# 연산자

## #01. 대입연산자 (=)

변수에 값이나 다른 연산결과를 대입하기 위해 사용한다.

항상 우변에서 좌변으로 대입된다.

### 1) 연산자의 사용 방법

선언된 변수에 새로운 값을 대입할 수 있다.

```
int x = 100;
int y;
y = 200;
```

이미 값이 대입되어 있는 변수의 값을 다른 값으로 변경할 수 있다.

```
int num = 100;
num = 200;
```

### 2) 두 개의 변수값을 맞교환 하기.

두 변수 a와 b가 있을 때 이 두 값을 직접적으로 교환하지는 못하기 때문에 새로운 변수 c를 등장시켜서 처리해야 한다.

1. c에 a를 복사.
2. a에 b를 복사.
3. b에 c를 복사.

## #02. 사칙연산자 (산술연산자)

덧셈(+), 뺄셈(-), 곱셈(*), 나눗셈(/, %)

### 1) 연산의 결과

연산 결과는 다른 변수에 대입하거나 직접 출력할 수 있다.

**변수간의 연산결과를 다른 변수에 대입 후 출력**

```
int num1 = 100;
int num2 = 200;
int result = num1 + num2;
System.out.println(result);
```

**변수간의 연산결과를 직접 출력**

```
int num1 = 100;
int num2 = 200;
System.out.println(num1 - num2);
```

**값들간의 연산결과를 직접 출력**

```
System.out.println(100 * 2)
```

### 2) 나눗셈

**/ 연산자**

나눗셈의 몫을 반환한다.

```
int a = 7;
int b = 3;
Sytem.out.println(a / b);  // 2가 출력됨
```

**% 연산자**

나눗셈의 나머지만 반환한다.

```
int a = 7;
int b = 3;
Sytem.out.println(a % b);  // 1이 출력됨
```

### 3) 연산자 사용시 주의사항

**데이터 타입 유지**

정수끼리의 연산결과는 정수형 데이터가 되고, 실수끼리의 연산은 실수형 데이터가 된다.

정수와 실수를 연산할 경우 표현 범위가 더 큰 실수형 데이터로 결과가 만들어 진다.

```
int x = 1;
double y = 3.5;

// 정수와 실수의 연산이므로 결과는 실수형 변수에만 할당 가능하다.
double z = x + y;
```

**모든 수는 0으로 나눌 수 없다**

```
int a = 10;
int b = 0;
System.out.println(a / b);
```

위의 연산은 아래와 같은 에러메시지를 표시한다.

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
at Test.main(Test.java:5)
```

### 4) 문자열의 연산

**문자열 덧셈**

문자열간에는 덧셈만 가능하고, 모든 내용을 연결하는 결과를 반환한다.

```
String a = "Hello";
String b = "World";
System.out.println(a + b);       // HelloWorld (띄어쓰기 없음)
```

**공백을 포함한 연산**

공백 문자열(띄어쓰기)도 하나의 글자로 취급된다.

```
String a = "Hello";
String b = "World";
String c = " ";                  // 띄어쓰기 한 칸
System.out.println(a + c + b);   // Hello World (띄어쓰기 있음)
```

**빈 문자열**

아무런 글자도 포함하지 않는 문자열. 쌍따옴표를 열고 즉시 닫는다.

💡 ex) 로그인 시에 아무런 값이 입력하지 않고 버튼을 누르면 사용자의 입력값은 빈 문자열로 취급된다.

```
String a = ""; // 아무런 내용도 포함하지 않음
System.out.println(a);   // 출력되는 내용 없이 줄바꿈만 수행함.
```

**문자열과 다른 데이터 타입간의 덧셈**

문자열과 연산되는 모든 변수값은 문자열로 변환되어 처리된다.

그 결과 문자열간의 연산으로 취급되어 단순히 연결된 결과값을 반환한다.

