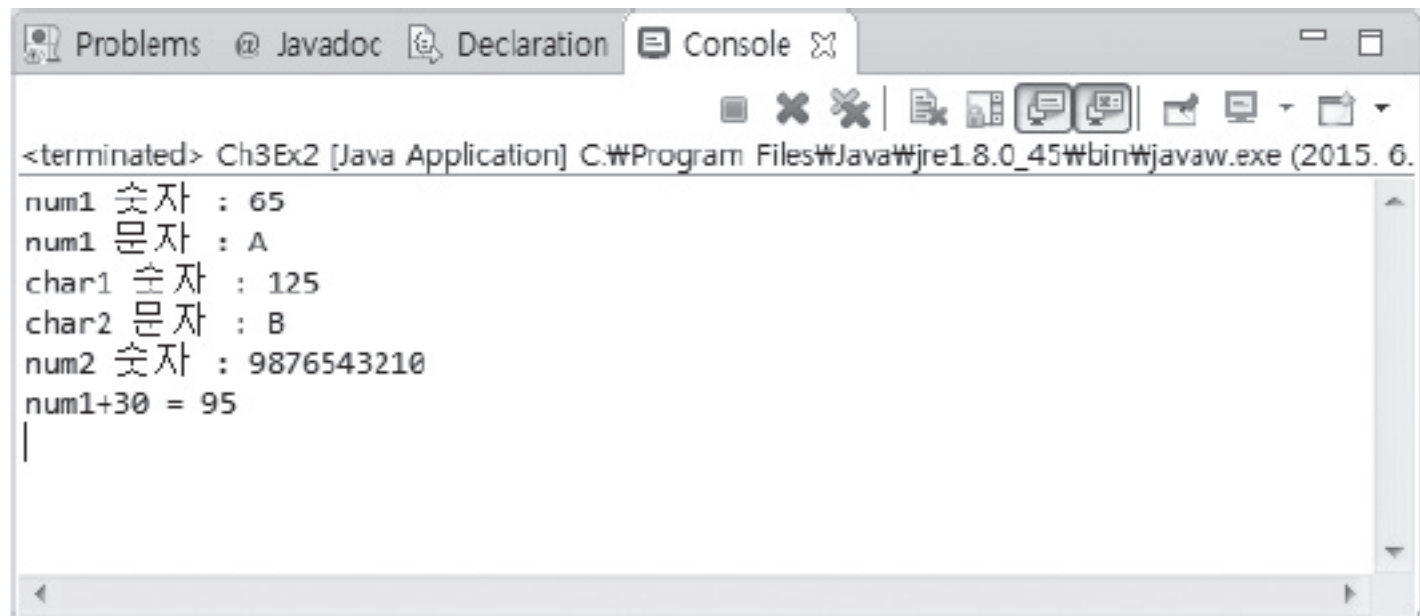
```
12      System.out.printf("num1 숫자 : %d \n", num1);
13      System.out.printf("num1 문자 : %c \n", num1);
14      System.out.printf("char1 숫자 : %d \n", char1);
15      System.out.printf("char2 문자 : %c \n", char2);
16      System.out.printf("num2 숫자 : %d \n", num2);
17
18      result = num1 + 30;
19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

# 자바기초

## 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  
|
```

# 자바 기초

## 01. 변수와 자료형

### ■ 기본 자료형 : 논리형

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

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

# 자바 기초

## 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.

# 자바기초

## • 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
```

# 자바기초

## 01. 변수와 자료형

### ■ 기본 자료형 : 논리형, 실수형을 사용한 예제

예제 3-3 논리형과 실수형을 사용한 예제

Ch3Ex3.java

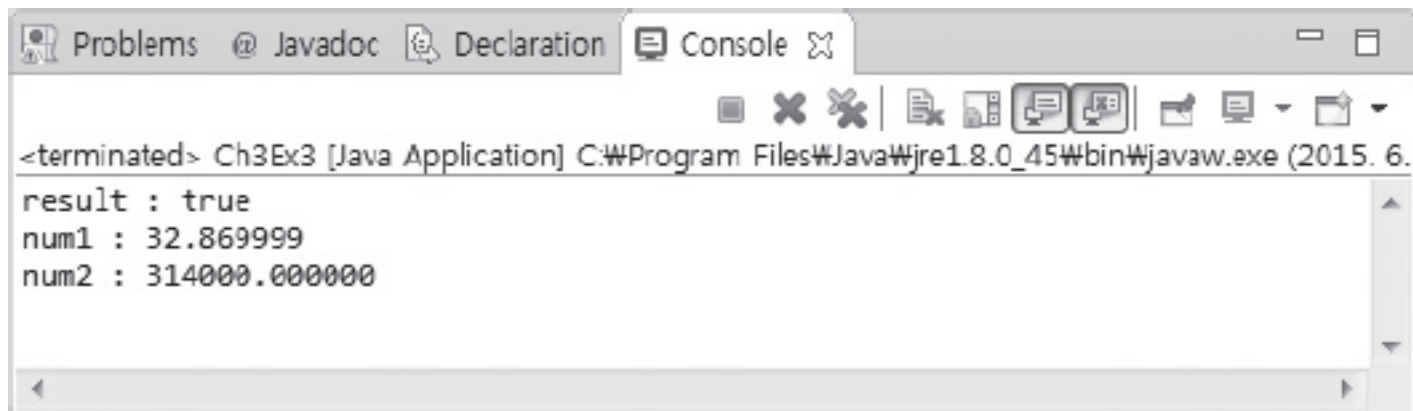
```
01 package javabook.ch3;
02
03 public class Ch3Ex3 {
04     public static void main(String[] args) {
05         boolean result = true;
06         float num1 = 32.87f;
07         double num2 = 3.14E5;
08
09         System.out.printf("result : %s \n", result);
10         System.out.printf("num1 : %f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

# 자바기초

## 01. 변수와 자료형

- 기본 자료형 : 논리형, 실수형을 사용한 예제

[실행결과]



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

```
<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
```

# 자바기초

## 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



# 자바기초

## 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;  
    }  
}
```

# 자바 기초

## 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.

# 자바 기초

## 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.

# 자바기초

## 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;

# 자바 기초

## 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.
```

