



# 예제 2-1 자바프로그램의 기본 구조

2

```
/*  
 * 소스 파일 : Hello.java  
 */
```

```
public class Hello {
```

```
    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("Hello"); // "Hello" 문자열 화면 출력
```

```
        System.out.println(s); // 정수 s 값 화면 출력
```

```
    }
```

```
}
```

클래스

```
?  
Hello  
30
```

메소드

# 예제 2-1 코드 설명

3

## □ 클래스 만들기

- Hello 이름의 클래스 선언

```
public class Hello {  
}
```

- class 키워드로 클래스 선언
- public 선언하면 다른 클래스에서 접근 가능
- 클래스 코드는 {} 내에 모두 작성

## □ 주석문

- // 한 라인 주석
- /\* 여러 행 주석 \*/

## □ main() 메소드

- 자바 프로그램은 main()에서 실행 시작

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

- public static void으로 선언
- String[] args로 실행 인자를 전달 받음

## □ 메소드

- C/C++에서의 함수를 메소드로 지칭

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

- 클래스 바깥에 작성할 수 없음

## □ 메소드 호출

- sum() 메소드 호출

```
int i = 20;  
s = sum(i, 10);
```

- sum() 호출 시 변수 i의 값과 정수 10을 전달
- sum()의 n, m에 각각 20, 10 값 전달
- sum()은 n과 m 값을 더한 30 리턴
- 변수 s는 정수 30을 전달받음



# 예제 2-1 설명 (계속)

4

## □ 변수 선언

- 변수 타입과 변수 이름 선언

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

- 메소드 내에서 선언된 변수는 지역 변수

- 지역 변수는 메소드 실행이 끝나면 자동 소멸

## □ 문장

- ;로 한 문장의 끝을 인식

```
int i=20;  
s = sum(i, 20);
```

## □ 화면 출력

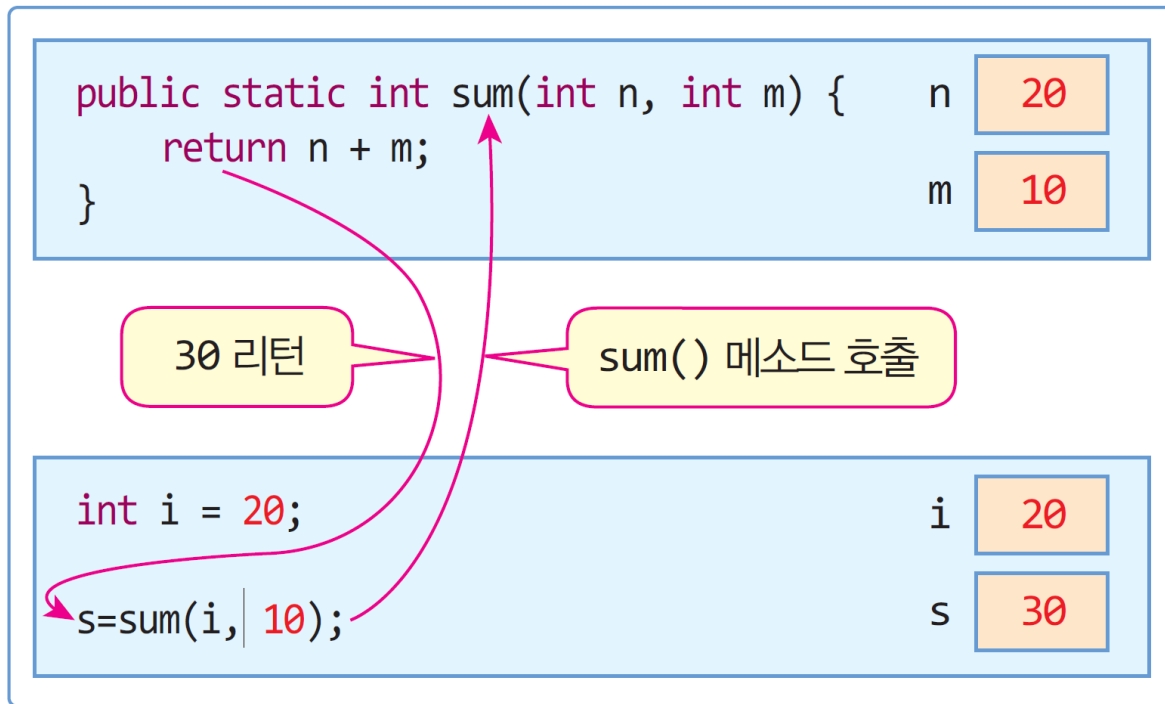
- 표준 출력 스트림에 메시지 출력

```
System.out.println("Hello"); // "Hello" 화면 출력
```

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

# sum() 메소드 호출과 리턴

5



클래스 Hello

# 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;             // 식별자의 첫문자로 숫자 사용 불가
class  if { }                 // 자바의 예약어 if 사용 불가
char   false;                 // false 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        |



# 좋은 이름 붙이는 언어 관습

9

- 기본 : 가독성 높은 이름
  - ▣ 목적을 나타내는 이름 붙이기 : s 보다 sum
  - ▣ 충분히 긴 이름으로 붙이기 : AVM보다 AutoVendingMachine
- 자바 언어의 이름 붙이는 관습 : 헝가리언 이름 붙이기
  - ▣ 클래스 이름
    - 첫 번째 문자는 대문자로 시작
    - 각 단어의 첫 번째 문자만 대문자
- 변수, 메소드 이름
  - 첫 단어 이후 각 단어의 첫 번째 문자는 대문자로 시작
- 상수 이름
  - 모든 문자를 대문자로 표시

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

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

```
int myAge;  
boolean isSingle;  
public int getAge() { }
```

