

제 2 장 자바 기본 프로그래밍

자바 프로그램 구조 - 맛보기 예제

2

```
/*  
 * 맛보기 예제.  
 * 소스 파일 : Hello2.java  
 */
```

```
public class Hello2 {
```

```
    public static int sum(int n, int m) {  
        return n + m;  
    }
```

메소드

```
// main() 메소드에서 실행 시작
```

```
public static void main(String[] args) {  
    int i = 20;  
    int s;  
    char a;
```

```
    s = sum(i, 10); // sum() 메소드 호출  
    a = '?';
```

```
    System.out.println(a); // 문자 '?' 화면 출력  
    System.out.println("Hello2"); // "Hello2" 문자열 화면 출력  
    System.out.println(s); // 정수 s 값 화면 출력
```

```
}
```

```
}
```

클래스

```
?  
Hello2  
30
```

메소드

맛보기 예제 설명

3

□ 클래스 만들기

- Hello2 이름의 클래스 선언

```
public class Hello2 {  
}
```

- class 키워드로 클래스 정의(4장 참고)
- public으로 선언하면 다른 클래스에서도 접근 가능
- 클래스 본문은 '{'으로 시작하여 '}'으로 끝남

□ main() 메소드

- public static void으로 선언되어야 함

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

- 자바 프로그램은 main() 메소드부터 실행 시작
- String[] args로 실행 인자를 전달 받음(3장 참고)

□ 멤버 메소드

- 메소드 sum() 정의

```
public static int sum(int n, int  
m) {  
...  
}
```

- 클래스 에 속한 함수, 클래스 내에서만 선언
- 인자들의 타입과 변수 명을 ','로 분리하여 나열
- 메소드 코드는 '{'과 '}' 사이에 작성

□ 변수 선언

- 변수 타입과 변수 이름 선언

```
int i=20;  
int s;  
char a;
```

- 메소드 내에서 선언된 변수는 지역 변수
 - 지역 변수는 메소드 실행이 끝나면 저장 공간 반환

□ 메소드 호출

- sum() 메소드 호출

```
s = sum(1,10); // 메소드 sum() 호출
```

- sum() 메소드의 호출 시 변수 i의 값과 정수 10을 전달
- sum() 메소드의 인자인 n, m에 각각 20, 10 값 전달
- sum() 메소드는 n과 m 값을 더한 30 리턴
- 변수 s는 정수 30을 전달받아 저장

sum() 메소드 호출과 리턴

4

```
public static int sum(int n, int m) {  
    return n + m; // 30 리턴  
}
```

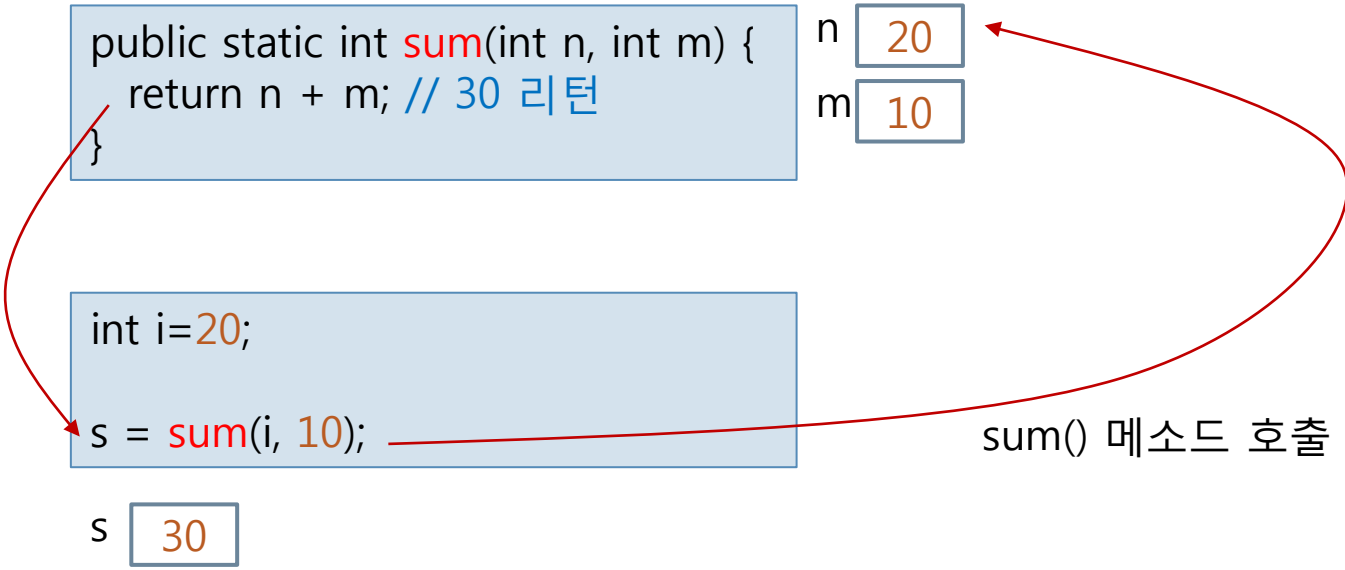
n 20
m 10

```
int i=20;
```

```
s = sum(i, 10);
```

s 30

sum() 메소드 호출



맛보기 예제 설명(계속)

5

주석문

- ▣ //을 만나면 행 끝날 때까지 한 라인을 주석문 처리
- ▣ /*를 만나면 */을 만날 때까지 여러 행을 주석문 처리

화면 출력

- ▣ 표준 출력 스트림에 메시지 출력

```
System.out.println(a); // 문자 ? 화면 출력
System.out.println("Hello2"); // "Hello2" 문자열 화면 출력
System.out.println(s); // 정수 s 값 화면 출력
```

- ▣ 표준 출력 스트림 System.out의 println() 메소드 호출
- ▣ println()은 여러 종류 데이터 타입 출력 가능
- ▣ println()은 출력 후 다음 행으로 커서 이동

문장

- ▣ ;로 한 문장의 끝을 인식

```
int i=20;
b = '?';
s = sum(i, 20);
```

- ▣ 한 문장을 여러 줄에 작성해도 무방

```
b
= '?';
```

- ▣ 주석문 끝에는 ';'를 붙이지 않음

블록

- ▣ 블록은 { 로 시작하여 } 로 끝남

```
public class Hello2 {
....
} // Hello2 클래스 선언문 끝

public static void main(String[] args) {
...
} // 메소드 main() 선언문 끝
```

- ▣ 클래스 선언과 메소드 선언 등은 블록으로 구성

식별자 (identifier)

6

- 식별자란?
 - ▣ 클래스, 변수, 상수, 메소드 등에 붙이는 이름
- 식별자의 원칙
 - ▣ '@', '#', '!'와 같은 특수 문자, 공백 또는 탭은 식별자로 사용할 수 없으나 '_', '\$'는 사용 가능
 - ▣ 유니코드 문자 사용 가능. 한글 사용 가능
 - ▣ 자바 언어의 키워드는 식별자로 사용불가
 - ▣ 식별자의 첫 번째 문자로 숫자는 사용불가
 - ▣ '' 또는 '\$'를 식별자 첫 번째 문자로 사용할 수 있으나 일반적으로 잘 사용하지 않는다.
 - ▣ 불린 리터럴 (true, false)과 널 리터럴(null)은 식별자로 사용불가
 - ▣ 길이 제한 없음
- 대소문자 구별
 - ▣ Test와 test는 별개의 식별자

식별자 이름 사례

7

□ 사용 가능한 예

```
int name;
char student_ID;           // '_' 사용 가능
void $func() { }           // '$' 사용 가능
class Monster3 { }         // 숫자 사용 가능
int whatsyournamemynameiskitae; // 길이 제한 없음
int barChart; int barchart; // 대소문자 구분. barChart와 barchart는 다름
int 가격;                  // 한글 이름 사용 가능
```

□ 잘못된 예

```
int 3Chapter;              // 숫자로 사용하였기 때문
class if { }               // if는 자바의 예약어임
char false;                // false는 사용 불가
void null() { }            // null 사용 불가
class %calc { }            // '%'는 특수문자
```