```
int number = 100;
String message = "HelloJava";

// "HelloJava" + 100 --> "HelloJava" + "100"으로 처리됨
System.out.println(message + number); // HelloJava100
```

# #03. 단항연산자

## 1) 단항연산자의 이해

어떤 변수가 연산 결과를 다시 자기 스스로에게 덮어 씌우는 경우

💡 대입연산자를 기준으로 항상 오른쪽이 먼저 수행되고 왼쪽으로 제어가 이동한다.

```
int x = 1;
x = x + 2;
System.out.println(x);
```

- **출력결과**

```
3
```

위의 연산식을 축약해서 표현한 형태가 **단항연산자**

```
int x = 1;
x += 2;    // x 스스로 2 증가한다고 이해.
```

- **출력결과**

```
3
```

## 2) 단항연산자의 종류

모든 사칙연산자는 단항연산자로 표현 가능하다.

**덧셈**

```
int a = 1;
a += 10;  // a 스스로 10 증가
System.out.println(a);
```

- 출력결과

```
11
```

**뺄셈**

```
int b = 10;
b -= 5;  // b스스로 5 감소
System.out.println(b);
```

- 출력결과

```
5
```

**곱셈**

```
int c = 5;
c *= 2;  // c스스로 2배 증가
System.out.println(c);
```

- 출력결과

```
10
```

**나눗셈 (몫)**

```
int d = 100;
d /= 2;  // d스스로 2로 나눈 몫을 취함
System.out.println(d);
```

- 출력결과

```

```

```
50
```

**나눗셈 (나머지)**

```
int e = 5;
e %= 2;
System.out.println(e);
```

- 출력결과

```
1
```

## #04. 증감연산자

💡 `1`에 대한 단항 연산의 또 한번의 축약

어떤 변수에 단항 연산으로 표현되어 계산되는 값이 1인 경우 **덧셈**과 **뺄셈**에 한해 다시 한번 축약이 가능하다.

### 1) 덧셈에 대한 연산식 변화

**사칙연산자 표현**

```
int k = 100;
k = k + 1;
```

**단항연산자 표현**

```
int k = 100;
k += 1;
```

**증감연산자 표현**

```
int k = 100;
k++;  // k 스스로 1증가
```

혹은

```
```

```
int k = 100;
++k;  // k 스스로 1증가
```

2) 뺄셈에 대한 연산식 변화

**사칙연산자 표현**

```
int k = 100;
k = k - 1;
```

**단항연산자 표현**

```
int k = 100;
k -= 1;
```

**증감연산자 표현**

```
int k = 100;
k--;  // k 스스로 1증가
```

혹은

```
int k = 100;
--k;  // k 스스로 1감소
```

3) 증감연산자의 위치에 따른 차이

증감연산자를 적용한 변수가 단독으로 사용된 경우는 위치에 따른 차이가 없지만 다른 수식에 포함된 경우는 위치에 따라 결과가 달라진다.

**전위 증감연산자**

`++`, `--`가 변수 앞에 적용된 경우.

증감연산자가 적용된 변수를 먼저 계산하고 전체 수식을 계산한다. **(앞북)**

```
int x = 1;
int y = 100 + ++x;  // x가 먼저 1증가하고 100을 더해 y를 확정한다.
System.out.println("x=" + x);
System.out.println("y=" + y);
```

- 출력결과

```
x=2
y=102
```

**후위 증감연산자**

`++`, `--`가 변수 뒤에 적용된 경우.

증감연산자가 적용되기 전 상태에서 전체 수식을 계산하여 결과값을 확정한 후 증감연산이 수행된다. (뒷북)

```
int x = 1;
int y = 100 + x++;  // 100+x가 먼저 수행되어 y를 101로 확정한 뒤 x가 1증가한다.
System.out.println("x=" + x);
System.out.println("y=" + y);
```

- 출력결과

```
x=2
y=101
```

# #05. 비교연산자

## 1) 비교연산자의 종류

수학에서의 부등식에서 사용되는 연산자

| 연산자 | 의미 |
|---|---|
| `==` | 같다 |
| `!=` | 다르다 |
| `<` | 작다(미만) |
| `<=` | 작거나 같다(이하) |
| `>` | 크다(초과) |
| `>=` | 크거나 같다(이상) |

## 2) 비교연산자의 결과값

비교연산자는 전체 수식이 성립되는지 아닌지를 판단한다.

그러므로 결과값은 참(`true`) 혹은 거짓(`false`)로 형성된다.

---