# 자바의 데이터 타입

10

## □ 자바의 데이터 타입

### ▣ 기본 타입 : 8 개

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

### ▣ 레퍼런스 타입 : 1 개이며 용도는 다음 3 가지

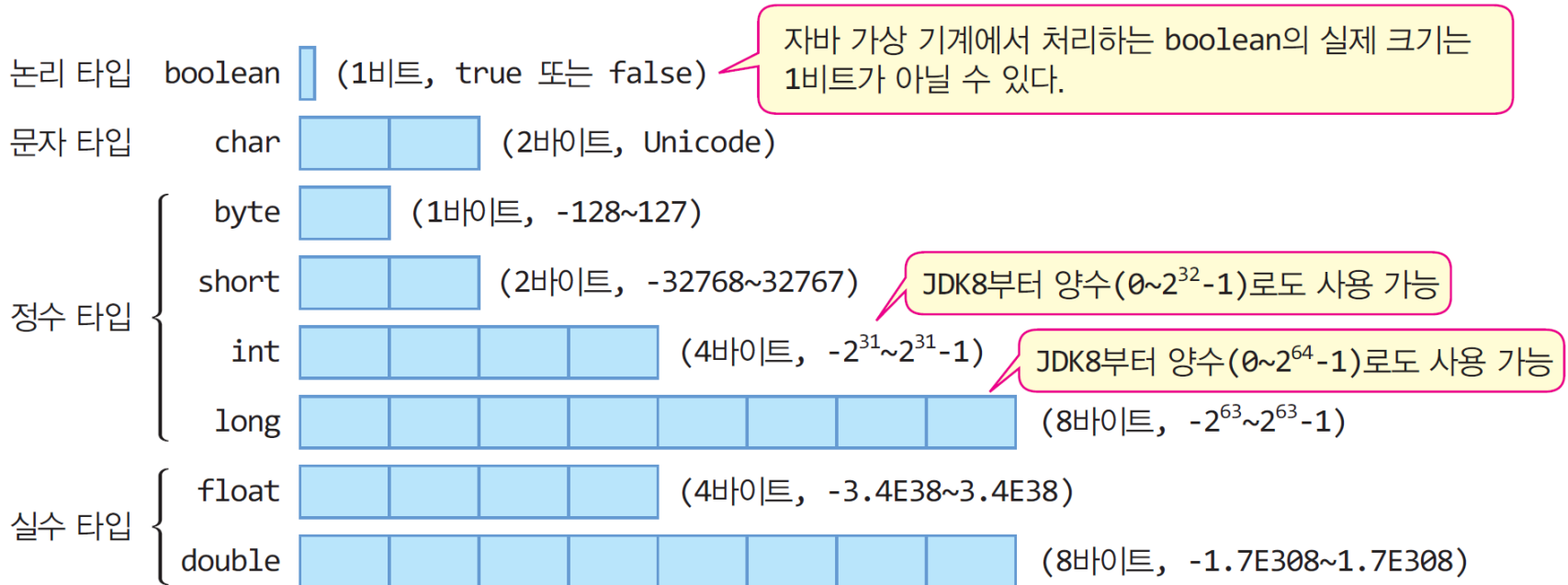
- 배열(array)에 대한 레퍼런스
- 클래스(class)에 대한 레퍼런스
- 인터페이스(interface)에 대한 레퍼런스

# 자바의 기본 타입

11

## 특징

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



# 문자열

12

- 문자열은 기본 타입이 아님
- String 클래스로 문자열 표현
  - ▣ 문자열 리터럴 - "JDK", "한글", "계속하세요"

```
String toolName="JDK";
```

- ▣ 문자열이 섞인 + 연산 -> 문자열 연결

```
toolName + 1.8          -> "JDK1.8"  
"(" + 3 + "," + 5 + ")" -> "(3,5)"  
System.out.println(toolName + "이 출시됨"); // "JDK1.8이 출시됨" 출력
```

# 변수와 선언

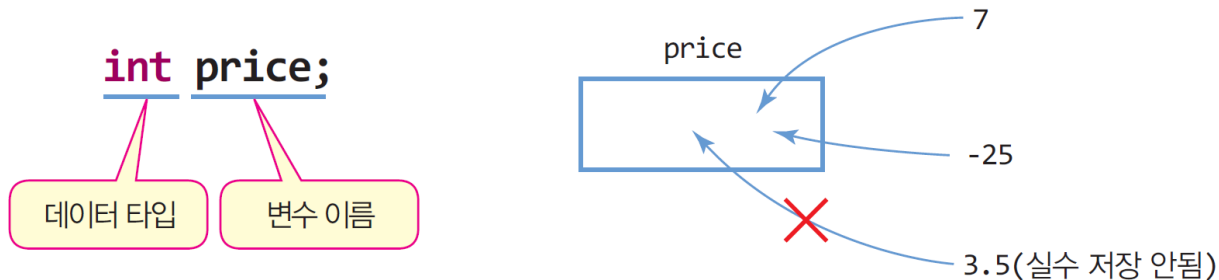
13

## □ 변수

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

## □ 변수 선언

- ▣ 변수의 타입 다음에 변수 이름을 적어 변수를 선언



# 변수 선언 사례

14

## □ 변수 선언 사례

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

## □ 변수 선언과 초기화

### ▣ 선언과 동시에 초기값 지정

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

## □ 변수 읽기와 저장

### ▣ 대입 연산자인 = 다음에 식(expression)

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



# 리터럴과 정수 리터럴

15

- 리터럴(literal)
  - ▣ 프로그램에서 직접 표현한 값
  - ▣ 정수, 실수, 문자, 논리, 문자열 리터럴 있음
    - 사례) 34, 42.195, ' % ', true, " hello "
- 정수 리터럴
  - ▣ 10진수, 8진수, 16진수, 2진수 리터럴

|        |                           |
|--------|---------------------------|
| 15     | -> 10진수 리터럴 15            |
| 015    | -> 0으로 시작하면 8진수. 십진수로 13  |
| 0x15   | -> 0x로 시작하면 16진수. 십진수로 21 |
| 0b0101 | -> 0b로 시작하면 2진수. 십진수로 5   |



```
int n = 15;  
int m = 015;  
int k = 0x15;  
int b = 0b0101;
```

- ▣ 정수 리터럴은 int 형으로 컴파일
- ▣ long 타입 리터럴은 숫자 뒤에 L 또는 l을 붙여 표시
  - ex) long g = 24L;

# 실수 리터럴

16

- 소수점 형태나 지수 형태로 표현한 실수

- 12. 12.0 .1234 0.1234 1234E-4

- 실수 타입 리터럴은 double 타입으로 컴파일

```
double d = 0.1234;  
double e = 1234E-4; // 1234E-4 = 1234x10-4이므로 0.1234와 동일
```

- 숫자 뒤에 f(float)나 d(double)을 명시적으로 붙이기도 함

```
float f = 0.1234f;  
double w = .1234D; // .1234D와 .1234는 동일
```

# 문자 리터럴

17

- 단일 인용부호(' ')로 문자 표현
  - 사례) 'w', 'A', '가', '\*', '3', '글', ₩u0041
  - ₩u다음에 4자리 16진수(2바이트의 유니코드)
    - \u0041 -> 문자 'A'의 유니코드(0041)
    - \uae00 -> 한글문자 '글'의 유니코드(ae00)

```
char a = 'A';  
char b = '글';  
char c = ₩u0041; // 문자 'A'의 유니코드 값(0041) 사용  
char d = ₩uae00; // 문자 '글'의 유니코드 값(ae00) 사용
```


- 특수문자 리터럴은 백슬래시(\)로 시작

| 종류   | 의미               | 종류   | 의미                      |
|------|------------------|------|-------------------------|
| '\b' | 백스페이스(backspace) | '\r' | 캐리지 리턴(carriage return) |
| '\t' | 탭(tab)           | '\"' | 이중 인용부호(double quote)   |
| '\n' | 라인피드(line feed)  | '\'' | 단일 인용부호(single quote)   |
| '\f' | 폼피드(form feed)   | '\\' | 백슬래시(backslash)         |

# 논리 리터럴과 boolean 타입

18

- 논리 리터럴은 2개뿐
  - ▣ true, false
  - ▣ boolean 타입 변수에 치환하거나 조건문에 이용

boolean a = true;  
boolean b = 10 > 0; // 10>0가 참이므로 b 값은 true  
 boolean c = 1; // 타입 불일치 오류. C/C++와 달리 자바에서 1,0을 참, 거짓으로 사용 불가  
  
while(**true**) { // 무한 루프. while(1)로 사용하면 안 됨  
...  
}

# Tip: 기본 타입 이외의 리터럴

19

## □ null 리터럴

- ▣ 레퍼런스에 대입 사용



```
int n = null; // 기본 타입에 사용 불가  
String str = null;
```

## □ 문자열 리터럴(스트링 리터럴)

- ▣ 이중 인용부호로 묶어 표현
  - 사례) "Good", "Morning", "자바", "3.19", "26", "a"
- ▣ 문자열 리터럴은 String 객체로 자동 처리

```
String str = "Good";
```