# 자바기초

## 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("-----");
```

# 자바기초

## 02. 연산자

### ■ 산술 연산자의 기본 사용 예제

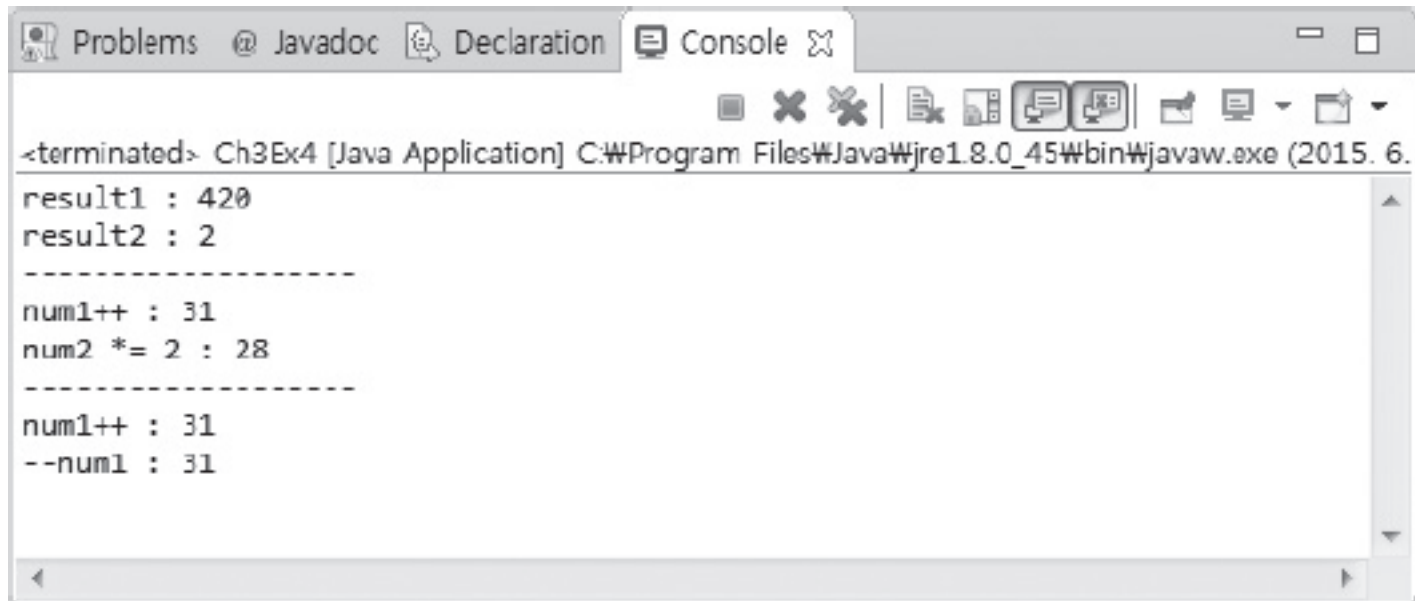
```
14
15     num1++;
16     num2 *= 2;
17     System.out.printf("num1++ : %d \n", num1);
18     System.out.printf("num2 *= 2 : %d \n", num2);
19     System.out.println("-----");
20
21     System.out.printf("num1++ : %d \n", num1++);
22     System.out.printf("--num1 : %d \n", --num1);
23 }
24 }
```

# 자바기초

## 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
```



# 자바기초

## 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.

# 자바기초

## 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;
```

# 자바 기초

## 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.

# 자바기초

## 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

# 자바기초

## 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;
```

# 자바기초

## 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;
```

# 자바기초

## 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);
```

# 자바기초

## 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
```



# 자바기초

## 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

---

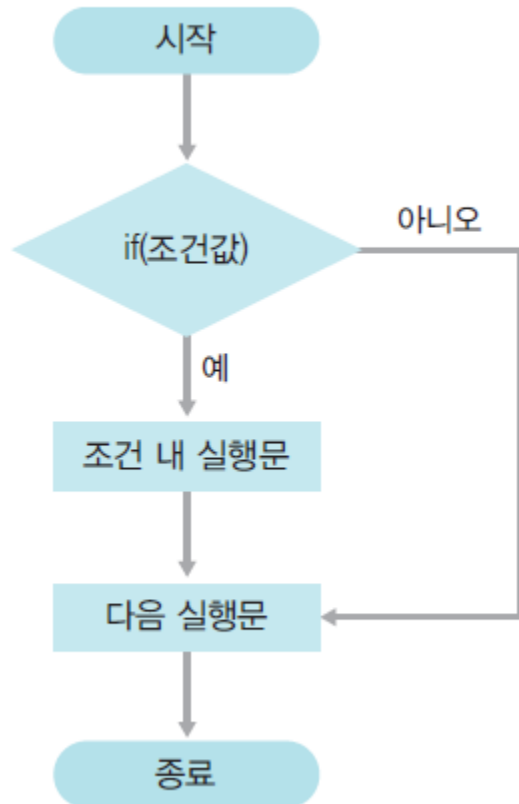
#### 기본 구조

```
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);
```

# 자바기초

## 03. 분기문

### ■ if 문 : if 문 단독 사용 예제

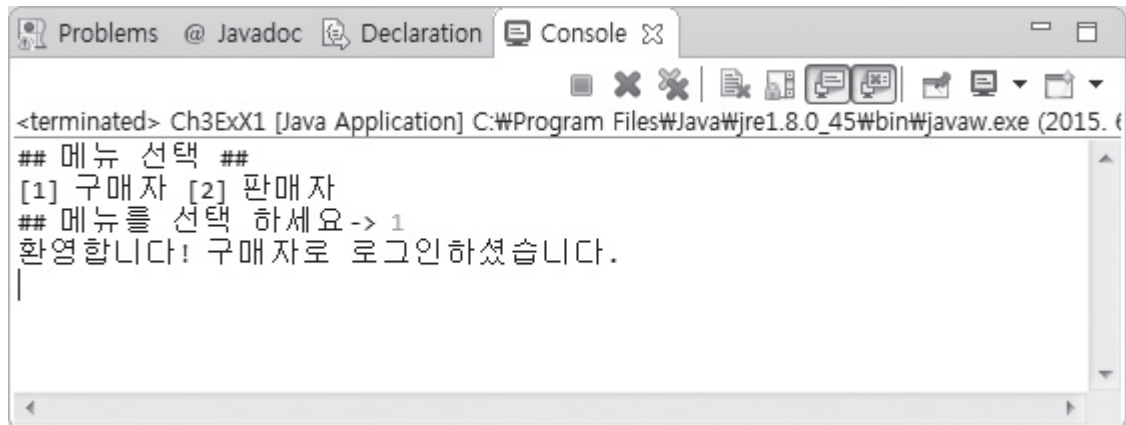
```
08
09     System.out.println("## 메뉴 선택 ##");
10     System.out.println("[1] 구매자 [2] 판매자");
11     System.out.print("## 메뉴를 선택 하세요-> ");
12
13     String sel = scan.next();
14
15     if(sel.equals("1")) {
16         System.out.println("환영합니다! 구매자로 로그인하셨습니다.");
17     }
18     if(sel.equals("2")) {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

