연산자

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류

1

연산자		연산기호	설명	비고
1	산술연산자	+, -, *, /, %	사칙연산 및 나머지연산	값
2	증감연산자	++, --	데이터 값의 1증가 및 감소	
3	비트연산자	&, |, ~, ^	비트 AND, OR, NOT, XOR	
4	쉬프트연산자	>>, <<, >>>	비트단위의 이동	
5	비교연산자	<, >, <=, >=, ==, !=	데이터의 크기 비교	참 또는 거짓
6	논리연산자	&&, ||, !, ^	논리적 AND, OR, NOT, XOR	
7	대입연산자	=, +=, -=, *=, /=, &=, |=, >>=, <<=, >>>=	산술연산 결과의 대입 (축약형 표현)	실행
8	삼항연산자	(참 또는 거짓)? x : y	참인 경우 x, 거짓인 경우 y	

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **산술연산자 (+, -, *, /, %)**

1

```
System.out.println(2+3);
System.out.println(8-5);
System.out.println(7*2);
System.out.println(7/2);
System.out.println(8%5);
```

```
5
3
14
3
3
```

2

TIP

나누기연산(/)
- 정수/정수 = 정수

나머지연산(%)
- 정수/정수의 형태에서 나눈 후 나머지 정수값

☞ 연산자의 종류 – **증감연산자 (++, --)**

- **전위형**: 연산(또는 실행) **전** 증감 수행 (++변수명)
- **후위형**: 연산(또는 실행) **후** 증감 수행 (변수명++)

3

```
int a=3;
++a;
System.out.println(a);

int b=3;
b++;
System.out.println(b);
```

```
4

4
```

4

TIP

증감연산자(++, --)
- a=a+1; → a+=1 → a++;
- a=a-1; → a-=1 → a--;

5

전위형

```
int a=3;
int b=++a;
System.out.println(a);
System.out.println(b);
```

```
4
4
```

후위형

```
int a=3;
int b=a++;
System.out.println(a);
System.out.println(b);
```

```
4
3
```

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **비트연산자 (&, |, ~, ^)**

1

값1	값2	&	|	^
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

값	~
0	1
0	1
1	0
1	0

- 비트 AND 연산자

2

```
  00000011 → 3
& 00001010 → 10
  00000010 → 2
```

```
System.out.println(3 & 10);
System.out.println(0b00000011 & 0b00001010);
System.out.println(0x03 & 0x0A);
```

```
2
2
2
```

- 비트 OR 연산자

3

```
  00000011 → 3
| 00001010 → 10
  00001011 → 11
```

```
System.out.println(3 | 10);
System.out.println(0b00000011 | 0b00001010);
System.out.println(0x03 | 0x0A);
```

```
11
11
11
```

- 비트 XOR 연산자

4

```
  00000011 → 3
^ 00001010 → 10
  00001001 → 9
```

```
System.out.println(3 ^ 10);
System.out.println(0b00000011 ^ 0b00001010);
System.out.println(0x03 ^ 0x0A);
```

```
9
9
9
```

TIP

코드에서의 2진수 표현
- 0b + 2진수
- ex. 0b0011 → 3

코드에서의 8진수 표현
- 0 + 8진수
- ex. 00011 → 9

코드에서의 10진수 표현
- 10진수
- ex. 1234 → 1234

코드에서의 16진수 표현
- 0x + 16진수
- ex. 0x0011→17

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **비트연산자 (&, |, ~, ^)**

TIP

- 10진수 → 2진수 / 8진수 / 16진수 **코드상에서 변환 방법**

1

```
int data = 10;

System.out.println(Integer.toBinaryString(data)); //1010 : 10진수 → 2진수
System.out.println(Integer.toOctalString(data)); //12 : 10진수 → 8진수
System.out.println(Integer.toHexString(data)); //a : 10진수 → 16진수
```

- 2진수 / 8진수 / 16진수 → 10진수 **코드상에서 변환 방법**

```
int data = 10;

System.out.println(Integer.parseInt("1010",2)); //10 : 2진수 → 10진수
System.out.println(Integer.parseInt("12",8)); //10 : 8진수 → 10진수
System.out.println(Integer.parseInt("a",16)); //10 : 16진수 → 10진수
```

Copyright 2020. Donghyung Kim, All rights reserved

연산자

값1	값2	&	|	^
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

값	~
0	1
0	1
1	0
1	0

☞ 연산자의 종류 – **비트연산자 (&, |, ~, ^)**

- 비트 NOT 연산자

1

~ 00000011 → 3
11111100 →-4

```
System.out.println(~3);
System.out.println(~0b00000011);
System.out.println(~0x03);
```

```
-4
-4
-4
```

~ 00000000 → 0
11111111 → -1

```
System.out.println(~0);
System.out.println(~0b00000000);
System.out.println(~0x00);
```

```
-1
-1
-1
```