# Tip. JDK7부터 숫자에 '\_' 허용, 가독성 높임

20

- 숫자 리터럴의 아무 위치에나 언더스코어('\_') 허용
  - ▣ 컴파일러는 리터럴에서 '\_'를 빼고 처리
- 사용 예

```
int price = 20_100;                // 20100과 동일
long cardNumber = 1234_5678_1357_9998L; // 1234567813579998L와 같음
long controlBits = 0b10110100_01011011_10110011_111110000;
long maxLong = 0x7fff_ffff_ffff_ffffL;
int age = 2____5;                  // 25와 동일
```

- 허용되지 않는 4가지 경우

```
int x = 15_;           // 오류. 리터럴 끝에 사용할 수 없다.
double pi = 3_.14;     // 오류. 소수점(.) 앞뒤에 사용할 수 없다.
long idNum = 981231_1234567_L; // 오류. _L(_F) 앞에 사용할 수 없다.
int y = 0_x15;         // 오류. 0x 중간이나 끝에 사용할 수 없다. 0x_15(오류)
```





# Tip. var 키워드를 사용하여 변수 타입 생략

21

## □ var 키워드

### ▣ Java 10부터 도입된 키워드

- var와 동일한 기능으로 C++(2011년 표준부터)의 auto 키워드
- ▣ 지역 변수의 선언에만 사용
- ▣ 변수 타입 선언 생략 : 컴파일러가 변수 타입 추론

## □ 사용 예

```
var price = 200;           // price는 int 타입으로 결정
var name = "kitae";        // name은 String 타입으로 결정
var pi = 3.14;             // pi는 double 타입으로 결정
var point = new Point();    // point는 Point 타입으로 결정(4장 참조)
var v = new Vector<Integer>(); // v는 Vector<integer> 타입으로 결정(7장 참조)
```

## □ 변수 선언문에 반드시 초기값 지정



```
var name;           // 컴파일 오류. 변수 name의 타입을 추론할 수 없음
```

# 상수

22

## □ 상수 선언

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

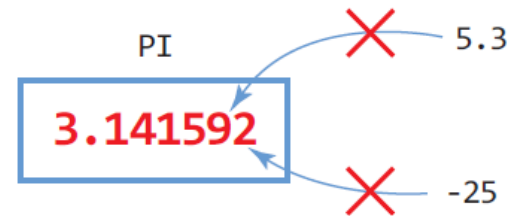
**final double PI = 3.141592;**

상수 선언

데이터 타입

상수 이름

초기화



## 예제 2-2 : 변수, 리터럴, 상수 활용

23

원의 면적을 구하는 프로그램을 작성해보자.

```
public class CircleArea {  
  
    public static void main(String[] args) {  
        final double PI = 3.14; // 원주율을 상수로 선언  
  
        double radius = 10.0; // 원의 반지름  
        double circleArea = radius*radius*PI; // 원의 면적 계산  
  
        // 원의 면적을 화면에 출력한다.  
        System.out.println("원의 면적 = " + circleArea);  
    }  
}
```

원의 면적 = 314.0

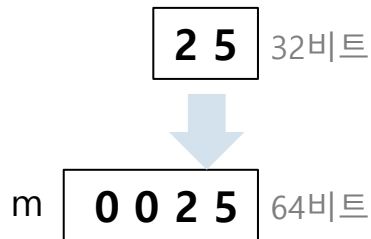
# 자동 타입 변환

24

## □ 자동 타입 변환

- ▣ 작은 타입은 큰 타입으로 자동 변환
  - 컴파일러에 의해 이루어짐
- ▣ 치환문(=)이나 수식 내에서 타입이 일치하지 않을 때

`long m = 25;` // 25는 int 타입. 25가 long 타입으로 자동 변환되는 사례



`double d = 3.14 * 10;` // 실수 연산을 하기 위해 10이 10.0으로 자동 변환  
// 다른 피연산자 3.14가 실수이기 때문

# 강제 타입 변환

25

- 자동 타입 변환이 안 되는 경우 : 큰 타입이 작은 타입으로 변환할 때

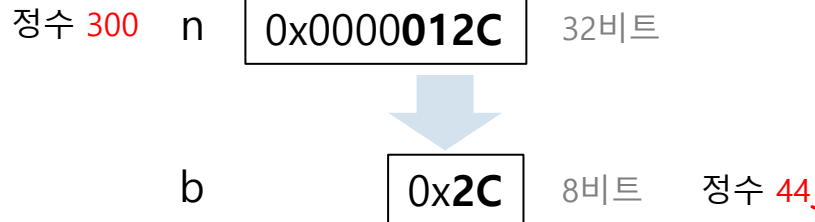
오류

```
int n = 300;  
byte b = n; // 컴파일 오류. int 타입이 byte로 자동 변환 안 됨
```

강제 타입 변환하려면, byte b = (byte)n; 로 수정

- 강제 타입 변환
  - ▣ 개발자가 필요하여 강제로 타입 변환을 지시
    - () 안에 변환할 타입 지정
  - ▣ 강제 변환은 값 손실 우려

byte b = (byte)n; 에 따른 손실



```
double d = 1.9;  
int n = (int)d; // n = 1
```

강제 타입 변환으로  
소숫점 이하 0.9 손실

$$44 = 300 \% 256$$

## 예제 2-3 : 타입 변환

26

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);  
    }  
}
```

강제 타입 변환 결과 0x41이 되며, 문자 A의 코드임

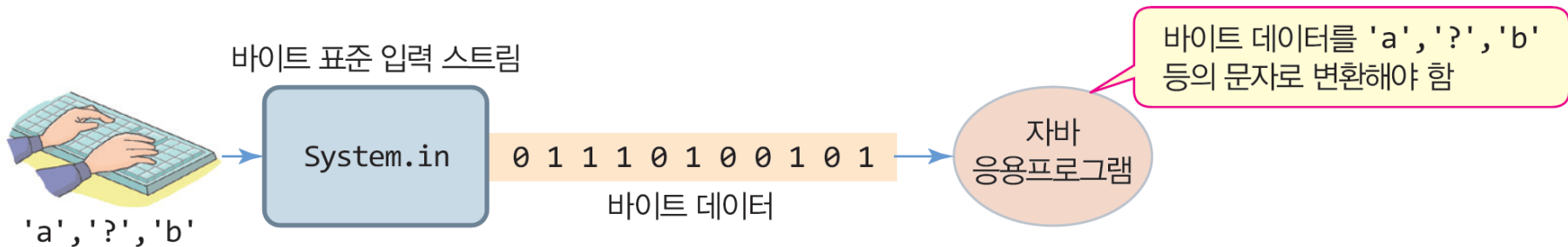
227  
2  
2.5  
A  
-29  
3.8  
4  
3



# 자바에서 키 입력

27

- System.in
  - ▣ 키보드로부터 직접 읽는 자바의 표준 입력 스트림
  - ▣ 키 값을 바이트(문자 아님)로 리턴
- System.in을 사용할 때 문제점
  - ▣ 키 값을 바이트 데이터로 넘겨주므로 응용프로그램이 문자 정보로 변환해야 함



# Scanner로 쉽게 키 입력

28

## □ Scanner 클래스

- System.in에게 키를 읽게 하고, 읽은 바이트를 문자, 정수, 실수, 불린, 문자열 등 다양한 타입으로 변환하여 리턴

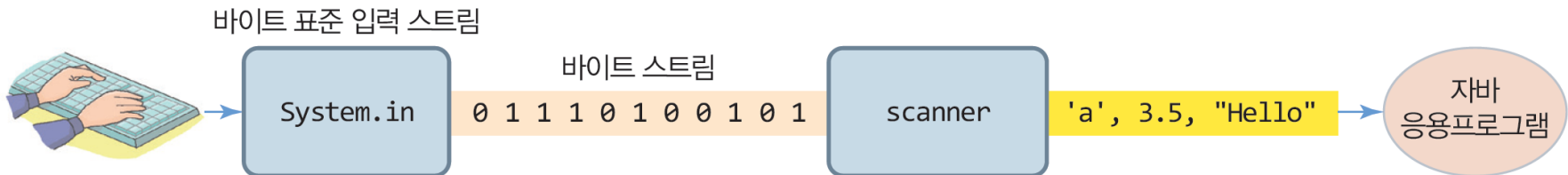
- java.util.Scanner 클래스

## □ 객체 생성

