

02

## 학습 목표

1. 자바의 프로그램의 기본 구조 이해
2. 자바의 데이터 타입 이해
3. 자바에서 키 입력 받는 방법 이해
4. 자바의 연산자 이해
5. 자바의 조건문(if-else와 switch) 이해

## 예제 2-1 : Hello, 자바 프로그램의 기본 구조

3

```
/*  
* 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 설명

4

## □ 클래스 만들기

- Hello 이름의 클래스 선언

```
public class Hello {  
}
```

- class 키워드로 클래스 선언
- public으로 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

```
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)

5

## □ 변수 선언

- 변수 타입과 변수 이름 선언

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

- 메소드 내에서 선언된 변수는 지역 변수

- 지역 변수는 메소드 실행이 끝나면 저장 공간 반환

## □ 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

# 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

- int **barChart**; 와 int **barchart**;는 서로 다른 식별자 선언

# 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
```

# 자바의 데이터 타입

8

## □ 자바의 데이터 타입

### ▣ 기본 타입 (Primitive Type) : 8 개

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

레퍼런스는 C/C++의 포인터와 유사한 개념  
그러나 메모리 주소는 아님

### ▣ 레퍼런스 타입 (Reference Type) : 1 개이며 용도는 다음 3 가지

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

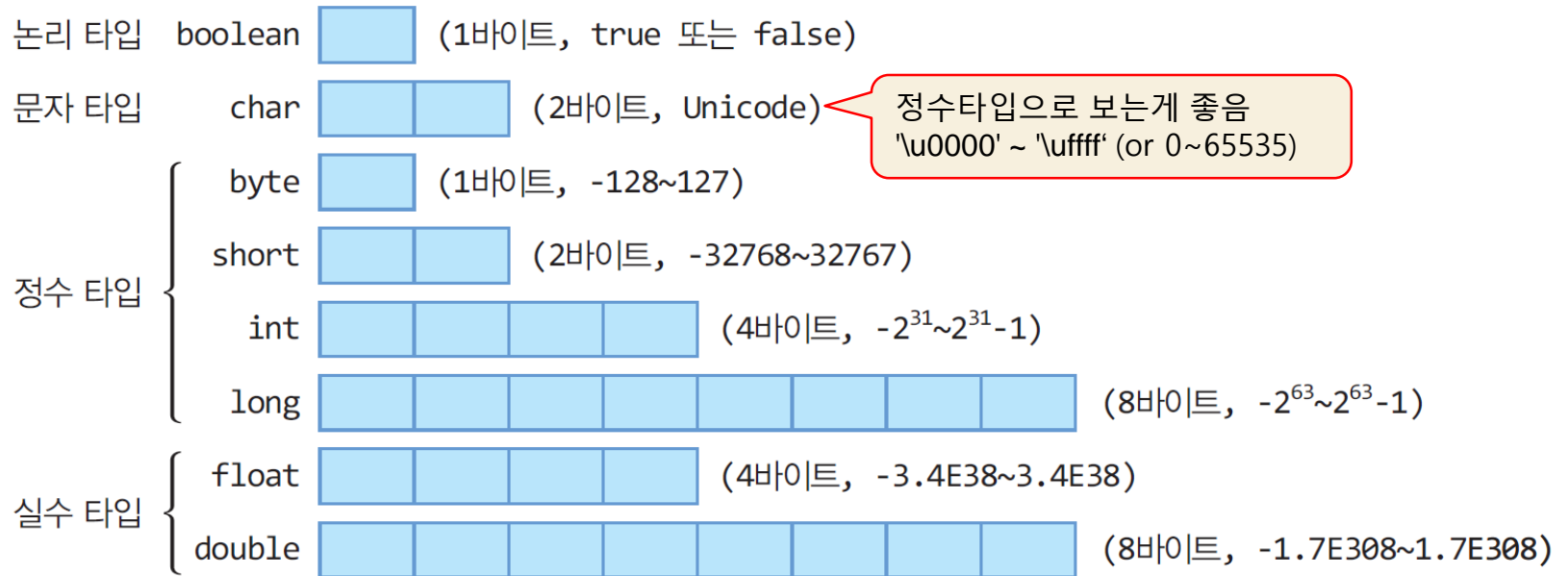


# 자바의 기본 타입

9

## □ 특징

- 기본 타입의 크기는 CPU나 운영체제에 따라 변하지 않음



Java SE 8 이후부터 unsigned int, long을 처리할 수 있는 방법 제공

# 문자열

10