```java
int x = 1;
int y = 2;

// 결과값을 다른 변수에 저장 후 출력하는 경우
boolean a = x == y;
boolean b = x != y;
System.out.println(a);
System.out.println(b);

// 직접 출력하는 경우
System.out.println(x < y);
System.out.println(x <= y);
System.out.println(x > y);
System.out.println(x >= y);
```

- 출력결과

```
false
true
true
true
false
false
```

## #06. 논리연산자

**참**이나 **거짓**을 판별할 수 있는 변수나 수식끼리 **AND(&&)**나 **OR(||)** 연산을 수행하여 논리값 형태의 결과를 만들어 내는 식

중,고등학교 수학 과정에서의 **명제** 단원에 해당하는 내용

### 1) **AND** 연산

**AND** 연산은 모든 값이 **true** 인 경우만 결과값이 **true**이고 그 외의 경우는 모두 **false**를 반환한다.

```java
boolean a = true && true;       // true
boolean b = true && false;      // false
boolean c = false && true;      // false
boolean d = false && false;     // false
```

### 2) **OR** 연산

**OR** 연산은 하나라도 **true** 라면 결과값이 **true**이고 그 외의 경우는 **false**를 반환한다.

```
boolean a = true || true;       // true
boolean b = true || false;      // true
boolean c = false || true;      // true
boolean d = false || false;     // false
```

### 3) 비교식과의 연계

비교 연산자도 결과값이 **true**, **false**로 반환되므로 비교식들끼리 논리연산을 수행할 수 있다.

비교 연산자와 논리 연산자가 함께 사용될 경우 비교 연산자가 우선적으로 판단된다.

```
int x = 10;
int y = 20;
int z = 30;

boolean a = x < y && y > z;     // true && false --> false
boolean b = x >= y && x < z;    // false && true --> false
boolean c = z > y && z > x;     // true && true --> true

boolean d = x < y || y > z;     // true || false --> true
boolean e = x >= y || x < z;    // false || true --> true
boolean f = z > y || z > x;     // true || true --> true
```

### 4) NOT 연산

느낌표(!)를 논리값이나 논리값을 저장하고 있는 변수 앞에 적용하여 반대의 결과값을 반환한다.

```
boolean a = !true;      // false
boolean b = !false;     // true;
boolean c = !a;         // true;
boolean d = !b;         // false;
boolean e = !!true;     // true의 반대의 반대값 --> true
```

## #07. 삼항연산자

?와 :을 사용하여 주어진 식이 참인 경우와 그렇지 않은 경우로 나누어 선택적으로 결과값을 반환한다.

> 조건식 ? 참인경우 : 거짓인경우

```
int a = 10;
int b = 5;
int c = a > b ? 1 : 2;  // a > b가 참이므로 c에 1이 대입된다.
```

```
```

```
boolean k = true;
int r = k ? 100 : 0;    // k는 그 자체가 참을 의미하므로 r에는 100이 대입된다.
int s = !k ? 100 : 0;   // !k는 false이므로 s에는 0이 대입된다.
```

```
int x = 10;
int y = 20;
int z = x > y ? x + y : x - y;  // x > y가 거짓이므로 z에는 x - y의 결과값이 대입된다.
```

## #08. 연산자 우선순위

| 연산자 | 의미 | 순위 |
|---|---|---|
| () | 괄호 | 1 |
| !, ++, -- | 부정, 증감연산자 | 2 |
| *, /, % | 곱셈, 나눗셈 | 3 |
| +, - | 덧셈, 뺄셈 | 4 |
| <, <=, >, >= | 크기를 비교하기 위한 비교연산자 | 5 |
| ==, != | 같음을 비교하는 비교연산자 | 6 |
| && | AND를 위한 논리연산자 | 7 |
| \|\| | OR를 위한 논리연산자 | 8 |
| =, +=, -=, *=, /=, %= | 대입, 단항 연산자 | 9 |