```
import java.util.Scanner; // import 문 필요
```

```
...
```

```
Scanner a = new Scanner(System.in); // Scanner 객체 생성
```

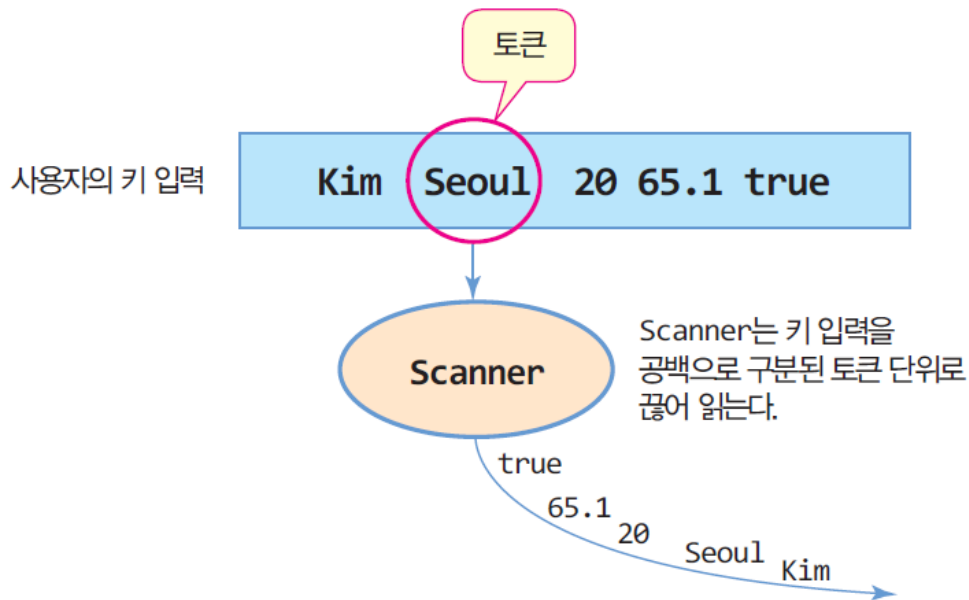


- System.in에게 키를 읽게 하고, 원하는 타입으로 변환하여 리턴

# Scanner를 이용한 키 입력

29

- Scanner에서 키 입력 받기
  - Scanner는 입력되는 키 값을 공백으로 구분되는 아이템 단위로 읽음
  - 공백 문자 : 'wt', 'wf', 'wr', ' ', 'wn'
- 개발자가 원하는 다양한 타입의 값으로 바꾸어 읽을 수 있음



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

String name = scanner.next();           // "Kim"
String city = scanner.next();           // "Seoul"
int age = scanner.nextInt();            // 20
double weight = scanner.nextDouble();  // 65.1
boolean single = scanner.nextBoolean(); // true
```

# Scanner 주요 메소드

30

| 메소드                                | 설명   |
|------------------------------------|--|
| <code>String next()</code>         | 다음 토큰을 문자열로 리턴   |
| <code>byte nextByte()</code>       | 다음 토큰을 byte 타입으로 리턴  |
| <code>short nextShort()</code>     | 다음 토큰을 short 타입으로 리턴   |
| <code>int nextInt()</code>         | 다음 토큰을 int 타입으로 리턴   |
| <code>long nextLong()</code>       | 다음 토큰을 long 타입으로 리턴  |
| <code>float nextFloat()</code>     | 다음 토큰을 float 타입으로 리턴   |
| <code>double nextDouble()</code>   | 다음 토큰을 double 타입으로 리턴  |
| <code>boolean nextBoolean()</code> | 다음 토큰을 boolean 타입으로 리턴   |
| <code>String nextLine()</code>     | '\n'을 포함하는 한 라인을 읽고 '\n'을 버린 나머지 문자열 리턴  |
| <code>void close()</code>          | Scanner의 사용 종료   |
| <code>boolean hasNext()</code>     | 현재 입력된 토큰이 있으면 true, 아니면 입력 때까지 무한정 대기, 새로운 입력이 들어올 때 true 리턴. ctrl-z 키가 입력되면 입력 끝이므로 false 리턴 |

## 예제 2-4 : Scanner를 이용한 키 입력 연습

31

Scanner를 이용하여  
이름, 도시, 나이, 체중,  
독신 여부를 입력 받고  
다시 출력하는  
프로그램을 작성하라.

```
import java.util.Scanner;

public class ScannerEx {
    public static void main(String args[]) {
        System.out.println("이름, 도시, 나이, 체중, 독신 여부를 빈칸으로 분리하여 입력하세요");
        Scanner scanner = new Scanner(System.in);

        String name = scanner.next(); // 문자열 읽기
        System.out.print("이름은 " + name + ", ");

        String city = scanner.next(); // 문자열 읽기
        System.out.print("도시는 " + city + ", ");

        int age = scanner.nextInt(); // 정수 읽기
        System.out.print("나이는 " + age + "살, ");

        double weight = scanner.nextDouble(); // 실수 읽기
        System.out.print("체중은 " + weight + "kg, ");

        boolean single = scanner.nextBoolean(); // 논리값 읽기
        System.out.println("독신 여부는 " + single + "입니다.");

        scanner.close(); // scanner 닫기
    }
}
```

이름, 도시, 나이, 체중, 독신 여부를 빈칸으로 분리하여 입력하세요.

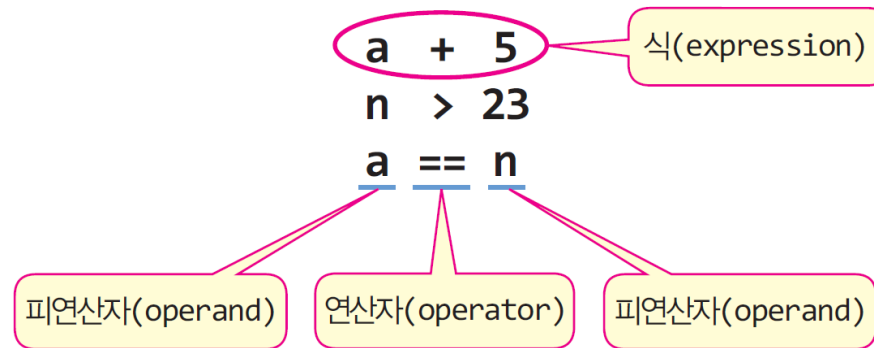
Kim Seoul 20 65.1 true

이름은 Kim, 도시는 Seoul, 나이는 20살, 체중은 65.1kg, 독신 여부는 true입니다.

