연산자

연산자

- 이항연산자 : 연산의 대상, 즉 피연산자가 둘인 연산자

연산기호	결합방향	우선순위
[ ], .	->	1 (높음)
expr++, expr--	<-	2
++expr, --expr, !, (type)	<-	3
*, /, %	->	4
+, -	->	5
<<, >>	->	6
<, >, <=, >=	->	7
==, !=	->	8
&	->	9
^	->	10
|	->	11
&&	->	12
||	->	13
? expr : expr	<-	14
=, +=, -=, *=, /=, %=	<-	15(낮음)

연산자

- 대입 연산자 및 산술 연산자

연산자	기능	결합방향
=	- 연산자 오른쪽에 있는 값을 연산자 왼쪽에 있는 변수에 대입한다. - 예) int val = 20;	<-
+	- 두 피연산자의 값을 더한다. - 예) int val = 5 + 4;	->
-	- 왼쪽의 피연산 값에서 오른쪽의 피연산자 값을 뺀다. - 예) int val = 5 – 4;	->
*	- 두 피연산자의 값을 곱한다. - 예) int val = 3 * 4;	->
/	- 왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나눈다. - 예) int val = 6 / 2;	->
%	- 왼쪽의 피연산자 값을 오른쪽의 피연산자 값으로 나누었을 때 나머지를 얻는다. - 예) int var = 7 % 2;	->

연산자

```
class ArithOp {
   public static void main(String[] args) {
      int num1 = 7;
      int num2 = 3;

      System.out.println("num1 + num2 = " + (num1 + num2));
      System.out.println("num1 - num2 = " + (num1 - num2));
      System.out.println("num1 * num2 = " + (num1 * num2));
      System.out.println("num1 / num2 = " + (num1 / num2));
      System.out.println("num1 / num2 = " + ((double) num1 / num2);
      System.out.println("num1 % num2 = " + (num1 % num2));
   }
}
```

연산자

- 복합 대입 연산자
  - "a = a + b" 와 "a += b"는 동일한 의미
  - "a = a - b" 와 "a -= b"는 동일한 의미
  - "a = a * b" 와 "a *= b"는 동일한 의미
  - "a = a / b" 와 "a /= b"는 동일한 의미
  - "a = a % b" 와 "a %= b"는 동일한 의미

```
class CompAssignOp {
   public static void main(String[] args) {
      int a = 5;
      int b = 3;
      a = a + b;
      System.out.println(a);
      a += b;
      System.out.println(a);
   }
}
```

연산자

- 관계 연산자
  - 두 개의 피연산자 사이에서 크기 및 동등 관계를 따져주는 이항 연산자
  - 비교 연산자라고도 함
  - 연산의 결과로 boolean 자료형(true, false) 반환

연산자	기능	결합방향
<	- 예) n1 < n2 - n1이 n2보다 작은가?	->
>	- 예) n1 > n2 - n1이 n2보다 큰가?	->
<=	- 예) n1 <= n2 - n1이 n2보다 작거나 같은가?	->
>=	- 예) n1 >= n2 - n1이 n2보다 크거나 같은가?	->
==	- 예) n1 == n2 - n1과 n2가 같은가?	->
!=	- 예) n1 != n2 - n1과 n2가 다른가?	->

연산자

```
class RelationalOp {
   public static void main(String[] args) {
      int num1 = 7;
      int num2 = 3;

      System.out.println("num1 >= num2 : " + (num1 >= num2));
      System.out.println("num1 <= num2 : " + (num1 <= num2));
      System.out.println("num1 == num2 : " + (num1 == num2));
      System.out.println("num1 != num2 : " + (num1 != num2));
   }
}
```

연산자

- 논리 연산자
  - 연산의 결과로 boolean 자료형(true, false) 반환

연산자	기능	결합방향
&&	- 예) a > b && c < b - "a > b"와 "c < b"의 결과가 모두 true이면 true 반환 - "a > b"와 "c < b"의 결과가 하나라도 false이면 false 반환	->
||	- 예) a > b || c < b - "a > b"와 "c < b"의 결과가 하나라도 true이면 true 반환 - "a > b"와 "c < b"의 결과가 모두 false이면 false 반환	->
!	- 예) !(a > b) - "a > b"가 true이면 false 반환 - "a > b"가 false이면 true 반환	<-

연산자

```
class RelationalOp1 {
   public static void main(String[] args) {
      int num1 = 11;
      int num2 = 22;
      boolean result;

      // 변수 num1에 저장된 값이 1과 100 사이의 수인가?
      result = (num1 > 1) && (num1 < 100);
      System.out.println("1 초과 100 미만인가? " + result);

      // 변수 num2에 저장된 값이 2 또는 3의 배수인가?
      result = ((num2 % 2) == 0) || ((num2 % 3) == 0);
      System.out.println("2 또는 3의 배수인가? " + result);

      // 변수 num1이 0인가?
      result = !(num1 != 0);
      System.out.println("0 인가? " + result);
   }
}
```