- 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

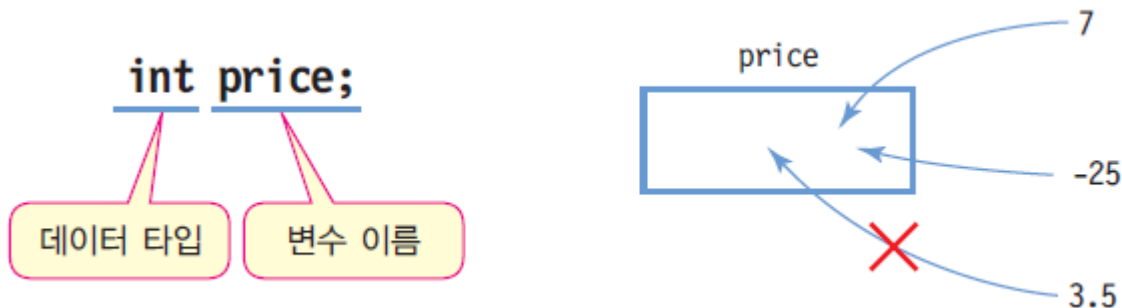
11

## □ 변수

- ▣ 프로그램 실행 중에 값을 임시 저장하기 위한 공간
  - 변수 값은 프로그램 수행 중 변경될 수 있음

## □ 변수 선언

- ▣ 데이터 타입에서 정한 크기의 메모리 할당



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

# 변수와 선언 (Advanced)

12

## □ 변수의 종류

### ▣ 클래스에 선언된 필드

- Instance Variable (Non-Static Field): 객체의 속성을 나타내는데 활용
- Class Variable (Static Field): 클래스의 속성을 표현 (여러 객체를 생성해도 동일한 값을 공유)

### ▣ 지역 변수

- 변수가 선언된 위치에서만 임시로 사용하는 변수 (즉, 중괄호 {} 내부)
- \*파라미터: 메소드를 통해 전달받는 값을 저장하는 지역변수

## □ 변수의 초기화

### ▣ 필드는 초기값을 지정하지 않으면 default 값으로 초기화 됨

- 정수나 실수타입: 0, 0L, 0.0f, 0.0d, 'Wu0000'
- boolean 타입: false
- 참조타입(객체): null

### ▣ 지역 변수는 자동으로 초기화 되지 않음

### ▣ 초기화 되지 않은 지역 변수를 사용하면 오류 발생

# 리터럴과 정수 리터럴

13

- 리터럴(literal)
  - ▣ 고정 값(fixed value)을 소스 코드에서 표현한 것
  - ▣ 정수, 실수, 문자, 논리, 문자열 리터럴 있음
- 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

14

- ▣ 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
```

```
//Java SE 7 이후부터 숫자 리터럴에 밑줄('_') 사용가능  
long creditCardNumber = 1234_5678_9012_3456L;  
float pi = 3.14_15F;
```

# 문자 리터럴

15

## ▣ 단일 인용부호(' ')로 문자 표현

- 'a', 'W', '가', '\*', '3', '7'

```
char a = 'W';  
char b = '글';  
char c = ₩uae00; // '글'의 유니코드 값(ae00) 사용
```

- ₩u다음에 4자리 16진수로, 2 바이트의 유니코드(Unicode)

