

Java Basic Grammar

Byeongjoon Noh

Dept. of AI/Bigdata, SCH Univ.

powernoh@sch.ac.kr

Contents

1. Variable
 1. Data type
 2. Literal & Casting
 3. Key input
2. Operations
 1. 대입, 부호, 산술, 복합대입, 증감 연산자
 2. 관계, 논리 연산자
 3. 조건, 비트 연산자

Note

문장

- ; (세미콜론)으로 한 문장의 끝을 인식

화면 출력

- 표준 출력 스트림에 메시지 출력

1. Variable

Variable

Declaration & Assignment

- Example

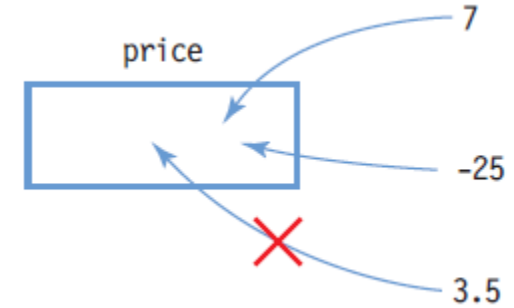
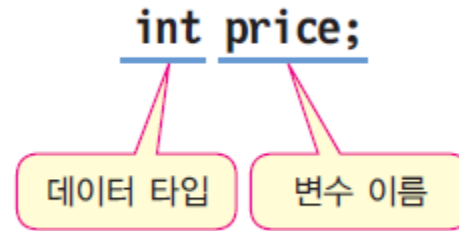
```
int radius;  
char c1, c2, c3; // 3 개의 변수를 한 번에 선언한다.  
double weight;
```

- 변수의 선언과 동시에 초기값 지정

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

- 값 대입

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



Identifier

사용가능 한 예

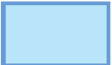

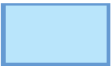





```
int    name;
char   student_ID;           // '_' 사용 가능
void   $func() { }           // '$' 사용 가능
class  Monster3 { }          // 숫자 사용 가능
int    whatsyournamemynameiskitae; // 길이 제한 없음
int    barChart;  int barchart; // 대소문자 구분. barChart와 barchart는 다름
int    가격;                 // 한글 이름 사용 가능
```

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
```

Data type

기본 타입 (8개)

논리 타입	boolean		(1바이트, true 또는 false)
문자 타입	char		(2바이트, Unicode)
정수 타입	byte		(1바이트, -128~127)
	short		(2바이트, -32768~32767)
	int		(4바이트, $-2^{31} \sim 2^{31}-1$)
	long		(8바이트, $-2^{63} \sim 2^{63}-1$)
실수 타입	float		(4바이트, $-3.4E38 \sim 3.4E38$)
	double		(8바이트, $-1.7E308 \sim 1.7E308$)

레퍼런스 타입

- Class
- Interface
- Array

String (문자열)

String class로 문자열 표현

- 문자열 리터럴 = “JDK”, “Korean”, “Hello World!!”

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

문자열이 섞인 + 연산

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


Literal

리터럴

- 프로그램에서 직접 표현한 값
- 정수, 실수, 문자, 논리, 문자열 리터럴

정수 리터럴

- 10진수, 8진수, 16진수, 2진수 리터럴
- 정수 리터럴 → int 형으로 컴파일
- Long 타입 리터럴 → 숫자 뒤에 L 또는 l 을 붙여 표시

15	-> 10진수 리터럴 15
015	-> 0으로 시작하면 8진수. 십진수로 13
0x15	-> 0x로 시작하면 16진수. 십진수로 21

```
int n = 15;  
int m = 015;  
int k = 0x15;  
int b = 0b0101;  
long g = 24L;
```

Literal

실수 리터럴

- 소수점 형태나 지수 형태로 표현한 실수
 - 12. 12.0 .1234 0.1234 1234E-4
- 실수 타입 리터럴 ➔ 기본이 double 타입으로 컴파일

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

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

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

Literal

문자 리터럴

- 단일 인용부호 (' ')로 문자 표현
 - 'a', 'W', '가', '*', '13'

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


- \u → Unicode로 표현
- 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)

Literal

논리 타입 리터럴

- 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  
...  
}
```