# 식과 연산자

32

- 연산 : 주어진 식을 계산하여 결과를 얻어내는 과정



| 연산의 종류 | 연산자   | 연산의 종류 | 연산자   |
|--------|---|--------|---|
| 증감     | <code>++ --</code>                          | 비트     | <code>&amp;   ^ ~</code>  |
| 산술     | <code>+ - * / %</code>                      | 논리     | <code>&amp;&amp;    ! ^</code>  |
| 시프트    | <code>&gt;&gt; &lt;&lt; &gt;&gt;&gt;</code> | 조건     | <code>? :</code>  |
| 비교     | <code>&gt; &lt; &gt;= &lt;= == !=</code>    | 대입     | <code>= *= /= += -= &amp;= ^=  = &lt;&lt;= &gt;&gt;= &gt;&gt;&gt;=</code> |



# 연산자 우선순위

33

|   |   |
|---|---|
| <div> <div>높음</div> <div>↓</div> <div>낮음</div> </div> | ++(postfix) --(postfix)                     |
|   | +(양수 부호) -(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

34

## □ 산술 연산자

- ▣ 더하기(+), 빼기(-), 곱하기(\*), 나누기(/), 나머지(%)
- ▣ /와 % 응용

- 10의 자리와 1의 자리 분리

|              |         |
|--------------|---------|
| $69/10 = 6$  | ← 몫 6   |
| $69\%10 = 9$ | ← 나머지 9 |

- n이 홀수인지 판단

```
int r = n % 2;      // r이 1이면 n은 홀수, 0이면 짝수
```

| 연산자 | 의미  | 예          | 결과   |
|-----|-----|------------|------|
| +   | 더하기 | 25.5 + 3.6 | 29.1 |
| -   | 빼기  | 3 - 5      | -2   |
| *   | 곱하기 | 2.5 * 4.0  | 10.0 |
| /   | 나누기 | 5/2        | 2    |
| %   | 나머지 | 5%2        | 1    |

## 예제 2-5 : /와 % 산술 연산

35

초 단위의 정수를 입력받고, 몇 시간, 몇 분, 몇 초인지 출력하는 프로그램을 작성하라.

```
import java.util.Scanner;

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

        System.out.print("정수를 입력하세요: ");
        int time = scanner.nextInt();    // 정수 입력
        int second = time % 60;        // 60으로 나눈 나머지는 초
        int minute = (time / 60) % 60;  // 60으로 나눈 몫을 다시 60으로 나눈 나머지는 분
        int hour = (time / 60) / 60;    // 60으로 나눈 몫을 다시 60으로 나눈 몫은 시간

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

        scanner.close();
    }
}
```

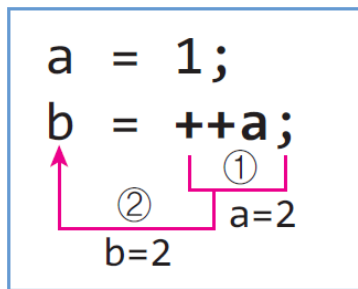
정수를 입력하세요:5000  
5000초는 1시간, 23분, 20초입니다.

# 증감 연산

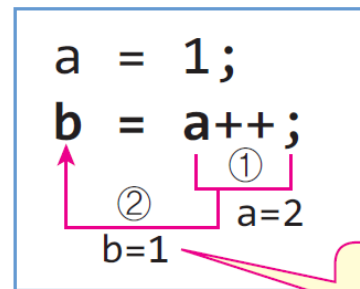
36

## □ 1 증가 혹은 감소 시키는 연산

▣ ++, --



(a) 전위 연산자



(b) 후위 연산자

a++의 연산은 증가 전의 값 1을 반환한다.

| 연산자 | 내용                   | 연산자 | 내용                 |
|-----|----------------------|-----|--------------------|
| a++ | a를 1 증가하고 증가 전의 값 반환 | ++a | a를 1 증가하고 증가된 값 반환 |
| a-- | a를 1 감소하고 감소 전의 값 반환 | --a | a를 1 감소하고 감소된 값 반환 |

# 대입 연산

37

## □ 연산의 오른쪽 결과는 왼쪽 변수에 대입

```
int a = 1, b = 3;  
a = b;           // b 값을 a에 대입하여 a=3  
a += b;          // a = a + b의 연산이 이루어져, a=6. b는 3 그대로
```

| 대입 연산자 | 내용            | 대입 연산자   | 내용              |
|--------|---------------|----------|-----------------|
| a = b  | b의 값을 a에 대입   | a &= b   | a = a & b와 동일   |
| a += b | a = a + b와 동일 | a ^= b   | a = a ^ b와 동일   |
| a -= b | a = a - b와 동일 | a  = b   | a = a   b와 동일   |
| a *= b | a = a * b와 동일 | a <<= b  | a = a << b와 동일  |
| a /= b | a = a / b와 동일 | a >>= b  | a = a >> b와 동일  |
| a %= b | a = a % b와 동일 | a >>>= b | a = a >>> b와 동일 |

## 예제 2-6 : 대입 연산자와 증감 연산자 사용

38

다음 코드의 실행 결과는 무엇인가?

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

        // 대입 연산자 사례
        a += 3;    // a=a+3 = 6
        b *= 3;    // b=b*3 = 9
        c %= 2;    // c=c%2 = 1
        System.out.println("a=" + a + ", b=" + b + ", c=" + c);

        int d=3;
        // 증감 연산자 사례
        a = d++; // a=3, d=4
        System.out.println("a=" + a + ", d=" + d);
        a = ++d; // d=5, a=5
        System.out.println("a=" + a + ", d=" + d);
        a = d--; // a=5, d=4
        System.out.println("a=" + a + ", d=" + d);
        a = --d; // d=3, a=3
        System.out.println("a=" + a + ", d=" + d);
    }
}
```

a=6, b=9, c=1  
a=3, d=4  
a=5, d=5  
a=5, d=4  
a=3, d=3

# 비교 연산과 논리 연산

39

## ▣ 비교 연산

- 두 피연산자를 비교하여 true 또는 false의 논리 값을 내는 연산

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

## ▣ 논리 연산

- 논리 값으로 NOT, OR, AND, XOR 논리 연산. 논리 값을 내는 연산

| 연산자             | 내용                                     | 예제                           | 결과    |
|-----------------|--|------------------------------|-------|
| $! a$           | a가 true이면 false, false이면 true          | $!(3 < 5)$                   | false |
| $a \parallel b$ | a와 b의 OR 연산. a와 b 모두 false인 경우에만 false | $(3 > 5) \parallel (1 == 1)$ | true  |
| $a \&\& b$      | a와 b의 AND 연산. a와 b 모두 true인 경우에만 true  | $(3 < 5) \&\& (1 == 1)$      | true  |
| $a \wedge b$    | a와 b의 XOR 연산. a와 b가 서로 다를 때 true       | $(3 > 5) \wedge (1 == 1)$    | true  |

# 비교 연산과 논리 연산의 복합 사례

40

```
// 나이(int age)가 20대인 경우  
(age >= 20) && (age < 30)
```

```
// 문자(char c)가 대문자인 경우  
(c >= 'A') && (c <= 'Z')
```

```
// (x,y)가 (0,0)과 (50,50)의 사각형 내에 있음  
(x>=0) && (y>=0) && (x<=50) && (y<=50)
```

```
20 <= age < 30    // 오류
```



## 예제 2-7 : 비교 연산자와 논리 연산자 사용하기

41

다음 소스의 실행 결과는 무엇인가?

