Operator

1. 연산자의 우선 순위와 종류

❖연산(operations): 프로그램에서 데이터를 처리하여 결과를 만들어내는 동작

- ✓ 연산자(Operator): 연산에 사용되는 기호나 명령어
- ✓ 피연산자(Operand): 연산에 사용되는 데이터
- ✓ 연산식(Expression): 연산자와 피연산자를 이용하여 연산의 과정을 기술한 것

❖연산의 분류

●연산의 결과에 따른 분류

- ●산술연산: 연산의 결과가 숫자로 리턴되는 연산
- ●논리연산: 연산의 결과과 boolean(true or false)로 리턴되는 연산

●피 연산자의 개수에 따른 연산자의 종류

- ✓ 단항 연산: 피 연산자가 1개인 연산자
- ✓ 이항 연산: 피 연산자가 2개인 연산자
- ✓ 삼항 연산: 피 연산자가 3개인 연산자

1. 연산자의 우선 순위와 종류

- 용도에 따른 연산자 및 우선 순위

종류	연산자	결합 방향
최우선	(), []	->
단항 연산자	++, --, +, -, (type), !, ~	<-
산술 연산자	%, /, *, +, -	->
시프트 연산자	>>, <<, >>>	->
관계 연산자	<, <=, >, >=, instanceof, ==, !=	->
논리 연산자	&, ^, |, ~ &&, ||	->
조건(삼항) 연산자	? :	<-
대입 연산자	=, +=, -=, *=, %=, /=, >>=, <<=	<-
콤마 연산자	,	

2. 최우선 연산자

1) ( ) 연산자
우선 순위 변경을 위해 사용

2) [ ] 연산자
배열의 크기나 첨자를 나타낼 때 사용

3.단항(Unary) 연산자

1)~ 틸드

- ✓ 1의 보수를 구해주는 연산자로 정수 데이터에만 사용이 가능
- ✓ 모든 비트의 값을 반대로 표현해서 정수로 리턴
- ✓ 양수의 1의 보수는 부호는 음수가 되고 절대값은 1증가
- ✓ 음수의 1의 보수는 부호는 양수가 되고 절대값은 1감소

2)! – 논리 부정 연산자

- ✓ boolean 데이터에만 사용이 가능
- ✓ NOT(true -> false로 false -> true)
- ✓ !true -> false

3)부호 +, -

- ✓ +: 양수를 의미하는 것으로 형식적으로 제공
- ✓ -: 2의 보수 연산자, 부호 바꿈 연산자, 연산의 기본 단위가 int

실습(Unary.java)

```
public class Unary {
	public static void main(String args[]) {
		int a = 10, b = -10;
		boolean c = true, d = false;
		System.out.println("a= " + a + "₩t b=" + b + "₩t c=" + c + "₩t d=" + d);
		System.out.println("~a=" + ~a);
		// -11 1의 보수 연산(양수-> 음수 부호가 바뀌고 1증가됨)
		System.out.println("~b=" + ~b);
		// 9 음수-> 양수 부호가 바뀌고 1감소됨
		System.out.println("!c=" + (!c) + "₩t !d = " + (!d));
		System.out.println("+a=" + (+a) + "₩t +b = " + (+b));
		System.out.println("-a= " + (-a) + "₩t -b = " + (-b));
	}
}
```

```
a= 10 b=-10 c=true d=false
~a=-11
~b=9
!c=false !d = true
+a=10 +b = -10
-a= -10 -b = 10
```

3.단항(Unary) 연산자

4) 증감연산자: ++, --

- ✓ 정수 타입의 변수에만 사용할 수 있는 연산자
- ✓ 단항 연산자는 피 연산자의 왼쪽에 위치하지만 ++와 --연산자는 왼쪽과 오른쪽 모두 가능
- ✓ ++: 변수의 값을 1 증가
- ✓ --: 변수의 값을 1 감소
- ✓ 증감 연산자는 어느 위치에 놓는가에 따라서 수행 결과가 다름
- ✓ 증감 연산자를 앞에 놓으면 전위 연산(prefix)이라 하고 뒤에 놓으면 후위 연산(postfix)
- ✓ 전위 연산은 증감을 먼저 하고 명령에 데이터를 사용하고 후위 연산은 명령에 데이터를 사용하고 증감