Literal

기본 타입 이외의 리터럴

- null 리터럴
 - 레퍼런스에 대입 사용



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

- 문자열 리터럴
 - 이중 인용부호 (“ ”)로 묶어 표현
 - “Good”, “Morning”, “Hello World!!!”, “3.19”, “a”
 - 문자열 리터럴은 String 객체에서 자동으로 처리됨

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

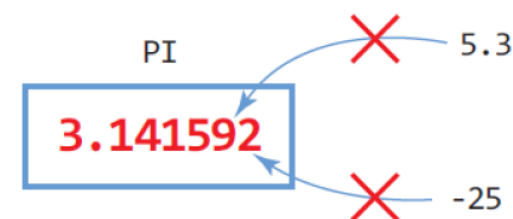
Constant

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



- 상수 선언 사례

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

```
static final double PI = 3.141592; // static으로 선언하는 것이 바람직
```

Example

Q. 원의 면적을 구하는 프로그램을 작성해보세요.

```
public class CircleArea {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        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("Radius = " + radius + ", ");  
        System.out.println("Area of Circle = " + circleArea);  
    }  
}
```

Radius = 10.2, Area of Circle = 326.68559999999997

Casting (자료형 변환)

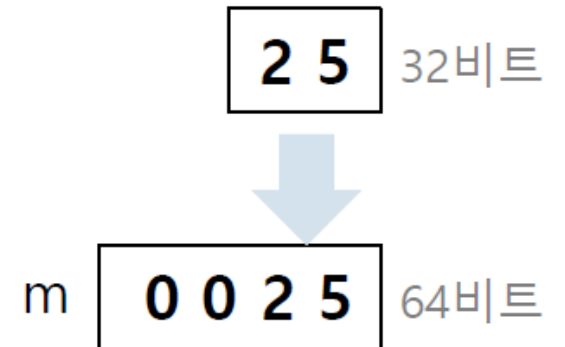
타입 변환

- 한 타입의 값을 다른 타입의 값으로 변환

자동 타입 변환

- 컴파일러에 의해 원래의 타입보다 큰 타입으로 자동 변환
- 할당(=)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
```



Casting (자료형 변환)

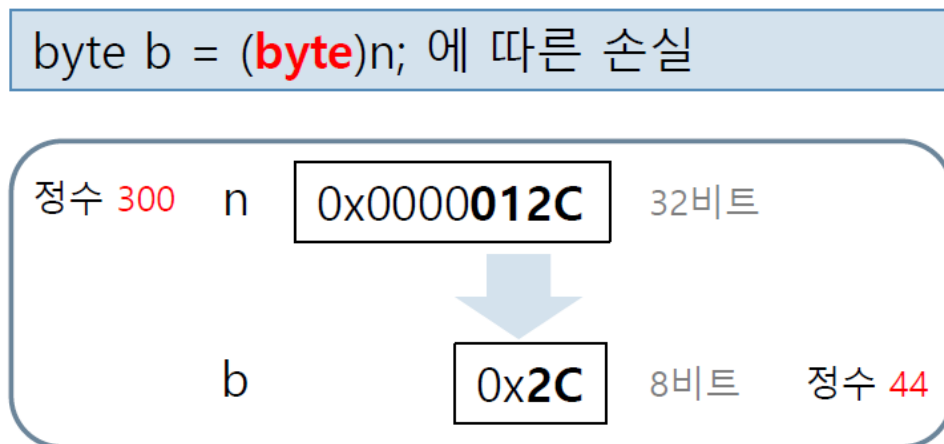
강제 타입 변환

- 개발자의 의도적 자료형 변환
- () 안에 명시적으로 자료형 변환 지정
- 강제 변환은 값 손실/변경 우려

오류

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

byte b = **(byte)**n; 로 수정



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

강제 타입 변환으로
소숫점 이하 0.9 손실

Example

자동 형 변환과 강제 형 변환 예시 및 결과 추측

