Ch2
연산자

객체지향프로그래밍(기본) 2019

경상대학교 항공우주및소프트웨어공학전공

2.4 연산자

어떠한 값에 대하여 계산하여 결과는 얻는 것을 연산이라 하며 연산에 사용되는 기호를 연산자라 한다.

-자바의 연산자 종류

- **산술 연산자**
- **증감 연산자**
- **대입 연산자**
- **비교 연산자**
- **논리 연산자**
- **비트 연산자**
- **삼항 연산자**

2.4.1 산술 연산자

산술연산자는 덧셈, 뺄셈, 곱셈, 나누기의 몫과 나머지 대한 결과를 얻기 위하여 사용한다.

연산자	사용법	의미
+	x + y	x와 y를 더한 결과 값
-	x - y	x와 y를 뺀 결과 값
*	x * y	x와 y를 곱한 결과 값
/	x / y	x와 y를 나눈 몫의 값
%	x % y	x와 y를 나눈 나머지의 값

```java
public class Example {
  public static void main(String[] args) {
    int result;

    result = 7+5;
    System.out.println("7 + 5 = " + result);

    result = 7-5;
    System.out.println("7 - 5 = " + result);

    result = 7*5;
    System.out.println("7 * 5 = " + result);

    result = 7/5;
    System.out.println("7 / 5 = " + result);

    result = 7%5;
    System.out.println("7 % 5 = " + result);
  }
}
```

결과

```
7 + 5 = 12
7 - 5 = 2
7 * 5 = 35
7 / 5 = 1
7 % 5 = 2
```

2.4.2 증감연산자

증감연산자는 하나의 항을 가지는 단항 연산자로 증감연산자가 실행될 때 항의 값을 1 증가 시키거나 감소시킬 때 사용된다.

연산자	사용법	의미
++	x++	실행 후에 x에 1을 증가
++	++x	실행 전에 x에 1을 증가
--	x--	실행 후에 x에 1을 감소
--	--x	실행 전에 x에 1을 감소

```java
public class Example {
  public static void main(String[] args) {
    int result;

    result = 10;
    System.out.println(result++);
    System.out.println(result++);
    System.out.println(result++);

    System.out.println();

    result = 10;
    System.out.println(++result);
    System.out.println(++result);
    System.out.println(++result);
  }
}
```

결과

```
10
11
12

11
12
13
```

2.4.3 대입연산자

대입연산자는 = 연산자를 사용하며 연산의 값을 저장 할 때 사용한다.

연산자	사용법	의미
+=	x += y	x에 y를 더한 결과 값을 x에 대입
-=	x -= y	x에 y를 뺀 결과 값을 x에 대입
*=	x *= y	x에 y를 곱한 결과 값을 x에 대입
/=	x /= y	x에 y를 나눈 몫의 값을 x에 대입
%=	x %= y	x에 y를 나눈 나머지의 값을 x에 대입

문제

- X=7, y=5를 대입한 후 다음의 산술연산자를 대입연산자로 바꾸어 계산하자 (교재 39페이지, 코드 2-10 참조).

  - Result = 7+5;
  - Result = 7-5;
  - Result = 7*5;
  - Result = 7/5;
  - Result = 7%5;

2.4.4 비교연산자

비교연산자는 두 개의 값을 서로 비교할 때 사용되는 연산자로 결과로 true나 false를 반환한다.

연산자	사용법	의미
==	x == y	x와 y 가 같으면 true 다르면 false
!=	x != y	x와 y 가 다르면 true 같으면 false
<	x < y	x가 y 보다 작으면 true 크면 false
>	x > y	x가 y 보다 크면 true 작으면 false
<=	x <= y	x가 y 보다 작거나 같으면 true 크면 false
>=	x >= y	x가 y 보다 크거나 같으면 true 작으면 false

```java
public class Example {
  public static void main(String[] args) {
    int x, y;

    x = 10;
    y = 20;

    System.out.println(x==y);
    System.out.println(x!=y);
    System.out.println(x<y);
    System.out.println(x>y);
    System.out.println(x<=y);
    System.out.println(x>=y);
  }
}
```

결과

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

2.4.5 논리연산자

논리연산자는 논리 타입에서만 사용할 수 있으며 항의 조건을 두 개의 값을 서로 비교할 때 사용되는 연산자로 결과로 true나 false를 반환한다.