## ▣ 특수문자 리터럴은 백슬래시(₩)로 시작


특수문자 리터럴	의미	특수문자 리터럴	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 타입 리터럴

16

## □ 논리 값 표시

- true 또는 false 뿐
- 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  
...  
}
```



# Tip: 기본 타입 이외 리터럴

17

## □ null 리터럴

- ▣ 레퍼런스에 대입 사용



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

## □ 문자열 리터럴(스트링 리터럴)

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

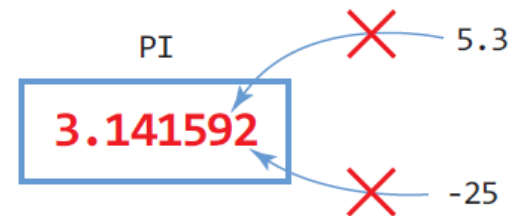
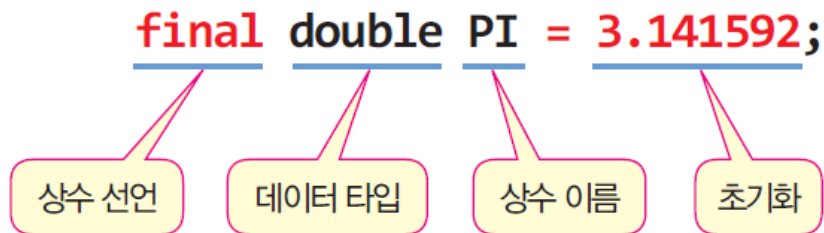
```
String str = "Good";
```

# 상수

18

## □ 상수 선언

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



## □ 상수 선언 사례

```
final int LENGTH = 20;
```

```
static final double PI = 3.141592; // static으로 선언하는 것이 바람직(5장 참조)
```

## 예제 2-2 : 리터럴, 상수 사용하기

19

원의 면적을 구하는 프로그램을 작성하라.

```
public class CircleArea {  
    public static void main(String[] args) {  
        final double PI = 3.14;           // 원주율을 상수로 선언  
        double radius = 10.2;           // 원의 반지름  
        double circleArea = radius*radius*PI; // 원의 면적 계산  
  
        // 원의 면적을 화면에 출력한다.  
        System.out.print("반지름 " + radius + ", ");  
        System.out.println("원의 면적 = " + circleArea); }  
}
```

반지름 10.2, 원의 면적 = 326.68559999999997

# 타입 변환과 자동 타입 변환

20

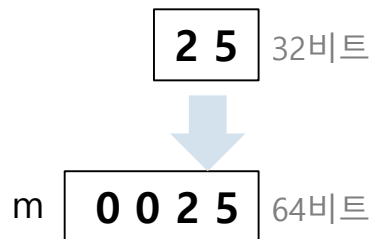
## □ 타입 변환

- ▣ 한 타입의 값을 다른 타입의 값으로 변환

## □ 자동 타입 변환

- ▣ 컴파일러에 의해 원래의 타입보다 큰 타입으로 자동 변환
- ▣ 치환문(=)이나 수식 내에서 타입이 일치하지 않을 때

```
long m = 25; // 25는 int 타입 25가 long 타입으로 자동 변환  
double d = 3.14 * 10; // 실수 연산 위해 10이 10.0으로 자동 변환
```



# 강제 타입 변환

21

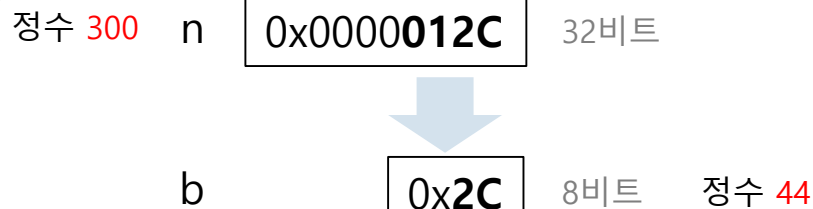
- 강제 타입 변환
  - ▣ 개발자의 의도적 타입 변환
  - ▣ () 안에 개발자가 명시적으로 타입 변환 지정

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

`byte b = (byte)n;` 로 수정

- ▣ 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

## 예제 2-3 : 타입 변환

22

자동 타입 변환과 강제 타입 변환이 들어 있는 코드이다.  
실행 결과는 무엇인가?