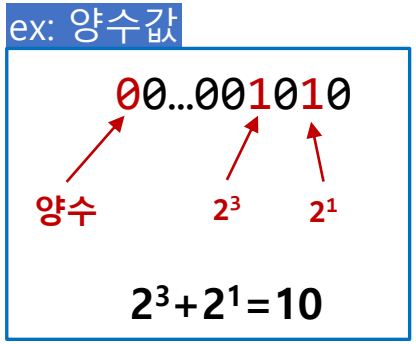
2

TIP

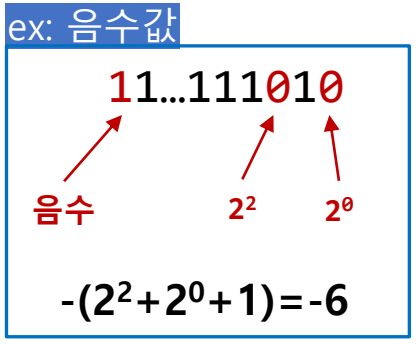
비트값 읽는 법
- 부호비트 : 첫번째 비트 (0:양수, 1: 음수)
- 양수 읽는 법 : 1을 기준으로 값 읽음
- 음수 읽는 법 : 0을 기준으로 값 읽음 + 1

3

ex: 양수값



ex: 음수값



Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 –**쉬프트연산자 (<<, >>, >>>)**

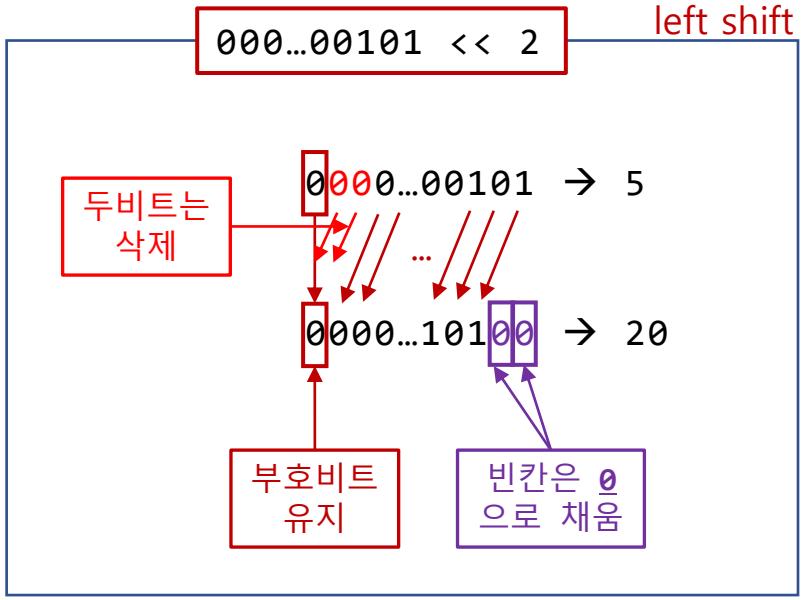
1

- **산술**쉬프트 (<<, >>)

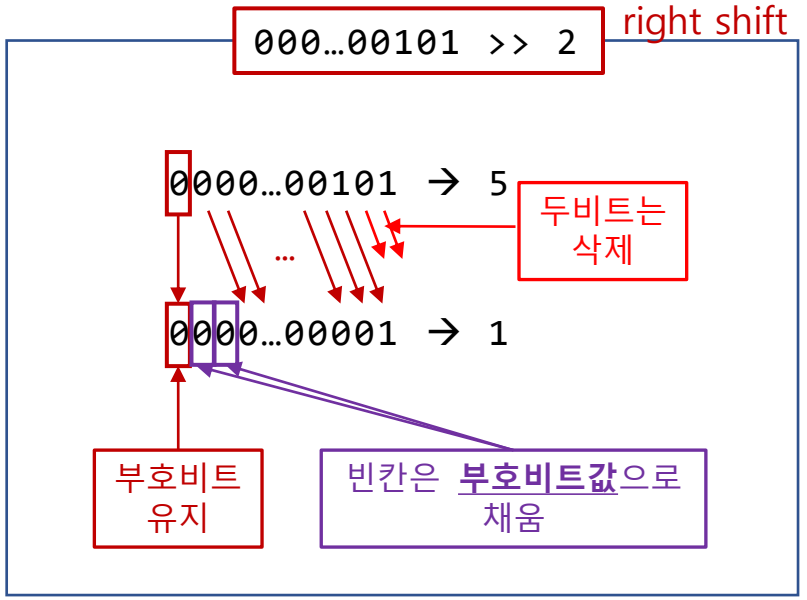
left shift　　right shift

- 부호비트 유지
- 쉬프트의 방향에 따라 1bit당 x2 또는 /2의 효과

2



3



4

TIP

양수 및 음수의 shift 연산

ex

Left shift: 양수와 음수 동일 방식
 3<<1 = 6; (1bit당 x 2) + 부호유지
-3<<1 = -6; (1bit당 x 2) + 부호유지
Right shift: 양수와 음수 다른 방식
 5>>2 = 1; (1bit당 / 2) + 부호유지 + 소수버림
-5>>2 = -2; (1bit당 / 2) + 부호유지 + 소수올림