연산자	사용법	의미
&&	x && y	x와 y의 값이 모두 true일 때 결과는 true 그 외의 연산 결과는 false
||	x || y	x와 y의 값이 모두 false일 때 결과는 false 그 외의 연산 결과는 true
^	x ^ y	x와 y의 값이 다르면 true 같으면 false
!	! x	x의 값이 true일 때 결과는 false, x의 값이 false일 때 결과는 true

2.4.5 논리연산자

논리연산자는 논리 타입에서만 사용할 수 있으며 항의 조건을 두 개의 값을 서로 비교할 때 사용되는 연산자로 결과로 true나 false를 반환한다.

&&		y	
		true	false
x	true	true	false
	false	false	false

||		y	
		true	false
x	true	true	true
	false	true	false

^		y	
		true	false
x	true	false	true
	false	true	false

!		
x	true	false
	false	false

```java
public class Example {
  public static void main(String[] args) {
    System.out.println("&&");
    System.out.println("true && true = "+ (true && true));
    System.out.println("true && false = "+ (true && false));
    System.out.println("false && true = "+ (false && true));
    System.out.println("false && false = "+ (false && false));

    System.out.println("||");
    System.out.println("true || true = "+ (true || true));
    System.out.println("true || false = "+ (true || false));
    System.out.println("false || true = "+ (false || true));
    System.out.println("false || false = "+ (false || false));

    System.out.println("^");
    System.out.println("true ^ true = "+ (true ^ true));
    System.out.println("true ^ false = "+ (true ^ false));
    System.out.println("false ^ true = "+ (false ^ true));
    System.out.println("false ^ false = "+ (false ^ false));

    System.out.println("!");
    System.out.println("!true = "+ (!true));
    System.out.println("!false = "+ (!false));
  }
}
```

결과

```
&&
true && true = true
true && false = false
false && true = false
false && false = false
||
true || true = true
true || false = true
false || true = true
false || false = false
^
true ^ true = false
true ^ false = true
false ^ true = true
false ^ false = false
!
!true = false
!false = true
```

2.4.6 비트연산자

비트 연산에는 비트간의 값을 비교하여 결과 값을 반환하는 연산자와 비트를 이동시킨 결과를 반환 하는 연산으로 이루어져 있다.

연산자	사용법	의미
&	x & y	x와 y의 비트 모두 1이면 연산 결과는 1
|	x | y	x 또는 y의 비트가 1이면 연산 결과는 1
^	x ^ y	x와 y가 다르면 연산 결과는 1이고, 같으면 연산 결과는 0
~	~x	x의 비트가 0이면 연산 결과는 1로, x의 비트가 1이면 연산 결과는 0
<<	x << y	x의 비트를 왼쪽으로 y번 쉬프트, 최하위 비트는 0으로 채움
>>	x >> y	x의 비트를 오른쪽으로 y번 쉬프트, 최상위 비트는 기존 비트와 동일한 값으로 채움
>>>	x >>> y	x의 비트를 오른쪽으로 y번 쉬프트, 최상위 비트는 0으로 채움

2.4.6 비트연산자

비트 연산에는 비트간의 값을 비교하여 결과 값을 반환하는 연산자와 비트를 이동시킨 결과를 반환 하는 연산으로 이루어져 있다.

&		y	
		1	0
x	1	1	0
	0	0	0

|		y	
		1	0
x	1	1	1
	0	0	0

^		y	
		1	0
x	1	0	1
	0	1	0

~		
x	1	0
	0	1

```java
public class Example {
  public static void main(String[] args) {
    System.out.println("10 & 2 = "+(10 & 2));
    System.out.println("10 | 2 = "+(10 | 2));
    System.out.println("10 ^ 2 = "+(10 ^ 2));
    System.out.println("~10 = "+(~10));

    System.out.println("10 << 2 = "+(10 << 2));
    System.out.println("10 >> 2 = "+(10 >> 2));
    System.out.println("10 >>> 2 = "+(10 >>> 2));

    System.out.println("-10 >> 2 = "+(-10 >> 2));
    System.out.println("-10 >>> 2 = "+(-10 >>> 2));
  }
}
```

결과

```
10 & 2 = 2
10 | 2 = 10
10 ^ 2 = 8
~10 = -11
10 << 2 = 40
10 >> 2 = 2
10 >>> 2 = 2
-10 >> 2 = -3
-10 >>> 2 = 1073741821
```

