

Java

제 3 강

연산자(Operator)

3강 연산자(Operator)

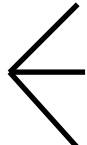
1. 연산자(Operator)란?
2. 연산자의 종류
3. 연산자의 우선순위
4. 증감연산자(++/--)
5. 부호연산자(+,-)와 논리부정연산자(!)
6. 비트전환연산자(~)
7. 이항연산자의 특징
8. 나머지 연산자(%)
9. 쉬프트연산자(<<,>>,>>>)
10. 비교연산자(>,<,>=,<=,==,!=)
11. 비트연산자(&|,^)
12. 논리연산자(&&||)
13. 삼항연산자(? :)
14. 대입연산자(=,op=)

1. 연산자(Operator)란?

- ▶ 연산자(Operator)
 - 어떠한 기능을 수행하는 기호(+, -, *, / 등)
- ▶ 피연산자(Operand)
 - 연산자의 작업 대상(변수, 상수, 리터럴, 수식)

$$a + b$$

2. 연산자의 종류

- ▶ 단항 연산자 : + - (타입) ++ -- ~ !
- ▶ 이항 연산자 
 - 산술 : + - * / % << >> >>>
 - 비교 : > < >= <= == !=
 - 논리 : && || & ^ |
- ▶ 삼항 연산자 : ? :
- ▶ 대입 연산자 : =

3. 연산자의 우선순위(1/4)

| 종 류 | 연산방향 | 연 산 자 | 우선순위 |
|--------|------|--|------|
| 단항 연산자 | ← | ++ -- + - ~ ! (타입) | 높음 |
| 산술 연산자 | → | * / % | |
| | → | + - | |
| | → | << >> >>> | |
| 비교 연산자 | → | < > <= >= instanceof | |
| | → | == != | |
| 논리 연산자 | → | & | |
| | → | ^ | |
| | → | | |
| | → | && | |
| | → | | |
| 삼항 연산자 | → | ?: | 낮음 |
| 대입 연산자 | ← | = *= /= %= += -= <<= >>= >>>= &= ^= = | |

[표3-1] 연산자의 종류와 우선순위

3. 연산자의 우선순위_(2/4)

- 괄호의 우선순위가 제일 높다.
- 산술 > 비교 > 논리 > 대입
- 단항 > 이항 > 삼항
- 연산자의 연산 진행방향은 왼쪽에서 오른쪽(\rightarrow)이다.
단, 단항, 대입 연산자만 오른쪽에서 왼쪽(\leftarrow)이다.

$$3 * 4 * 5$$

$$x = y = 3$$

3. 연산자의 우선순위_(3/4)

- 상식적으로 생각하라. 우리는 이미 다 알고 있다.

ex1) $-x + 3$ 단항 > 이항

ex2) $x + 3 * y$ 곱셈, 나눗셈 > 덧셈, 뺄셈

ex3) $x + 3 > y - 2$ 산술 > 비교

ex4) $x > 3 \ \&\& \ x < 5$ 비교 > 논리

ex5) `int result = x + y * 3;` 항상 대입은 맨 끝에

4. 연산자의 우선순위_(4/4)

- 그러나 몇 가지 주의해야 할 것이 있다.

1. $<<$, $>>$, $>>>$ 는 덧셈연산자보다 우선순위가 낮다.

ex5) $x << 2 + 1$ $x << (2 + 1)$ 과 같다.

2. \parallel , $|$ (OR)는 $\&\&$, $\&$ (AND)보다 우선순위가 낮다.

ex6) $x < -1 \parallel x > 3 \&\& x < 5$

$x < -1 \parallel (x > 3 \&\& x < 5)$ 와 같다.

