제2장 – 연산자

Outline

2.1 산술 연산자(1)

- 수치 계산에 사용하는 연산자

연산자	기능	사용법	의미
+	+ 더하기	a = b + c	b와 c를 더한 값을 a에 대입한다
−	− 빼기	a = b − c	b에서 c를 뺀 값을 a에 대입한다
*	× 곱하기	a = b * c	b와 c를 곱한 값을 a에 대입한다
/	÷ 나누기	a = b / c	b를 c로 나눈 값을 a에 대입한다 (c가 0이면 에러)
%	· · 나머지	a = b % c	b를 c로 나눈 나머지를 a에 대입한다 (정수형에서만 유효)
=	= 대입	a = b	b 값을 a에 대입한다

예 ››

```
class Calculation {
   public static void main(String [] args) {
      System.out.println("5+5는" +(5+5) + "입니다." );
      System.out.println("5 - 5는" +(5-5) + "입니다." );
      System.out.println("5×5는" + 5*5 + "입니다." );
      System.out.println("5÷5는" + 5/5 + "입니다." );
      System.out.println("5÷3의 나머지는"+ 5%3 + "입니다" );
   }
}
```

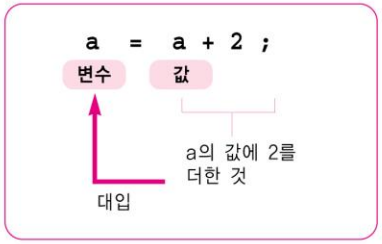
+ 연산자와 − 연산자는 우선순위가 동일하기 때문에 ( )가 필요합니다. (62페이지 참조)

실행결과 ››

```
5+5는 10입니다.
5-5는 0입니다.
5x5는 25입니다.
5÷5는 1입니다.
5÷3의 나머지는 2입니다.
```

2.1 산술 연산자(1)

- 대입 연산자



a 값을 2 증가시킨다는 것을 아래와 같이 쓸 수도 있습니다.

```
a += 2;
```

2.1 산술 연산자(1)

- 대입 연산자

연산자	기능	사용법	의미
+=	더한 값을 대입	a += b	a+b의 결과를 a에 대입 (a=a+b와 동일)
-=	뺀 값을 대입	a -= b	a-b의 결과를 a에 대입 (a=a-b와 동일)
*=	곱한 값을 대입	a *= b	a*b의 결과를 a에 대입 (a=a*b와 동일)
/=	나눈 값을 대입	a /= b	a/b의 결과를 a에 대입 (a=a/b와 동일)
%=	나머지를 대입	a %= b	a%b의 결과를 a에 대입 (a=a%b와 동일)

<< 예

```
class Plus {
   public static void main( String[] args) {
      int  a = 90 ;
      a += 10 ;          a = a + 10이라고 써도 동일합니다.
      System.out.println("90에 10을 더하면 "+a+"입니다.") ;
   }
}
```

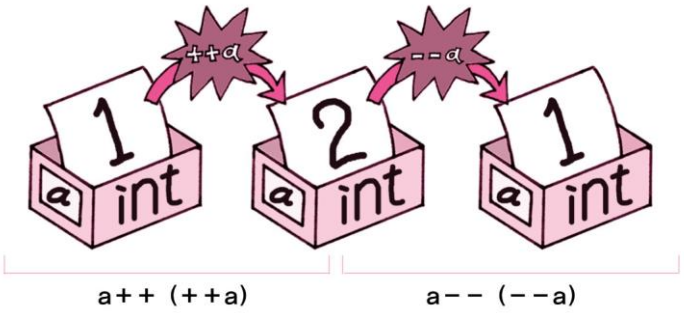
<< 실행결과

```
90에 10을 더하면 100입니다.
■
```

2.2 산술 연산자(2)

