연산자
(Operator)

자바에서 제공하는 이항 연산자들

연산기호	결합 방향	우선순위
[ ], .	➡	1(높음)
expr++, expr−−	⬅	2
++expr, −− expr, +expr, −expr, ~, !, (type)	⬅	3
*, /, %	➡	4
+, −	➡	5
〈〈, 〉〉, 〉〉〉	➡	6
〈, 〉, 〈=, 〉=, instanceof	➡	7
==, !=	➡	8
&	➡	9
^	➡	10
|	➡	11
&&	➡	12
||	➡	13
? expr : expr	⬅	14
=, +=, −=, *=, /=, %=, &=, ^=, |=, 〈〈=, 〉〉=, 〉〉〉=	⬅	15(낮음)

우선순위 적용

2 − 1 − 3 × 2

결합 방향 적용

= 2 − 1 − 6

= 1 − 6

결합 방향은 우선순위가 같을 때 적용하는 기준.

연산자의 우선순위와 결합 방향

연산자	연산자의 기능	결합 방향
=	연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입한다. 예) val = 20;	⬅
+	두 피연산자의 값을 더한다. 예) val = 4 + 3;	➡
−	왼쪽의 피연산자 값에서 오른쪽의 피연산자 값을 뺀다. 예) val = 4 − 3;	➡
*	두 피연산자의 값을 곱한다. 예) val = 4 * 3;	➡
/	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눈다. 예) val = 7 / 3;	➡
%	왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눴을 때 얻게 되는 나머지를 반환한다. 예) val = 7 % 3;	➡

대입 연산자와 산술 연산자

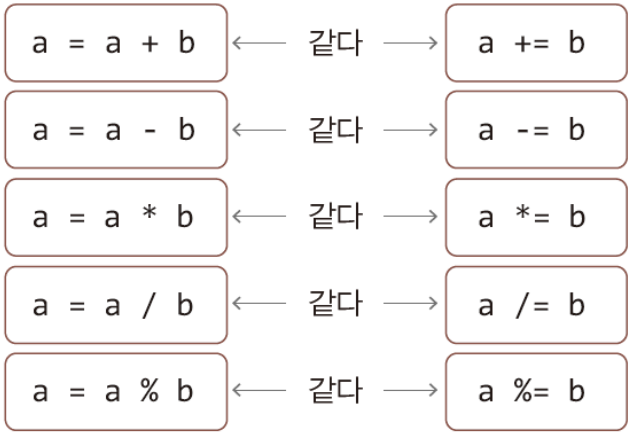
정수형 나눗셈과 실수형 나눗셈

```
int num1 = 7;
int num2 = 3;
System.out.println("num1 / num2 = " + (num1 / num2));
```

정수형 나눗셈 진행

```
System.out.println("num1 / num2 = " + (7.0 / 3.0));
```

실수형 나눗셈 진행



ex1)

num = num + 5;

→ num += 5;

ex2)

num = num * 3;

→ num *= 3;

복합 대입 연산자

A &= B	↔	A = A & B
A ^= B	↔	A = A ^ B
A <<= B	↔	A = A << B
A >>>= B	↔	A = A >>> B

복합 대입 연산자 추가

연산자	연산자의 기능	결합 방향
<	예) n1 < n2 n1이 n2보다 작은가?	➡
>	예) n1 > n2 n1이 n2보다 큰가?	➡
<=	예) n1 <= n2 n1이 n2보다 같거나 작은가?	➡
>=	예) n1 >= n2 n1이 n2보다 같거나 큰가?	➡
==	예) n1 == n2 n1과 n2가 같은가?	➡
!=	예) n1 != n2 n1과 n2가 다른가?	➡

관계 연산자

논리 연산자

연산자	연산자의 기능	결합 방향
&&	예) A && B A와 B 모두 true이면 연산 결과는 true (논리 AND)	➡
||	예) A || B A와 B 둘 중 하나라도 true이면 연산 결과는 true (논리 OR)	➡
!	예) !A 연산 결과는 A가 true이면 false, A가 false이면 true (논리 NOT)	⬅

피연산자 1(OP1)	피연산자 2(OP2)	연산 결과(OP1 && OP2)
true	true	true
true	false	false
false	true	false
false	false	false

피연산자 1(OP1)	피연산자 2(OP2)	연산 결과(OP1 || OP2)
true	true	true
true	false	true
false	true	true
false	false	false