```
public class LogicalOperator {  
    public static void main (String[] args) {  
        // 비교 연산  
        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));  
  
        // 비교 연산과 논리 연산 복합  
        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  
true  
false  
false  
true  
true
```


# 조건 연산자 ?:

42

- `condition ? opr2 : opr3`
  - ▣ 세 개의 피연산자로 구성된 삼항(ternary) 연산자
    - `condition`이 `true`이면, 연산식의 결과는 `opr2`, `false`이면 `opr3`
  - ▣ `if-else`을 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 2-8 : 조건 연산

43

다음은 조건 연산자의 사례이다. 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)));  
    }  
}
```

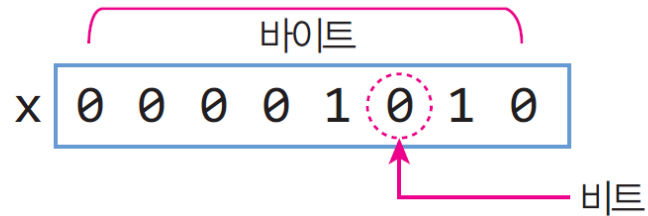
두 수의 차는 2

# 비트 연산

44

## □ 비트 개념

byte x = 10;



## □ 비트 연산

### ▣ 비트 논리 연산

- 비트끼리 AND, OR, XOR, NOT 연산

### ▣ 비트 시프트 연산

- 비트를 오른쪽이나 왼쪽으로 이동

# 비트 논리 연산

45

## □ 피 연산자의 각 비트들의 논리 연산

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

모두 1이므로  
결과는 1

둘 중 하나라도  
0이면 결과는 0

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

모두 0이므로  
결과는 0

둘 중 하나라도  
1이면 결과는 1

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

두 비트가 같으므로  
결과는 0

두 비트가 다르므로  
결과는 1

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

1은 0으로 변환

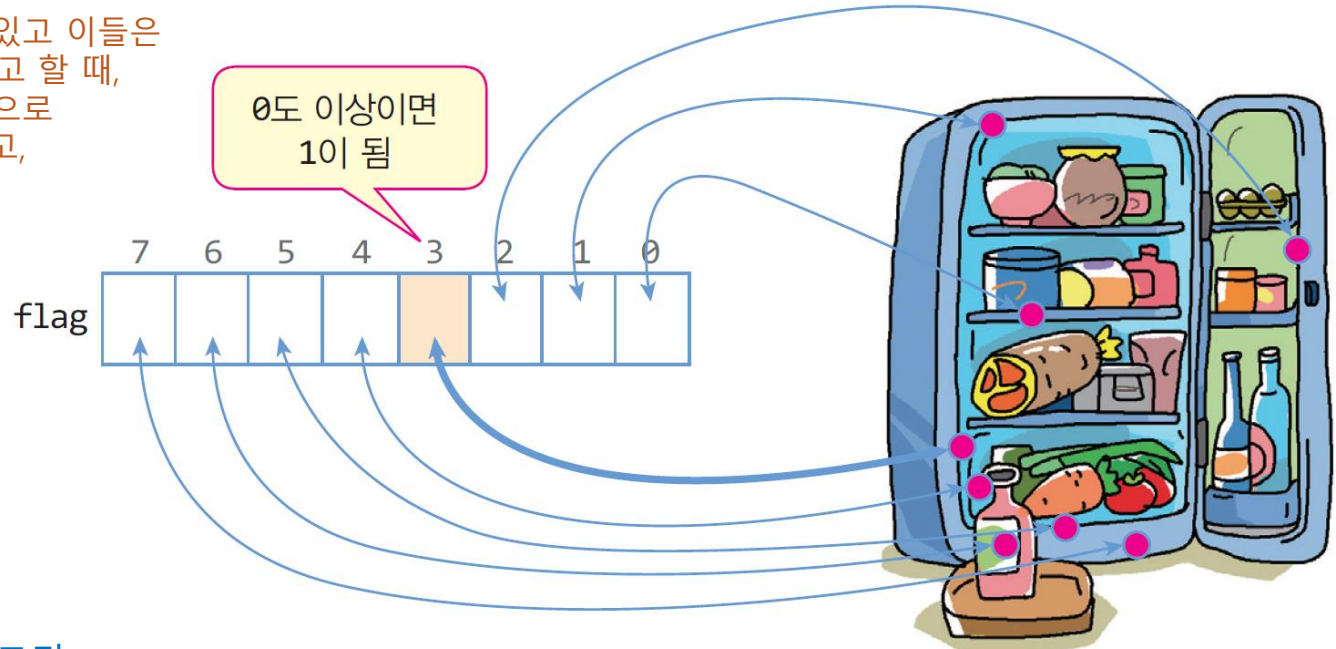
0은 1로 변환

| 연산자      | 별칭     | 내용                       |
|----------|--------|--------------------------|
| $a \& b$ | AND 연산 | 두 비트 모두 1이면 1, 그렇지 않으면 0 |
| $a   b$  | OR 연산  | 두 비트 모두 0이면 0, 그렇지 않으면 1 |
| $a ^ b$  | XOR 연산 | 두 비트가 다르면 1, 같으면 0       |
| $\sim a$ | NOT 연산 | 1을 0으로, 0을 1로 변환         |

# 비트 논리 연산 응용

46

냉장고에는 8개의 센서가 있고 이들은 flag 변수와 연결되어 있다고 할 때, 냉장고의 온도가 0도 이상으로 올라가면 비트 3이 1이 되고, 0도 이하이면 비트 3이 0을 유지한다.



문제) 현재 냉장고의 온도가 0도 이상인지 판단하는 코드를 작성하라.

```
byte flag = 0b00001010; // 각 비트는 8개의 센서 값을 가리킴
if(flag & 0b00001000 == 0)
    System.out.print("온도는 0도 이하");
else
    System.out.print("온도는 0도 이상");
```

온도는 0도 이상

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
|   | x | x | x | x | Y | x | x | x |
| & | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|   | 0 | 0 | 0 | 0 | Y | 0 | 0 | 0 |

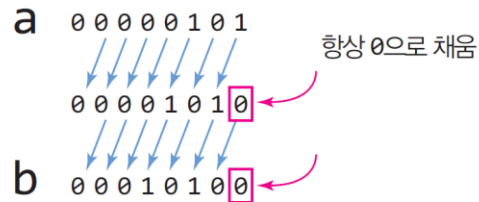
Y비트가 0이면  
& 결과는 0

# 시프트 연산자의 사례

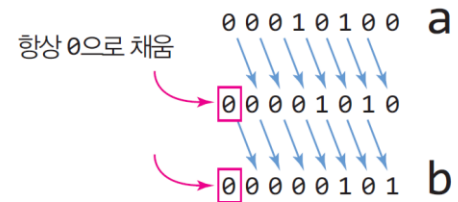
47

## □ 피 연산자의 비트들을 이동 연산

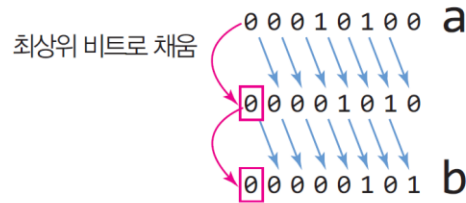
```
byte a = 5; // 5
byte b = (byte)(a << 2); // 20
```