```
public class TypeConversion {  
    public static void main(String[] args) {  
        byte b = 127;  
        int i = 100;  
  
        System.out.println(b+i);           // b가 int 타입으로 자동 변환  
        System.out.println(10/4);  
        System.out.println(10.0/4);        // 4가 4.0으로 자동 변환  
        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

# 자바의 키 입력과 System.in

23

## □ System.in

- ▣ 키보드와 연결된 자바의 표준 입력 스트림
- ▣ 입력되는 키를 바이트(문자 아님)로 리턴하는 저수준 스트림
- ▣ System.in을 직접 사용하면 바이트를 문자나 숫자로 변환하는 많은 어려움 있음

# Scanner와 Scanner 객체 생성

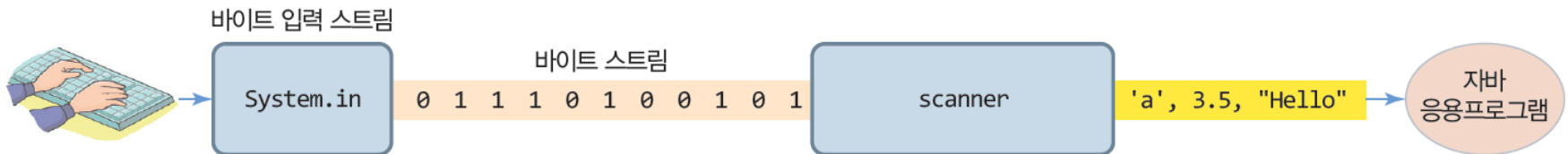
24

## □ Scanner 클래스

- ▣ 읽은 바이트를 문자, 정수, 실수, 불린, 문자열 등 다양한 타입으로 변환하여 리턴
  - java.util.Scanner
- ▣ 객체 생성

```
import java.util.Scanner; // 임포트 문 필요  
...  
Scanner a = new Scanner(System.in); // Scanner 객체 생성
```