```
public class TypeConversion {  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        byte b = 127;  
        int i = 100;  
        System.out.println(b+i); // b가 int타입으로 자동 변환 227  
        System.out.println(10/4); 2  
        System.out.println(10.0/4); // 4가 4.0으로 자동 변환 2.5  
        System.out.println((char)0x12340041); A  
        System.out.println((byte)(b+i)); -29  
        System.out.println((int)2.9 + 1.8); 3.8  
        System.out.println((int)(2.9 + 1.8)); 4  
        System.out.println((int)2.9 + (int)1.8); 3  
    }  
}
```

키 입력과 System.in

System.in

- 키보드와 연결된 자바의 표준 입력 스트림
- 입력되는 키를 바이트로 리턴하는 저수준 스트림
- System.in을 직접 사용하면 바이트를 문자나 숫자로 변환해줘야함 ➔ 어려움

Scanner와 Scanner 객체

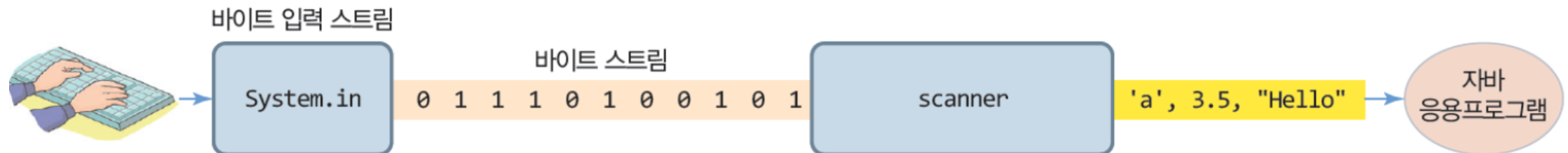
Scanner 클래스

- 읽은 바이트를 문자, 정수, 실수, Boolean, String 등 다양한 타입으로 변환하여 리턴
 - java.util.Scanner

- Scanner 객체 생성

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

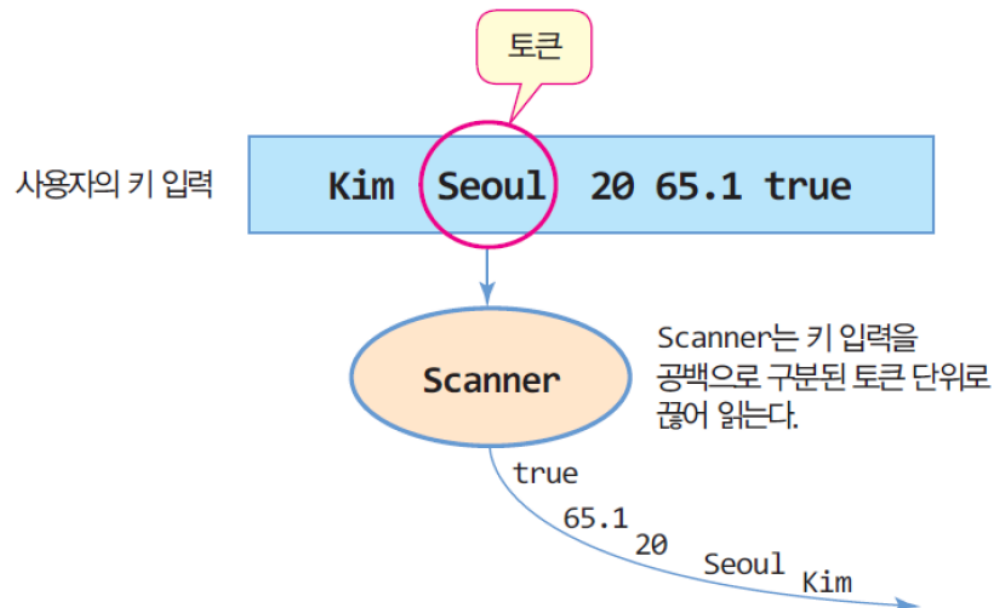
- 키보드에 연결된 System.in에게 키를 읽게하고, 원하는 타입으로 변환하여 리턴



Scanner를 이용한 키 입력

Scanner에서 키 입력 받기

- Scanner는 입력되는 키 값을 공백으로 구분되는 token 단위로 읽음
- 공백 문자: '\t' '\f' '\r' '\n'
- 개발자가 원하는 데이터 형으로 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

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

Example

Q. 이름, 도시, 나이, 체중, 독신 여부를 입력받고 다시 출력하는 프로그램을 작성하세요.