- 증가 연산자 감소 연산자

연산자	명칭	기능	사용법	의미
++	증가(increment) 연산자	변수의 값을 1증가	a++ 또는 ++a	a의 값을 1 증가시킨다
--	감소(decrement) 연산자	변수의 값을 1감소	a-- 또는 --a	a의 값을 1 감소시킨다



2.2 산술 연산자(2)

- 증가 연산자 감소 연산자

```
int x, a = 1 ;
x = ++a ;
```

a에 1을 더한 후, x에 그 값을 대입한다
→ x의 값은 2가 된다

```
int x, a = 1 ;
x = a++ ;
```

x에 값을 대입한 후, a에 1을 더한다
→ x의 값은 여전히 1이다.

예 ››

```
class One {
   public static void main(String[] args) {
      int a = 1 ;
      System.out.println("처음 값은 "+ a +"이었습니다.") ;

      a++ ;
      System.out.println("1 증가해서 "+ a +"가 됩니다.") ;


      a-- ;
      System.out.println("1 감소해서"+ a +"로 돌아왔습니다.") ;
   }
}
```

‹‹ 실행결과

```
처음 값은 1이었습니다.
1 증가해서 2가 됩니다.
1 감소해서 1로 돌아왔습니다.
■
```

2.2 산술 연산자(2)

- 증가 연산자 감소 연산자

```
class Position {
   public static void main (String [] args) {
      int a = 1, b = 1;
      System.out.println("전치일 경우 "+ ++a + "가 됩니다.") ;
      System.out.println("후치일 경우 "+ b++ + "이 됩니다.") ;
   }
}
```

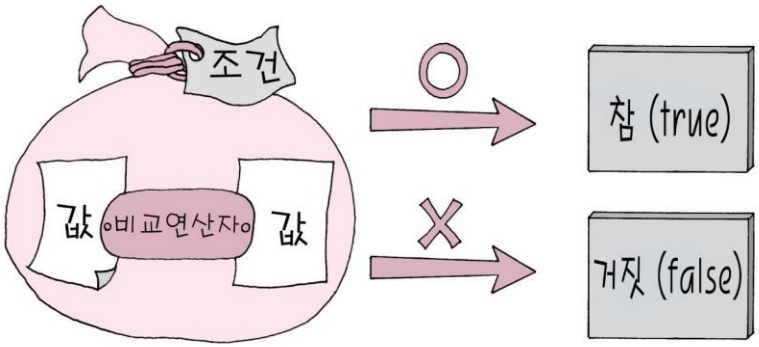
<< 예

<< 실행결과

```
전치일 경우 2가 됩니다.
후치일 경우 1이 됩니다.
■
```

2.3 변수

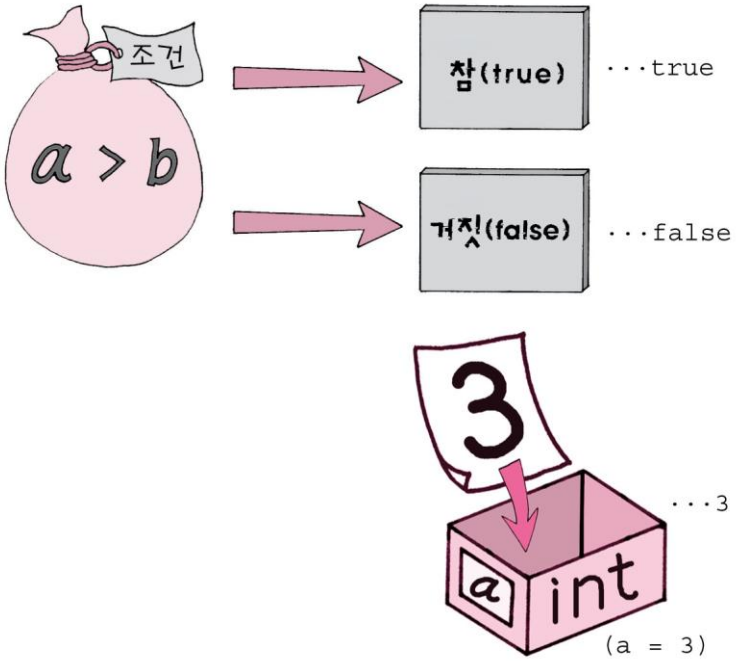
- 비교 연산자란?