4. 증감연산자 - ++, --

- ▶ 증가연산자(++): 피연산자의 값을 1 증가시킨다.
- ▶ 감소연산자(--): 피연산자의 값을 1 감소시킨다.

```
int i = 5;
```

```
int j = 0;
```

| | | | |
|-----|-----------------------|--|-------------------|
| 전위형 | <code>j = ++i;</code> | <code>++i;</code> <code>j = i;</code> | 값이 참조되기 전에 증가시킨다. |
| 후위형 | <code>j = i++;</code> | <code>j = i;</code> <code>i++;</code> | 값이 참조된 후에 증가시킨다. |

5. 부호연산자(+,-)와 논리부정연산자(!)

- ▶ 부호연산자(+,-) : '+'는 피연산자에 1을 곱하고
'-'는 피연산자에 -1을 곱한다.
- ▶ 논리부정연산자(!) : true는 false로, false는 true로
피연산자가 boolean일 때만 사용가능

```
int i = -10;
```

```
i = +i;
```

```
i = -i;
```

```
boolean power = false;
```

```
power = !power;
```

```
power = !power;
```

6. 비트전환연산자 - ~

- 정수를 2진수로 표현했을 때, 1을 0으로 0은 1로 바꾼다.
정수형에만 사용가능.

| 2진수 | 10진수 | | | | | | | | |
|--|------|---|---|---|---|---|---|---|------|
| <table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr></table> | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 10 |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | | |
| <table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table> | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | -11 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | | |
| <table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table> | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | -11 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | | |
| <table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr></table> | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | +) 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | | |
| <table><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table> | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | -10 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | | |

【표3-5】 음수를 2진수로 표현하는 방법

7. 이항연산자의 특징^(1/7)

이항연산자는 연산을 수행하기 전에 피연산자의 타입을 일치시킨다.

- int보다 크기가 작은 타입은 int로 변환한다.

(byte, char, short → int)

- 피연산자 중 표현범위가 큰 타입으로 형변환 한다.

byte + short → int + int → int

char + int → int + int → int

float + int → float + float → float

long + float → float + float → float

float + double → double + double → double

7. 이항연산자의 특징_(2/7)

```
byte a = 10;
```

```
byte b = 20;           byte + byte → int + int → int
```

```
byte c = a + b;
```

```
byte c = (byte)a + b;    // 에러
```

```
byte c = (byte)(a + b);  // OK
```

7. 이항연산자의 특징_(3/7)

```
int a = 1000000; // 1,000,000
```

```
int b = 2000000; // 2,000,000
```

```
long c = a * b; // c는 2,000,000,000,000 ?  
              // c는 -1454759936 !!!
```

$\text{int} * \text{int} \rightarrow \text{int}$

```
long c = (long)a * b; // c는 2,000,000,000,000 !
```

$\text{long} * \text{int} \rightarrow \text{long} * \text{long} \rightarrow \text{long}$

7. 이항연산자의 특징_(4/7)

`long a = 1000000 * 1000000; // a는 -727,379,968`

`long b = 1000000 * 1000000L; // b는 1,000,000,000,000`

`int c = 1000000 * 1000000 / 1000000; // c는 -727`

`int d = 1000000 / 1000000 * 1000000; // d는 1,000,000`

7. 이항연산자의 특징^(5/7)

```
char c1 = 'a';
```

```
char c2 = c1 + 1; // 에러
```

```
char c2 = (char)(c1 + 1); // OK
```

```
char c2 = ++c1; // OK
```

```
int i = 'B' - 'A';
```

```
int i = '2' - '0';
```

| 문자 | 코드 |
|-----|-----|
| ... | ... |
| 0 | 48 |
| 1 | 49 |
| 2 | 50 |
| ... | ... |
| A | 65 |
| B | 66 |
| C | 67 |
| ... | ... |
| a | 97 |
| b | 98 |
| c | 99 |
| ... | ... |

7. 이항연산자의 특징_(6/7)

```
float pi = 3.141592f;
```

```
float shortPi = (int)(pi * 1000) / 1000f;
```

```
(int)(3.141592f * 1000) / 1000f;
```

```
(int)(3141.592f) / 1000f;
```

```
3141 / 1000f;
```

```
3141.0f / 1000f
```

```
3.141f
```