```
import java.util.Scanner;
```

```
public class ScannerEx{  
    public static void main(String args[]) {  
        System.out.println("Please enter name, city, age, weight, single information with space");  
  
        Scanner scanner= new Scanner(System.in);  
        String name = scanner.next();  
  
        String city = scanner.next();  
  
        int age = scanner.nextInt();  
  
        double weight = scanner.nextDouble();  
  
        boolean single = scanner.nextBoolean();  
        scanner.close();  
  
        System.out.println("Name: " + name);  
        System.out.println("City: " + city);  
        System.out.println("Age: " + age);  
        System.out.println("Weight: " + weight);  
        System.out.println("Single: " + single);  
    }  
}
```

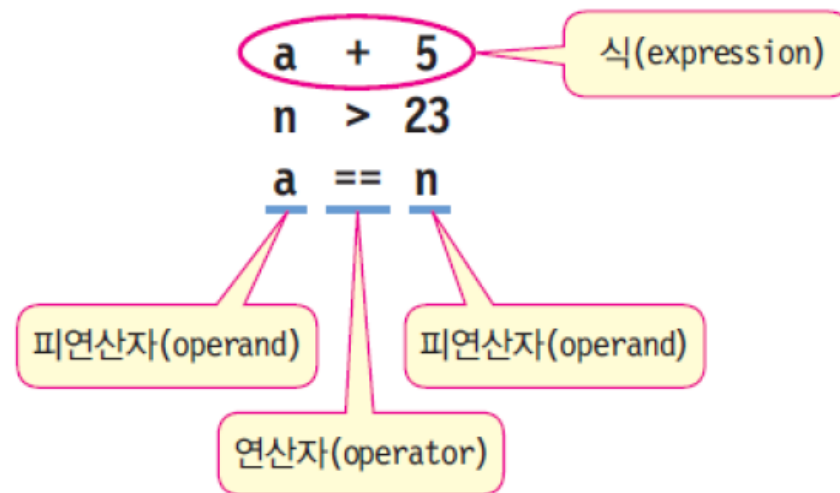
```
Please enter name, city, age, weight, single information with space  
ByeongjoonNoh  
Sejong  
30  
98.9  
true  
Name: ByeongjoonNoh  
City: Sejong  
Age: 30  
Weight: 98.9  
Single: true
```

2. Operations

식과 연산자

연산

- 주어진 식을 계산하여 결과를 얻어내는 과정



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

산술 연산자

더하기, 빼기, 곱하기, 나누기, 나머지

- $+$, $-$, $*$, $/$, $\%$
- $/$ 와 $\%$ 응용
 - 10의 자리와 1의 자리 분리

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

- x 가 홀수인지 판단

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

- n 의 값이 3의 배수인지 확인

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

Example

Q. 초 단위의 정수를 입력 받고, 몇 시간, 몇 분, 몇 초인지 구하여 출력하는 프로그램을 작성하세요.