연산자	기능	사용법	의미
= =	= (같다)	a == b	a와 b는 같다
〈	〈 (작다)	a 〈 b	a는 b보다 작다
〉	〉 (크다)	a 〉 b	a는 b보다 크다
〈 =	≦ (이하)	a 〈= b	a는 b 보다 작거나 같다
〉 =	≧ (이상)	a 〉= b	a는 b 보다 크거나 같다
! =	≠(같지 않다)	a != b	a와 b는 같지 않다

2.3 비교 연산자

- 식이 가지는 값



2.3 비교 연산자

- 식이 가지는 값

<< 예

```
class Compare {
   public static void main (String[] args) {
      int a = 10, b = 20 ;

      System.out.println("a = "+ a + "  b = " + b ) ;
      System.out.println("a < b · · · "+ ( a < b)) ;
      System.out.println("a > b · · · "+ ( a > b)) ;
      System.out.println("a == b · · · "+ ( a == b)) ;
      System.out.println("a = b · · · "+ ( a = b)) ;
   }
}
```

비교연산자는 +연산보다 우선순위가 낮기 때문에 ( )가 필요합니다(62페이지 참조).

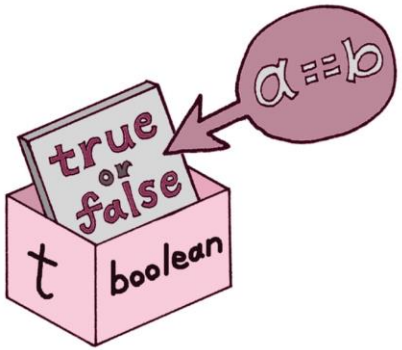
<< 실행결과

```
a = 10  b = 20
a 〈 b · · · true
a 〉 b · · · false
a == b · · · false
a = b · · · 20
■
```

2.4 논리형

- 논리형

```
boolean  t =  ( a == b) ;
```



2.4 논리형

- 논리형

예 ››

```
class Compare {
   public static void main(String [] args ) {
       int  a = 10, b =20 ;
       boolean  c, d, e ;

       c = a < b ;
       d = a > b ;
       e  = a == b ;
       System.out.println("a = "+ a + "  b = "+ b ) ;
       System.out.println("a < b · · ·"+ c ) ;
       System.out.println("a > b · · ·"+ d ) ;
       System.out.println("a == b · · ·"+ e ) ;
   }
}
```

실행결과 ››

```
a = 10  b = 20
a 〈 b · · · true
a 〉 b · · · false
a == b · · · false
■
```

2.4 논리형

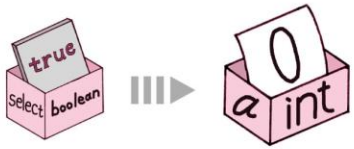
- 조건 연산자

```
boolean  select ;
            :
int  a = select ? 0 : 1 ;
```

조건식 / 조건식이 true일 경우 / 조건식이 false일 경우

조건에 맞는 경우 (true)

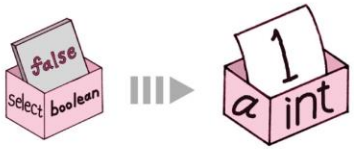
':' 의 좌측을 선택합니다.



a = 0

조건에 맞지 않는 경우(false)

':' 의 우측을 선택합니다.