실습(Fix.java)

```
public class Fix {
    public static void main(String args[]) {
        int a=10;
        System.out.println("a = " + a);
        System.out.println("후위 연산 - a = " + a++);
        System.out.println("a = " + a);
        // 증감을 하기 전에 명령문을 먼저 처리하므로 10 출력
        // 연산 후의 결과는 11
        a = 10;

        System.out.println("전위 연산 - a = " + ++a);
        System.out.println("a = " + a);
        // 증감을 한 후에 명령문을 처리하므로 11 출력
        // 연산 후의 결과는 11
    }
}
```

```
a = 10
후위 연산 - a = 10
a = 11
전위 연산 - a = 11
a = 11
```

4.산술 연산자

❖사칙연산자(+, -, *, /), 나머지 연산자(%)는 두 개의 피 연산자를 가지는 이항 연산자이며 산술 연산자는 피 연산자의 크기가 int보다 작으면 int(4byte)로 변환한 다음에 연산을 수행

❖산술 연산자는 연산을 수행하기 전에 피 연산자들의 데이터의 자료형을 일치 시킨 후 연산을 하며 연산의 결과도 변경한 자료형으로 리턴

❖서로 다른 자료형 사이의 연산의 경우에는 더 큰 자료형으로 자료형을 변경해서 연산을 수행

❖정수와 정수의 나눗셈을 하면 결과가 정수로 나와야 하므로 몫만 나오게 되며 소수는 버림

❖% 연산자는 정수 사이의 연산에서만 의미를 갖음

❖% 연산자는 주기적인 반복 작업이나 Transcription Error 제거를 위한 곳에서 사용

❖양수의 나머지는 양수만 나오지만 음수의 나머지를 구하게 되면 양수로 나오지 않고 음수로도 출력됨

실습(Error.java)

```
public class Error {
	public static void main(String[] args) {
		byte a = 10;
		byte b = 20;
		byte c = a + b;

		// 이 때 a 와 b가 모두 int 형으로 변환되어 연산하고
		// 따라서 연산의 결과도 int 형이 되므로 대입 불가능
		// byte c = (byte)(a + b); 으로 수정하던지
		// int c = a + b;로 수정하면 에러가 없어짐
		System.out.println(c);
	}
}
```

실습(Arithmetic.java)

```
public class Arithmetic {
	public static void main(String args[]) {
		int a = 11, b = 2;
		int add = a + b;
		int sub = a - b;
		int mul = a * b;
		//정수 와 정수 연산이므로 결과는 정수
		int div = a / b;
		//나머지 연산
		int mod = a % b;

		System.out.println("a+b=" + add);
		System.out.println("a-b=" + sub);
		System.out.println("a*b=" + mul);
		System.out.println("a/b=" + div);
		System.out.println("a/b=" + (float) a / b);
		System.out.println("a%b=" + mod);
```

```
a+b=13
a-b=9
a*b=22
a/b=5
a/b=5.5
a%b=1
5%-2=1
-5%2=-1
-5%-2=-1
```

실습(Arithmetic.java)

```
        //양수와 음수 사이에서의 나머지
        System.out.println("5%-2=" + (5%-2));
        //음수와 양수 사이에서의 나머지
        System.out.println("-5%2=" + (-5%2));
        //음수와 음수 사이에서의 나머지
        System.out.println("-5%-2=" + (-5%-2));
    }
}
```

4.산술 연산자

- ❖산술연산을 수행할 때 되도록 이면 실수를 사용하지 않는 것이 바람직
- ❖컴퓨터는 2진수를 사용하기 때문에 소수는 정확하게 표현하지 못하는 경우가 발생
- ❖실수 연산을 수행하는 경우에는 실수를 정수로 변환해서 연산을 하는 것이 바람직
- ❖/나 %연산을 할 때는 나누는 수의 값을 확인
  - ✓ 정수의 경우는 0으로 나누게 되면 예외가 발생
  - ✓ 실수의 경우는 예외가 발생하지 않고 Infinity 나 NaN의 값이 나와서 다음 연산을 수행할 때 잘못된 결과가 리턴
  - ✓ Infinity나 NaN 데이터와 산술 연산을 하면 결과는 Infinity나 NaN