```
import java.util.Scanner;

public class ArithmeticOperator{
    public static void main(String[] args) {

        Scanner scanner= new Scanner(System.in);
        System.out.print("Enter the integer value: ");

        int time = scanner.nextInt(); // 정수 입력

        int second = time % 60; // 60으로나눈나머지는초
        int minute = (time / 60) % 60; // 60으로나눈몫을다시60으로나눈나머지는분
        int hour = (time / 60) / 60; // 60으로나눈몫을다시60으로나눈몫은시간

        System.out.print(time + " second is ");
        System.out.print(hour + "hour, ");
        System.out.print(minute + "min, ");
        System.out.println(second + "sec");

        scanner.close();
    }
}
```

Enter the integer value: 4000
4000 second is 1hour, 6min, 40sec

비트 연산자

피 연산자의 각 비트를 대상으로 하는 연산

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.
$a \wedge b$	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 연산자

비트 연산자 예시

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

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 같으므로
결과는 0

두 비트가 다르므로
결과는 1

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

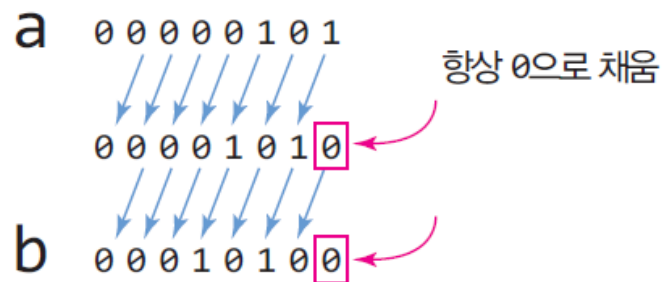
1은 0으로 변환

0은 1로 변환

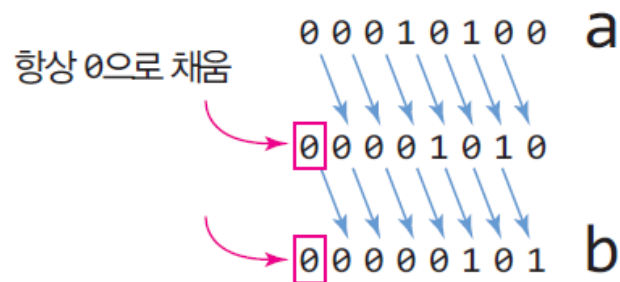
시프트 연산자

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.

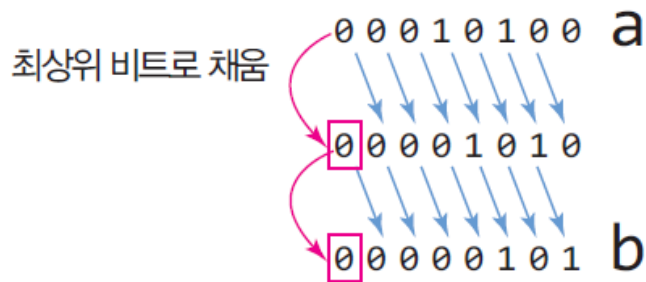
```
byte a = 5; // 5
byte b = (byte)(a << 2); // 20
```



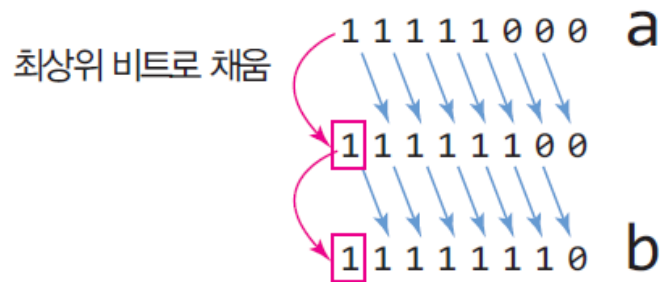
```
byte a = 20; // 20
byte b = (byte)(a >>> 2); // 5
```



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



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



Example

Q. 다음 코드의 실행 결과는?