# 자바기초

## 03. 분기문

### ■ if 문 : if 문 단독 사용 예제

[실행결과]



```
<terminated> Ch3ExX1 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. 10. 14:00:00)
## 메뉴 선택 ##
[1] 구매자 [2] 판매자
## 메뉴를 선택 하세요 -> 1
환영합니다! 구매자로 로그인하셨습니다.
|
```

# 자바기초

## 03. 분기문

### ■ if 문 : if ~ else 문

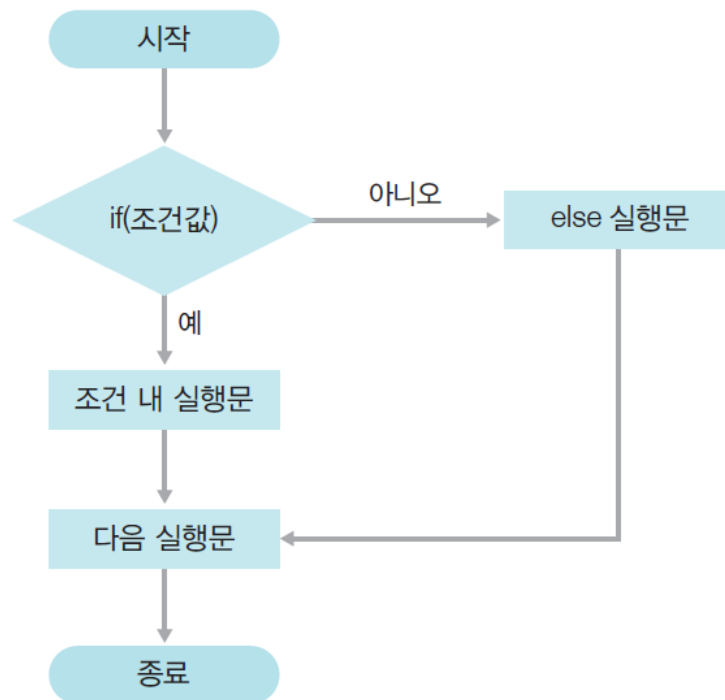
**사용 분야** 특정 조건이 참(true)일 때 처리해야 할 일과 거짓(false)일 때 처리해야 할 일이 다르다면 사용한다.

**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



# 자바기초

## 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     }
```



# 자바기초

## 03. 분기문

### ■ if 문 : if ~ else 문 예제

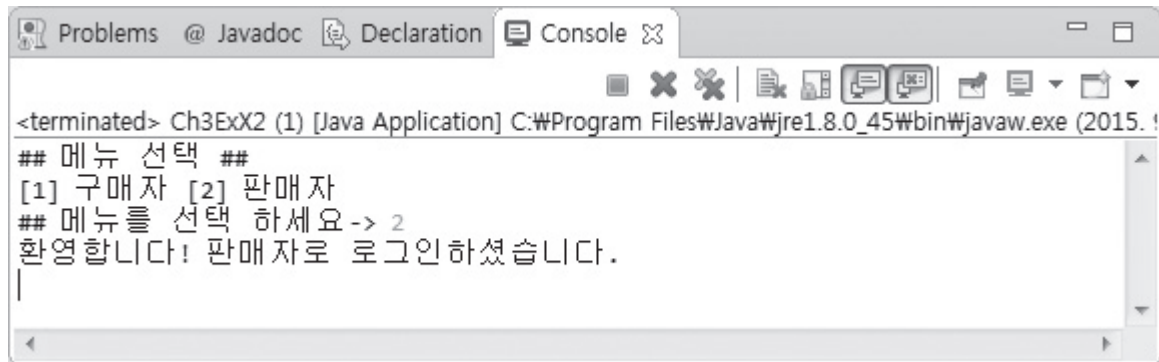
```
15         if(sel.equals("1")) {  
16             System.out.println("환영합니다! 구매자로 로그인하셨습니다.");  
17         }  
18         else {  
19             System.out.println("환영합니다! 판매자로 로그인하셨습니다.");  
20         }  
21     }  
22 }
```

# 자바기초

## 03. 분기문

### ■ if 문 : if ~ else 문 예제

[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> 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

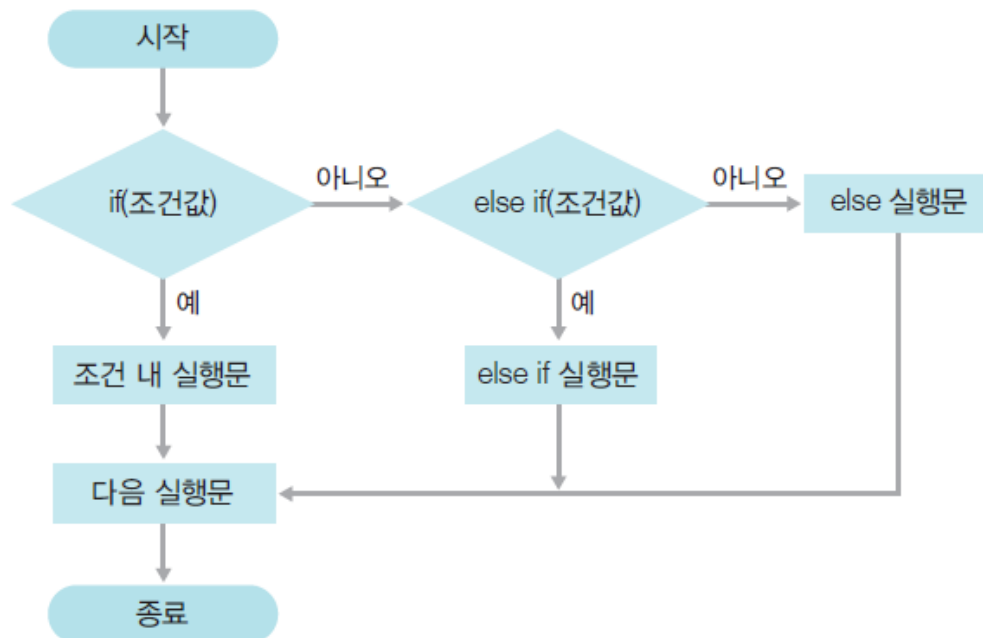
**사용 분야** 여러 조건이 있고, 각 조건마다 처리해야 하는 내용이 다를 때 사용한다. 조건이 세분화되어 있으면 else if 문을 여러 개 사용할 수 있다.

**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



# 자바기초

## 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
```