a = 1

2.4 논리형

- 조건 연산자

<< 예

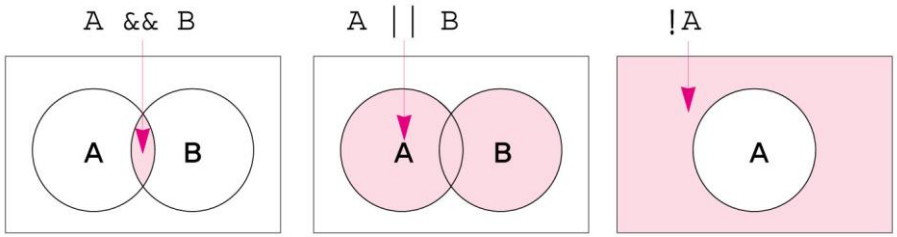
```
class Truth {
   public static void main(String [] args) {
      String right = "맞음", wrong = "틀림" ;
      boolean value ;

      value = true ;
      String answer = value ? right : wrong ;
      System.out.println(answer) ;

      value = false ;
      String answer = value ? right : wrong ;
      System.out.println(answer) ;
   }
}
```

<< 실행결과

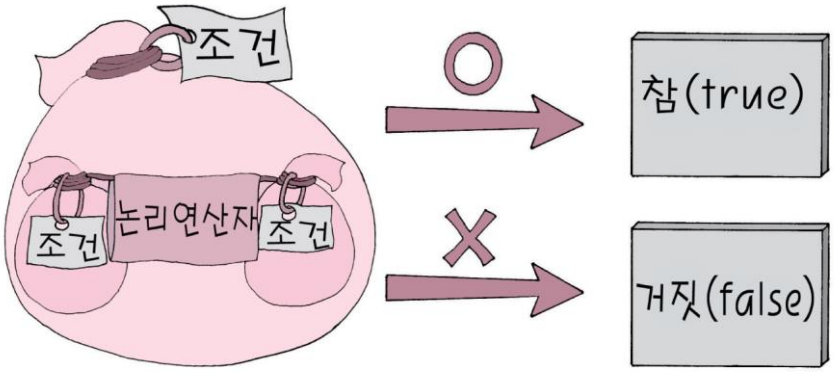
```
맞음
틀림
■
```

2.5 논리 연산자

- 논리 연산자란?



연산자	기능	사용법	의미
&&	그리고	(a >= 10) && (a < 50)	a는 10 이상이고 50 미만
||	또는	(a == 1) || (a == 100)	a 값이 1 또는 100
!	~ 가 아니다	!(a == 100)	a는 100이 아니다



2.5 논리 연산자

- 논리 연산자란?

a가 50 이상 100 미만이다

```
(50 <= a ) && ( a < 100 )
```

b가 0도 1도 아니다

```
!(b==0) || (b==1)) · · · 'b=0 또는 b=1' 이 아니다.
!(b==0) && !(b==1) · · · b=0이 아니고 b=1도 아니다
```

2.5 논리 연산자

- 논리형의 이용

C가 5 또는 9 이다

```
boolean x = ( c == 5 ) ··· x는 c = 5 이면 true, c≠5 이면 false
boolean y = ( c == 9 ) ··· x는 c = 9 이면 true, c≠9 이면 false
x || y                 ··· x = true 또는 y = true 이다
```

c = 5 c = 9

2.5 논리 연산자

- 논리형의 이용

```
class Or {
   public static void main(String [] args) {
      int a = 3 , b = 4 ;
      boolean x, y ;

      x = ( a < 0 ) ;
      y = ( b > 0 ) ;
      System.out.println((a == 3) && (b == 3 ));
      System.out.println(x || y)
   }
}
```