$10_{(10)}$ & $2_{(10)}$ = $2_{(10)}$

0 0 0 0 1 0 1 0 & 0 0 0 0 0 0 1 0 = 0 0 0 0 0 0 1 0

$10_{(10)}$ | $2_{(10)}$ = $10_{(10)}$

0 0 0 0 1 0 1 0 | 0 0 0 0 0 0 1 0 = 0 0 0 0 1 0 1 0

$10_{(10)}$ ^ $2_{(10)}$ = $8_{(10)}$

0 0 0 0 1 0 1 0 ^ 0 0 0 0 0 0 1 0 = 0 0 0 0 1 0 0 0

2의 보수

- 0 0000 0000 0000 0000 0
- 1 0000 0001 1111 1111 -1
- 2 0000 0010 1111 1110 -2
- 3 0000 0011 1111 1101 -3
- 4 0000 0100 1111 1100 -4
- 5 0000 0101 1111 1011 -5
- 6 0000 0110 1111 1010 -6
- 7 0000 0111 1111 1001 -7

2진 음의 정수 표현: 1의 보수와 2의 보수

- 최상위 비트는 부호 표시: 양수는 0, 음수는 1
  - 그런데, 컴퓨터는 연산의 용이성을 위해 보수를 사용함

- 1의 보수로 변환하는 방법
  - 0→1, 1→0으로 변환
    - 예: 0000 0011 → 1의 보수 = 1111 1100

- 2의 보수로 변환하는 방법
  - 1의 보수 + 1 = 2의 보수
    - 0000 0011 -> 2의 보수 = 1의 보수 + 1 = 1111 1100 + 1 = 1111 1101

2의 보수

	양수	1의 보수	2의 보수	
• 0	0000 0000	1111 1111 + 1	0000 0000	0
• 1	0000 0001	1111 1110 + 1	1111 1111	-1
• 2	0000 0010	1111 1101 + 1	1111 1110	-2
• 3	0000 0011	1111 1100 + 1	1111 1101	-3
• 4	0000 0100	1111 1011 + 1	1111 1100	-4
• 5	0000 0101	1111 1010 + 1	1111 1011	-5
• 6	0000 0110	1111 1001 + 1	1111 1010	-6
• 7	0000 0111	1111 1000 + 1	1111 1001	-7

문제

- 10101100이라는 2의 보수가 있을 때 10진수로 어떤 값인가?

해설 p.49

- -10 >>> 2 = 1073741821
- Java에서 숫자는 기본 정수로 취급, 32비트 크기

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

$$= 2^{29} + 2^{28} + 2^{27} + \ldots + 2^{2} + 1 = 1073741821$$

2.4.7 삼항연산자

삼항연산자는 조건연산자라하여 다음과 같이 ? :의 식으로 사용된다.

```
변수 = 조건 ?  값A : 값B ;
```

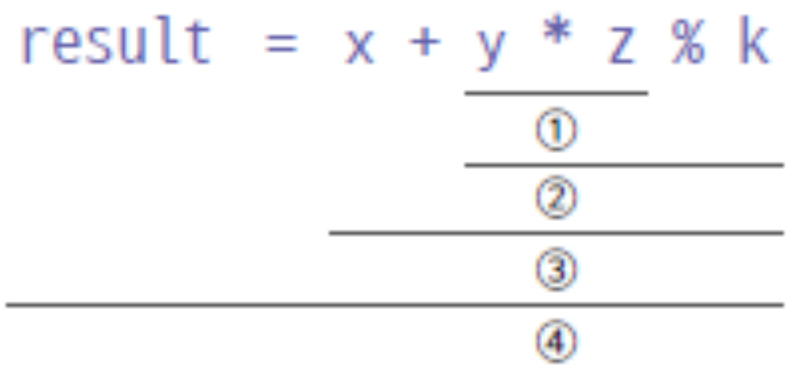
```
1  public class Example {
2    public static void main(String[] args) {
3      int x = 2;
4
5      x = ( x%2 == 0) ? x+1 : x-1;
6      System.out.println(x);
7
8      x = ( x%2 == 0) ? x+2 : x-2;
9      System.out.println(x);
10   }
11 }
```

결과

```
3
1
```

2.4.8 연산자 우선순위

연산자들의 우선순위와 결합성



우선순위	결합성	연산자
높음	right	[ ] . ( ) ++ --
↑	left	+ - ++ -- ~ !
	left	(형변환) new
	left	* / %
	left	+ - .
	left	<< >> >>>
	left	< <= > >=
	left	== !=
	left	&
	left	^
	left	|
	left	&&
	left	||
↓	right	?:
낮음	right	= += -= *= /= .= %= != ~= <<= >>=