7. 이항연산자의 특징_(7/7)

* Math.round() : 소수점 첫째자리에서 반올림한 값을 반환

```
float pi = 3.141592f;
```

```
float shortPi = Math.round(pi * 1000) / 1000f;
```

```
Math.round(3.141592f * 1000) / 1000f;
```

```
Math.round(3141.592f) / 1000f;
```

```
3142 / 1000f;
```

```
3142.0f / 1000f
```

```
3.142f
```

8. 나머지 연산자 - %

- 나누기한 나머지를 반환한다.
- 홀수, 짝수 등 배수검사에 주로 사용.

```
int share = 10 / 8;
```

```
int remain = 10 % 8;
```

$10 \% 8 \rightarrow 2$

$10 \% -8 \rightarrow 2$

$-10 \% 8 \rightarrow -2$

$-10 \% -8 \rightarrow -2$

9. 쉬프트연산자 - <<, >>, >>>

- 2^n 으로 곱하거나 나눈 결과를 반환한다.
- 곱셈, 나눗셈보다 빠르다.

$x \ll n$ 은 $x * 2^n$ 과 같다.

$x \gg n$ 은 $x / 2^n$ 과 같다.

$8 \ll 2$ 는 $8 * 2^2$ 과 같다.

$8 \gg 2$ 는 $8 / 2^2$ 과 같다.

* 참고 : p.54 표3-9 쉬프트연산의 예

10. 비교연산자 - > < >= <= == !=

- 피연산자를 같은 타입으로 변환한 후에 비교한다.
결과 값은 true 또는 false이다.
- 기본형(boolean제외)과 참조형에 사용할 수 있으나
참조형에는 ==와 !=만 사용할 수 있다.

| 수 식 | 연 산 결 과 |
|------------|-----------------------------------|
| $x > y$ | x가 y보다 클 때 true, 그 외에는 false |
| $x < y$ | x가 y보다 작을 때 true, 그 외에는 false |
| $x \geq y$ | x가 y보다 크거나 같을 때 true, 그 외에는 false |
| $x \leq y$ | x가 y보다 작거나 같을 때 true, 그 외에는 false |
| $x == y$ | x와 y가 같을 때 true, 그 외에는 false |
| $x != y$ | x와 y가 다를 때 true, 그 외에는 false |

【표3-11】 비교연산자의 연산결과

10. 비교연산자 - > < >= <= == !=

'A' < 'B' → 65 < 66 → true

'0' == 0 → 48 == 0 → false

'A' != 65 → 65 != 65 → false

10.0d == 10.0f → 10.0d == 10.0d → true

0.1d == 0.1f → 0.1d == 0.1d → true? false?

```
double d = (double)0.1f;
```

```
System.out.println(d); // 0.10000000149011612
```

(float)0.1d == 0.1f → 0.1f == 0.1f → true

11. 비트연산자 - & | ^

- 피연산자를 비트단위로 연산한다.

실수형(float, double)을 제외한 모든 기본형에 사용가능

- ▶ OR연산자(|) : 피연산자 중 어느 한 쪽이 1이면 1이다.
- ▶ AND연산자(&) : 피연산자 양 쪽 모두 1이면 1이다.
- ▶ XOR연산자(^) : 피연산자가 서로 다를 때 1이다.

| x | y | x y | x & y | x ^ y |
|---|---|-------|-------|-------|
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 |

11. 비트연산자 - & | ^

| 식 | 2진수 | 10진수 |
|------------------|--|------|
| $3 \mid 5 = 7$ | <div>0 0 0 0 0 0 1 1</div> | 3 |
| | <div>) 0 0 0 0 0 1 0 1</div> | 5 |
| | <div>0 0 0 0 0 1 1 1</div> | 7 |
| $3 \& 5 = 1$ | <div>0 0 0 0 0 0 1 1</div> | 3 |
| | <div>&) 0 0 0 0 0 1 0 1</div> | 5 |
| | <div>0 0 0 0 0 0 0 1</div> | 1 |
| $3 \wedge 5 = 6$ | <div>0 0 0 0 0 0 1 1</div> | 3 |
| | <div>^) 0 0 0 0 0 1 0 1</div> | 5 |
| | <div>0 0 0 0 0 1 1 0</div> | 6 |