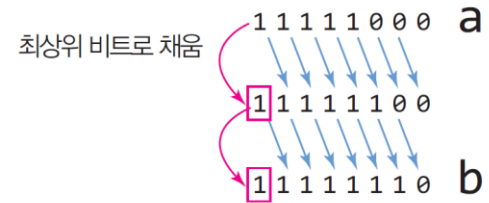
```
byte a = 20; // 20
byte b = (byte)(a >>> 2); // 5
```



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



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



| 시프트 연산자 | 내용   |
|---------|--|
| a >> b  | a의 각 비트를 오른쪽으로 b번 시프트한다. 최상위 비트의 빈자리는 시프트 전의 최상위 비트로 다시 채운다. 산술적 오른쪽 시프트라고 한다. |
| a >>> b | a의 각 비트를 오른쪽으로 b번 시프트한다. 최상위 비트의 빈자리는 항상 0으로 채운다. 논리적 오른쪽 시프트라고 한다.            |
| a << b  | a의 각 비트를 왼쪽으로 b번 시프트한다. 최하위 비트의 빈자리는 항상 0으로 채운다. 산술적 왼쪽 시프트라고 한다.              |

# 예제 2-9 : 비트 논리 연산과 비트 시프트 연산

48

다음 소스의 실행 결과는 무엇인가?

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

        // 비트 논리 연산
        System.out.println("[비트 연산 결과]");
        System.out.printf("%04x\n", (short)(a & b)); // 비트 AND
        System.out.printf("%04x\n", (short)(a | b)); // 비트 OR
        System.out.printf("%04x\n", (short)(a ^ b)); // 비트 XOR
        System.out.printf("%04x\n", (short)(~a)); // 비트 NOT

        byte c = 20; // 0x14
        byte d = -8; // 0xf8

        // 비트 시프트 연산
        System.out.println("[시프트 연산 결과]");
        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("%x\n", (d >>> 2)); // d를 2비트 오른쪽 시프트. 0 삽입
    }
}
```

printf("%x\n", ...)는 결과 값을 16진수 형식으로 출력

[비트 연산 결과]  
00ff  
55ff  
5500  
aa00  
[시프트 연산 결과]  
80  
5  
-2  
3ffffffe



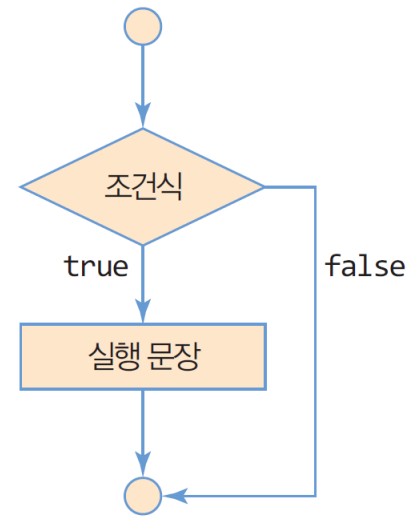
# 단순 if문

49

## □ 단순 if 문

- ▣ if의 괄호 안에 조건식(논리형 변수나 논리 연산)
  - 실행문장이 단일 문장인 경우 둘러싸는 { } 생략 가능

```
if (조건식) {  
    ...실행 문장... // 조건식이 참인 경우  
}
```



```
if(n%2 == 0) {  
    System.out.println(n + "은 짝수입니다.");  
}
```

## 예제 2-10 : if문 사용하기

50

시험 점수가 80점이 이상이면 합격 판별을 하는 프로그램을 작성하시오.

```
import java.util.Scanner;

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

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

        scanner.close();
    }
}
```

점수를 입력하시오: 95  
축하합니다! 합격입니다.

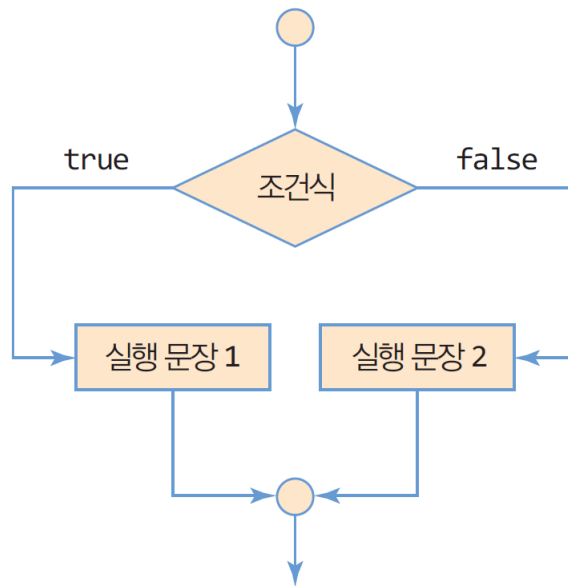
# 조건문 – if-else

51

## □ if-else 문

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

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



## 예제 2-11 : if-else 사용하기

52

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.");

        scanner.close();
    }
}
```

수를 입력하시오: 129  
3의 배수입니다.

# 다중 if-else 문

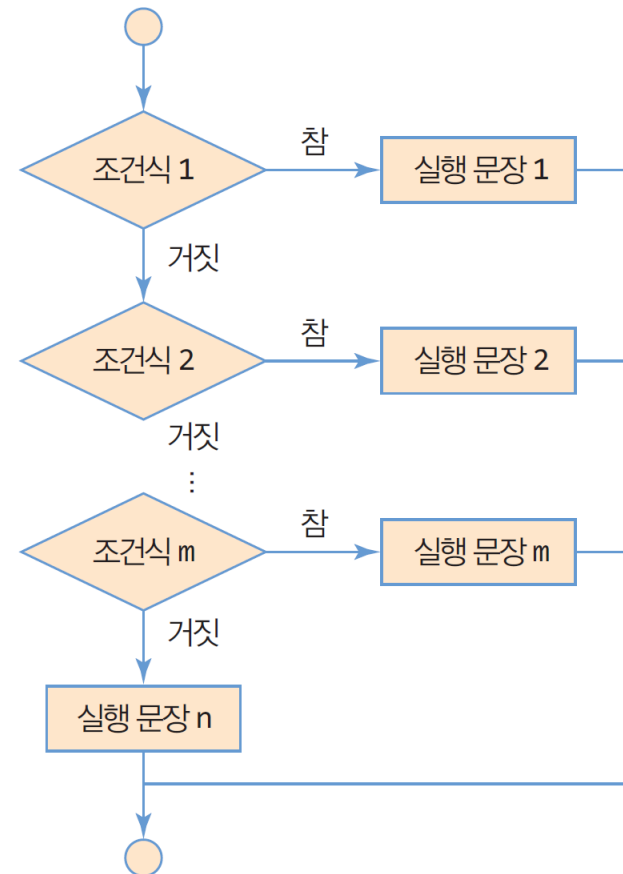
53

## □ 다중 if-else 문

### ▣ if-else가 연속되는 모양

- 조건문이 너무 많은 경우, switch 문 사용 권장

```
if (조건식 1) {  
    실행 문장 1; // 조건식 1이 참인 경우  
}  
else if (조건식 2) {  
    실행 문장 2; // 조건식 2가 참인 경우  
}  
else if (조건식 m) {  
    ..... // 조건식 m이 참인 경우  
}  
else {  
    실행 문장 n; // 앞의 모든 조건이 거짓인 경우  
}
```