<< 예

```
false
true
■
```

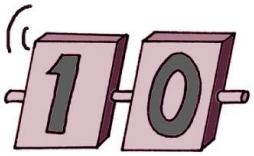
<< 실행결과

2.6 수치와 단위

- n진수

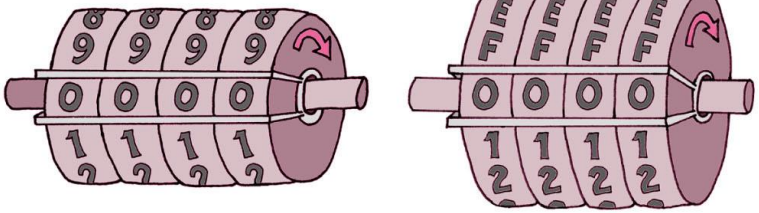
2진수

1과 0의 두 가지 상태로 나타냅니다. 컴퓨터 내부에서 가장 기본적인 표기법입니다



10진수

일반적으로 사용하고 있는 표기방법으로 0에서 9까지의 숫자를 사용합니다.



16진수

16마다 단위가 올라가며, 9 다음에는 A~F의 문자를 사용합니다.

2.6 수치와 단위

- n진수

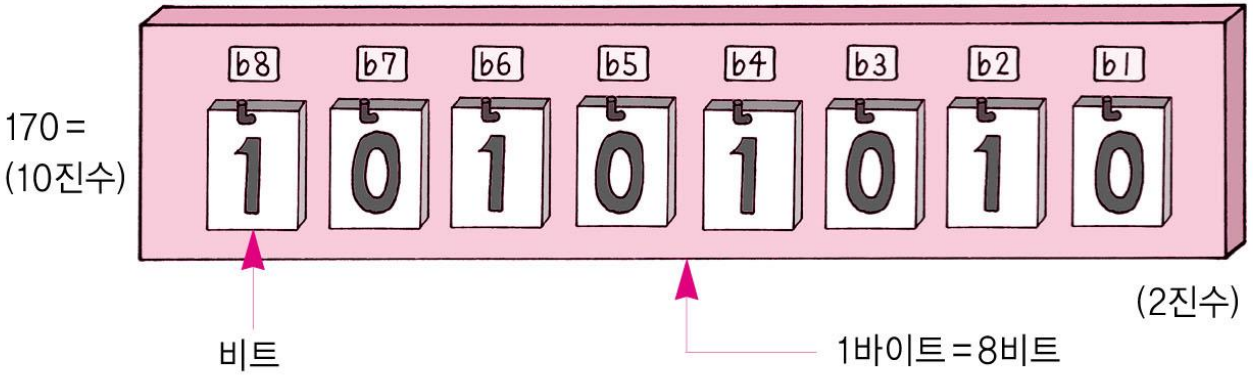
2진수	10진수	16진수
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10

2.6 수치와 단위

- 16진수의 표기방법

2.6 수치와 단위

- 비트와 바이트



2.6 수치와 단위

- 바이트의 단위

단위	읽는 법	의미
KB	킬로바이트	1 KB = 1024 바이트
MB	메가바이트	1 MB = 1024 KB
GB	기가바이트	1 GB = 1024 MB
TB	테라바이트	1 TB = 1024 GB

2.7 형의 변환

- 계산 중의 형 변환

3 ÷ 2의 결과를 구한다 (틀림)

```
3 / 2  →  1
정수  정수    정수
```

정수가 되도록 자동적으로 소수점 이하가 잘려집니다.

바른 값인 1.5를 산출해 내기 위해서는 실수표기로 하여 계산할 필요가 있습니다.

3 ÷ 2의 결과를 구한다 (맞음)

```
3.0 / 2.0  →  1.5
실수  실수     실수
```

2.7 형의 변환

- 계산 중의 형 변환

예〉〉

```
class Type {
   public static void main(String [] args ) {
      System.out.println("3 ÷ 2 = " + 3/2 ) ;
      System.out.println("3.0 ÷ 2.0 = " + 3.0/2.0 ) ;
      System.out.println("3.0 ÷ 2 = " + 3.0/2 ) ;
      System.out.println("3 ÷ 2.0 = " + 3/2.0 ) ;
   }
}
```