실습(DoubleProblem.java)

```
public class DoubleProblem {
	public static void main(String[] args) {
		// 실수 연산 - 연산의 결과가 이상
		double d = 0.3;
		double result = 1 - 3 * d;
		System.out.println("결과:" + result);

		//정수로 변환한 후 연산
		result = (1 * 10 - (d * 10 * 3)) / 10.0;
		System.out.println("결과:" + result);

		int x = 5;
		double y = 0.0;
		//0으로 나누는 것이 에러가 아닐 수도 있음
		double r1 = x / y;
		double r2 = x % y;

		//연산의 결과가 Infinity 나 NaN
		System.out.println(r1);
		System.out.println(r2);
		System.out.println(r1 + 2);
	}
}
```

```
결과:0.10000000000000009
결과:0.1
Infinity
NaN
Infinity
```

5.Shift 연산자

❖<<, >>, >>>

- ✓ shift 연산자는 정수형 데이터에만 사용할 수 있으며 피 연산자의 각 자리(2진수로 표현했을 때)를 오른쪽 또는 왼쪽으로 이동(shift)
- ✓ 오른쪽으로 n 비트를 이동하면 피 연산자를 2의 n승으로 나눈 것과 같은 결과를 얻을 수 있고 왼쪽으로 n자리를 이동하면 2의 n승으로 곱한 것과 같은 결과
- ✓ 왼쪽으로 shift 할 때는왼쪽의 첫번째 비트부터 n개를 제거하고 맨 뒷 자리에 0을 n개 만큼 삽입
- ✓ 오른쪽으로 shift 할 때는 첫번째 비트가 삽입
- ✓ 반면에 " >>> " 연산자는 첫번재 비트부터 shift 하고 앞의 빈 자리에 0으로 빈자리를 채워줍니다.
- ✓ 음수에 ">>>"연산을 수행하면 양수로 결과가 리턴
- ✓ shift 연산에서 연산을 수행하는 비트 수 n의 값이 32보다 크면 32로 나눈 나머지 만큼만 이동
- ✓ int형의 경우 4byte(32bit)이므로 1000 >> 32는 수행하면 아무 일도 하지 않음
- ✓ 1000 >> 35는 35를 32로 나눈 나머지인 3만큼만 이동하는 1000 >> 3을 수행

실습(Shift.java)

```
8
1000
8 << 2 = 32
100000
8 << 28 = -2147483648
10000000000000000000000000000000
-8
11111111111111111111111111111000
-8 << 2 = -32
11111111111111111111111111100000
-8
11111111111111111111111111111000
-8 >> 2 = -2
11111111111111111111111111111110
-8
11111111111111111111111111111000
-8 >>> 2 = 1073741822
11111111111111111111111111111110
4 << 2 = 16
10000
```

실습(Shift.java)

```
public class Shift {
	public static void main(String args[]) {
		// Integer.toBinaryString은 매개변수를 2진 문자열로 변경해주는 메소드

		int temp; // 계산 결과를 담기 위한 변수

		System.out.println(8);
		System.out.println(Integer.toBinaryString(8));
		//2번 왼쪽으로 shift 하면 4배가 되서 32
		temp = 8 << 2;
		System.out.println("8 << 2 = " + temp);
		System.out.println(Integer.toBinaryString(temp));

		//28번을 밀면 1이 맨 앞으로 이동을 하게 되서 음수
		temp = 8 << 28;
		System.out.println("8 << 28 = " + temp);
		System.out.println(Integer.toBinaryString(temp));
```

실습(Shift.java)

```
System.out.println(-8);
System.out.println(Integer.toBinaryString(-8));

temp = -8 << 2;
System.out.println("-8 << 2 = " + temp);
System.out.println(Integer.toBinaryString(temp));

System.out.println(-8);
System.out.println(Integer.toBinaryString(-8));

//오른쪽으로 shift 하면 첫번째 비트가 맨 앞에 추가
temp = -8 >> 2;
System.out.println("-8 >> 2 = " + temp);
System.out.println(Integer.toBinaryString(temp));

System.out.println(-8);
System.out.println(Integer.toBinaryString(-8));
```

