JAVA 프로그래밍

연산자(Operator)

연산자(Operator)

- 연산자(Operator) : 어떠한 기능을 수행하는 기호
- 피연산자(Operand) : 연산자의 작업 대상(변수, 상수, 리터럴, 수식)
- 연산자의 종류

① 단항 연산자 : -(음수), ++, --, ~, !

② 이항 연산자

- 산술 : +, -, *, /, %, <<, >>
- 비교 : >, <, >=, <=, ==, !=
- 논리 : &&,||, &, ^, |

③ 삼항 연산자 : 조건식 ? 식1 : 식2

④ 대입 연산자 : =

연산자의 우선순위

종 류	연산방향	연산자	우선순위
단항 연산자	←	++ -- + - ~ ! (타입)	높음
산술 연산자	→	* / %	
	→	+ -	
	→	<< >> >>>	
비교 연산자	→	< > <= >= instanceof	
	→	== !=	
논리 연산자	→	&	
	→	^	
	→	|	
	→	&&	
	→	||	
삼항 연산자	→	?:	
대입 연산자	←	= *= /= %= += -= <<= >>= >>>= &= ^= |=	낮음

연산자의 우선순위

- 괄호의 우선순위가 제일 높다.
- 산술 > 비교 > 논리 > 대입
- 단항 > 이항 > 삼항
- 연산자의 연산 진행방향 : (→) 왼쪽에서 오른쪽
- 단항 연산자와 대입 연산자의 진행방향 : (←) 오른쪽에서 왼쪽

  * 예) 3*4*5 : 3*4가 먼저 계산되고, 3*4의 결과인 12*5가 계산된다.
  * 예) x=y=3 : y에 3이 먼저 저장되고, y의 값(3)이 x에 저장된다.
  * 예) -x+3 : 단항이 이항보다 먼저 계산된다.
  * 예) x+3*y : 곱셈과 나눗셈이 덧셈과 뺄셈보다 먼저 계산된다.
  * 예) x+3>y-2 : 산술 연산자가 비교 연산자보다 먼저 계산된다.
  * 예) x>3&&x<5 : 비교 연산자가 논리 연산자보다 먼저 계산된다.
  * 예) int result=x+y*3; 대입 연산자는 가장 마지막에 대입된다.

연산자의 우선순위

- **주의해야 할 연산자의 우선순위**

아래와 같은 수식은 괄호를 사용하여 우선순위를 구분하는 것이 좋다.

① 쉬프트 연산자(<<, >>)는 덧셈 연산자보다 우선순위가 낮다.

예) x<<2+1 → x << (2+1)과 같다.

② ||, |(OR)는 &&, &(AND)보다 우선순위가 낮다.

예) x<-1 || x>3 && x<5 → x<-1 || (x>3 && x<5)와 같다.

실습 - 산술 연산자

```
import java.util.Scanner;  // 입력을 받기 위한 패키지  임포트
public class OperatorTest1 {
public static void main(String[] args) {
   Scanner input = new Scanner(System.in);    // 값을 입력 받기 위한 객체 생성
   int x, y;
   int sum = 0;

   System.out.print("첫 번째 정수 입력 : ");
   x = input.nextInt();          // 정수를 입력 받아 변수 x에 저장
   System.out.print("두 번째 정수 입력 : ");
   y = input.nextInt();          // 정수를 입력 받아 변수 y에 저장

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

증감 연산자

- 증가 연산자(++) : 피연산자의 값을 1 증가시킨다.
- 감소 연산자(--) : 피연산자의 값을 1 감소시킨다.

전위형	**j = ++i;**	++i; j = i;	값이 참조되기 전에 증가시킨다.
후위형	**j = i++;**	j = i; i++;	값이 참조된 후에 증가시킨다.

예) int i = 2;

int j = 0;

전위형 : j = ++i; → i=3, j=3

후위형 : j = i++; → i=3, j=2

※ ++i와 i = i+1은 같은 결과를 얻지만, ++i가 연산 속도가 더 빠르고,

코드를 간결하게 작성할 수 있기 때문에 더 많이 사용한다.

실습 - 증감 연산자

```
public class OperatorTest2 {
   public static void main(String[] args) {
     int num1 = 10, num2 = 10;
     int a, b;

     a = ++num1;       // 전위 방식, 먼저 증가 후 연산
     System.out.println("a=" + a + ", num1=" + num1);   // 11, 11

     b = num2++ ;      // 후위 방식, 먼저 연산 후 증가
     System.out.println("b=" + b + ", num2=" + num2);   // 10, 11
   }
}
```

비트 전환 연산자

- 비트 연산자 : ~
- 2진수 표현에서 1은 0으로, 0은 1로 바꾸는 연산자
- 정수형에서만 사용 가능

〈양의 정수를 음의 정수로 표현하는 방법〉

2진수	10진수
0 0 0 0 1 0 1 0	10
1 1 1 1 0 1 0 1	-11
1 1 1 1 0 1 0 1	-11
0 0 0 0 0 0 0 1	+) 1
1 1 1 1 0 1 1 0	-10

이항 연산자

- 이항 연산자의 특징
  : **이항 연산자는 연산을 수행하기 전에 피연산자의 타입을 일치시킨다.**
- int 보다 크기가 작은 타입은 int로 변환한다 : (byte, char, short) → int
- **피연산자 중 표현범위가 큰 타입으로 형 변환한다.**

  byte + short → int

  char + int → int

  float + int → float

  long + float → float

  float + double → double

(예) byte + byte → int