```
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

비교 연산자, 논리 연산자

비교 연산자

- 두 개의 값을 비교하여 true/false 결과

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

논리 연산자

- 두 개의 논리 값의 논리 연산 (논리 결과)

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

비교 연산자, 논리 연산자

Java에서 비교/논리 식 표현 방법

- $20 \leq \text{age} < 30 \rightarrow (20 \leq \text{age}) \ \&\& \ (\text{age} < 30)$

```
(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
```

Example

Q. 다음 코드의 실행 결과는?

```
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

대입 연산자	내용	대입 연산자	내용
<code>a = b</code>	b의 값을 a에 대입	<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 *= 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 >>>= b</code>	<code>a = a >>> 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
```

Example

Q. 다음 코드의 실행 결과는?

```
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

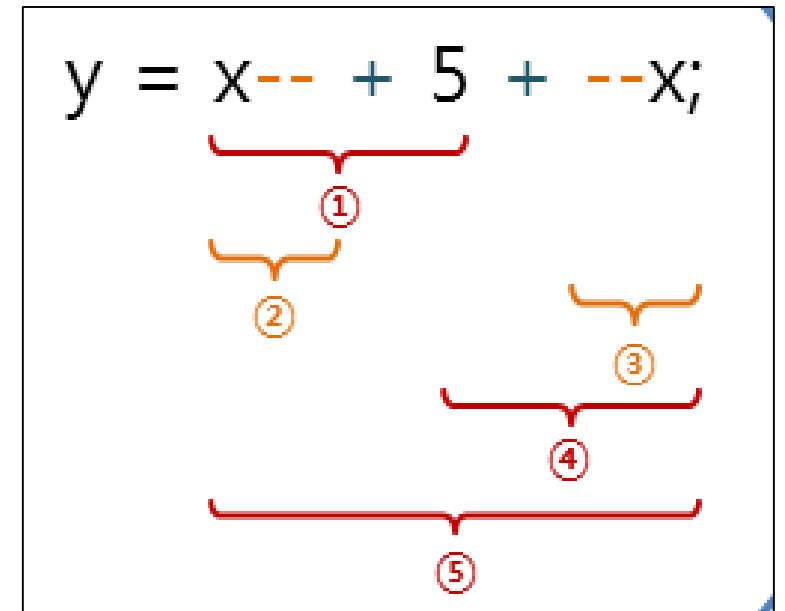
증감 연산자 (Hard)

전위/후위 연산이 단독으로 있을 때

- $a++$, $++a \rightarrow a$ 의 값이 1 증가 되는 효과 (주변에 영향X)

증감 연산자가 다른 연산과 함께 있을 때

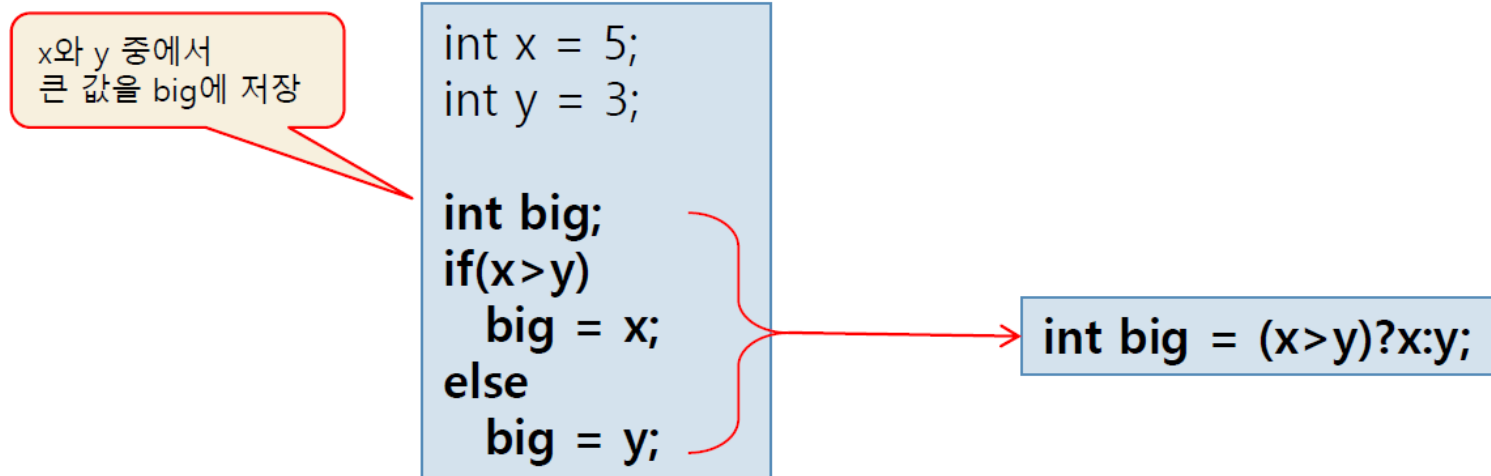
- `int x=3;`
- `y = x-- + 5 + --x;`
- `x, y = ?`



조건 연산자 (삼항 연산자)

삼항 연산자

- 세 개의 피연산자(operand)로 구성된 ternary 연산자
- (opr1)?(opr2):(opr3)
- opr1이 true이면, 연산식의 결과는 opr2, false 이면 opr3
- if-else를 조건 연산자로 간결하게 표현가능



Example

Q. 다음 코드의 실행 결과는?