실습(Shift.java)

```
        //>>>은 무조건 0이 맨 앞에 추가되므로 음수를 가지고 하게 되면 양수로 변경
        temp = -8 >>> 2;
        System.out.println("-8 >>> 2 = " + temp);
        System.out.println(Integer.toBinaryString(temp));

        //33번 shift 인 경우 34가 32보다 크거나 같으므로 32로 나눈 나머지인 2번만 수행
        temp = 4 << 34;
        System.out.println("4 << 2 = " + temp);
        System.out.println(Integer.toBinaryString(temp));
    }
}
```

6. 관계 연산자

❖관계 연산자는 이항 연산자로 피 연산자들을 비교하는 연산자

- ✓ 연산의 결과는 boolean(true 아니면 false)으로 리턴
- ✓ <, >, <=, >=
  - 두 피 연산자의 크기를 비교하는 연산자
  - 기본형 중에서는 boolean형을 제외한 나머지 자료형에 사용할 수 있고 참조형에는 사용할 수 없음

- ✓ ==, !=
  - 두 피 연산자에 저장되어 있는 값이 같은지 또는 다른지를 비교하는 연산자
  - 대소 비교연산자(<, >, <=, >=)와는 달리 기본형은 물론 참조형에 사용 가능
  - 기본형의 경우 변수에 저장되어 있는 값이 같은지를 알 수 있고 참조형의 경우 객체의 참조(해시) 값을 저장하기 때문에 두 개의 피 연산자(참조변수)가 같은 메모리의 참조를 가리키는 지 확인 가능
  - 참조형 데이터의 내용을 비교할 때는 equals() 메소드를 사용
  - 기본형과 참조형 간에는 저장되는 데이터의 종류가 다르기 때문에 등가비교 연산자(==, !=)의 피 연산자로 기본형과 참조형을 함께 사용할 수는 없음
  - boolean은 다른 자료형과 비교할 수 없음

- ✓ instanceof: 참조형의 경우 어떤 타입인지 확인 해주는 연산자

실습(Relation.java)

```
public class Relation
{
	public static void main(String[] args)
	{
		boolean pan;
		int n = 10;
		float f = 10.0f;
		pan = 10 > 3;
		System.out.println("pan:" + pan);
		pan = 10 == 3;
		System.out.println("pan:" + pan);
		//서로 다른 자료형이지만 숫자 타입이므로 비교가능
		pan = n == f;
		System.out.println("pan:" + pan);
		//자료형이 다르므로 다르다고 리턴
		pan = 0.3f == 0.3;
		System.out.println("pan:" + pan);
	}
}
```

```
pan:true
pan:false
pan:true
pan:false
```

7.논리 연산자

❖논리 연산자는 이항 연산자로 조건의 참, 거짓을 비교하는 조건 논리 연산자와 비트 단위로 연산을 수행해서 결과를 정수로 리턴하는 비트 논리 연산자로 나눔

❖||, && 연산자(조건 논리 연산자)

- ✓ ||, && 연산자는 피 연산자로 boolean형 또는 boolean형 값을 결과로 하는 연산식이나 메소드 호출 만을 허용
- ✓ 조건문과 반복문에서 조건식 간의 결합에 사용
- ✓ "&&"가 "||" 연산보다 우선순위가 높으므로 한 조건식에 "&&"와 "||" 가 함께 사용될 때는 괄호를 사용해서 우선순위를 명확히 해주는 것이 좋음
- ✓ || (OR결합) - 피 연산자 중 한 쪽만 true이면 true를 결과로 얻게 됨
- ✓ &&(AND결합) - 피 연산자 양쪽 모두 true이어야 true를 결과로 얻게 됨
- ✓ ||의 앞쪽 조건이 참이 되면 뒤의 조건은 조사할 필요가 없어지며 &&의 앞쪽 조건이 거짓이면 뒤의 조건은 조사할 필요가 없어집니다. Don't Care라고도 함

실습(Logical.java)

```
public class Logical
{
	public static void main(String[] args)
	{
		boolean pan;
		int ten = 10;
		int three = 3;
		pan = ten > three && three < ten;
		System.out.println("pan:" + pan);
		pan = ten > three || ten++ < three;
		System.out.println("pan:" + pan);
		//|| 조건에서 앞의 조건이 true이므로 뒤의 조건은 확인하지 않음
		//ten 과 three의 1증가 작업은 수행되지 않음
		System.out.println("ten:" + ten);
		System.out.println("three:" + three);
	 }
}
```

```
pan:true
pan:true
ten:10
three:3
```