자바 키워드

8

abstract	continue	for	new	switch
assert	default	if	package	synchronized
boolean	do	goto	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

식별자 이름 붙이는 관습

9

- 기본 : 헝그리안 이름 붙이기 관습
- 클래스 이름

```
public class HelloWorld {}  
class Vehicle {}  
class AutoVendingMachine {}
```

- 첫 번째 문자는 대문자로 시작
- 여러 단어가 복합되어 있을 때는 각 단어의 첫 번째 문자만 대문자로 표시
- 변수, 메소드 이름

```
int iAge;           // iAge의 i는 int의 i를 표시  
boolean bIsSingle; // bIsSingle의 처음 b는 boolean의 b를 표시  
String strName;     // strName의 str은 String의 str을 표시  
public int iGetAge() {} // iGetAge의 i는 int의 i를 표시
```

- 첫 단어 이후 각 단어의 첫 번째 문자는 대문자로 시작
- 상수 이름

```
final static double PI = 3.141592;
```

- 모든 문자를 대문자로 표시

자바의 데이터 타입

10

□ 자바의 데이터 타입

▣ 기본 타입 : 8 개

- boolean
- char
- byte
- short
- int
- long
- float
- double

▣ 레퍼런스 타입 : 3 개

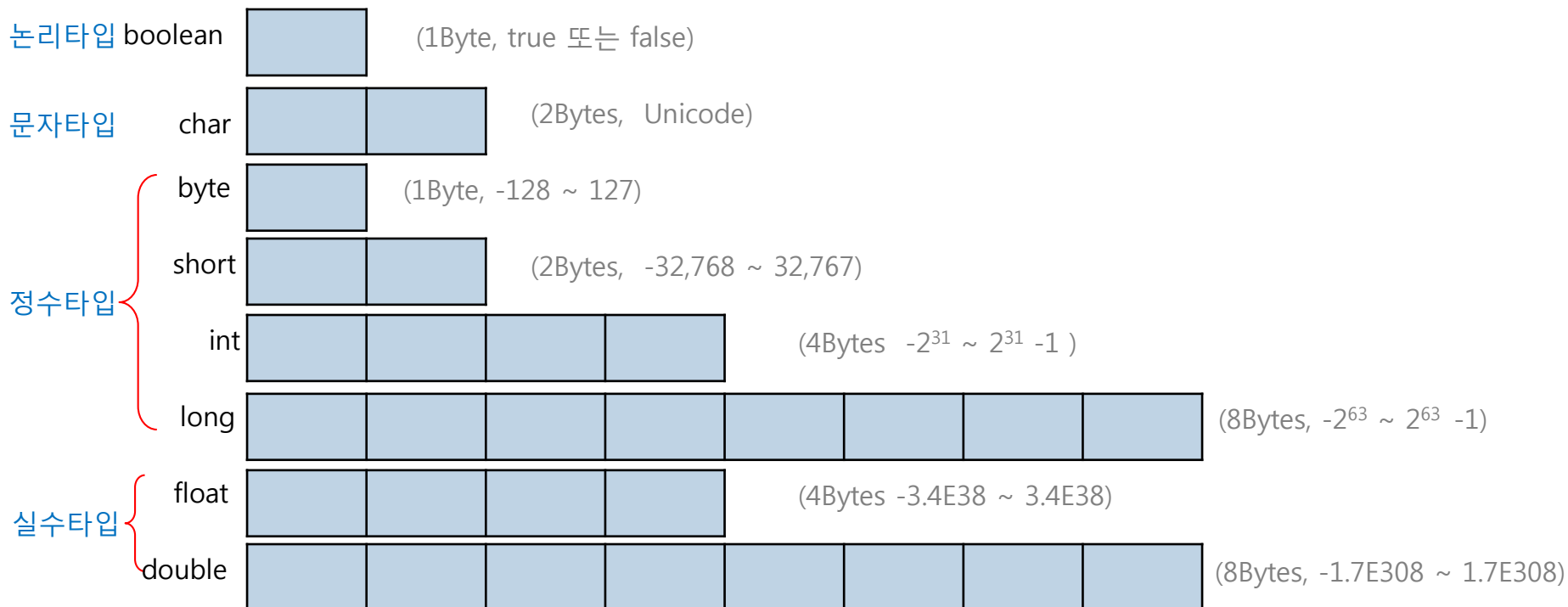
- 클래스(class)에 대한 레퍼런스
- 인터페이스(interface)에 대한 레퍼런스
- 배열(array)에 대한 레퍼런스

자바의 기본 데이터 타입

11

□ 특징

- 기본 데이터 타입의 크기가 정해져 있음
- 기본 데이터 타입의 크기는 CPU나 운영체제에 따라 변하지 않음



변수와 선언

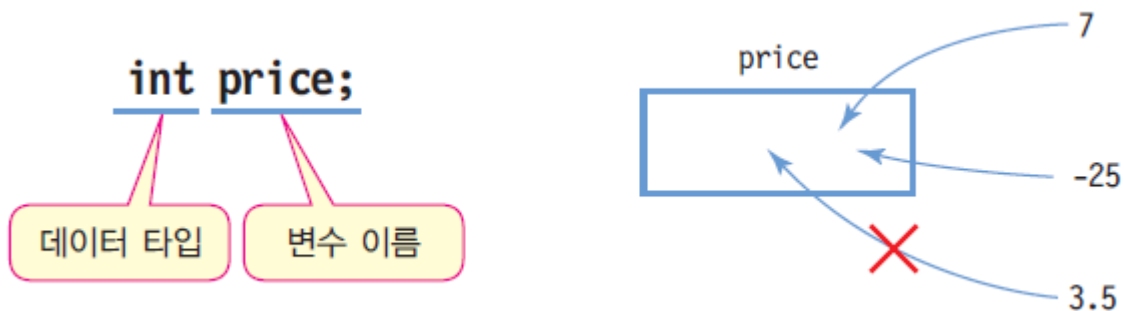
12

□ 변수

- ▣ 프로그램 실행 중에 값을 임시 저장하기 위한 공간
 - 변수 값은 프로그램 수행 중 변경될 수 있음
- ▣ 데이터 타입에서 정한 크기의 메모리 할당
- ▣ 반드시 변수 선언과 값을 초기화 후 사용

□ 변수 선언

- ▣ 변수의 타입 다음에 변수 이름을 적어 변수를 선언



변수 선언 사례

13

□ 변수 선언 사례

```
int radius;  
char c1, c2, c3; // 3 개의 변수를 한 번에 선언한다.  
double weight;
```

□ 변수 선언과 초기화

▣ 선언과 동시에 초기값 지정

```
int radius = 10;  
char c1 = 'a', c2 = 'b', c3 = 'c';  
double weight = 75.56;
```

□ 변수에 값 대입

▣ 대입 연산자인 = 다음에 식(expression)

```
radius = 10 * 5;  
c1 = 'r';  
weight = weight + 5.0;
```

정수 타입 리터럴

14

- 정수 타입 리터럴 : 정수 직접 표시
 - ▣ 8진수 : 0으로 시작하는 숫자는 모두 8진수
 - `int n = 015;` // 10진수로 13
 - ▣ 16진수 : 0x로 시작하는 숫자는 16진수
 - `int n = 0x15;` // 10진수로 21
 - ▣ 10진수 : 0으로 시작하지 않는 숫자는 10진수
 - 15, 3, 20, 55, 88
 - ▣ 모든 정수타입 리터럴은 int형으로 컴파일함
 - ▣ long 타입 리터럴은 숫자 뒤에 L 또는 l을 붙임
 - ex) 24L, 3578l

실수 타입 리터럴

15

- 부동 소수점을 갖는 수 직접 표시
 - ▣ 소수점을 찍은 실수, 지수(exponent)식으로 표현한 실수
 - 12, 또는 12.0
 - .1234 또는 0.1234 또는 1234E-4
 - ▣ 숫자 뒤에 f(float)나 d(double)을 명시적으로 붙이기도 함
 - 0.1234 또는 0.1234D 또는 0.1234d → double 타입
 - 0.1234f 또는 0.1234F → float 타입
 - 1234D 또는 1234d → 1234.0과 같으며 double 타입
 - 1234F 또는 1234f → 1234.0과 같으며 float 타입
 - ▣ 실수 타입 리터럴은 double 타입으로 컴파일됨