# 자바기초

## 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         }
```

# 자바기초

## 03. 분기문

### ■ if 문 : if ~ else if ~ else 문 예제

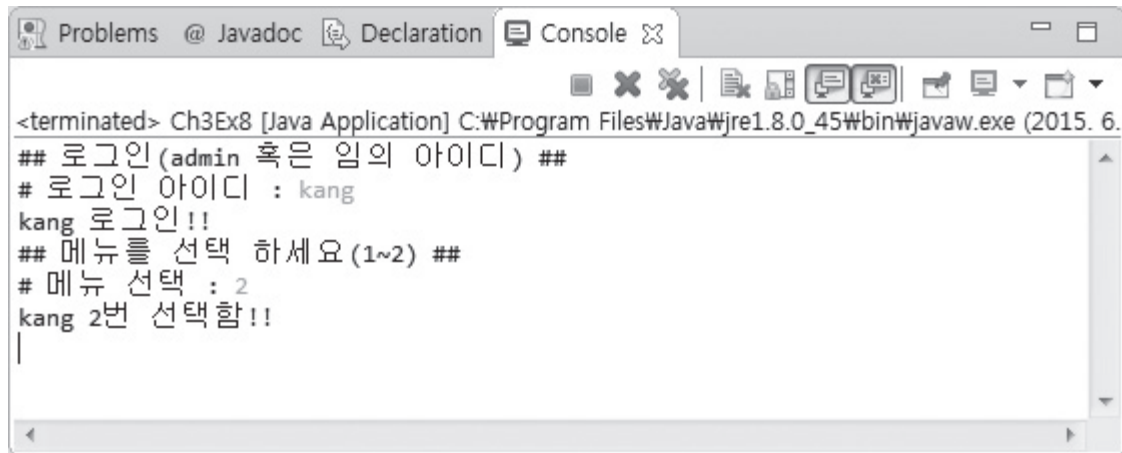
```
28         else if(sel.equals("2") && user.equals("admin")) {
29             System.out.println("관리자 2번 선택함!!");
30         }
31         else if(sel.equals("1") && !user.equals("admin")) {
32             System.out.println(user + " 1번 선택함!!");
33         }
34         else if(sel.equals("2") && !user.equals("admin"))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

# 자바기초

## 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!!
|
```

# 자바기초

## 03. 분기문

### ■ if 문 : 중첩 if 문

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

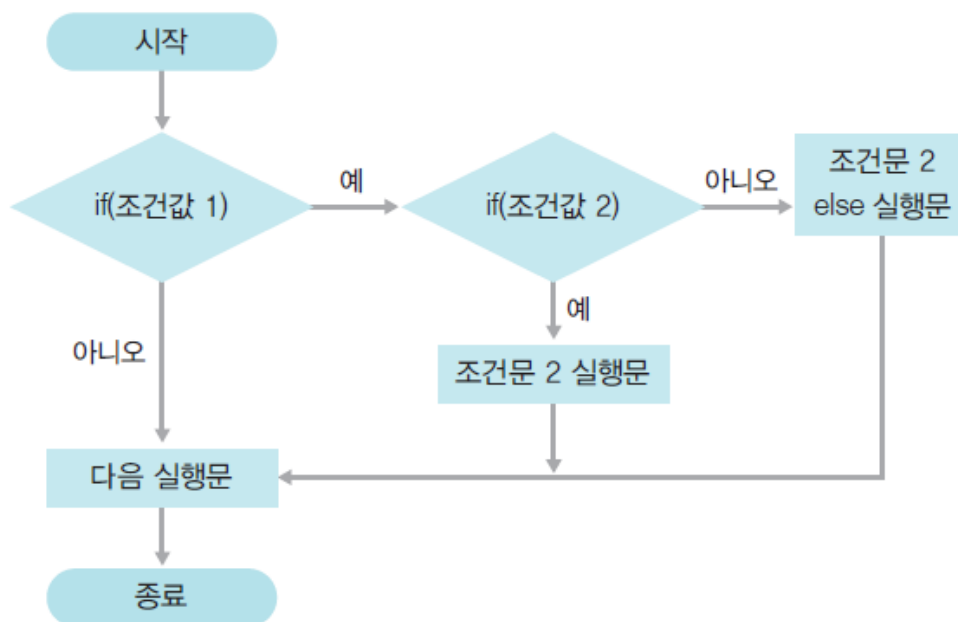
**사용 예**

- 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();
```

# 자바기초

## 03. 분기문

### ■ if 문 : 중첩 if 문 예제 (뒷장 계속)

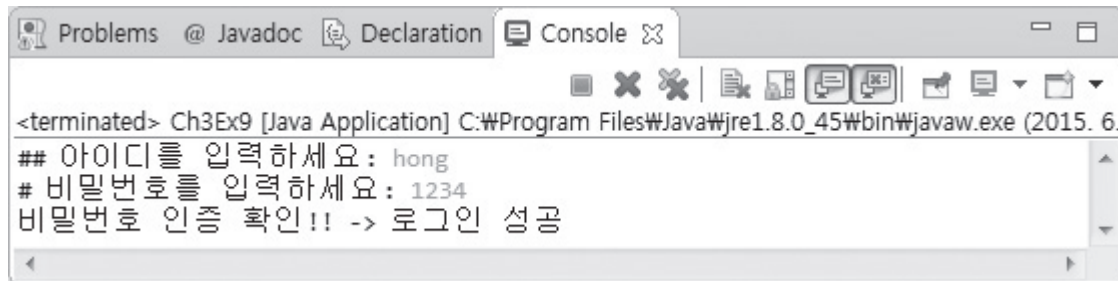
```
16
17         if(pwd.equals("1234")) {
18             System.out.println("비밀번호 인증 확인!! -> 로그인 성공");
19         }
20         else
21             System.out.println("비밀번호가 틀렸습니다.!!");
22     }
23     else
24         System.out.println("아이디가 틀렸습니다.!!");
25 }
26 }
```

# 자바기초

## 03. 분기문

### ■ if 문 : 중첩 if 문 예제 (뒷장 계속)

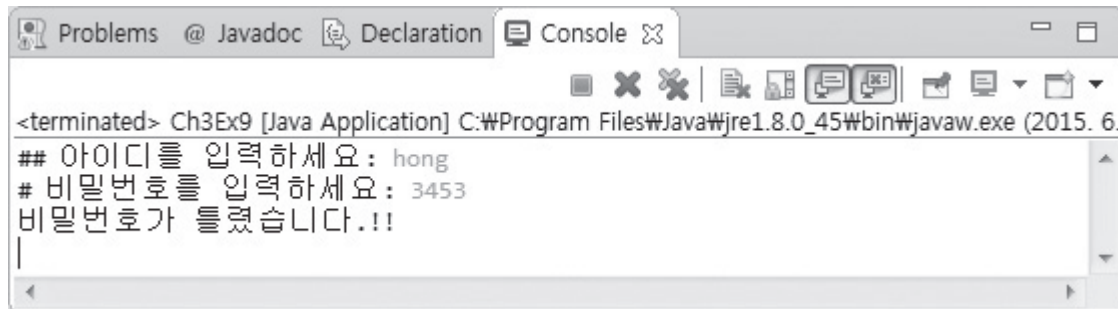
#### [실행결과]