7.논리 연산자

❖|, &, ^, ~ 연산자(비트 논리 연산자)

- ✓ 피 연산자끼리 이진 비트연산을 수행해서 결과를 돌려주는 연산자
- ✓ 실수형인 float와 double을 제외한 모든 숫자형(정수형)에 사용 가능하며 정수끼리 연산을 수행하면 정수로 boolean끼리 연산을 하면 boolean으로 결과를 리턴
- ✓ |(OR연산자) - 피 연산자 중 한 쪽의 값이 1이면 1을 결과로 1을 얻고 그 외에는 0을 리턴
- ✓ &(AND연산자) - 피 연산자 양 쪽이 모두 1이어야 1을 결과로 얻고 그 외에는 0을 리턴
- ✓ ^(XOR연산자) - 피 연산자의 값이 서로 다를 때만 1을 결과로 얻고 그 외에는 0을 리턴
- ✓ ~(NOT 연산자) – 1의 보수를 리턴해주는 연산자

실습(Bit.java)

```
public class Bit
{
	public static void main(String[] args)
	{
		int x = 3;  //011
		int y = 5;  //101
		System.out.println("x는 " + x + "이고, y는 " + y +"일 때, ");
		//011 | 101 = 111
		System.out.println("x | y = " + (x|y));
		//011 & 101 = 001
		System.out.println("x & y = " + (x&y));
		//011 ^ 101 = 110
		System.out.println("x ^ y = " + (x^y));
		//~011 -> ~100
		System.out.println("~x  = " + ~x);

		System.out.println("true | false = " + (true|false));
		System.out.println("true & false = " + (true&false));
		System.out.println("true ^ false = " + (true^false));
	}
}
```

```
x는 3이고, y는 5일 때
x | y = 7
x & y = 1
x ^ y = 6
~x  = -4
true | false = true
true & false = false
true ^ false = true
```

8.삼항 연산자

❖삼항 연산자는 조건 연산자라고도 하며 3개의 피 연산자를 필요로 하게 되는데 첫 번째 피 연산자는 조건이며 두 번째 피 연산자는 조건이 true일 때 리턴되는 값이고 세 번째 피 연산자는 조건이 false일 때 리턴되는 값

❖기본형식
조건식? 식1: 식2

실습(Ternary.java)

```
public class Ternary {
	public static void main(String[] args) {
		int a = 10, b = 11;
		int max, min;
		//a의 값이 크다면 a 그렇지 않으면 b
		max = a > b ? a : b;
		min = a < b ? a : b;
		System.out.println("두 수 중 큰 값:" + max);
		System.out.println("두 수 중 작은 값:" + min);
	}
}
```

```
두 수 중 큰 값:11
두 수 중 작은 값:10
```

9. 대입 연산자

❖대입 연산자는 변수에 값 또는 수식의 연산결과 또는 메소드의 수행 결과를 저장하는데 사용

❖대입 연산자의 왼쪽에는 반드시 변수가 위치해야 하며 오른쪽에는 리터럴이나 변수 또는 수식 및 메소드 호출구문이 올 수 있음

❖대입 연산자는 모든 연산자들 중에서 가장 낮은 연산순위를 가지고 있기 때문에 제일 마지막에 수행

❖연산 진행 방향이 오른쪽에서 왼쪽이기 때문에 x=y=3; 에서 y=3이 먼저 수행되고 그 다음에 x=y가 수행

❖대입연산자는 다른 연산자와 결합하여 "op="와 같은 방식으로 사용될 수 있다. 예를 들면 i = i + 3은 i +=3과 같이 표현

실습(Assign.java)

```
public class Assign
{
	public static void main(String[] args)
	{
	  int add,sub,mul, div;
		add = sub = mul = div =10;
		//add = add + 10
		add +=10;
		sub -=10;
		mul *=10;
		div /=10;
		System.out.println("add:" + add);
		System.out.println("sub:" + sub);
		System.out.println("mul:" + mul);
		System.out.println("div:" + div);
	}
}
```

```
add:20
sub:0
mul:100
div:1
```