문자 타입 리터럴

16

- 문자 한 자를 나타낸다.
 - ▣ 단일 인용부호(")로 문자 하나 표현
 - 'a', 'W', '가', '*', '3', '7'
 - ▣ ₩다음에 숫자는 8진수로서 0 ~ 337사이의 8진수만 가능
 - \102 -> 문자 'B' 를 나타내는 8진수
 - \337 -> 문자 ' β ' 를 나타내는 8진수
 - ▣ ₩u다음에 4자리 16진수, 2 바이트의 유니코드(Unicode)
 - \u0041 -> 문자 'A'의 유니코드(0041)
 - \uae00 -> 한글문자 '글'의 유니코드(ae00)
 - ▣ 특수 기호는 ₩로 시작

'\b'	백스페이스	'\t'	탭
'\n'	라인피드	'\f'	폼피드
'\r'	캐리지리턴	'\"'	이중 인용 부호
'\''	단일 인용 부호	'\\'	백슬래시

논리 타입 리터럴

17

- 논리 값 표시
 - ▣ true 또는 false 뿐
- 논리 타입과 정수타입 사이의 타입 변환 허용 안 됨

```
int i;  
if ((boolean)i){ }    // 정수 i를 논리 타입으로 변환할 수 없음. 컴파일 에러
```

- ▣ (i==1) 또는 (i!=0)과 같은 논리 연산을 사용해야 함

Tip: 기본 데이터 타입 이외 리터럴

18

□ null 리터럴

- 어떠한 레퍼런스 타입의 값으로도 사용 가능

- ~~int n = null;~~ // 기본 데이터 타입에는 사용 불가
- String str = null;

□ 문자열 리터럴

- 이중 인용부호로 묶어서 표현

- "Good", "Morning", "자바", "3.19", "26", "a"

- 자바에서 문자열은 객체이므로 기본 타입이 아님

- 문자열 리터럴은 String 객체로 생성됨

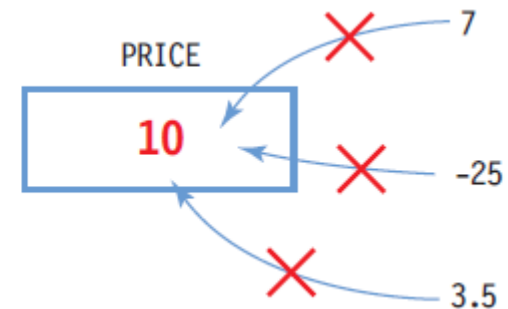
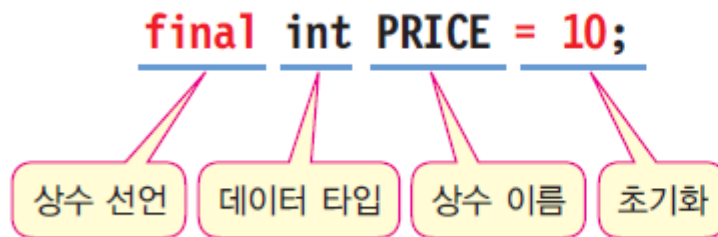
```
String str1 = "Welcome";  
String str2 = null;  
System.out.println(str1);
```

상수

19

□ 상수 선언

- ▣ final 키워드 사용
- ▣ 값 변경 불가
- ▣ 선언 시 초기값 지정



□ 상수 선언 사례

```
final double PI = 3.141592;  
final int LENGTH = 20;
```

예제 2-1 : 변수, 리터럴, 상수 사용하기

20

원의 면적을 구하는 프로그램을 작성해보자.

```
public class CircleArea {  
    public static void main(String[] args) {  
        final double PI = 3.14; // 원주율을 상수로 선언  
        double radius = 10; // 원의 반지름  
        double circleArea = 0; // 원의 면적  
  
        circleArea = radius*radius*PI; // 원의 면적 계산  
  
        // 원의 면적을 화면에 출력한다.  
        System.out.print("원의 면적 = ");  
        System.out.println(circleArea);  
    }  
}
```

원의 면적 = 314.0

타입 변환 - 자동 타입 변환

21

- 자동타입 변환이 발생하는 경우
 - ▣ 원래의 타입보다 큰 자료타입으로 바뀔 때

byte >> short/char >> int >> long >> float >> double

- ▣ 원본 데이터 그대로 보존
- 자동 타입 변환 사례

```
byte a;  
int price;  
price = a;
```

정수타입 변수 자동타입변환 바이트타입 변수

```
long var;  
int n = 32555;  
byte b = 25;  
var = n; // int 타입에서 long 타입으로 자동 변환. var 값은 32555  
var = b; // byte 타입에서 long 타입으로 자동 변환. var 값은 25
```

강제 타입 변환

22

- 강제 타입 변환 : 개발자의 의도적으로 타입 변환
- 강제 타입 변환 방법

```
byte a;  
int price;  
a = (byte) price;
```

바이트타입 변수

price 정수 값을 byte
타입으로 강제타입 변환

정수타입 변수

□ 강제 타입 변환 사례

- 실수 타입이 정수 타입으로 강제 변환 시 소수점 아래가 버려짐
 - 데이터 손실

```
short var;  
int n = 855638017; // n의 16진수 값은 0x33000001  
var = (short) n; // int 타입에서 short 타입으로 강제 변환. var 값은 1  
  
double d = 1.9;  
int n = (int)d; // n은 1이 된다.
```

예제 2-2 : 자동 타입 변환, 강제 타입 변환

23

자동 타입 변환과 강제 타입 변환의 이해를 위한 예제이다.
다음 소스의 실행 결과는 무엇인가?

```
public class TypeConversion {  
    public static void main(String[] args) {  
        byte b = 127;  
        int i = 100;  
        System.out.println(b+i);  
        System.out.println(10/4);  
        System.out.println(10.0/4);  
        System.out.println((char)0x12340041);  
        System.out.println((byte)(b+i));  
        System.out.println((int)2.9 + 1.8);  
        System.out.println((int)(2.9 + 1.8));  
        System.out.println((int)2.9 + (int)1.8);  
    }  
}
```

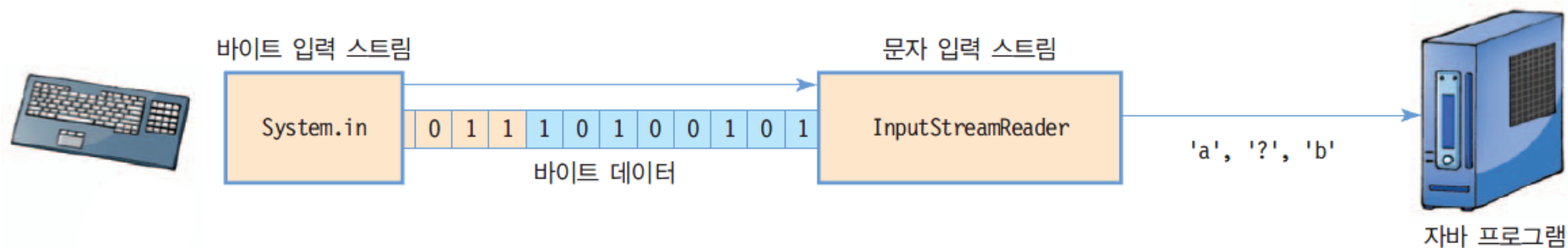
227
2
2.5
A
-29
3.8
4
3

자바에서 키 입력, System.in

24

□ System.in

- 자바의 표준 입력 스트림
- java.io 패키지의 InputStream 클래스
 - System.in은 바이트 스트림으로서 키 값을 바이트로 리턴
 - 문자로 변환하려면 InputStreamReader 클래스를 이용



- 입력 동안 문제가 발생하면 IOException 발생
 - try-catch를 이용한 예외 처리 필요(3장 참조)

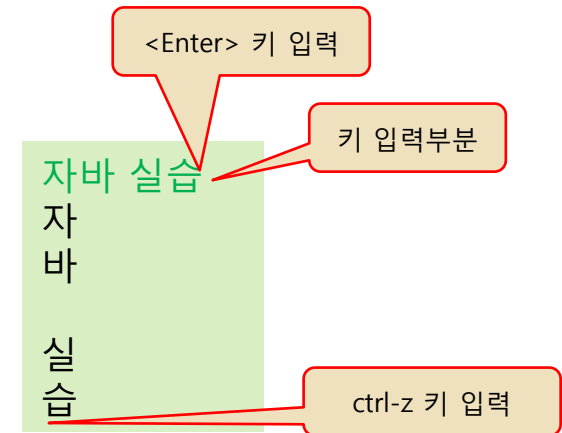
예제 2-3 : 키보드로부터 문자 입력, 화면 출력

25

다음 소스의 실행 결과는 무엇인가?