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.!!  
|
```

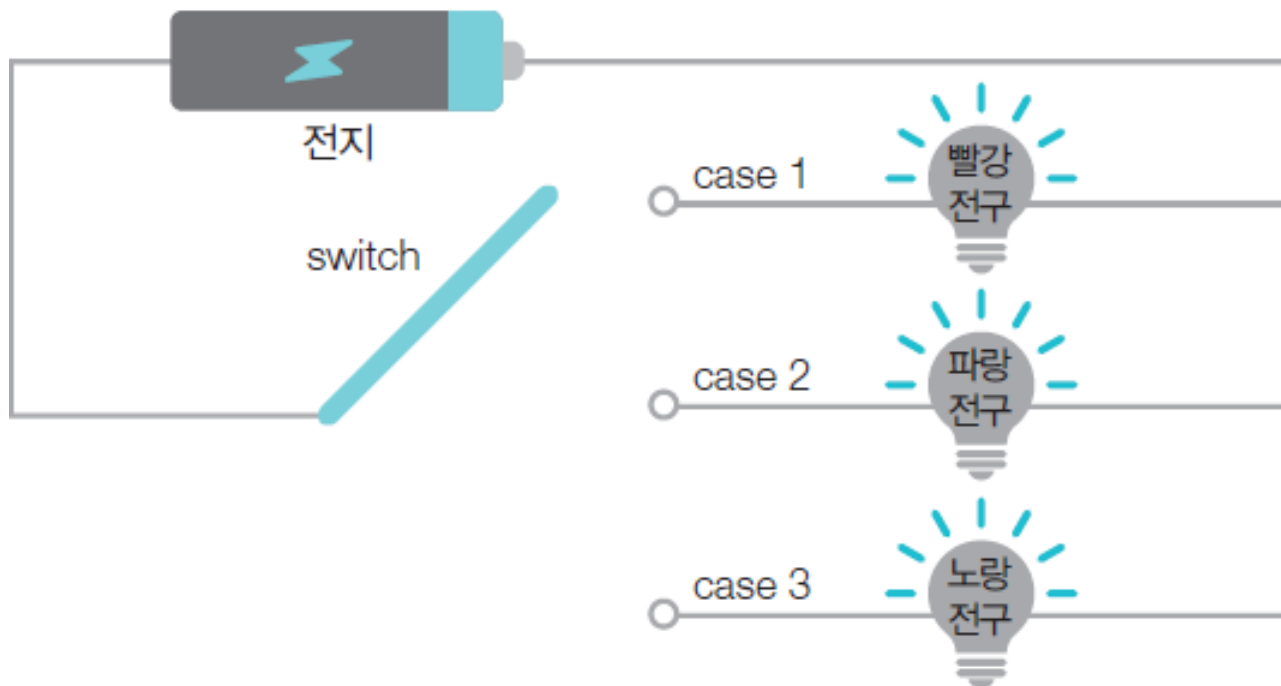
# 자바기초

## 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



# 자바기초

## 03. 분기문

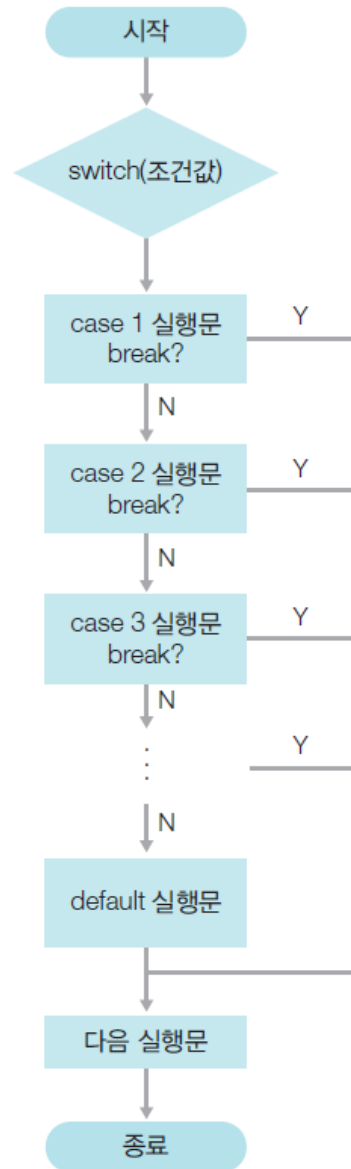
### ■ switch 문

- **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



## 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;
```