[표3-14] 비트연산자의 연산결과

11. 비트연산자 - & | ^

0x185C

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0x185C >> 4 → 0x0185

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0x000F

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0x0185 & 0x000F → 0x0005

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

0x185C >> 4 & 0x000F → 0x0005

0x185C >> 8 & 0x000F → 0x0008

12345 / 100 % 10 → 3

12345 / 1000 % 10 → 2

12. 논리연산자 - && ||

-피연산자가 반드시 boolean이어야 하며 연산결과도 boolean이다.

&&가 || 보다 우선순위가 높다. 같이 사용되는 경우 괄호를 사용하자

- ▶ OR연산자(||) : 피연산자 중 어느 한 쪽이 true이면 true이다.
- ▶ AND연산자(&&) : 피연산자 양 쪽 모두 true이면 true이다.

| x | y | x y | x && y |
|-------|-------|--------|--------|
| true | true | true | true |
| true | false | true | false |
| false | true | true | false |
| false | false | false | false |

12. 논리연산자 - && ||

```
int i = 7;
```

```
i > 3 && i < 5
```

```
i > 3 || i < 0
```

| x | y | x y | x && y |
|-------|-------|--------|--------|
| true | true | true | true |
| true | false | true | false |
| false | true | true | false |
| false | false | false | false |

```
char x = 'j';
```

```
x >= 'a' && x <= 'z'
```

```
(x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')
```

13. 삼항연산자 - ? :

- 조건식의 연산결과가 true이면 '식1'의 결과를 반환하고
false이면 '식2'의 결과를 반환한다.

(조건식) ? 식1 : 식2

```
int x = -10;
```

```
int absX = x >= 0 ? x : -x;
```

```
int score = 50;
```

```
char grade = score >= 90 ? 'A' : (score >= 80 ? 'B' : 'C');
```

```
if(x >= 0) {  
    absX = x;  
} else {  
    absX = -x;  
}
```

14. 대입연산자 - = op=

- 오른쪽 피연산자의 값을 왼쪽 피연산자에 저장한다.
단, 왼쪽 피연산자는 상수가 아니어야 한다.

```
int i = 0;
```

```
i = i + 3;
```

```
final int MAX = 3;
```

```
MAX = 10; // 에러
```

| op= | = |
|--------------|------------------|
| i += 3; | i = i + 3; |
| i -= 3; | i = i - 3; |
| i *= 3; | i = i * 3; |
| i /= 3; | i = i / 3; |
| i %= 3; | i = i % 3; |
| i <<= 3; | i = i << 3; |
| i >>= 3; | i = i >> 3; |
| i >>>= 3; | i = i >>> 3; |
| i &= 3; | i = i & 3; |
| i ^= 3; | i = i ^ 3; |
| i = 3; | i = i 3; |
| i *= 10 + j; | i = i * (10+j) ; |

키보드입력을 받는 방법

```
1 import java.util.Scanner;
2
3 public class KeyInput {
4
5     public static void main(String[] args) {
6
7         Scanner scan = new Scanner(System.in);
8         System.out.print("이름을 입력하세요: ");
9         String name = scan.nextLine();
10
11         System.out.print("정수를 입력하세요?");
12         int number = scan.nextInt();
13
14         System.out.print("실수를 입력하세요?");
15         double real = scan.nextDouble();
16
17         System.out.println("이름은 " + name);
18         System.out.println("정수는 " + number);
19         System.out.println("실수는 " + real);
20
21     }
22
23 }
```

감사합니다.