## 예제 2-12 : 다중 if-else로 학점 매기기

54

다중 if-else문을 이용하여 입력받은 성적에 대해 학점을 부여하는 프로그램을 작성해보자.

```
import java.util.Scanner;
public class Grading {
    public static void main(String[] args) {
        char grade;
        Scanner scanner = new Scanner(System.in);

        System.out.print("점수를 입력하세요(0~100): ");
        int score = scanner.nextInt(); // 점수 읽기
        if(score >= 90) // score가 90 이상
            grade = 'A';
        else if(score >= 80) // score가 80 이상 90 미만
            grade = 'B';
        else if(score >= 70) // score가 70 이상 80 미만
            grade = 'C';
        else if(score >= 60) // score가 60 이상 70 미만
            grade = 'D';
        else // score가 60 미만
            grade = 'F';
        System.out.println("학점은 " + grade + "입니다.");

        scanner.close();
    }
}
```

점수를 입력하세요(0~100): 89  
학점은 B입니다.

## 예제 2-13 : 중첩 if-else 문 사례

55

점수와 학년을 입력 받아 60점 이상이면 합격,  
미만이면 불합격을 출력한다. 4학년의 경우 70점 이상이어야 합격이다.

```
import java.util.Scanner;
public class NestedIf {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("점수를 입력하세요(0~100): ");
        int score = scanner.nextInt();

        System.out.print("학년을 입력하세요(1~4): ");
        int year = scanner.nextInt();

        if(score >= 60) { // 60점 이상
            if(year != 4)
                System.out.println("합격!"); // 4학년 아니면 합격
            else if(score >= 70)
                System.out.println("합격!"); // 4학년이 70점 이상이면 합격
            else
                System.out.println("불합격!"); // 4학년이 70점 미만이면 불합격
        }
        else // 60점 미만 불합격
            System.out.println("불합격!");

        scanner.close();
    }
}
```

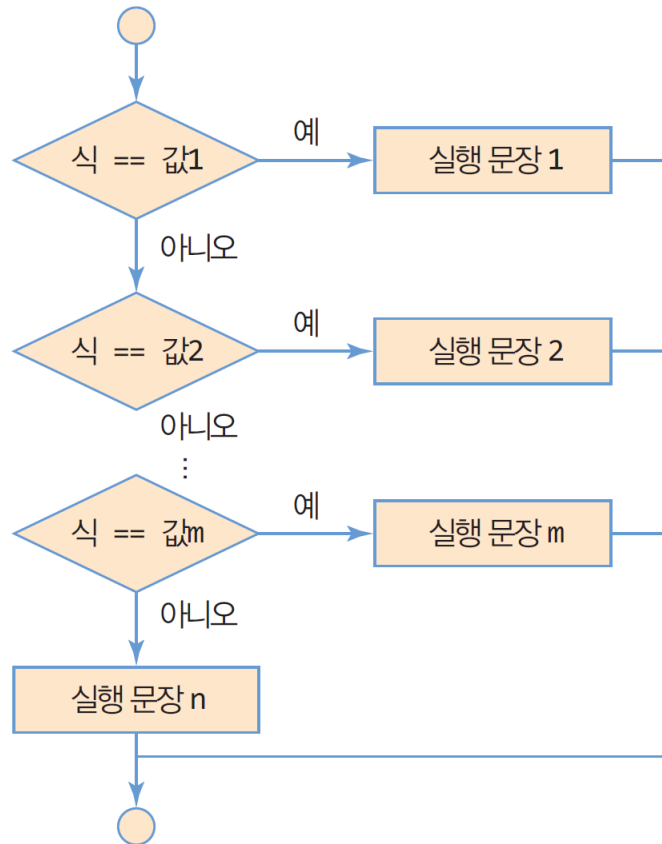
점수를 입력하세요(0~100): 65  
학년을 입력하세요(1~4): 4  
불합격!

# switch문

56

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

```
switch (식) {  
  case 값1:  
    실행 문장 1;  
    break;  
  case 값2:  
    실행 문장 2;  
    break;  
  ...  
  case 값m:  
    실행 문장 m;  
    break;  
  default:  
    실행 문장 n;  
}
```





# 예제 2-14 switch 문으로 학점 매기기

57

예제 2-12의 성적 매기는  
코드를 switch 문으로  
작성하라.

```
import java.util.Scanner;
public class GradingSwitch {
    public static void main (String[] args) {
        Scanner scanner = new Scanner(System.in);

        char grade;
        System.out.print("점수를 입력하세요(0~100): ");
        int score = scanner.nextInt();
        switch (score/10) {
            case 10: // score = 100
            case 9: // score는 90~99
                grade = 'A';
                break;
            case 8: // score는 80~89
                grade = 'B';
                break;
            case 7: // score는 70~79
                grade = 'C';
                break;
            case 6: // score는 60~69
                grade = 'D';
                break;
            default: // score는 59 이하
                grade = 'F';
        }
        System.out.println("학점은 "+grade+"입니다");
        scanner.close();
    }
}
```

점수를 입력하세요(0~100): 89  
학점은 B입니다.

# switch문에서 벗어나기

58

- switch문 내의 break문
  - ▣ break문 만나면 switch문 벗어남
  - ▣ case 문에 break문이 없다면, 다음 case문으로 실행 계속
    - 언젠가 break를 만날 때까지 계속 내려 가면서 실행

```
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.

# case 문의 값

59

- case 문의 값
  - ▣ 문자, 정수, 문자열 리터럴만 허용
  - ▣ 실수 리터럴은 허용되지 않음

```
int b;  
switch(b%2) {  
    case 1 : ...; break;  
    case 2 : ...; break;  
}
```

정수 리터럴  
사용 가능

```
char c;  
switch(c) {  
    case '+' : ...; break;  
    case '-' : ...; break;  
}
```

문자 리터럴  
사용 가능

```
String s = "예";  
switch(s) {  
    case "예" : ...; break;  
    case "아니요" : ...; break;  
}
```

문자열 리터럴  
사용 가능

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

오류

## 예제 2-15 : switch 문 활용

60

switch 문을 이용하여 커피 메뉴의 가격을 알려주는 프로그램을 작성하라.  
에스프레소, 카푸치노, 카페라떼는 3500원이고, 아메리카노는 2000원이다.

```
import java.util.Scanner;
public class CoffeePrice {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("무슨 커피 드릴까요? ");
        String order = scanner.next();
        int price=0;
        switch (order) {
            case "에스프레소":
            case "카푸치노":
            case "카페라떼":
                price = 3500;
                break;
            case "아메리카노" :
                price = 2000;
                break;
            default:
                System.out.println("메뉴에 없습니다!");
        }
        if(price != 0)
            System.out.print(order + "는 " + price + "원입니다");
        scanner.close();
    }
}
```

무슨 커피 드릴까요? 에스프레소  
에스프레소는 3500원입니다