```
byte a = 10;
byte b = 20;
byte c = a + b;        // 오류
int c = a + b;         // 정상
```

실습 - 이항 연산자

```
class  OperatorTest3 {
   public static void main(String[] args) {
      byte b1 = 10;
      byte b2 = 20;
      char c = 'A';
      int i = 10;
      float f = 3.14f;

      //byte b3 = b1+b2;            // 오류
      int b3 = b1+b2;     // 정상

      //byte b4 = (byte)b1+b2;    // 오류
      byte b4 = (byte)(b1+b2);    // 정상

      //char ci1 = c+i;             // 오류
      char ci2 = (char)(c+i);       // 정상
      int ci = (int)c+i;            // 정상

      //int fi1 = f+i;          // 오류
      int fi2 = (int)f+I;     // 정상
      float fi = f+i;         // 정상

      System.out.println("b1의 값: " + b1);
      System.out.println("b2의 값: " + b2);
      System.out.println("b3의 값: " + b3);
      System.out.println("b4의 값: " + b4);
      System.out.println("ci의 값: " + ci);
      System.out.println("ci2의 값: " + ci2);
      System.out.println("fi의 값: " + fi);
      System.out.println("fi2의 값: " + fi2);
   }
}
```

쉬프트 연산자

- 쉬프트 연산자 : <<(왼쪽 쉬프트), >>(오른쪽 쉬프트)
- 쉬프트 연산자는 2의 n승으로 곱하거나 나눈 결과와 같다.
- 사칙연산의 곱셈이나 나눗셈보다 연산 속도가 빠르다.

예) $x<<n = x*2^n$

$x>>n = x/2^n$

$8<<2 = 8*2^2$

$8>>2 = 8/2^2$

실습 - 쉬프트 연산자

```
class  OperatorShift  {
  public static void main(String[] args)  {
      int a=3;
      int b=12;

      System.out.println("a의 값: " + a);
      System.out.println("a<<1의 값: " + (a<<1));        // 3*2^1
      System.out.println("a<<2의 값: " + (a<<2));        // 3*2^2
      System.out.println("a<<3의 값: " + (a<<3));        // 3*2^3

      System.out.println("b의 값: " + b);
      System.out.println("b>>1의 값: " + (b>>1));        // 12/2^1
      System.out.println("b>>2의 값: " + (b>>2));        // 12/2^2
      System.out.println("b>>3의 값: " + (b>>3));        // 12/2^3
  }
}
```

Quiz - 쉬프트 연산자

〈 OperatorShiftExample.java〉

① a = 2라는 변수에 515을 곱하는 연산을 쉬프트 연산자를 사용하여 구하시오.

② b = 128이라는 변수를 32로 나누는 연산을 쉬프트 연산자를 사용하여 구하시오.

예) a = 2라는 변수에 320을 곱하는 쉬프트 연산 : (a<<8)+(a<<6)

[실행 결과]

2 * 515 = 1030

128 / 32 = 4

비교 연산자

- 비교 연산자 : >, <, >=, <=, ==, !=
- 피연산자를 같은 타입으로 변환한 후에 비교한다.
- 결과 값은 true 또는 false이다.
- 기본형(boolean제외)과 참조형에 사용할 수 있으나, 참조형에는 '=='와 '!='만 사용할 수 있다.

수 식	연 산 결 과
x > y	x가 y보다 클 때 true, 그 외에는 false
x < y	x가 y보다 작을 때 true, 그 외에는 false
x >= y	x가 y보다 크거나 같을 때 true, 그 외에는 false
x <= y	x가 y보다 작거나 같을 때 true, 그 외에는 false
x == y	x와 y가 같을 때 true, 그 외에는 false
x != y	x와 y가 다를 때 true, 그 외에는 false

실습 - 비교 연산자

```
class OperatorCompare {
   public static void main(String[] args)   {
    System.out.println(" 'A'<'B'의 결과: " + ('A'<'B'));
    System.out.println(" '0'==0의 결과: " + ('0'==0));
    System.out.println(" 'A'!=65의 결과: " + ('A'!=65));
    System.out.println(" 10.0d==10.0f의 결과: " + (10.0d==10.0f));

   // 실수는 정수처럼 정확한 값이 아닌 근사값으로 표현되기 때문에 float를 double로
      형 변환했을 때 값이 달라질 수 있다. 굉장히 작은 오차(1억 분의 1 정도)이지만
     비교 연산자로 비교했을 경우에는 다른 값이기 때문에 결과는 false이다.
    System.out.println(" 0.1d==0.1f의 결과: " + (0.1d==0.1f));