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

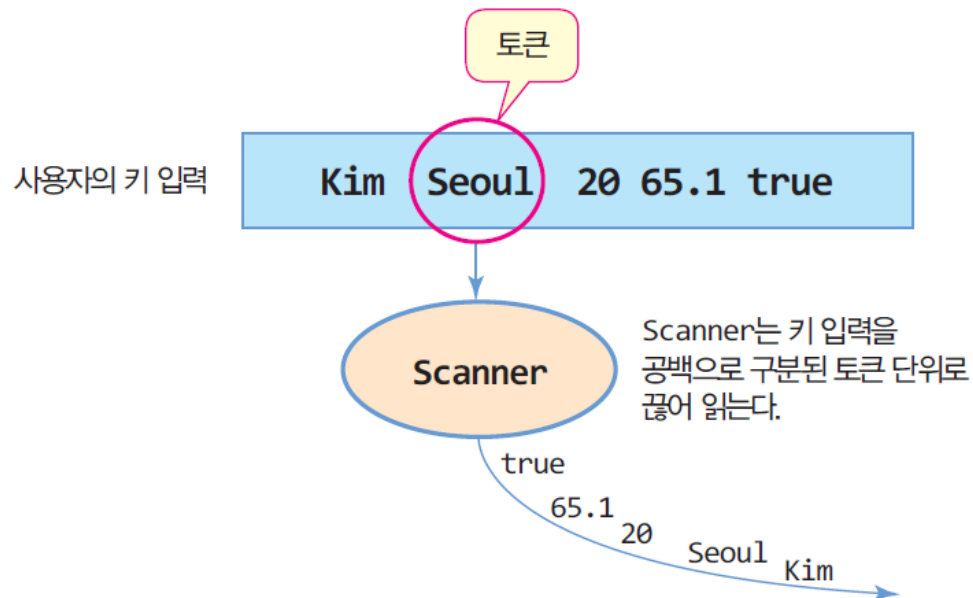




# Scanner를 이용한 키 입력

25

- Scanner에서 키 입력 받기
  - Scanner는 입력되는 키 값을 공백으로 구분되는 토큰 단위로 읽음
  - 공백 문자 : 'wt', 'wf', 'wr', ' ', 'wn'
- 개발자가 원하는 타입 값으로 쉽게 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

26

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>String nextLine()</code>	'\n'을 포함하는 한 라인을 읽고 '\n'을 버린 나머지만 리턴
<code>void close()</code>	Scanner의 사용 종료
<code>boolean hasNext()</code>	현재 입력된 토큰이 있으면 true, 아니면 새로운 입력이 들어올 때까지 무한정 기다려서, 새로운 입력이 들어오면 그 때 true 리턴. ctrl-z 키가 입력되면 입력 끝이므로 false 리턴

## 예제 2-4 : Scanner를 이용한 키 입력 연습

27

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();
        System.out.println("당신의 이름은 " + name + "입니다.");
        String city = scanner.next();
        System.out.println("당신이 사는 도시는 " + city + "입니다.");
        int age = scanner.nextInt();
        System.out.println("당신의 나이는 " + age + "살입니다.");
        double weight = scanner.nextDouble(); // 실수 토큰 읽기
        System.out.println("당신의 체중은 " + weight + "kg입니다.");
        boolean single = scanner.nextBoolean();
        System.out.println("당신은 독신 여부는 " + single + "입니다.");

        scanner.close();
    }
}
```

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

Kim Seoul 20 65.1 true

당신의 이름은 Kim입니다.

당신이 사는 도시는 Seoul입니다.

당신의 나이는 20살입니다.

당신의 체중은 65.1kg입니다.

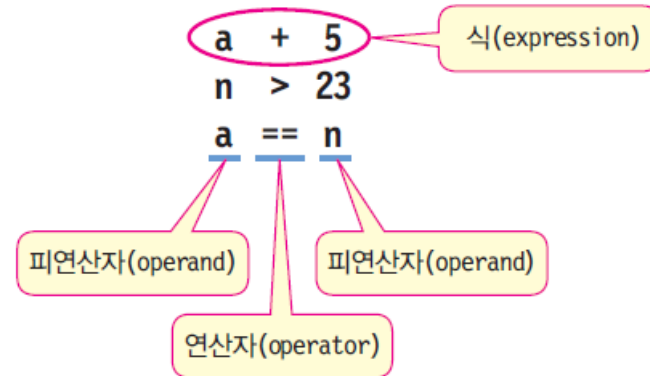
당신은 독신 여부는 true입니다.

# 식과 연산자

28

## □ 연산

- ▣ 주어진 식을 계산하여 결과를 얻어내는 과정



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

# 산술 연산자

29

## □ 산술 연산자

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

- 10의 자리와 1의 자리 분리

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

- x가 홀수인지 판단

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

- n의 값이 3의 배수인지 확인