System.in을 InputStreamReader에 연결하여 사용자로부터 키 입력.
입력 받은 문자를 화면에 출력하고, ctrl-z 키를 누르면 읽기 종료

```
import java.io.*;
public class InputExample {
    public static void main (String args[]) {
        InputStreamReader rd = new InputStreamReader(System.in);
        try {
            while (true) {
                int a = rd.read();
                if (a == -1) // ctrl-z가 입력되면 read()는 -1을 리턴
                    break;
                System.out.println((char)a);// 입력된 문자 출력
            }
        }
        catch (IOException e) {
            System.out.println("입력 에러 발생");
        }
    }
}
```



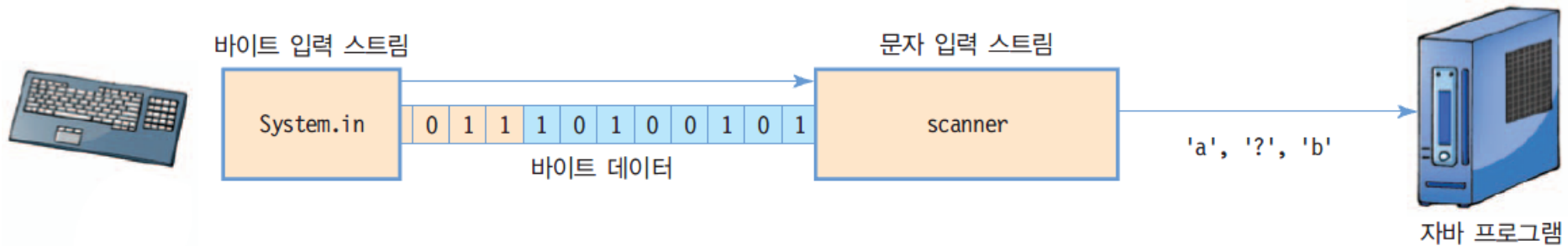
Scanner를 이용한 키 입력

26

□ Scanner 클래스

- java.util.Scanner 클래스
- Scanner 객체 생성

```
Scanner a = new Scanner(System.in);
```



□ import문 필요

- 소스 맨 앞줄에 사용
- ```
import java.util.Scanner;
```

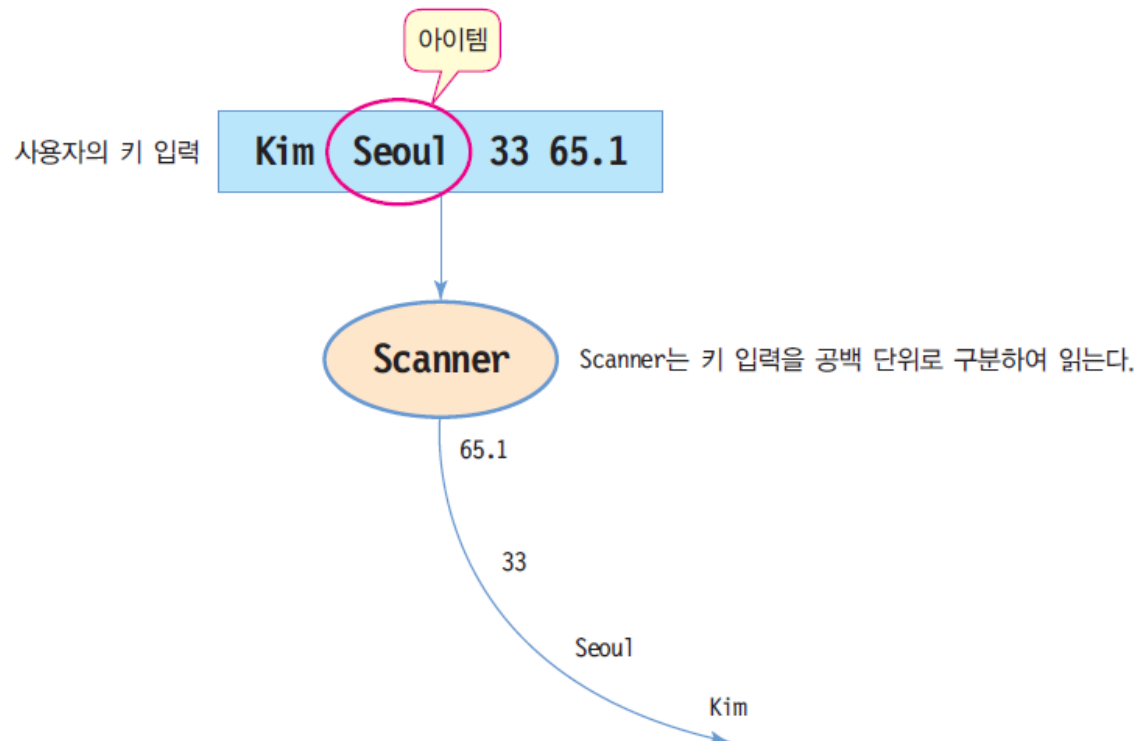
## □ Scanner에서 키 입력 받기

- Scanner는 입력되는 키 값을 공백으로 구분되는 아이템 단위로 읽음
- 공백 문자: '\t', '\f', '\r', ' ', '\n'

# Scanner를 이용한 키 입력

27

```
Scanner scanner = new Scanner(System.in);
String name = scanner.next(); // "Kim"
String addr = scanner.next(); // "Seoul"
int age = scanner.nextInt(); // 33
double weight = scanner.nextDouble(); // 65.1
```



# Scanner 주요 메소드

28

| 생성자/메소드               | 설명                             |
|-----------------------|--------------------------------|
| String next()         | 다음 아이템을 찾아 문자열로 반환             |
| boolean nextBoolean() | 다음 아이템을 찾아 boolean으로 변환하여 반환   |
| byte nextByte()       | 다음 아이템을 찾아 byte로 변환하여 반환       |
| double nextDouble()   | 다음 아이템을 찾아 double로 변환하여 반환     |
| float nextFloat()     | 다음 아이템을 찾아 float로 변환하여 반환      |
| int nextInt()         | 다음 아이템을 찾아 int로 변환하여 반환        |
| long nextLong()       | 다음 아이템을 찾아 long으로 변환하여 반환      |
| short nextShort()     | 다음 아이템을 찾아 short로 변환하여 반환      |
| String nextLine()     | 한 라인 전체('\\n' 포함)를 문자열 타입으로 반환 |

## 예제 2-4 : Scanner를 이용한 키 입력 연습

29

Scanner를 이용하여 나이, 체중, 신장 데이터를 키보드에서 입력 받아 다시 출력하는 프로그램을 작성해보자.

```
import java.util.Scanner;

public class ScannerExam {
 public static void main (String args[]) {
 Scanner a = new Scanner(System.in);
 System.out.println("나이, 체중, 신장을 빈칸으로 분리하여 순서대로 입력하세요");
 System.out.println("당신의 나이는 " + a.nextInt() + "살입니다.");
 System.out.println("당신의 체중은 " + a.nextDouble() + "kg입니다.");
 System.out.println("당신의 신장은 " + a.nextDouble()+ "cm입니다.");
 }
}
```

나이, 체중, 신장을 빈칸으로 분리하여 순서대로 입력하세요

35 75 175

당신의 나이는 35살입니다.