실행결과 〉〉

```
3   2 = 1
3.0   2.0 = 1.5
3.0   2 = 1.5
3   2.0 = 1.5
■
```

2.7 형의 변환

- 다른 유효범위를 가진 자료형끼리의 대입

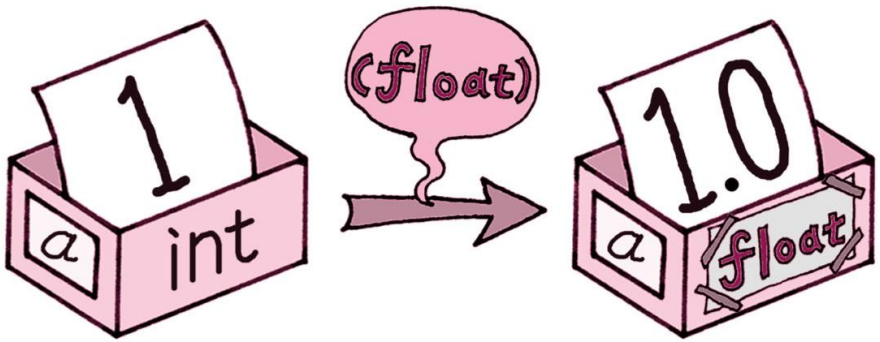
형	유효범위
double	넓다
float	
long	
int	
short	
byte	좁다

○
```
int  i  =  8 ;
float f = i ;
```

×
```
double  d = 2.8 ;
long l = d ;
```

2.7 형의 변환

- 케스트 연산자



2.7 형의 변환

- 케스트 연산자

《 예

```
class Cast {
   public static void main (String [] args) {
      System.out.println("3 ÷ 2 = " + (float)3/2 ) ;
      System.out.println("3 ÷ 2 = " + 3/(float)2 ) ;
   }
}
```

float형으로 캐스트

《 실행결과

```
3 ÷ 2 = 1.5
3 ÷ 2 = 1.5
■
```

2.8 연산의 우선 순위

- 연산자의 우선순위

우선순위	연산자	우선순위가 같은 경우 연산 방향
1	[ ] . (피리오드, 오브젝트 멤버를 선택) ( ) ++(후치) --(후치)	→
2	++(전치) --(전치) +(부호) -(부호) ~ !	←
3	new 캐스트 연산자	←
4	* / %	→
5	+ -	→
6	<< >> >>>	→
7	< > >= <= instanceof	→
8	== !=	→
9	& (비트연산)	→
10	^	→
11	|	→
12	&&	→
13	||	→
14	? :	←
15	= += -= *= /= %=>>= <<= >>>= &= ^= |=	←

2.8 연산의 우선 순위

- 연산자의 우선순위

우선순위가 다를때

```
a + b * b
    ①
  ②
```

+나 – 보다 *와 /를 먼저 계산합니다.

```
(a + b) * c
 ①      ②
```

( )로 둘러싸면, 그 안을 먼저 계산합니다.

```
c = a == b
    ①
  ②
```

a와 b가 같으면 true, 다르면 false를 boolean 형인 c에 대입합니다.

우선순위가 다를때

```
a + b - c
 ①    ②
```

사칙 연산은 좌측부터 계산합니다.

```
a = b = c = 1
        ①
    ②
③
```

대입은 오른쪽부터 실행합니다. a, b, c의 값은 모두 1이 됩니다.

2.8 연산의 우선 순위

- 연산자의 우선순위

```
class Priority {
   public static void main(String[] args) {
      System.out.println("2x8-6÷2 = "+(2*8-6/2)) ;
      System.out.println("2x(8-6)÷2 = "+2*(8-6)/2) ;
      System.out.println("1-2+3 = " + (1-2+3)) ;
      System.out.println("1-(2+3) = " + (1-(2+3))) ;
   }
}
```

<< 예

```
2x8-6÷2 = 13
2x(8-6)÷2 = 2
1-2+3 = 2
1-(2+3) = -4
■
```

<< 실행결과