```
int s = n % 3;      // s가 0이면 n은 3의 배수
```

## 예제 2-5 : /와 % 산술 연산자 응용

30

초 단위의 정수를 입력 받고, 몇 시간, 몇 분, 몇 초인지 구하여 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:4000  
4000초는 1시간, 6분, 40초입니다.

# 비트 연산자

31

- 피 연산자의 각 비트들을 대상으로 하는 연산

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

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

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

두 비트가 같으므로  
결과는 0

두 비트가 다르므로  
결과는 1

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

1은 0으로 변환

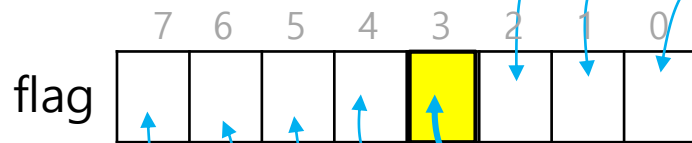
0은 1로 변환

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

# 비트 연산 응용(옵션)

32

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 코드는?

```
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");
```

	x	x	x	x	Y	x	x	x
&	0	0	0	0	1	0	0	0
<hr/>								
	0	0	0	0	Y	0	0	0

Y비트가 0 이면  
& 결과는 0

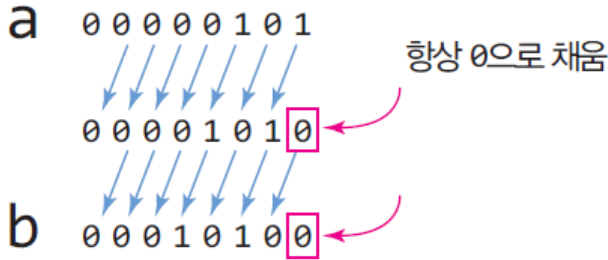
온도는 0도 이상



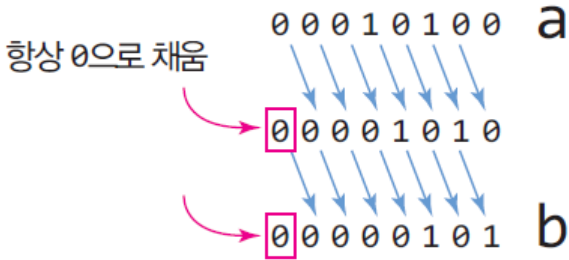
# 시프트 연산자

시프트 연산자	내용
$a \gg b$	a의 각 비트를 오른쪽으로 b번 시프트한다. 최상위 비트의 빈자리는 시프트 전의 최상위 비트로 다시 채운다. 산술적 오른쪽 시프트라고 한다.
$a \ggg b$	a의 각 비트를 오른쪽으로 b번 시프트한다. 그리고 최상위 비트의 빈자리는 0으로 채운다. 논리적 오른쪽 시프트라고 한다.
$a \ll b$	a의 각 비트를 왼쪽으로 b번 시프트한다. 그리고 최하위 비트의 빈자리는 0으로 채운다. 산술적 왼쪽 시프트라고 한다.

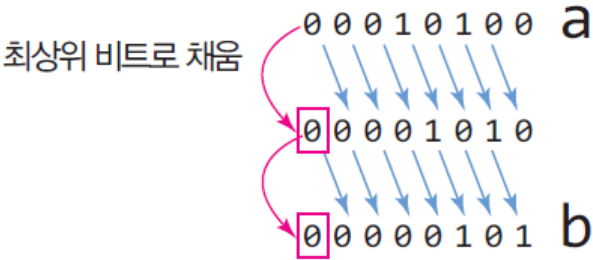
```
byte a = 5; // 5
byte b = (byte)(a << 2); // 20
```



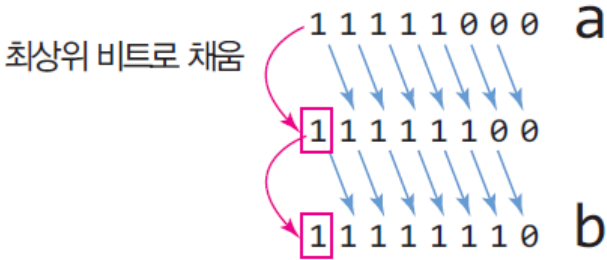
```
byte a = 20; // 20
byte b = (byte)(a >>> 2); // 5
```



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



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



# 예제 2-6 : 비트 연산자와 시프트 연산자 사용하기

34

다음 코드의 실행 결과는 무엇인가?

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

        // 비트 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

        // 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. 양수이므로 0 삽입
        System.out.println(d >> 2); // d를 2비트 오른쪽 시프트. 음수이므로 1 삽입
        System.out.printf("%x\n", (d >>> 2)); // d를 2비트 오른쪽 시프트. 0 삽입
    }
}
```