```
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

조건 연산자 (if-else문)

단순 if문

- if의 괄호 안에 조건식 (논리형 변수나 논리 연산)
 - 실행 문장이 단일 문장인 경우 block {} 생략 가능

```
if(조건식) {  
    ... 실행 문장 ... // 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.");
```


조건 연산자 (if-else문)

if-else문

- 조건식이 true면 실행문자 1, false면 실행문장 2이 실행됨

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

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

Example

Q. 나이를 입력 받아 20대인지 판별하는 프로그램을 작성해보세요.

```
public class Twenties {  
    public static void main(String[] args) {  
        Scanner scanner= new Scanner(System.in);  
        System.out.print("Enter the age: ");  
        int age = scanner.nextInt();  
  
        if((age>=20) && (age<30)){  
            System.out.print("Yes");  
        }  
        else  
            System.out.println("No");  
  
        scanner.close();  
    }  
}
```

조건 연산자 (if-else문)

다중 if-else문

- 조건문이 무분별하게 많은 경우 → switch문 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';  
}
```

Example

Q. 다중 if-else문을 사용하여 입력받은 성적의 학점을 부여하는 프로그램을 작성하세요.

```
public class Grading {
    public static void main(String[] args) {
        char grade;
        Scanner scanner= new Scanner(System.in);
        System.out.print("Enter the score (0~100):");

        int score = scanner.nextInt(); // 점수읽기

        if(score >= 90) // score가90 이상
            grade = 'A';
        else if(score >= 80) // score가80 이상90 미만
            grade = 'B';
        else if(score >= 70) // score가70 이상80 미만
            grade = 'C';
        else if(score >= 60) // score가60 이상70 미만
            grade = 'D';
        else // score가60 미만
            grade = 'F';

        System.out.println("Grade is " + grade);

        scanner.close();
    }
}
```

조건 연산자 (if-else문)

중첩 if-else문

- if/else문 혹은 if-else문 내에 if-else문을 내포할 수 있음
- Ex) 점수/학년 입력받은 후 60점 이상이면 합격, 미만이면 불합격을 출력하라.
다만, 4학년은 70점 이상이어야 합격이다.