연산자

- SCE : 연산의 특성 중 하나로 연산의 효율 및 속도를 높이기 위해 불필요한 연산 생략

```
class SCECheck {
    public static void main(String[] args) {
        int num1 = 0;
        int num2 = 0;
        boolean result;

        result = ((num1 += 10) < 0) && ((num2 += 10) > 0);
        System.out.println("result = " + result);
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2 + "₩n");

        result = ((num1 += 10) > 0) || ((num2 += 10) > 0);
        System.out.println("result = " + result);
        System.out.println("num1 = " + num1);
        System.out.println("num2 = " + num2 + "₩n");
    }
}
```

연산자

- 부호 연산자 : +, -는 덧셈과 뺄셈을 하는 연산자이지만 부호를 나타내기도 함

```
class UnaryAddMin {
    public static void main(String[] args) {
        int num1 = 7;
        int num2 = +num2;
        int num3 = -num2;
        System.out.println(num1)
        System.out.println(num2);
        System.out.println(num3);
    }
}
```

연산자

- 전위 증감 연산자 : Prefix ++, Prefix --

연산자	기능	결합방향
++	- 피연산자에 저장된 값을 1 증가 - 예) val = ++n;	<-
--	- 피연산자에 저장된 값을 1 감소 - 예) val = --n;	<-

```
class PrefixOp {
   public static void main(String[] args) {
      int num = 7;
      System.out.println(++num);
      System.out.println(++num);
      System.out.println(num);
   }
}
```

연산자

- 후위 증감 연산자 : Postfix ++, Postfix --

연산자	기능	결합방향
++	- 피연산자에 저장된 값을 1 증가 - 예) val = n++;	<-
--	- 피연산자에 저장된 값을 1 감소 - 예) val = n--;	<-

```
class PostfixOp {
   public static void main(String[] args) {
      int num = 5;
      System.out.println(num++);
      System.out.println(num++);
      System.out.println(num--);
      System.out.println(num--);
   }
}
```

연산자

- 비트 연산자

연산자		기능	결합방향
&	-	비트 단위로 AND 연산, 예) n1 & n2;	->
|	-	비트 단위로 OR 연산, 예) n1 | n2;	->
^	-	비트 단위로 XOR 연산, 예) n1 ^ n2;	->
~	-	피 연산자의 모든 비트를 반전시켜서 얻은 결과 반환, 예) ~n;	->

```
class BitOpMeans {
   public static void main(String[] args) {
      int num1 = 13;
      int num2 = 7;
      int num3 = num1 & num2
      System.out.print(num3);
   }
}
```

연산자

- 비트 연산자

연산자		기능	결합방향
&	-	비트 단위로 AND 연산, 예) n1 & n2;	->
|	-	비트 단위로 OR 연산, 예) n1 | n2;	->
^	-	비트 단위로 XOR 연산, 예) n1 ^ n2;	->
~	-	피 연산자의 모든 비트를 반전시켜서 얻은 결과 반환, 예) ~n;	->

```
class BitOpMeans1 {
   public static void main(String[] args) {
      int num1 = 5;          // 00000101
      int num2 = 3;          // 00000011

      System.out.print(num1 & num2);
      System.out.print(num1 | num2);
      System.out.print(num1 ^ num2);
      System.out.print(~num1);
   }
}
```

연산자

- 비트 쉬프트 연산자

연산자	기능	결합방향
<<	- 피연산자의 비트 열을 왼쪽으로 이동 - 이동에 따른 빈 공간은 0으로 - 예) n << 2; -> n의 비트 열을 두 칸 왼쪽으로 이동 시킨 결과 반환	->
>>	- 피연산자의 비트열을 오른쪽으로 이동 - 이동에 따른 빈 공간은 음수인 경우 1, 양수의 경우 0으로 채움 - 예) n >> 2; -> n의 비트열을 두 칸 오른쪽으로 이동 시킨 결과 반환	->

```
class BitOpMeans2 {
   public static void main(String[] args) {
      int num = 2;// 00000010
      System.out.print(num << 1);
      System.out.print(num << 2);
      System.out.print(num >> 1);
      System.out.print(num >> 2);
   }
}
```

연산자

1. int형 변수 num1, num2, num3에 각각 10, 20, 30이 저장된 상태에서 "num1 = num2 = num3;" 코드를 실행하고 num1, num2, num3에 저장된 값을 출력하는 프로그램 코드를 작성하시오.
2. {(25 * 5) + (36 – 4) – 72} / 5 의 계산 결과를 출력하는 프로그램을 작성해 보시오.
3. 다음 계산 결과를 출력하는 프로그램을 작성하되, 덧셈 연산의 횟수를 최소화하여 작성해 보시오.

   3 + 6 3 + 6 + 9 3 + 6 + 9 + 12
4. 변수 n1, n2, n3가 다음과 같을 때 n1 > n2 > n3 이면 true, 아니면 false를 출력하는 프로그램을 작성하시오.

   n1 = {(25 + 5) + (36 / 4) - 72} * 5

   n2 = {(25 * 5) + (36 - 4) + 71} / 4

   n3 = (128 / 4) * 2