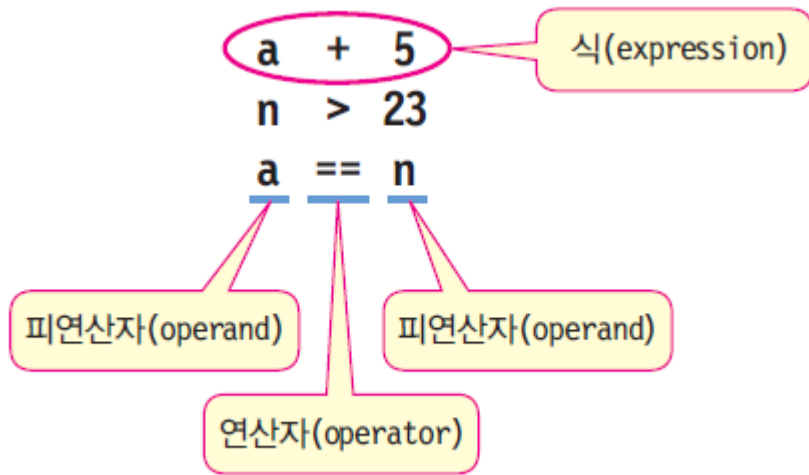
당신의 체중은 75.0kg입니다.

당신의 신장은 175.0cm입니다.

# 식과 연산자

30

- 연산 : 주어진 식을 계산하여 결과를 얻어내는 과정



| 연산자의 종류 | 연산자                                    |
|---------|----------------------------------------|
| 증감      | ++ --                                  |
| 산술      | + - * / %                              |
| 시프트     | >> << >>>                              |
| 비교      | > < >= <= == !=                        |
| 비트      | &   ^ ~                                |
| 논리      | &&    ! ^                              |
| 조건      | ? :                                    |
| 대입      | = *= /= += -= &= ^=<br> = <<= >>= >>>= |

# 연산자 우선 순위

31

|                                                       |                                                 |
|-------------------------------------------------------|-------------------------------------------------|
| <div> <div>높음</div> <div>↓</div> <div>낮음</div> </div> | ++(postfix) -- (postfix)                        |
|                                                       | +(양수 부호) -(양수, 음수 부호) ++(prefix) --(prefix) ~ ! |
|                                                       | 형 변환(type casting)                              |
|                                                       | * / %                                           |
|                                                       | +(덧셈) -(뺄셈)                                     |
|                                                       | << >> >>>                                       |
|                                                       | < > <= >= instanceof                            |
|                                                       | == !=                                           |
|                                                       | &(비트 AND)                                       |
|                                                       | ^(비트 XOR)                                       |
|                                                       | (비트 OR)                                         |
|                                                       | &&(논리 AND)                                      |
|                                                       | (논리 OR)                                         |
|                                                       | ? : (조건)                                        |
|                                                       | = += -= *= /= %= &= ^=  = <<= >>= >>>=          |

- 같은 우선순위의 연산자
  - ▣ 왼쪽에서 오른쪽으로 처리
  - ▣ 예외)오른쪽에서 왼쪽으로
    - 대입 연산자, --, ++, +,-(양수 음수 부호), !, 형 변환은 오른쪽에서 왼쪽으로 처리
- 괄호는 최우선순위
  - ▣ 괄호가 다시 괄호를 포함한 경우는 가장 안쪽의 괄호부터 먼저 처리

# 산술 연산자

32

## □ /와 % 연산자

- ▣ 정수 연산시 /은 정수 몫. %는 정수 나머지
- ▣ %의 이용 사례 : 홀수 짝수 판별
  - `int r = x % 2; r이 1이면 x는 홀수`

| 산술 연산자 | 의미  | 예          | 결과값  |
|--------|-----|------------|------|
| +      | 더하기 | 25.5 + 3.6 | 29.1 |
| -      | 빼기  | 3 - 5      | -2   |
| *      | 곱하기 | 2.5 * 4    | 10.0 |
| /      | 나누기 | 5/2        | 2    |
| %      | 나머지 | 5%2        | 1    |



## 예제 2-5 : 산술 연산 예제

33

정수를 입력 받고 입력 받은 정수의 초를 몇 시간, 몇 분, 몇 초인가를 구하는 프로그램을 작성하시오.

```
import java.util.Scanner;

public class ArithmeticOperator {
 public static void main (String[] args) {
 int time;
 int second;
 int minute;
 int hour;
 Scanner sc = new Scanner(System.in);
 System.out.print("정수를 입력하세요:"); // 시,분,초로 변환될 정수 입력

 time = sc.nextInt();
 second = time % 60; // 60으로 나눈 나머지는 초를 의미
 minute = (time / 60) % 60; // 60으로 나눈 몫을 다시 60으로 나눈 나머지는 분을 의미
 hour = (time / 60) / 60; // 60으로 나눈 몫을 다시 60으로 나눈 몫은 시간을 의미

 System.out.print(time + "초는 ");
 System.out.print(hour + "시간, ");
 System.out.print(minute + "분, ");
 System.out.println(second + "초입니다.");
 }
}
```

정수를 입력하세요:500  
500초는 0시간, 8분, 20초입니다.

# 비트 연산자

34

## □ 피 연산자의 각 비트들을 대상으로 하는 연산

| 비트 연산자       | 내용                                              |
|--------------|-------------------------------------------------|
| $a \& b$     | a와 b의 각 비트들의 AND 연산. 두 비트 모두 1일 때만 1이 되며 나머지는 0 |
| $a   b$      | a와 b의 각 비트들의 OR 연산. 두 비트 모두 0일 때만 0이 되며 나머지는 1  |
| $a \wedge b$ | a와 b의 각 비트들의 XOR 연산. 두 비트가 서로 다르면 1, 같으면 0      |
| $\sim a$     | 단항 연산자로서, a의 각 비트들에 NOT 연산. 1을 0으로, 0을 1로 변환    |

# 비트 연산자의 사례

35

$$\begin{array}{r} 01101010 \\ \& 11001101 \\ \hline 01001000 \end{array}$$

모두 1이므로  
결과는 1

둘 중 하나라도  
0이 되면 결과는 0

$$\begin{array}{r} 01101010 \\ | 11001101 \\ \hline 11101111 \end{array}$$

모두 0이므로  
결과는 0

둘 중 하나라도  
1이 되면 결과는 1

$$\begin{array}{r} 01101010 \\ \wedge 11001101 \\ \hline 10100111 \end{array}$$

두 비트가 모두  
같으므로 결과는 0

두 비트가 서로  
다르므로 결과는 1

$$\begin{array}{r} \sim 01101010 \\ \hline 10010101 \end{array}$$

0은 1로 변환

1은 0으로 변환

# 시프트 연산자

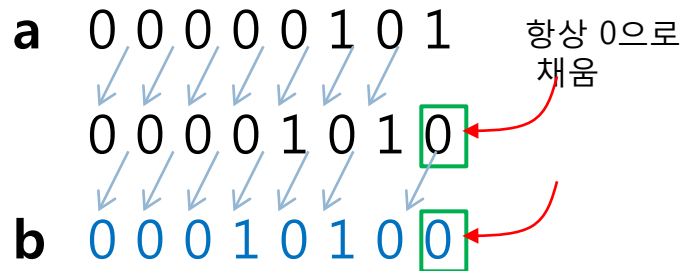
36

| 시프트 연산자    | 내용                                                                               |
|------------|----------------------------------------------------------------------------------|
| $a \gg b$  | a의 각 비트를 오른쪽으로 b 번 시프트한다.<br>최상위 비트의 빈자리는 시프트 전의 최상위 비트로 다시 채운다.<br>산술적 오른쪽 시프트. |
| $a \ggg b$ | a의 각 비트를 오른쪽으로 b 번 시프트한다.<br>그리고 최상위 비트의 빈자리는 0으로 채운다. 논리적 오른쪽 시프트.              |
| $a \ll b$  | a의 각 비트를 왼쪽으로 b 번 시프트한다.<br>그리고 최하위 비트의 빈자리는 0으로 채운다. 산술적 왼쪽 시프트.                |