```
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();
```

switch-case

switch문은 식과 case 문의 값을 비교

- case의 비교값과 일치하면 해당 case의 실행문장 수행
 - “break”를 만나면 switch문을 탈출
- case의 비교값과 일치하는 것이 없으면 default문 실행 (default문은 생략가능)

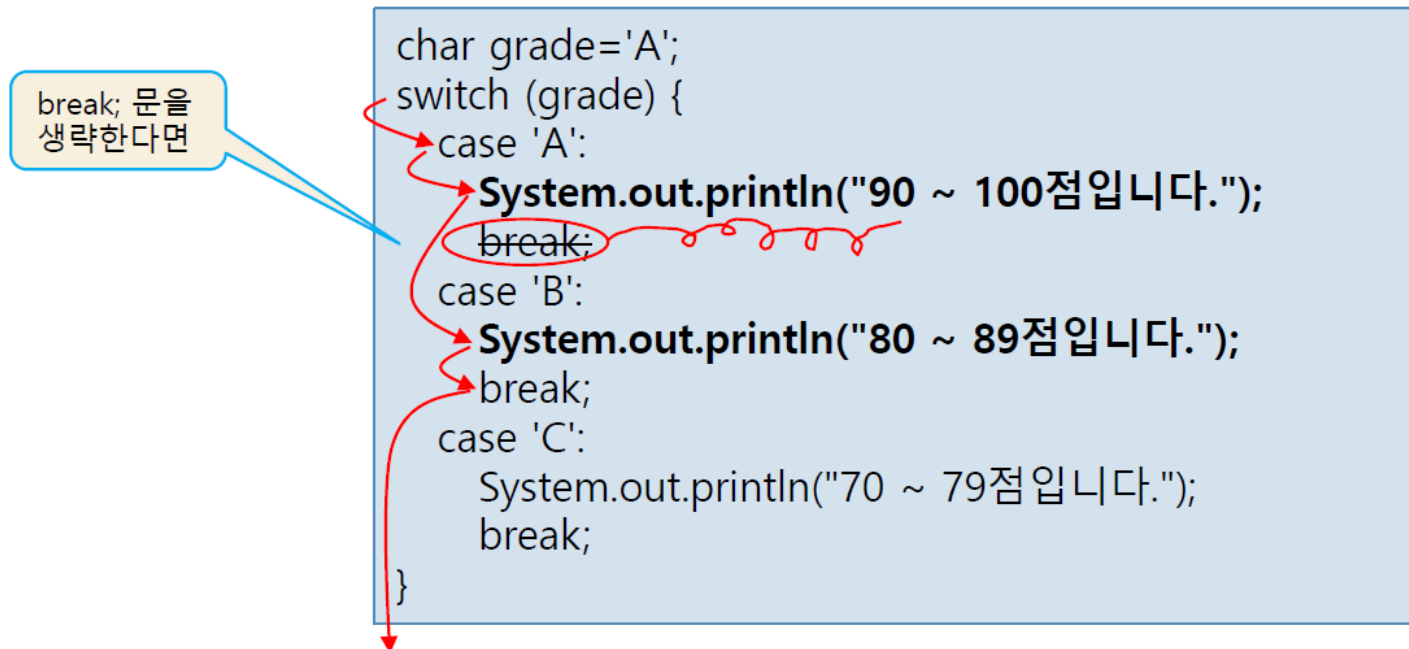
```
switch(식) {  
    case 값1: // 식의 결과가 값1과 같을 때  
        실행 문장 1;  
        break;  
    case 값2: // 식의 결과가 값2와 같을 때  
        실행 문장 2;  
        break;  
    ...  
    case 값m:  
        실행 문장 m; // 식의 결과가 값m과 같을 때  
        break;  
    default: // 어느 것과도 같지 않을 때  
        실행 문장 n;  
}
```

```
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!");  
}
```

switch-case

switch문 내의 break

- “break”를 만나면 switch문을 탈출
- case문에 break문이 없다면 ➔ 다음 case문으로 실행 계속
 - 언젠가 break를 만날 때 까지...



90 ~ 100점입니다.
80 ~ 89점입니다.

switch-case

case문의 값

- 문자, 정수, 문자열 리터럴(JDK>=1.7)만 허용
- 실수 리터럴 허용X

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

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

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

정수 리터럴
사용 가능

문자 리터럴
사용 가능

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

Example

Q. 1~12사이의 month를 입력받아 봄/여름/가을/겨울을 판단하여 출력해보세요.

```
Scanner scanner= new Scanner(System.in);
System.out.print("Enter the month (1-12)");

int month = scanner.nextInt(); // 정수로월입력

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("잘못된입력입니다.");
}
scanner.close();
```

End of slide