1. 연산자(Operator)

1. 연산자는 산술(덧셈, 뺄셈, 곱셈, 나눗셈), 비교(숫자의 크기 비교)... 등을 할 때 사용되는 부호 => +, -, *, /, %, >, <, ...
2. Java에서 연산자는 단항연산자, 이항연산자, 삼항연산자로 구분할 수 있다. 항은 피연산자(연산이 되는 값을 의미). 단항, 이항, 삼항은 피연산자의 개수를 의미한다.

2. 단항연산자

1. 단항연산자는 피연산자의 개수가 하나인 연산자이다. 대표적으로 부호 연산자와 증감연산자가 있다.
2. 부호연산자
    - 값의 부호를 변경해주는 연산자. +, -가 존재한다.
    - +는 아무 역할을 하지 않지만 -는 현재 부호를 반대로 변경해주는 역할을 한다.

```
int iNum1 = 10;
int iNum2 = +iNum1;
int iNum3 = -iNum1;
```

3. 증감연산자
    - 증감연산자는 변수의 값의 1을 더하거나 1을 뺀 값을 다시 변수에 저장할 때 사용한다. ++, -- 연산자가 존재한다.
    - ++, -- 연산자는 연산자의 위치에 따라서 불리는 이름도 다르고 동작하는 방식도 달라진다.
    - 변수 앞에 증감연산자가 붙으면 그 증감연산자를 전위 증감연산자라고 부르고 변수 뒤에 증감연산자가 붙으면 그 증감연산자를 후위 증감연산자라고 부른다.

```
int num1 = 10;
//num1 = num1 + 1;
//전위 증감연산자
++num1;
//후위 증감연산자
num1++;

//num1 = num1 - 1;
//전위 증감연산자
--num1;
//후위 증감연산자
num1--;
```

- 증감연산자가 다른 연산이나 메소드에서 사용되게 되면 전위 증감연산자와 후위 증감연산자의 동작이 달라지게 된다.
- 전위 증감연산자는 무조건 증감연산이 먼저 진행되고 다른 연산이나 메소드가 진행된다.
- 후위 증감연산자는 다른 연산이나 메소드가 진행되고 난 후에 증감연산을 진행한다.

```
int num1 = 10;
int num2 = 10;

//전위 증감연산자
System.out.println(++num1); => 출력: 11, num1의 값도

//후위 증감연산자
System.out.println(num2++); => 출력: 10, num2 값은 1
```

- **예제: chap03_operator._01_OneOperator.java**

3. 이항연산자

1. 피연산자의 개수가 2개인 연산자를 이항연산자라고 부른다. 대입연산자(=), 산술연산자(+, -, *, /, %), 복합대입연산자(+=, -=, *=, /=, %=), 관계연산자(<, >, <=, >=, ==, !=), 논리연산자(&&, ||, !)
2. 이항연산자는 연산자 왼쪽에 위치하는 피연산자가 기준이 된다.

```
int a = 10;
//왼쪽에 있는 b에 a의 값을 대입한다.
int b = a;

//왼쪽에 있는 b가 a보다 큰지 비교
b > a;
```

3. 대입연산자(=)
    - 왼쪽의 있는 변수에 새로운 값을 저장(대입)할 때 사용하는 연산자.
4. 산술연산자(+, -, *, /, %)
    - 왼쪽에 위치한 피연산자에서 오른쪽에 위치한 피연산자를 산술연산해주는 연산자

```
int num1 = 10;
int num2 = 3;

int result = num1 + num2;
result = num1 - num2;
result = num1 * num2;
result = num1 / num2;
//나머지 연산(%)
//왼쪽의 피연산자를 오른쪽 피연산자로 나눴을 때 남는 나
result = num1 % num2;
```

5. 복합대입연산자
    - 왼쪽 피연산자에서 오른쪽 피연산자를 산술 연산한 값을 다시 왼쪽 피연산자에 저장

```
int num1 = 10;
int num2 = 3;

num1 += num2; // num1 = num1 + num2;
num1 -= num2; // num1 = num1 - num2;
num1 *= num2; // num1 = num1 * num2;
num1 /= num2; // num1 = num1 / num2;
num1 %= num2; // num1 = num1 % num2;
```

- **예제: chap03_operator._02_ArithmeticOperator.java**

6. 관계연산자(비교연산자)
    - 왼쪽 피연산자의 값을 기준으로 오른쪽 피연산자의 값과 크기 비교 또는 동일한지 여부를 비교해주는 연산자. 결과 값은 true나 false로 리턴된다.

```
//값의 크기 비교
10 > 100 => false
10 < 100 => true
10 <= 100 => true
10 >= 100 => false

//값의 동일여부 비교
//값이 같은지 비교(같으면 true, 다르면 false)
10 == 100 => false
//값이 다른지 비교(다르면 true, 같으면 false)
10 != 100 => true
```

7. 논리연산자
    - 두 개의 피연산자를 논리곱(&&), 논리합(||) 해주는 연산자
    - 단항연산자인 부정연산자(!)도 포함된다.
    - 논리곱(&&)
        - 왼쪽 피연산자와 오른쪽 피연산자가 모두 true일 경우에만 true를 반환하는 연산자. 양쪽 하나라도 false가 있으면 false가 된다.

```
10 < 100 && 200 > 20 => true && true => true
10 > 100 && 200 > 20 => false && true => false
```

    - 논리합(||)
        - 왼쪽 피연산자나 오른쪽 피연산자 어디에도 true가 하나라도 존재하면 true를 결과로 리턴

```
10 > 100 || 200 > 20 => false || true => true
10 > 100 || 200 < 20 => false || false => fals
```

    - 부정(!)
        - 피연산자의 값을 반전 시켜주는 연산자. 피연산자의 값이 true면 false를 리턴. 피연산자의 값이 false면 true를 리턴.

```
!(10 < 100) => !true => false
```

- **예제: chap03_operator._03_LogicalOperator.java**

4. 삼항연산자

1. 삼항연산자에는 삼항조건연산자 하나만 존재한다.
2. 삼항 조건연산자: A ? B : C;
    - A에는 조건(true나 false가 나오는 연산)

- A의 결과가 true 면 B가 실행
- A의 결과가 false면 C가 실행

```
int num1 = 10;
int num2 = 20;
                    A              B              C
int result = num1 > num2 ? num1 - num2 : num2 - num1
//result = num2 - num1
```

3. 삼항 조건연산자의 중첩
    - A ? B : C ? D : E;
    - A가 true면 B가 실행. A가 false면서 C가 true D가 실행. A가 false면서 C도 false면 E가 실행.
    - 항상 두 번만 중첩되는 건 아니고 여러 번 반복 중첩하여 사용할 수 있다.

- **예제: chap03_operator._04_ThreeOperator.java**

5. 연산자의 우선순위

1. 단항연산자 > 이항연산자 > 삼항연산자
2. 대입연산자는 항상 마지막에 실행된다.
3. 산술연산자 > 관계연산자(비교연산자) > 논리연산자 > 대입연산자 순으로 진행된다. 하지만 소괄호()로 묶인 연산이 최우선적으로 일어난다.