Copyright 2020. Donghyung Kim, All rights reserved

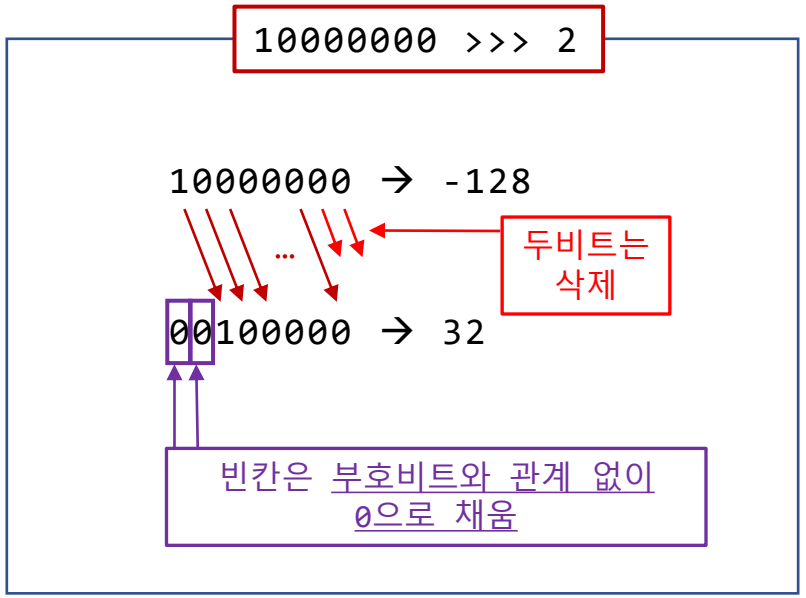
연산자

☞ 연산자의 종류 –**쉬프트연산자 (<<, >>, >>>)**

- **논리**쉬프트 (>>>)

  - 부호비트 상관없이 쉬프트 수행
  - 부호비트 변화에 따라 부호변동 생길 수 있음

1



2

TIP

Java는 최소 int 단위로 연산 수행
- 쉬프트 연산자를 비트에서 확인하기 위해서는 32bit(int의 바이트수)로 변환하여 고려하여야 함

3

TIP

이진수→16진수 표현 (2진수 4개→16진수 1개)

ex

```
0b00111000 = 0x38
0b11110000 = 0xf0
0b11111111111111111111111111111111 = 0xffff
```

4

ex

```
System.out.println(5<<2);
System.out.println(0b00000101<<2);

System.out.println(5>>2);
System.out.println(0b00000101>>2);

System.out.println(-5>>2);
System.out.println((byte)0b11111011>>2);

System.out.println(0xffffffff>>>31);
```

```
20
20

1
1

-2
-2

1
```

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **비교연산자 (<, >, <=, >=, ==, !=)**

- **크기**비교 (>, <, >=, <=)

  - 데이터 크기의 대소 비교
  - 연산결과는 **true/false**

1

ex

```
System.out.println(5<2);
System.out.println(5>2);
System.out.println(5>5);
System.out.println(5<=5);
System.out.println(5>=5);
```

```
false
true
false
true
true
```

- **등가**비교 (==, !=)

  - 데이터 크기의 등가 비교
  - 연산결과는 **true/false**

2

ex

```
System.out.println(5==2);
System.out.println(5!=2);
System.out.println(5==5);
System.out.println(5!=5);
```

```
false
true
true
false
```

3

TIP

등가비교는 stack메모리의 값을 비교
- 기본자료형 (값비교) / 참조자료형 (번지비교)

ex

```
int a=3, b=3;
System.out.println(a==b); //true

String a = new String("안녕");
String b = new String("안녕");
System.out.println(a==b); //false
```

4

TIP

자바에서 등호(=)가 다른 부호와 함께 사용되는 경우 등호는 항상 오른쪽에 위치함
- <=, >=, !=, ==

5

TIP

등가비교 (==)는 대입연산자(=)와 반드시 구분

ex

```
int a=3;
System.out.println(a==5);
System.out.println(a=5);
```

```
false
5
```

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **논리연산자 (&&, ||, !, ^)**

1

값1	값2	&&	||	^
false	false	false	false	false
false	true	false	true	true
true	false	false	true	true
true	true	true	true	false