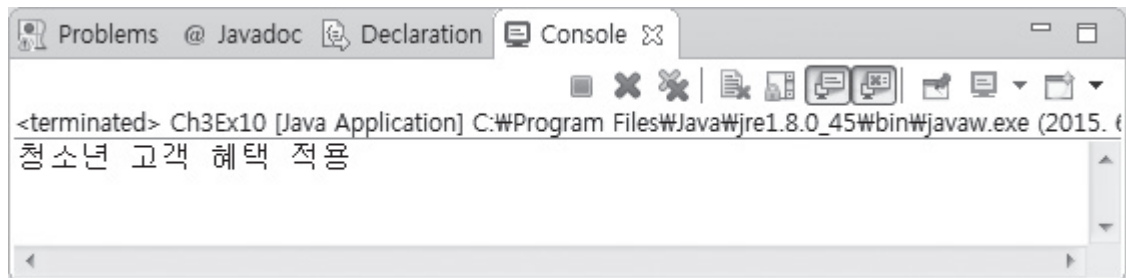
# 자바기초

## 03. 분기문

### ■ switch 문

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

#### [실행결과]



# 자바기초

## 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]

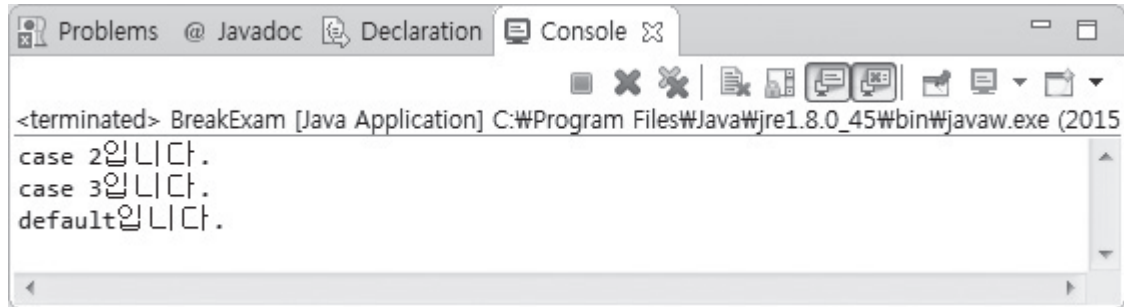


# 자바기초

## 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.
```

## 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.

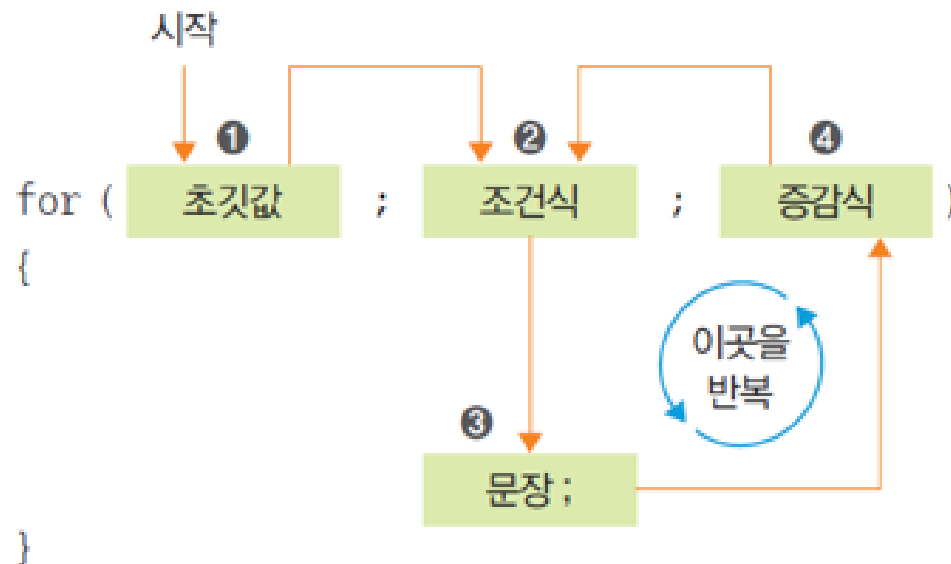


그림 for 문의 실행 순서

# 자바기초

## 04. 반복문

### ■ 반복문 : for 문

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

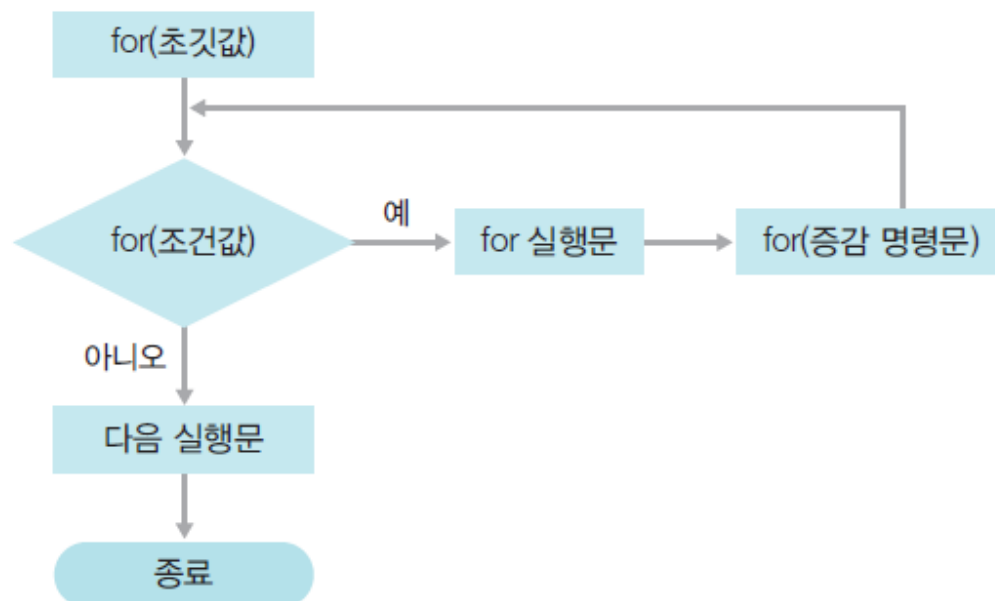
**사용 예**

- 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



# 자바기초

## 04. 반복문

- for 문의 실행 순서
- 반복문 : for 문

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

# 자바기초

## 04. 반복문

### ■ 반복문 : for 문



# 자바기초

## 04. 반복문

### ■ 반복문 : for 문

① 초기값 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

# 자바기초

## 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

# 자바기초

## 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.



# 자바기초

## 04. 반복문

### ■ 반복문 : for 문

예제 3-13 for 문으로 구구단 출력하기

Ch3Ex11.java

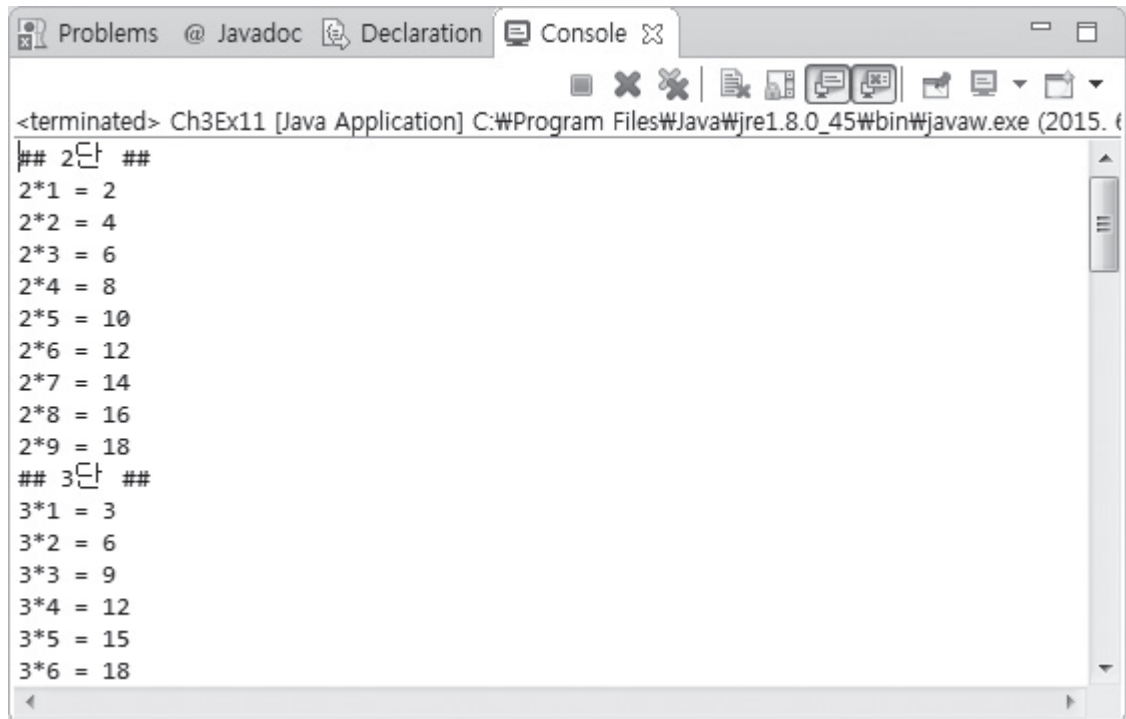
```
01 package javabook.ch3;
02
03 public class Ch3Ex11 {
04     public static void main(String[] args) {
05         for(int i = 2; i < 10; i++) {
06             System.out.println("## " + i + "단 ##");
07             for(int j = 1; j < 10; j++) {
08                 System.out.printf("%d * %d = %d\n",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