   // float형과 double형을 비교할 때는 float을 double로 형 변환 하는 것이 아니라
      오히려 double을 float으로 형 변환한 다음에 비교해야 한다.
    System.out.println(" (float)0.1d==0.1f의 결과: " + ((float)0.1d==0.1f));
   }
}
```

비트 연산자

- **비트 연산자 : &, |, ^**
- 피연산자를 비트 단위로 연산하는 연산자
- 실수형(float, double)을 제외한 모든 기본형에 사용 가능하다.
- OR연산자(|) : 피연산자 중 어느 한 쪽이 1이면 1이다.
- AND연산자(&) : 피연산자 양 쪽 모두 1이면 1이다.
- XOR연산자(^) : 피연산자가 서로 다를 때 1이다.

x	y	x | y	x & y	x ^ y
1	1	1	1	0
1	0	1	0	1
0	1	1	0	1
0	0	0	0	0

비트 연산자

식	2진수	10진수
3 | 5 = 7	0 0 0 0 0 0 1 1	3
	|) 0 0 0 0 0 1 0 1	5
	0 0 0 0 0 1 1 1	7
3 & 5 = 1	0 0 0 0 0 0 1 1	3
	&) 0 0 0 0 0 1 0 1	5
	0 0 0 0 0 0 0 1	1
3 ^ 5 = 6	0 0 0 0 0 0 1 1	3
	^) 0 0 0 0 0 1 0 1	5
	0 0 0 0 0 1 1 0	6

논리 연산자

- **논리 연산자 : &&, ||**
- 피연산자가 반드시 boolean이어야 하며 연산결과도 boolean이다.
- &&가 || 보다 우선순위가 높다. 같이 사용하는 경우에는 괄호로 묶어야 한다.
- OR연산자(||) : 피연산자 중 하나 이상이 true이면 true이다.
- AND연산자(&&) : 피연산자 모두 true이면 true이다.

x	y	x **||** y	x **&&** y
true	true	true	true
true	false	true	false
false	true	true	false
false	false	false	false

실습 - 논리 연산자

```
class OperatorLogic  {
   public static void main(String[] args)  {
      int i = 7;
      char x = 'j';

      System.out.println(" i>3 && i<5의 결과 : " + (i>3 && i<5));

      System.out.println(" i>3 || i<0의 결과 : " + (i>3 || i<0));

      System.out.println( "(x>='a' && x<='z') || (x>='A' && x<='Z')의 결과 : "
                              + ((x>='a' && x<='z') || (x>='A' && x<='Z')) );
   }
}
```

삼항 연산자

- **삼항 연산자: 조건식 ? 식1 : 식2**
- 조건식의 연산 결과가 true이면 '식1'의 결과를 반환하고, false이면 '식2'의 결과를 반환한다. (※ if ~ else문과 같은 형식)

```
class ConditionalOperator {
  public static void main(String[] args)  {
    int x = -10;
    int y = 50;
    int absX = x>=0 ? x : -x;
    char grade = y>=90 ? 'A' : y>=80 ? 'B' : 'C';

    System.out.println("x의 절대값: " +absX);
    System.out.println("grade의 값: " +grade);
  }
}
```

대입 연산자

- **대입 연산자 : =(단순 대입), op=(결합 대입, 복합 대입 연산자)**
- 오른쪽 피연산자의 값을 왼쪽 피연산자에 저장한다.
- 왼쪽 피연산자는 상수가 아니어야 한다.

예1) int i = 0;

i = i+3;

예2) final int MAX = 3;

MAX = 10; // 오류(상수)

op=	=
i +=3;	i = i + 3;
i -= 3;	i = i - 3;
i *= 3;	i = i * 3;
i /= 3;	i = i / 3;
i %= 3;	i = i % 3;
i <<= 3;	i = i << 3;
i >>= 3;	i = i >> 3;
i >>>= 3;	i = i >>> 3;
i &= 3;	i = i & 3;
i ^= 3;	i = i ^ 3;
i |= 3;	i = i | 3;
i *= 10 + j;	i = i * (10+j);

Quiz - 삼항 연산자(1)

〈 ConditionalOperatorQuiz1.java〉 점수를 입력 받은 후, 그 점수를 비교하여 실행 결과 예시와 같이 수, 우, 미, 양, 가로 출력되도록 코딩하세요.(삼항 연산자를 사용)

[실행결과 예시]

점수를 입력하세요 :

score〉=90 → grade : 수

score〉=80 → grade : 우

score〉=70 → grade : 미

score〉=60 → grade : 양

그 외 나머지 → grade : 가

Quiz - 삼항 연산자(2)

〈 ConditionalOperatorQuiz2.java〉 연도를 입력 받아서 윤년인지를 검사하는 소스 코드를 작성하세요.(삼항 연산자 사용)

- 윤년의 조건 : 4의 배수이면서 동시에 100의 배수가 아니어야 한다.
  또는 400의 배수이어야 한다.
- 삼항 연산자의 결과 : 윤년의 조건이 만족하면 "윤년입니다."를 출력,
  그렇지 않으면 "윤년이 아닙니다."를 출력

[실행 결과]

연도를 입력하세요 : 2019

윤년입니다. (또는 윤년이 아닙니다.)