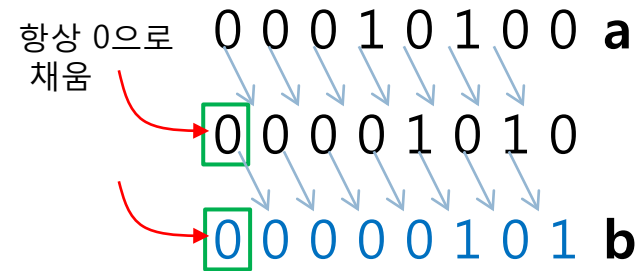
# 시프트 연산자의 사례

37

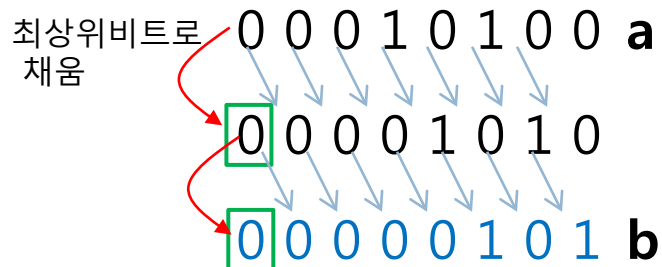
```
byte a = 5; // 5
byte b = (byte)(a << 2); // 20
```



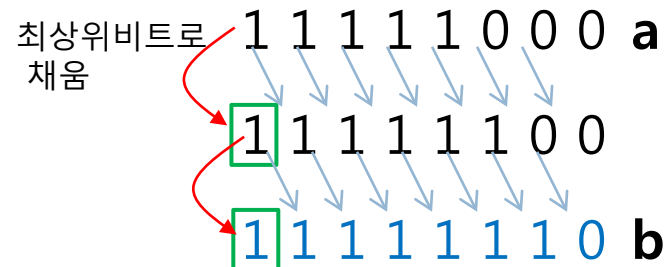
```
byte a = 20; // 20
byte b = (byte)(a >>> 2); // 5
```



```
byte a = 20; // 20
byte b = (byte)(a >> 2); // 5
```



```
byte a = (byte)0xf8; // -8
byte b = (byte)(a >> 2); // -2
```



# Tip: 산술적 시프트와 논리적 시프트

38

- 산술적 오른쪽 시프트
  - ▣ >>는 1비트 오른쪽으로 시프트할 때마다 2로 나누기하는 결과
- 산술적 왼쪽 시프트
  - ▣ << 연산자는 1비트 시프트할 때마다 2로 곱하는 결과
  - ▣ 음수(최상위 비트가 1)는 시프트 결과 최상위 비트가 0인 양수가 되는 오버플로 발생 가능 주의
- 논리적 오른쪽 시프트
  - ▣ >>>는 시프트 시 최상위 비트에 항상 0이 삽입
  - ▣ 나누기의 산술적 효과가 나타나지 않음
- byte, short, char 타입의 시프트 연산 시 주의 사항
  - ▣ int 타입으로 변환되어 연산이 일어나므로 원하지 않는 결과 발생 가능

# 예제 2-6 : 비트 연산자와 시프트 연산자 사용 예

39

다음 소스의 실행 결과는 무엇인가?

```
public class BitShiftOperator {
 public static void main (String[] args) {
 short a = (short)0x55ff;
 short b = 0x00ff;

 // 비트 연산
 System.out.printf("%xWn", a & b);
 System.out.printf("%xWn", a | b);
 System.out.printf("%xWn", a ^ b);
 System.out.printf("%xWn", ~a);
 byte c = 20; // 0x14
 byte d = -8; // 0xf8

 // 시프트 연산
 System.out.println(c << 2); // c를 2비트 왼쪽 시프트
 System.out.println(c >> 2); // c를 2비트 오른쪽 시프트. 0 삽입
 System.out.println(d >> 2); // d를 2비트 오른쪽 시프트. 1 삽입
 System.out.printf("%xWn", d >>> 2); // d를 2비트 오른쪽 시프트. 0 삽입
 }
}
```

printf("%xWn", ...)는 결과 값을 16진수 형식으로 출력

ff  
55ff  
5500  
ffffaa00  
80  
5  
-2  
3ffffffe

# 비교연산자

40

| 비교 연산자     | 내용                            | 예제           | 결과    |
|------------|-------------------------------|--------------|-------|
| $a < b$    | a가 b보다 작으면 true 아니면 false     | $3 < 5$      | true  |
| $a > b$    | a가 b보다 크면 true 아니면 false      | $3 > 5$      | false |
| $a \leq b$ | a가 b보다 작거나 같으면 true 아니면 false | $1 \leq 0$   | false |
| $a \geq b$ | a가 b보다 크거나 같으면 true 아니면 false | $10 \geq 10$ | true  |
| $a == b$   | a가 b와 같으면 true 아니면 false      | $1 == 3$     | false |
| $a != b$   | a가 b와 같지 않으면 true 아니면 false   | $1 != 3$     | true  |



# 논리 연산자

| a     |  | !a    | 예제              |
|-------|--|-------|-----------------|
| true  |  | false | !(3 < 5)는 false |
| false |  | true  | !(3 > 5)는 true  |

| a     | b     | a ^ b | 예제                    |
|-------|-------|-------|-----------------------|
| true  | true  | false | (3<5) ^ (1==1)은 false |
| true  | false | true  | (3<5) ^ (1==2)은 true  |
| false | true  | true  | (3>5) ^ (1==1)은 true  |
| false | false | false | (3>5) ^ (1==2)은 false |

| a     | b     | a    b | 예제                     |
|-------|-------|--------|------------------------|
| true  | true  | true   | (3<5)    (1==1)은 true  |
| true  | false | true   | (3<5)    (1==2)은 true  |
| false | true  | true   | (3>5)    (1==1)은 true  |
| false | false | false  | (3>5)    (1==2)은 false |

| a     | b     | a && b | 예제                     |
|-------|-------|--------|------------------------|
| true  | true  | true   | (3<5) && (1==1)은 true  |
| true  | false | false  | (3<5) && (1==2)은 false |
| false | true  | false  | (3>5) && (1==1)은 false |
| false | false | false  | (3>5) && (1==2)은 false |

## 예제 2-7 : 비교 연산자와 논리 연산자 사용하기

42

다음 소스의 실행 결과는 무엇인가?

```
public class LogicalOperator {
 public static void main (String[] args) {
 System.out.println('a' > 'b');
 System.out.println(3 >= 2);
 System.out.println(-1 < 0);
 System.out.println(3.45 <= 2);
 System.out.println(3 == 2);
 System.out.println(3 != 2);
 System.out.println(!(3 != 2));
 System.out.println((3 > 2) && (3 > 4));
 System.out.println((3 != 2) || (-1 > 0));
 System.out.println((3 != 2) ^ (-1 > 0));
 }
}
```

false  
true  
true  
false  
false  
false  
true  
false  
false  
true  
true

# 대입 연산자, 증감 연산자

43

| 대입 연산자                         | 내용                                     |
|--------------------------------|----------------------------------------|
| <code>a = b</code>             | b의 값을 a에 대입                            |
| <code>a += b</code>            | <code>a = a + b</code> 과 동일            |
| <code>a -= b</code>            | <code>a = a - b</code> 과 동일            |
| <code>a *= b</code>            | <code>a = a * b</code> 과 동일            |
| <code>a /= b</code>            | <code>a = a / b</code> 과 동일            |
| <code>a %= b</code>            | <code>a = a % b</code> 과 동일            |
| <code>a &amp;= b</code>        | <code>a = a &amp; b</code> 과 동일        |
| <code>a ^= b</code>            | <code>a = a ^ b</code> 과 동일            |
| <code>a  = b</code>            | <code>a = a   b</code> 과 동일            |
| <code>a &lt;&lt;= b</code>     | <code>a = a &lt;&lt; b</code> 과 동일     |
| <code>a &gt;&gt;= b</code>     | <code>a = a &gt;&gt; b</code> 과 동일     |
| <code>a &gt;&gt;&gt;= b</code> | <code>a = a &gt;&gt;&gt; b</code> 과 동일 |

| 증감 연산자           | 내용                |
|------------------|-------------------|
| <code>a++</code> | a을 먼저 사용한 후에 1 증가 |
| <code>a--</code> | a을 먼저 사용한 후에 1 감소 |
| <code>++a</code> | a을 먼저 1 증가한 후에 사용 |
| <code>--a</code> | a을 먼저 1 감소한 후에 사용 |