값	!
false	true
false	true
true	false
true	false

- 논리 AND 연산자

2

ex

```
System.out.println(true && true);
System.out.println(true && false);
System.out.println(false && (5<3));
System.out.println((5>=5) && (7>2));
```

```
true
false
false
true
```

- 논리 OR 연산자

3

ex

```
System.out.println(true || true);
System.out.println(true || false);
System.out.println(false || (5<3));
System.out.println((5>=5) || (7>2));
```

```
true
true
false
true
```

- 논리 XOR 연산자

4

ex

```
System.out.println(true ^ true);
System.out.println(true ^ false);
System.out.println(false ^ (5<3));
System.out.println((5>=5) ^ (7>2));
```

```
false
true
false
false
```

- 논리 NOT 연산자

5

ex

```
System.out.println(!true);
System.out.println(!false);
System.out.println(false || !(5<3));
System.out.println((5>=5) || !(7>2));
```

```
false
true
true
true
```

6

TIP

논리연산자의 좌우에는 반드시 true/false 만 올 수 있음

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **논리연산자 (&&, ||, !, ^)**

- 비트연산자(&, |)를 이용한 논리연산

ex

1

```
System.out.println(true & true);
System.out.println(true & false);
System.out.println(true | (5<3));
System.out.println((5>=5) | (7>2));
```

```
true
false
true
true
```

Short Circuit

2

연산 수행시 결과가 이미 확정된 경우 나머지 연산과정을 수행하지 않는 것

ex

(5>3) || (3<2) : (5>3)이 true값을 가져 (3<2)는 검사하지 않고 true를 리턴함

TIP

3

비트연산자의 XOR(^)와 논리연산의 XOR은 모두 '^'의 기호를 사용함. (단, 모두 쇼트서킷(short circuit) 미적용)
XOR은 구조상 반드시 두개를 다 확인해야 결과값 결정

4 **비트연산자를 이용한 경우 논리연산과의 차이점**

논리연산자

5

```
int a=3, b=3, c=3;
System.out.println(false && a++>6);
System.out.println(true || b++>6);
System.out.println(true ^ c++>6);
System.out.println(a);
System.out.println(b);
System.out.println(c);
```

```
false
true
true
3
3
4
```

비트연산자

6

```
int a=3, b=3, c=3;
System.out.println(false & a++>6);
System.out.println(true | b++>6);
System.out.println(true ^ c++>6);
System.out.println(a);
System.out.println(b);
System.out.println(c);
```

```
false
true
true
4
4
4
```

Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **대입연산자 (=, +=, -=, *=, /=, &=, |=, >>=, <<=, >>>=)**

- 대입연산자 (=)

- 오른쪽의 연산결과를 왼쪽에 대입

ex

1

```
int a = 3;
a=5;
System.out.println(a);
a=a+3;
System.out.println(a);
```

```
5

8
```

TIP

2

수학에서는 불가능하고 자바코드에서는 가능한 이유

ex

a = a+3

step 1. a+3 연산

step 2. step1의 연산결과를 a에 입력

- 대입연산과 다른 연산의 축약표현

3

일반표현	a=a+b	a=a-b	a=a*b	a=a/b	a=a&b	a=a|b	a=a>>b	a=a>>b	a=a>>>b
축약표현	a+=b	a-=b	a*=b	a/=b	a&=b	a|=b	a>>=b	a<<=b	a>>>=b

ex

```
int a = 3;
a*=5;
System.out.println(a);
```

```
15
```

4

TIP

증감연산자(++, --)

- a=a+1; → a+=1 → a++;
- a=a-1; → a-=1 → a--;

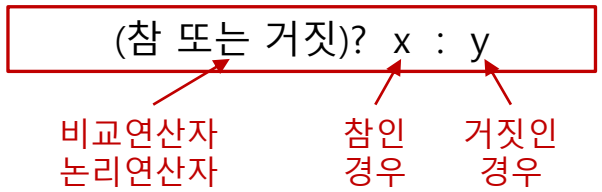
Copyright 2020. Donghyung Kim, All rights reserved

연산자

☞ 연산자의 종류 – **삼항연산자 ((참 또는 거짓)? x : y)**

- 삼항연산 처리방식

1



2

ex

```
int a = (3>5)? 6:9;
System.out.println(a);
int b = (5>3)? 10:20;
System.out.println(b);
```

```
9

10
```

3

TIP

삼항연산자는 if-else 제어문으로 변경 가능

삼항연산자

```
int a = (3>5)? 6:9;
System.out.println(a);
```

if-else → 제어문 참조

```
int a;
if(3>5) {
    a=6;
} else {
    a=9;
}
System.out.println(a);
```

Copyright 2020. Donghyung Kim, All rights reserved

The End

Copyright 2020. Donghyung Kim, All rights reserved