java

# 6강_연산자

프로그램이 실행될 때 컴퓨터(CPU)한테 계산(연산) 업무를 시키기 위한 방법에 대해서 학습합니다.

6-1 피연산자 개수에 의한 연산자 구분
6-2 대입 연산자
6-3 산술 연산자
6-4 복합 대입연산자
6-5 관계 연산자
6-6 증감 연산자
6-7 논리 연산자
6-8 조건(삼항) 연산자
6-9 비트 연산자

2018. 01 / blogstudy@naver.com

# 6-1 : 피연산자 개수에 의한 연산자 구분

**피연산자 개수에 따라서 단항, 이항 그리고 삼항 연산자로 구분할 수 있다.**

## 단항 연산자

피연산자가 하나 존재

+x, -x, !x

## 이항 연산자

피연산자가 두개 존재

x = y, x < y, x != y

## 삼항 연산자

피연산자가 세개 존재

조건식 : true ? false

# 6-2 : 대입 연산자

## 대입 연산자는 오른쪽의 결과를 왼쪽에 대입(할당) 한다.

'='는 수학에서 '오른쪽 값과 왼쪽 값이 같다' 라는 의미 이지만,
프로그램에서는 '오른쪽 값을 왼쪽에 대입' 하는 의미로 쓰인다.

프로그램에서 '오른쪽와 왼쪽이 같다' 의미는 '=='이다.

```java
int x = 10;
int y = 20;

// 대입 연산자
System.out.println("x = " + x);
System.out.println("y = " + y);

x = y;
System.out.println("x = " + x);
System.out.println("y = " + y);
```

```
<terminated>
x = 10
y = 20
x = 20
y = 20
```

# 6-3 : 산술 연산자

**피연산자를 이용해서 +, -, *, /, % 등을 수행한다.**

| | |
|---|---|
| + | 덧셈 |
| - | 뺄셈 |
| * | 곱셈 |
| / | 나눗셈 |
| % | 나머지 |

```
x = 10; y = 20;
// 산술 연산자
System.out.println("x + y = " + (x + y));
System.out.println("x - y = " + (x - y));
System.out.println("x * y = " + (x * y));
System.out.println("x / y = " + (x / y));
System.out.println("x % y = " + (x % y));
```

```
x + y = 30
x - y = -10
x * y = 200
x / y = 0
x % y = 10
```

# 6-4 : 복합 대입연산자

**산술 연산자와 대입 연산자를 결합한 연산자이다.**

| | |
|---|---|
| += | 더하고 대입 |
| -= | 빼고 대입 |
| *= | 곱하고 대입 |
| /= | 나누고 대입 |
| %= | 나머지를 대입 |

```
// 복합대입 연산자
x = 10;
System.out.println("x += 10 : " + (x += 10));
x = 10;
System.out.println("x -= 10 : " + (x -= 10));
x = 10;
System.out.println("x *= 10 : " + (x *= 10));
x = 10;
System.out.println("x /= 10 : " + (x /= 10));
x = 10;
System.out.println("x %= 10 : " + (x %= 10));
```

```
x += 10 : 20
x -= 10 : 0
x *= 10 : 100
x /= 10 : 1
x %= 10 : 0
```

# 6-5 : 관계 연산자

**두개의 피연산자를 비교해서 참/거짓의 결론을 돌출한다.**

| | |
|---|---|
| 〉 | a〉b : a가 b보다 크면 참 |
| 〈 | a〈b : a가 b보다 작으면 참 |
| 〉= | a〉=b : a가 b보다 크거나 같으면 참 |
| 〈= | a〈=b : a가 b보다 작거나 같으면 참 |
| == | a==b : a와 b가 같으면 참 |
| != | a!=b : a와 b가 같지 않으면 참 |

```
// 관계 연산자
x = 10; y = 20;
System.out.println("x 〉 y : " + (x 〉 y));
System.out.println("x 〈 y : " + (x 〈 y));
System.out.println("x 〉= y : " + (x 〉= y));
System.out.println("x 〈= y : " + (x 〈= y));
System.out.println("x == y : " + (x == y));
System.out.println("x != y : " + (x != y));
```

```
x 〉 y : false
x 〈 y : true
x 〉= y : false
x 〈= y : true
x == y : false
x != y : true
```

# 6-6 : 증감 연산자

## 1만큼 증가 하거나 감소를 수행한다.

| | |
|---|---|
| ++ | 1만큼 증가 |
| -- | 1만큼 감소 |

```
// 증감 연산자
x = 10;
System.out.println("++x : " + (++x));
x = 10;
System.out.println("--x : " + (--x));
x = 10;
System.out.println("x++ : " + (x++));
System.out.println("x : " + x);
x = 10;
System.out.println("x-- : " + (x--));
System.out.println("x : " + x);
```

```
++x : 11
--x : 9
x++ : 10
x : 11
x-- : 10
x : 9
```

# 6-7 : 논리 연산자

**피연산자의 논리곱(AND), 논리합(OR), 논리부정(NOT)을 수행한다.**

| | |
|---|---|
| && 논리곱(AND) | a && b : a와 b가 모두 참이면 참 |
| \|\| 논리합(OR) | a \|\| b : a와 b중 하나라도 참이면 참 |
| ! 논리부정(NOT) | ! a : a의 상태를 부정 |

```java
// 논리 연산자
boolean b1 = false;
boolean b2 = true;
System.out.println("b1 && b2 : " + (b1 && b2));
System.out.println("b1 || b2 : " + (b1 || b2));
System.out.println("!b1 : " + !b1);
System.out.println("!b2 : " + !b2);
```

```
b1 && b2 : false
b1 || b2 : true
!b1 : true
!b2 : false
```

# 6-8 : 조건(삼항) 연산자

**삼항 연산자로 두개의 피연산자 연산 결과에 따라서 나머지 피연산자가 결정된다.**

| | |
|---|---|
| 조건식 ? 식1 : 식2 | 조건식이 참이면 식1이 실행되고,<br>조건식이 거짓이면 식2가 실행된다. |

```java
// 조건(삼항) 연산자
x = 10; y = 20;
int result = 0;
result = (x > y) ? 100 : 200;
System.out.println("result : " + result);

result = (x < y) ? 100 : 200;
System.out.println("result : " + result);

result = (x == y) ? 100 : 200;
System.out.println("result : " + result);
```

```
result : 200
result : 100
result : 200
```

# 6-9 : 비트 연산자

**데이터를 비트(bit)단위로 환산하여 연산을 수행하며, 다른 연산자보다 연산 속도가 향상된다.**

| | |
|---|---|
| & : AND 연산 | a & b : a와 b가 모두 1이면 1 |
| \| : OR 연산 | a \| b : a와 b중 하나라도 1이면 1 |
| ^ : XOR 연산 | a^b : a와 b가 같지 않으면 1 |

```
// 비트 연산자
x = 2;
y = 3;
System.out.println("x & y : " + (x & y));
System.out.println("x | y : " + (x | y));
System.out.println("x ^ y : " + (x ^ y));
```

```
x & y : 2
x | y : 3
x ^ y : 1
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| x=2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| y=3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| & | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| \| | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| ^ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

2

3

1