# 자바기초

## 04. 반복문

### ■ 반복문 : for 문

[실행결과]



```
<terminated> Ch3Ex11 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015. 6. 11. 11:11:11 AM)
## 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
```

# 자바 기초

## 04. 반복문

### ■ 반복문 : while 문

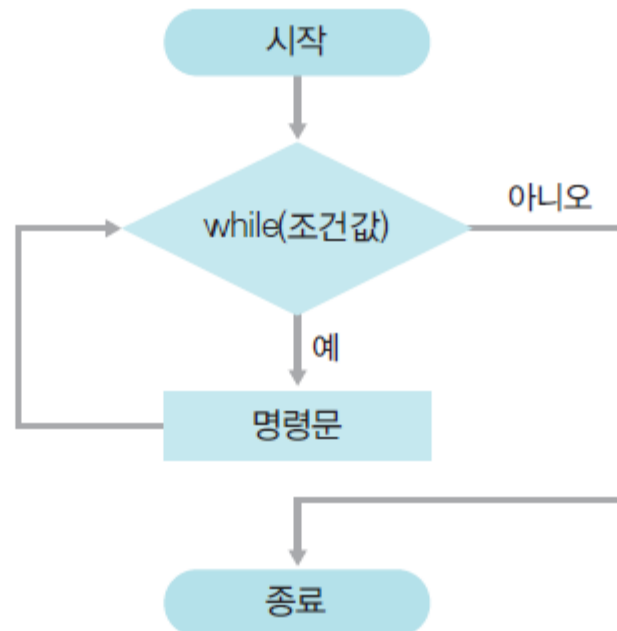
**사용 분야** 특정 조건을 만족하는 동안 반복할 때 사용한다.

- 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



## 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) {
```

## 04. 반복문

### ■ 반복문 : while 문

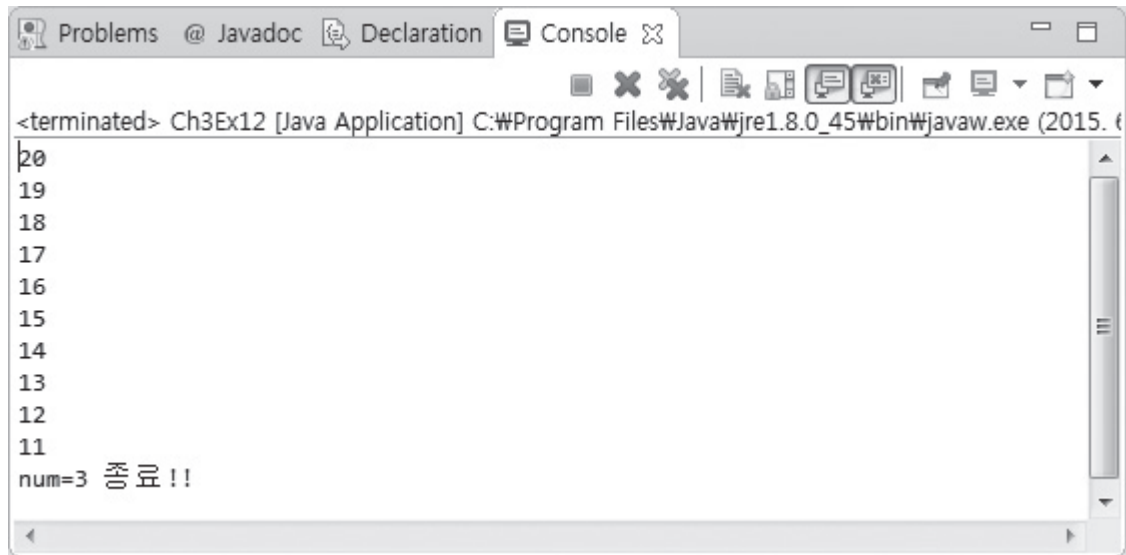
```
14         flag = false;
15         System.out.println("num=3 종료!!");
16     }
17 }
18 }
19 }
```

# 자바기초

## 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 a sequence of numbers from 20 down to 11, followed by the text 'num=3 종료!!'. The window title is '<terminated> Ch3Ex12 [Java Application] C:\Program Files\Java\jre1.8.0\_45\bin\javaw.exe (2015.6.12)'.