피연산자(OP)	연산 결과(!OP)
true	false
false	true

논리 연산자 사용시 주의점 : SCE

```
result = ((num1 += 10) < 0) && ((num2 += 10) > 0);

result = ((num1 += 10) > 0) || ((num2 += 10) > 0);
```

num1과 num2의 값이 모두 증가할 수 있을까?

false 이면...

```
(num1 += 10) < 0   &&   (num2 += 10) > 0
```

이 부분 생략

true 이면...

```
(num1 += 10) > 0   ||   (num2 += 10) > 0
```

이 부분 생략

자바에서 제공하는 단항 연산자들

부호 연산자

```
double e1 = 3.5;

double e2 = -e1;        // e1에 저장되는 값은 -3.5
```

부호 연산자 -는 변수에 저장된 값의 부호를 바꾸어 반환한다.

연산자	연산자의 기능	결합 방향
++ (prefix)	피연산자에 저장된 값을 1 증가 예) val = ++n;	←
-- (prefix)	피연산자에 저장된 값을 1 감소 예) val = --n;	←

연산자	연산자의 기능	결합 방향
++ (postfix)	피연산자에 저장된 값을 1 증가 예) val = n++;	←
-- (postfix)	피연산자에 저장된 값을 1 감소 예) val = n--;	←

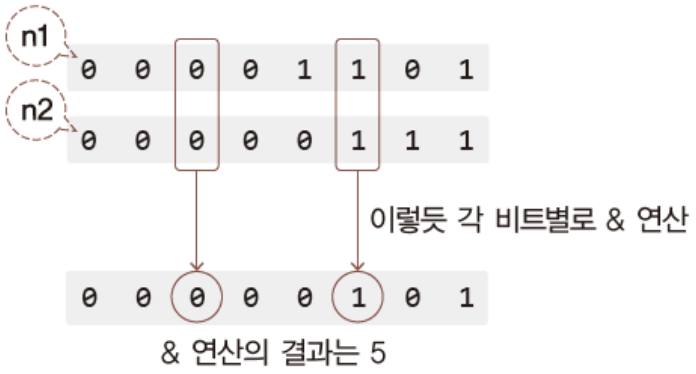
증가 감소 연산자

비트를 대상으로 하는 연산자들

비트 연산자의 이해

```
public static void main(String[] args) {
    byte n1 = 13;
    byte n2 = 7;
    byte n3 = (byte)(n1 & n2);
    System.out.println(n3);
}
```

연산자	연산자의 기능
&	비트 단위로 AND 연산을 한다. 예) n1 & n2;



각각의 비트를 대상으로 연산을 진행, 그리고 각 비트를 대상으로 진행된 연산 결과를 묶어서 하나의 연산 결과 반환

비트 연산자

연산자	연산자의 기능	결합 방향
&	비트 단위로 AND 연산을 한다. 예) n1 & n2;	
|	비트 단위로 OR 연산을 한다. 예) n1 | n2;	
^	비트 단위로 XOR 연산을 한다. 예) n1 ^ n2;	
~	피연산자의 모든 비트를 반전시켜서 얻은 결과를 반환 예) ~n;	

비트	~비트
1	0
0	1

비트 A	비트 B	비트 A & 비트 B
1	1	1
1	0	0
0	1	0
0	0	0

비트 A	비트 B	비트 A | 비트 B
1	1	1
1	0	1
0	1	1
0	0	0

비트 A	비트 B	비트 A ^ 비트 B
1	1	0
1	0	1
0	1	1
0	0	0

연산자	연산자의 기능	결합 방향
<<	• 피연산자의 비트 열을 왼쪽으로 이동 • 이동에 따른 빈 공간은 0으로 채움 • 예) n << 2; → n의 비트 열을 두 칸 왼쪽으로 이동 시킨 결과 반환	➡
>>	• 피연산자의 비트 열을 오른쪽으로 이동 • 이동에 따른 빈 공간은 음수의 경우 1, 양수의 경우 0으로 채움 • 예) n >> 2; → n의 비트 열을 두 칸 오른쪽으로 이동 시킨 결과 반환	➡
>>>	• 피연산자의 비트 열을 오른쪽으로 이동 • 이동에 따른 빈 공간은 0으로 채움 • 예) n >>> 2; → n의 비트 열을 두 칸 오른쪽으로 이동 시킨 결과 반환	➡

비트 쉬프트 연산자