연습문제

❖2개의 변수에 저장된 값을 교환하는 swap 코드를 완성
(데이터를 순서대로 배치하는 정렬을 할 때 자주 사용되는 코드이므로 반드시 기억)
❖출력-> 콜라는 800원 사이다는 1000원

❖기본코드

```
public class Practice1 {

        public static void main(String[] args) {
                int coke = 1000;
                int cider = 800;
                //이부분에 코드를 작성해서 coke 와 cider의 값을 변경해보세요


                System.out.println("콜라는 " + coke + "원 사이다는 " + cider + "원");
        }
}
```

연습문제

❖초를 입력받아서 시분초로 변환하는 코드를 완성하세요.
(/ 연산자와 % 연산자를 사용)

- ✓ 입력-> 변환하고자 하는 초를 입력하세요: 3700
- ✓ 출력-> 입력한 3700초는 1시간 1분 40초 입니다.

❖기본코드

```
import java.util.Scanner;
public class Practice2 {
        public static void main(String[] args) {
                //콘솔로부터 입력받기 위한 객체 생성
                Scanner sc = new Scanner(System.in);
                System.out.print("변환하고자 하는 초를 입력하세요:");
                //콘솔에서 정수를 입력하면 input에 저장됩니다.
                int input = sc.nextInt();
                //이 부분에 코드를 작성해서 입력한 초가 몇시간 몇분 몇초인지 출력


                sc.close();
        }
}
```

연습문제

❖섭씨 온도를 화씨 온도로 변환하는 프로그램을 작성

(산술 연산에서 동일한 자료형의 연산은 연산을 수행한 자료형으로 나오므로 이 부분에 주의해서 연산을 해야합니다. 잘못하면 이상한 결과가 나올 수 있으므로 적절한 형 변환을 이용해서 수행)

- ✓ 화씨 온도 = 9 / 5 * 섭씨온도 + 32
- ✓ 입력->변환하고자 하는 섭씨 온도를 정수로 입력하세요:32
- ✓ 출력->섭씨 32도는 화씨로는 89.6도

❖기본코드

```
import java.util.Scanner;
public class Practice3 {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                System.out.print("변환하고자 하는 섭씨 온도를 정수로 입력하세요:");
                int celsius = sc.nextInt();
                //이 부분에 코드를 작성해서 출력하세요

                sc.close();
        }
}
```

연습문제

❖입력한 년도가 윤년인지 판별하는 코드를 완성하세요.

✓ 조건문 괄호 안을 작성해보세요.

- 윤년의 조건(아래 2개의 조건 중 하나만 만족하면 윤년입니다.)
  - 4의 배수이고 100의 배수가 아닌 경우
  - 400의 배수인 경우
- 입력->판별하고자 하는 년도를 입력하세요:2017
- 출력->2017년은 윤년이 아닙니다.

```
import java.util.Scanner;
public class Practice4 {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.print("판별하고자 하는 년도를 입력하세요:");
		int year = sc.nextInt();
		if(요기에 윤년을 판별하는 연산식을 삽입하세요)
			System.out.println(year +"년은 윤년입니다.");
		else
			System.out.println(year +"년은 윤년이 아닙니다.");
		sc.close();
	}
}
```

연습문제

❖상점에서 물건을 사고 지폐로 돈을 내면 거스름 돈을 주어야 하는데 이 때 동전을 어떻게 해서 줘야 하는지 계산하시오.

❖돈은 반드시 1000 원을 내며 거스름 돈은 10 원 , 50 원 , 100 원 동전으로 하고 큰 동전이 우선

✓ 입력

물건 값으로 세자리 자연수가 입력으로 주어집니다.

일의 자리는 0

✓ 출력

동전 100 원 , 50 원 , 10 원의 개수를 출력

✓ 입력 예

입력: 530

✓ 출력 예

거스름 돈은 470원

100원:4

50원:1

10원:2

연습문제

```
import java.util.Scanner;
public class Practice5 {
		public static void main(String[] args) {
			Scanner sc = new Scanner(System.in);
			System.out.print("물건 가격을 입력하세요:");
			int money = sc.nextInt();

			//이 부분에 코드를 작성하시오.

			sc.close();
		}
}
```