# 증감 연산자

44

## □ 증감 연산의 순서

- ▣ 연산자가 피연산자 뒤에 붙는 경우

```
int a, b = 4;
a = b++;
// 결과 a=4, b=5
```

- ▣ 연산자가 피연산자 앞에 붙는 경우

```
int a, b = 4;
a = ++b;
// 결과 a=5, b=5
```

## 예제 2-8 : 대입 연산자와 증감 연산자 사용하기

45

다음 소스의 실행 결과는 무엇인가?

```
public class UnaryOperator {
 public static void main(String[] args) {
 int opr = 0;
 opr += 3; // opr = opr + 3
 System.out.println(opr++); // opr 출력 후 증가
 System.out.println(opr);
 System.out.println(++opr); // opr 증가 후 출력
 System.out.println(opr);
 System.out.println(opr--); // opr 출력 후 감소
 System.out.println(opr);
 System.out.println(--opr); // opr 감소 후 출력
 System.out.println(opr);
 }
}
```

3  
4  
5  
5  
5  
4  
3  
3

# 조건 연산자 ?:


46

## □ opr1?opr2:opr3

- 세 개의 피연산자로 구성된 삼항(ternary) 연산자
  - opr1° | true°이면, 연산식의 결과는 opr2, false°이면 opr3
- if-else를 간결하게 표현할 수 있음

```
int x = 5;
int y = 3;

int s;
if(x>y)
 s = 1;
else
 s = -1;
```



```
int s = (x>y)?1:-1;
```

## 예제 2-9 : 조건 연산자 사용하기

47

다음 소스의 실행 결과는 무엇인가?

```
public class TernaryOperator {
 public static void main (String[] args) {
 int a = 3, b = 5;

 System.out.println("두 수의 차는 " + ((a>b)?(a-b):(b-a)));
 }
}
```

두 수의 차는 2

# 조건문 - if문

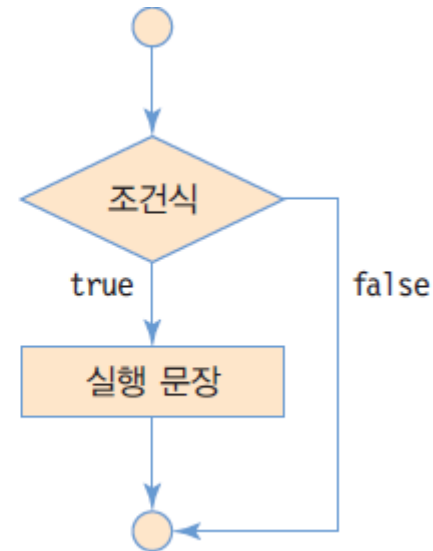
48

## □ 단순 if 문

- ▣ if 다음의 괄호 안에는 조건식(논리형 변수나 논리 연산)
- ▣ 조건식의 값
  - true인 경우, if문을 벗어나 다음 문장이 실행된다.
  - false인 경우에는 if 다음의 문장이 실행되지 않고 if 문을 빠져 나온다.
- ▣ 실행문장이 단일 문장인 경우 둘러싸는 { } 생략 가능

```
if(조건식) {
 ...실행 문장...
}
```

if 키워드





## 예제 2-10 : if문 사용하기

49

시험 점수가 80점이 이상이면 합격 판별을 하는 프로그램을 작성하시오.

```
import java.util.Scanner;

public class SuccessOrFail {
 public static void main (String[] args) {
 Scanner in = new Scanner(System.in);

 System.out.print("점수를 입력하시오: ");
 int score = in.nextInt();
 if (score >= 80)
 System.out.println("축하합니다! 합격입니다.");
 }
}
```

점수를 입력하시오: 95  
축하합니다! 합격입니다.

# 조건문 – if-else

50

## □ if-else 문

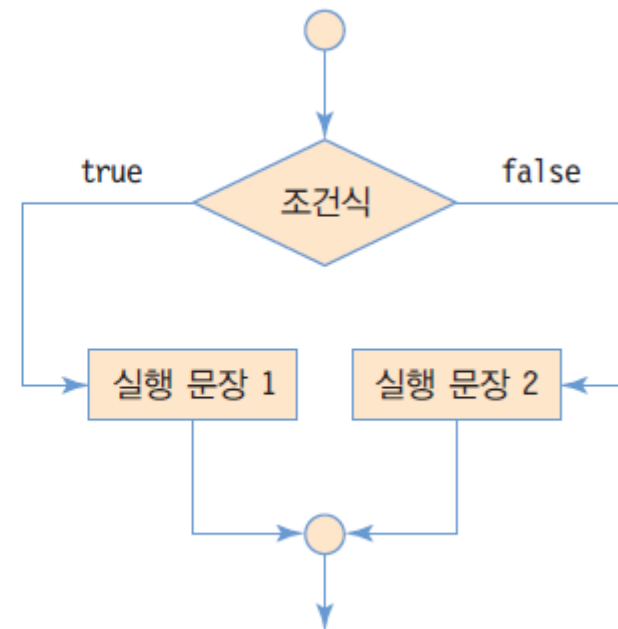
- 조건식이 true면 실행문장1 실행 후 if-else문을 벗어남
- false인 경우에 실행문장2 실행후, if-else문을 벗어남

2/

```
if(조건식) {
 ...실행 문장 1...
}
else {
 ...실행 문장 2...
}
```

if 키워드

else 키워드



## 예제 2-11 : if-else 사용하기

51

입력된 수가 3의 배수인지 판별하는 프로그램을 작성하시오.

```
import java.util.Scanner;

public class MultipleOfThree {
 public static void main (String[] args) {
 Scanner in = new Scanner(System.in);

 System.out.print("수를 입력하시오: ");
 int number = in.nextInt();

 if (number % 3 == 0)
 System.out.println("3의 배수입니다.");
 else
 System.out.println("3의 배수가 아닙니다.");
 }
}
```

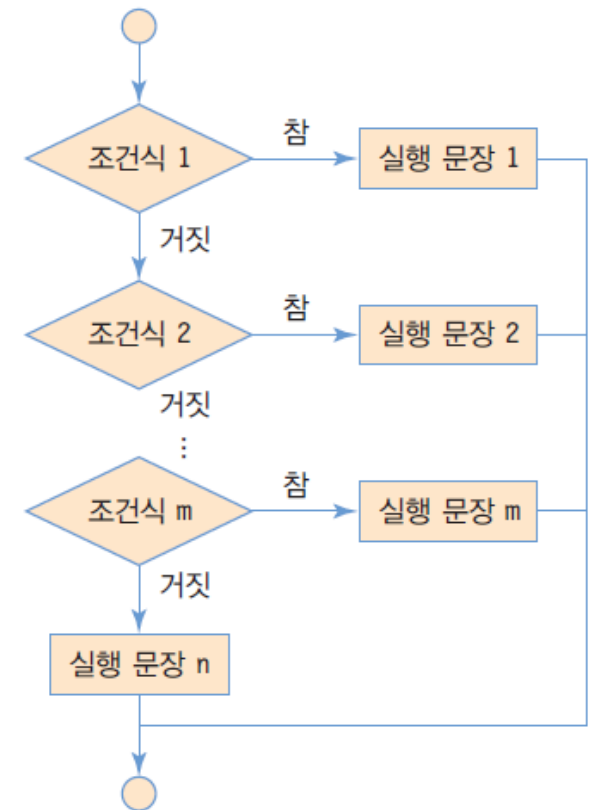
수를 입력하시오: 129  
3의 배수입니다.

