

Java



연산자

자바의 연산자에 대하여
여러 가지 연산자들의 사용 방법
연산자의 우선순위



연산자

01. 자바의 연산자

용어 설명

• 연산(operation)

- 정해진 규칙에 따라 데이터를 처리하여 결과를 산출하는 것

• 연산자(operator)

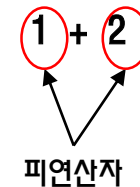
- 연산에 사용되는 표시나 기호

• 피연산자(operand)

- 연산자가 처리하는 데이터

• 식(expression)

- 연산자와 피연산자를 이용하여 연산의 과정을 기술한 것

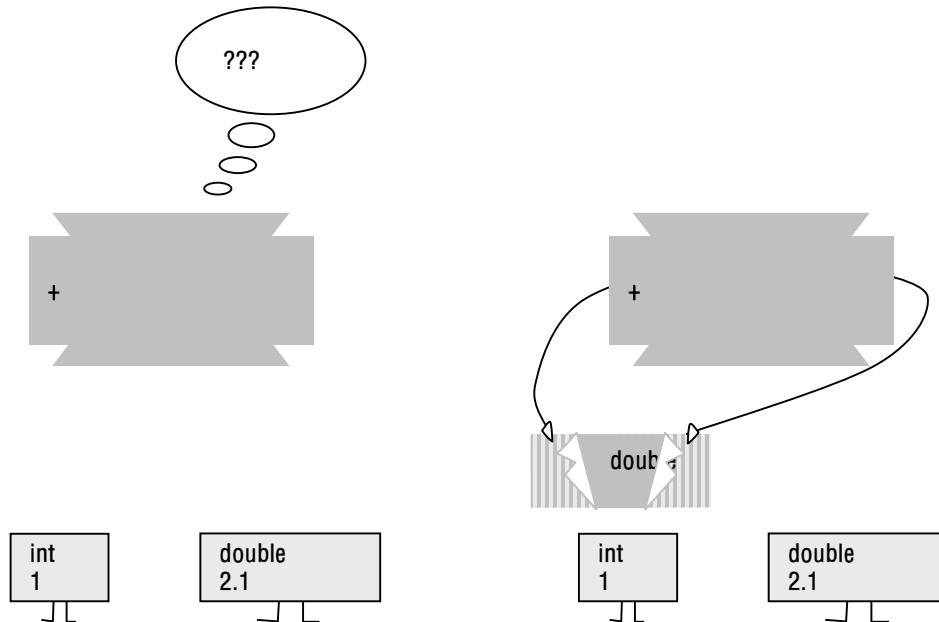


연산자

01. 자바의 연산자

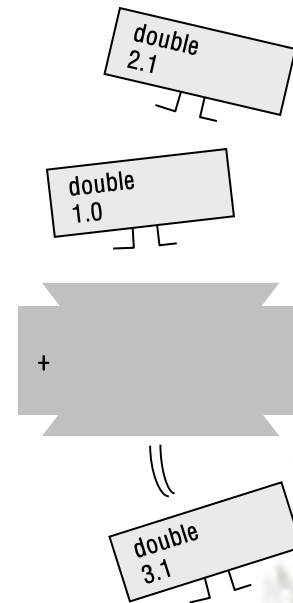
데이터 타입과 연산자

- 데이터 타입과 밀접한 관련이 있는 연산자의 기능



덧셈 연산자는 타입이 다른 데이터를 처리할 수 없습니다.

그래서 먼저
데이터 타입부터 통일합니다.



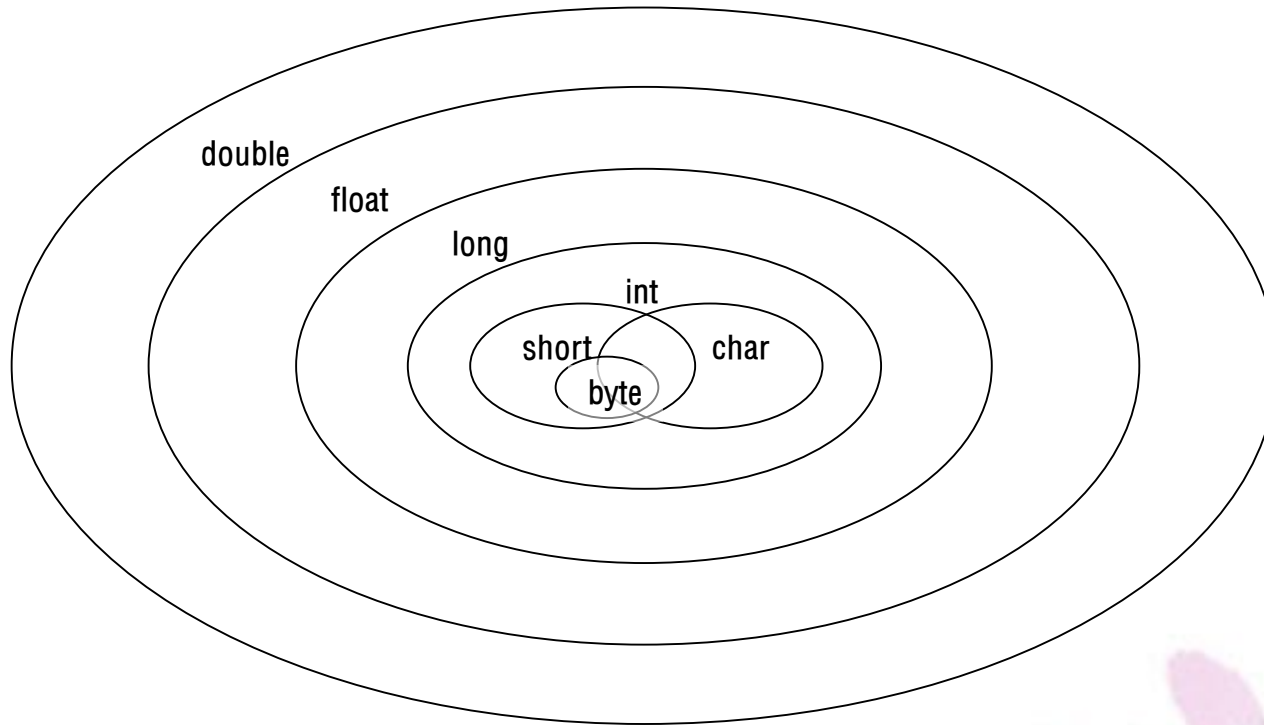
double 타입과 double 타입을
더하여 double 타입의 결과를 산출합니다.