```
<terminated> Ch3Ex12 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015.6.12)
20
19
18
17
16
15
14
13
12
11
num=3 종료!!
```

# 자바기초

## 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

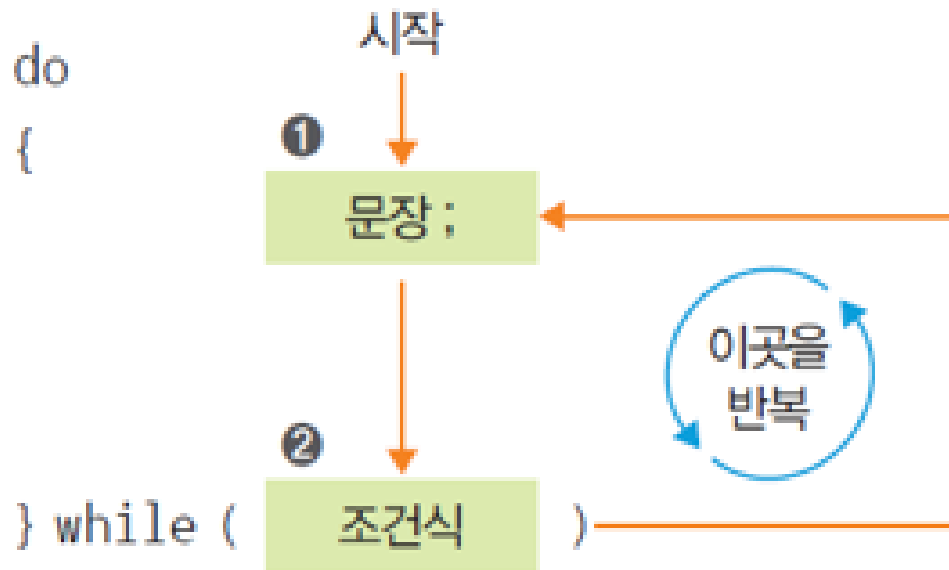


그림 do~while 문의 형식과 실행 순서

## 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.



# 자바 기초

## 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

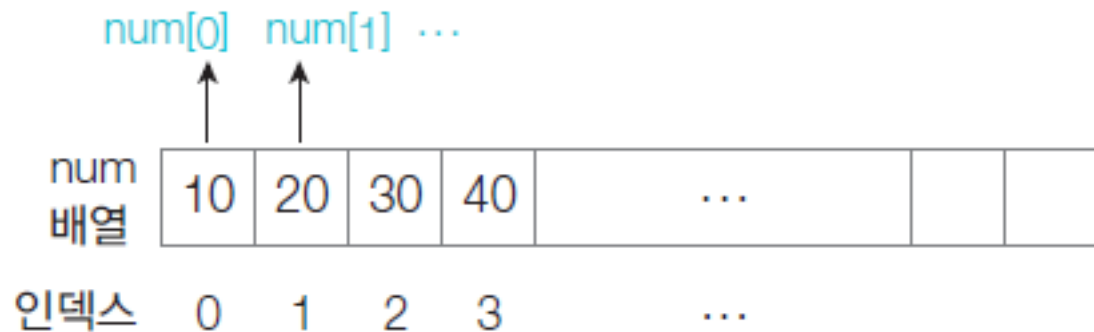
# 자바기초

## 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



# 자바기초

## 05. 배열

### ■ 자바 배열의 특징

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

# 자바 기초

## 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

# 자바기초

## 05. 배열

### ■ 자바 배열의 특징

예제 3-15 배열을 사용하여 제품 목록 출력하기

Ch3ExX3.java

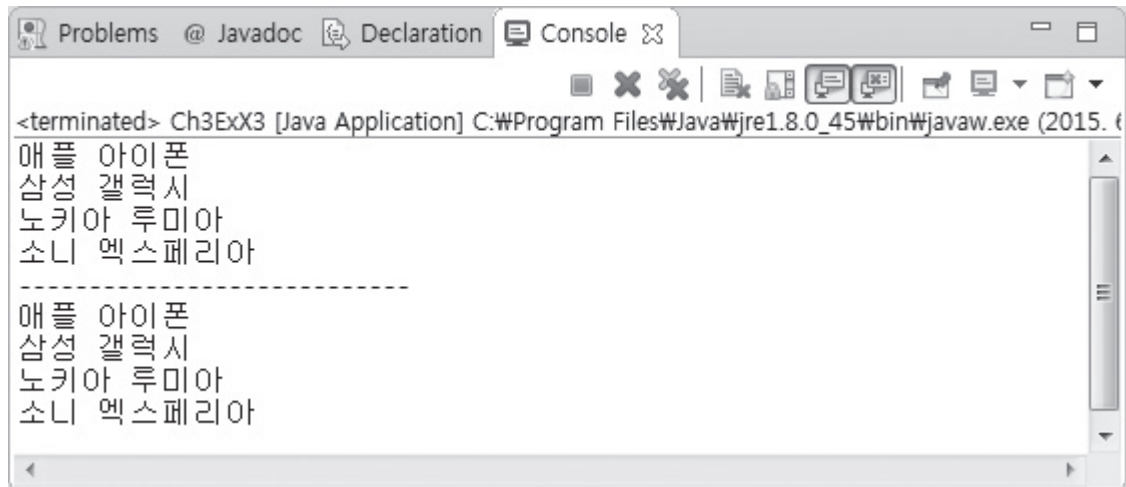
```
01 package javabook.ch3;
02
03 public class Ch3ExX3 {
04     public static void main(String[] args) {
05         String[] products = {"애플 아이폰", "삼성 갤럭시", "노키아 루미아", "소니 엑스페리아"};
06
07         for(int i = 0; i < products.length; i++) {
08             System.out.println(products[i]);
09         }
10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

# 자바기초

## 05. 배열

### ■ 자바 배열의 특징

[실행결과]



```
<terminated> Ch3ExX3 [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015.6.12)
애플 아이폰
삼성 갤럭시
노키아 루미아
소니 엑스페리아
-----
애플 아이폰
삼성 갤럭시
노키아 루미아
소니 엑스페리아
```

# 자바기초

## 05. 배열

### ■ 자바 배열의 특징

예제 3-16 2차원 배열을 사용한 자료구조 처리하기

Ch3Ex13.java

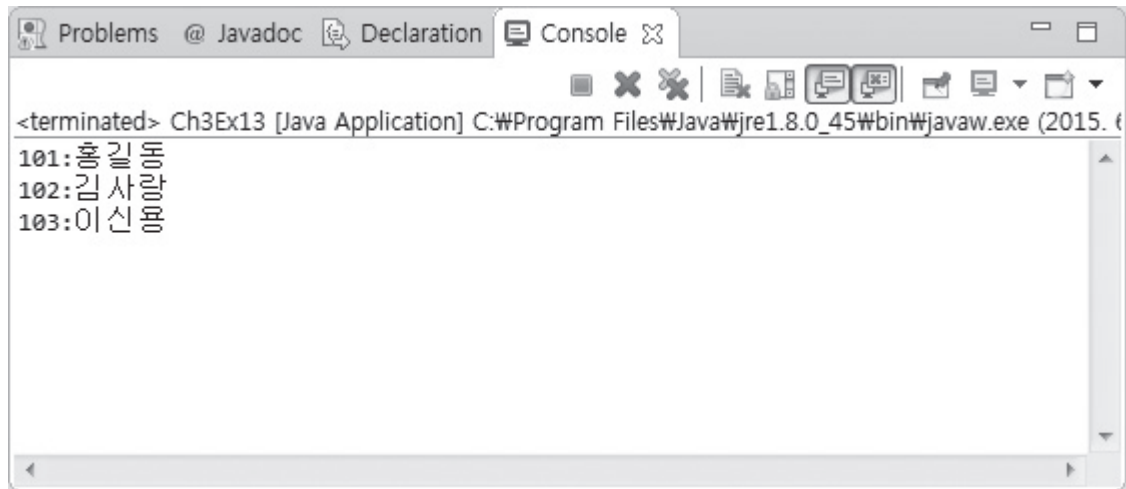
```
01 package javabook.ch3;
02
03 public class Ch3Ex13 {
04     public static void main(String[] args) {
05         String[][] members = {"101", "홍길동"}, {"102", "김사랑"}, {"103", "이신용"};
06
07         for(int i = 0; i < members.length; i++) {
08             System.out.println(members[i][0] + ":" + members[i][1]);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

# 자바기초

## 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
```