# 조건문 - 다중 if

52

## □ 다중 if문

- 실행문장이 다시 if문 또는 if-else문을 포함
- else 문은 바로 전의 if문과 짝을 이룬다.
- 조건문이 너무 많은 경우, switch 문 사용 권장



## 예제 2-12 : 학점 매기기

53

if-else문을 이용하여 키보드 입력된 성적에 대해  
학점을 부여하는 프로그램을 작성해보자.

```
import java.util.Scanner;

public class Grading {
 public static void main (String[] args) {
 char grade;
 Scanner a = new Scanner(System.in);
 while (a.hasNext()) {
 int score = a.nextInt();
 if(score >= 90.0) // score가 90.0 이상인 경우
 grade = 'A';
 else if(score >= 80.0) // score가 80.0 이상이면서 90.0 미만인 경우
 grade = 'B';
 else if(score >= 70.0) // score가 70.0 이상이면서 80.0 미만인 경우
 grade = 'C';
 else if(score >= 60.0) // score가 60.0 이상이면서 70.0 미만인 경우
 grade = 'D';
 else // score가 60.0 미만인 경우
 grade = 'F';
 System.out.println("학점은 "+grade+"입니다");
 }
 }
}
```

키가 입력될 때까지 기다리며, 입력된 키가 있는 경우 true 리턴. ctrl-z 키가 입력되면 false 리턴

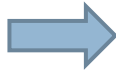
80  
학점은 B입니다  
90  
학점은 A입니다  
76  
학점은 C입니다

# Tip: if문과 조건 연산자 ?:

54

- 조건 연산자 ?:는 if-else로 바꿀 수 있음

```
i = a>b?a-b:b-a;
```

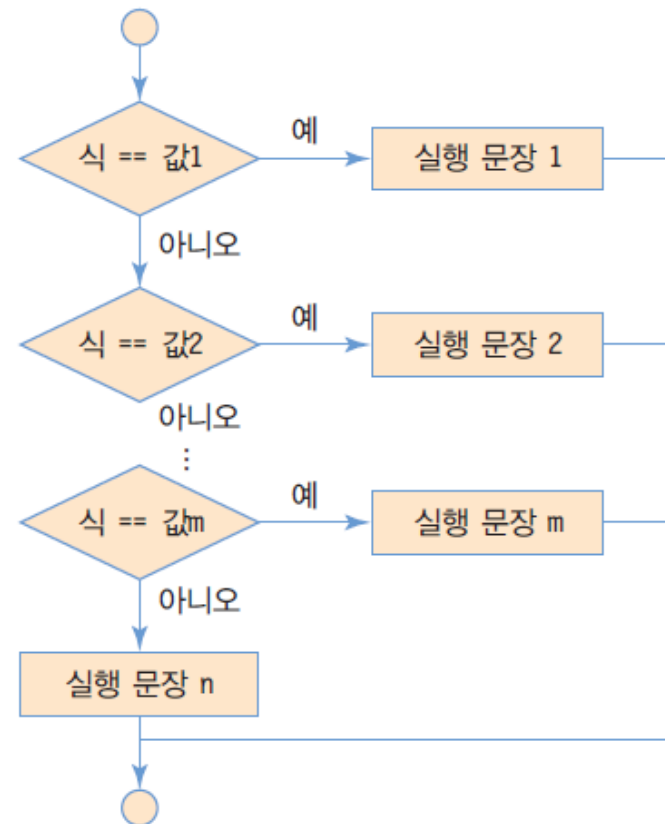
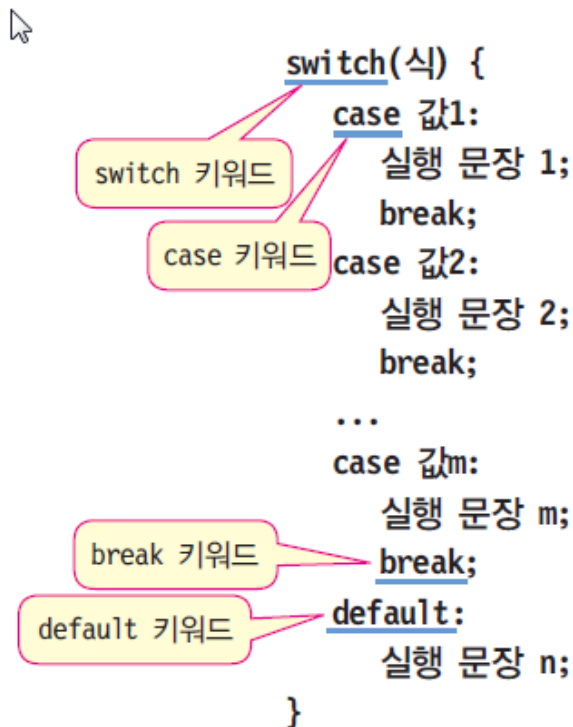


```
if (a>b)
 i = a - b;
else
 i = b - a;
```

# switch문

55

- switch문은 식과 case 문의 값과 비교
  - ▣ case의 비교 값과 일치하면 해당 case문의 실행문장 수행.
    - break를 만나면 switch문을 벗어남
  - ▣ case의 비교 값과 일치하는 것이 없으면 default 문 실행
- default문은 생략 가능



# switch문에서 벗어나기

56

## □ switch문 내의 break문

- ▣ break 문장을 만나면 switch문을 벗어남
- ▣ 만일 case 문에 break문이 없다면, 다음 case문의 실행 문장으로 실행을 계속하게 되며, 언젠가 break를 만날 때까지 계속 내려감

```
char grade='A';
switch (grade) {
 case 'A':
 System.out.println("90 ~ 100점입니다.");
 break;
 case 'B':
 System.out.println("80 ~ 89점입니다.");
 break;
 case 'C':
 System.out.println("70 ~ 79점입니다.");
 break;
}
```

90 ~ 100점입니다.  
80 ~ 89점입니다.



## 예제 2-13 : switch문의 break 사용하기

57

학점이 A, B 인 학생에게는 "참 잘하였습니다.", 학점이 C, D인 학생에게는 " 좀 더 노력하세요. ", 학점이 F인 학생에게는 " 다음 학기에 다시 수강하세요. "를 출력하는 프로그램을 switch문의 break를 잘 활용하여 작성하여라.

```
public class GradeSwitch {
 public static void main(String[] args) {
 char grade='C';
 switch (grade) {
 case 'A':
 case 'B':
 System.out.println("참 잘하였습니다.");
 break;
 case 'C':
 case 'D':
 System.out.println("좀 더 노력하세요.");
 break;
 case 'F':
 System.out.println("다음 학기에 다시 수강하세요.");
 break;
 default:
 System.out.println("잘못된 학점입니다.");
 }
 }
}
```

좀 더 노력하세요.

# case 문의 값

58

## □ case 문의 값의 특징

- ▣ switch 문은 식의 결과 값을 case 문과 비교
- ▣ 사용 가능한 case문의 비교 값
  - 정수 타입 리터럴, JDK 1.7부터는 문자열 리터럴도 허용

```
int a = 0;
int b = 1;
int c = 25;
switch(c%2) {
 case 1 : // 정수 리터럴
 ...;
 break;
 case 2: // 정수 리터럴
}

String s = "예";
switch(s) {
 case "예" : // 문자열 리터럴 사용 가능. JDK1.7부터 적용
 ...;
 break;
 case "아니요" : // 문자열 리터럴 사용 가능. JDK1.7부터 적용
 ...;
 break;
}
```

```
switch(a) {
 case a : // 오류. 변수 사용 안됨
 case a > 3 : // 오류. 수식 안됨
 case a == 1 : // 오류. 수식 안됨
}
```

잘못된  
case 문

정상적인  
case 문

## 예제 2-14 : 성적 분류

59

앞의 다중 if문을 이용한  
성적 분류 프로그램을  
switch문으로 바꾸시오.

100  
학점은 A입니다  
55  
학점은 F입니다  
76  
학점은 C입니다

```
import java.util.Scanner;

public class Grading2 {
 public static void main (String[] args) {
 char grade;
 Scanner a = new Scanner(System.in);
 while (a.hasNext()) {
 int score = a.nextInt();
 switch (score/10) {
 case 10:
 case 9:
 grade = 'A';
 break;
 case 8:
 grade = 'B';
 break;
 case 7:
 grade = 'C';
 break;
 case 6:
 grade = 'D';
 break;
 default:
 grade = 'F';
 }
 System.out.println("학점은 "+grade+"입니다");
 }
 }
}
```