printf("%04", ...) 메소드는 값을 4자리의 16진수로 출력하고 빈 곳에는 0을 삽입한다.

c에 4를 곱한 결과가 나타난다

4로 나누기 효과

d가 음수이어도 0 삽입.  
4로 나누기 효과 없음

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

# 비교연산자, 논리연산자

35

- 비교연산자 : 두 개의 값을 비교하여 true/false 결과
- 논리연산자 : 두 개의 논리 값에 논리 연산. 논리 결과

비교 연산자	내용
a < b	a가 b보다 작으면 true
a > b	a가 b보다 크면 true
a <= b	a가 b보다 작거나 같으면 true
a >= b	a가 b보다 크거나 같으면 true
a == b	a가 b와 같으면 true
a != b	a가 b와 같지 않으면 true

논리 연산자	내용
! a	a가 true이면 false, false이면 true
a ^ b	a와 b의 XOR 연산. a, b가 같으면 false
a    b	a와 b의 OR 연산. a와 b 모두 false인 경우만 false
a && b	a와 b의 AND 연산. a와 b 모두 true인 경우만 true

```
(age >= 20) && (age < 30) // 나이(int age)가 20대인 경우
(c >= 'A') && (c <= 'Z') // 문자(char c)가 대문자인 경우
(x>=0) && (y>=0) && (x<=50) && (y<=50) // (x,y)가 (0,0)과 (50,50)의 사각형 내에 있음
```



20 <= age < 30 // 조건식 문법 오류

## 예제 2-7 : 비교 연산자와 논리 연산자 사용하기

36

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  
true  
false  
false  
true  
true

# 대입 연산자, 증감 연산자

37

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

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

```
int a, b = 4;  
a = b++; // b가 a에 대입된 후 b 증가. 결과 a=4, b=5
```

```
int a, b = 4;  
a = ++b; // b가 증가한 후, b 값이 a에 대입. 실행 결과 a=5, b=5
```

## 예제 2-8 : 대입 연산자와 증감 연산자 사용하기

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

# 조건 연산자 ?:

39

## □ opr1?opr2:opr3

- 3 개의 피연산자로 구성된 삼항(ternary) 연산자
  - opr1이 true이면, 연산식의 결과는 opr2, false이면 opr3
- if-else를 조건연산자로 간결하게 표현 가능

x와 y 중에서  
큰 값을 big에 저장

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

```
int big;  
if(x>y)  
    big = x;  
else  
    big = y;
```

```
int big = (x>y)?x:y;
```

## 예제 2-9 : 조건 연산자 사용하기

40

다음 코드의 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 – 단순 if 문, if-else 문

41

## □ 단순 if 문

### ▣ if의 괄호 안에 조건식(논리형 변수나 논리 연산)

- 실행문장이 단일 문장인 경우 둘러싸는 {, } 생략 가능

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

```
if(n%2 == 0) {  
    System.out.print(n);  
    System.out.println("은 짝수입니다.");  
}  
if(score >= 80 && score <= 89)  
    System.out.println("학점은 B입니다.");
```

## □ if-else 문

### ▣ 조건식이 true면 실행문장1, false이면 실행문장2 실행

```
if(조건식) {  
    ... 실행문장 1 ... // 조건식이 참인 경우  
}  
else {  
    ... 실행문장 2 ... // 조건식이 거짓인 경우  
}
```

```
if(score >= 90)  
    System.out.println("합격입니다.");  
else  
    System.out.println("불합격입니다.");
```

## 예제 2-10 : if-else 사용하기

42

나이를 입력 받아 20대인지 판별하는 프로그램을 작성하라

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

        System.out.print("나이를 입력하시오:");
        int age = scanner.nextInt();
        if((age>=20) && (age<30)) { // age가 20~29 사이인지 검사
            System.out.print("20대입니다. ");
            System.out.println("20대라서 행복합니다!");
        }
        else
            System.out.println("20대가 아닙니다.");

        scanner.close();
    }
}
```

나이를 입력하시오:23  
20대입니다. 20대라서 행복합니다!

# 다중 if-else 문

43

## □ 다중 if문

- ▣ 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  
}
```

실행 문장이 실행된 후 맨 아래  
else 코드 밑으로 벗어남

```
if(score >= 90) { // score가 90 이상  
    grade = 'A';  
}  
else if(score >= 80) { // 80 이상 90 미만  
    grade = 'B';  
}  
else if(score >= 70) { // 70 이상 80 미만  
    grade = 'C';  
}  
else if(score >= 60) { // 60 이상 70 미만  
    grade = 'D';  
}  
else { // 60 미만  
    grade = 'F';  
}
```

## 예제 2-11 : 다중 if-else를 이용하여 학점 매기기

44

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

점수를 입력하세요(0~100):89  
학점은 B입니다.

```
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();
    }
}
```

# 중첩 if-else문, 예제 2-12 중첩 if-else문 사례

45

- if 문이나 else 문, 혹은 if-else 문에 if문이나 if-else문을 내포할 수 있다.

점수와 학년을 입력 받아  
60점 이상이면 합격,  
미만이면 불합격을 출력하라.  
다만 4학년은 70점 이상이어야  
합격이다. 60점 이상인 경우  
4학년을 구분하여  
처리해야 한다.

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

```
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();
    }
}
```

# switch문

46

- 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: // 식의 결과가 값1과 같을 때  
        실행 문장 1;  
        break;  
    case 값2: // 식의 결과가 값2와 같을 때  
        실행 문장 2;  
        break;  
    ...  
    case 값m:  
        실행 문장 m; // 식의 결과가 값m과 같을 때  
        break;  
    default: // 어느 것과도 같지 않을 때  
        실행 문장 n;  
}
```

```
char grade='B';  
switch(grade) {  
    case 'A':  
        System.out.println("축하합니다.");  
        System.out.println("잘했습니다.");  
        break;  
    case 'B':  
        System.out.println("좋아요.");  
        break;  
    case 'C':  
        System.out.println("노력하세요.");  
        break;  
    default:  
        System.out.println("탈락입니다!");  
}
```

# switch문에서 break문의 역할

47

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

break; 문을  
생략한다면

```
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

48

- case 문의 값
  - ▣ 문자, 정수, 문자열 리터럴(JDK 1.7부터)만 허용
  - ▣ 실수 리터럴은 허용되지 않음

```
int b;  
switch(c%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 2-13 : switch 문 사용하기

49

1~12 사이의 월을 입력 받아 봄, 여름, 가을, 겨울을 판단하여 출력하라.

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

        System.out.print("월(1~12)을 입력하시오:");
        int month = scanner.nextInt(); // 정수로 월 입력
        switch(month) {
            case 3:
            case 4:
            case 5:
                System.out.println("봄입니다.");
                break;
            case 6: case 7: case 8:
                System.out.println("여름입니다.");
                break;
            case 9: case 10: case 11:
                System.out.println("가을입니다.");
                break;
            case 12: case 1: case 2:
                System.out.println("겨울입니다."); break;
            default:
                System.out.println("잘못된 입력입니다.");
        }
        scanner.close();
    }
}
```

월(1~12)을 입력하시오:3  
봄입니다.