연산자

01. 자바의 연산자

데이터 타입과 연산자

• 수치 타입의 표현 범위

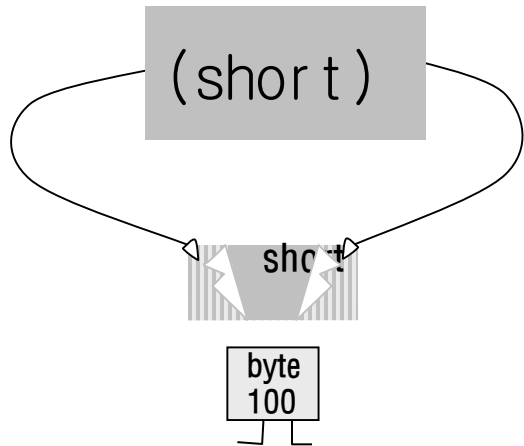


연산자

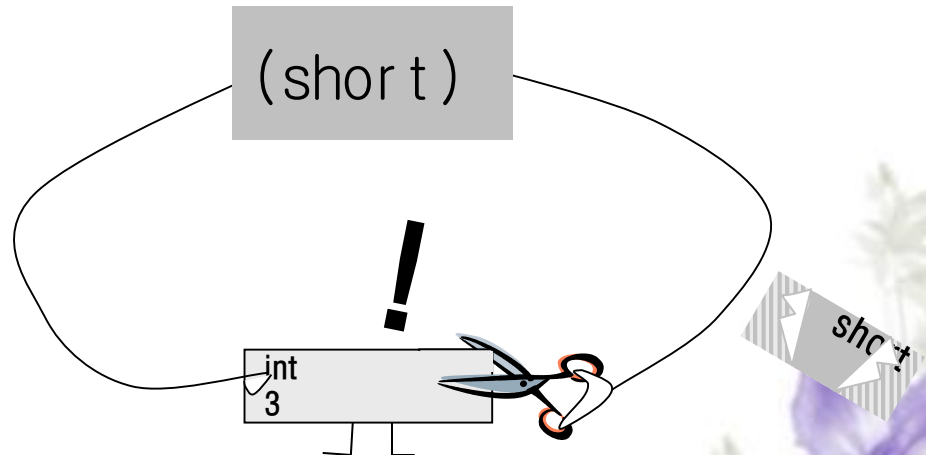
01. 자바의 연산자

데이터 타입과 연산자

• 데이터의 타입을 바꾸는 캐스트 연산자



캐스트 연산자는 타입 변환만
전문으로 수행하는 연산자입니다.



캐스트 연산자는 큰 범위 타입의 데이터를
작은 범위 타입으로 바꿀 수도 있습니다.

02. 여러 가지 연산자들

자바의 연산자들

구분	연산자	기능 설명
사칙 연산자	+ - * / %	사칙연산 및 나눗셈의 나머지 계산
부호 연산자	+ -	음수와 양수의 부호
문자열 연결 연산자	=	두 문자열을 연결
단순 대입 연산자	++ --	우변의 값을 좌변의 변수에 대입
증가/감소 연산자	< > <= >=	변수 값을 1만큼 증가/감소
수치 비교 연산자	== !=	수치의 크기 비교
동등 연산자	== !=	데이터의 동일 비교
논리 연산자	& ^ !	논리적 AND, OR, XOR, NOT 연산
조건 AND/OR 연산자	&&	최적화된 논리적 AND, OR 연산
조건 연산자	?:	조건에 따라 두 값 중 하나를 택일
비트 연산자	& ^ ~	비트 단위의 AND, OR, XOR, NOT 연산
쉬프트 연산자	<< >> >>>	비트를 좌측/우측으로 밀어서 이동
복합 대입 연산자	+= -= *= /= %= &= = ^= <<= >>= >>>=	+ - * / % & ^ << >> >>> 연산자와 =의 기능을 함께 수행
캐스트 연산자	(타입 이름)	타입의 강제 변환

02. 여러 가지 연산자들

사칙 연산자

• 사칙연산자 : 덧셈, 뺄셈, 곱셈, 나눗셈을 하는 연산자

피연산자1 + 피연산자2

이 값과 이 값을 더합니다

피연산자1 - 피연산자2

이 값에서 이 값을 뺍니다.

피연산자1 * 피연산자2

이 값과 이 값을 곱합니다.

피연산자1 / 피연산자2

이 값을 이 값으로
나눈 몫을 계산합니다.

피연산자1 % 피연산자2

이 값을 이 값으로
나눈 나머지를 계산합니다.

연산자

02. 여러 가지 연산자들

사칙 연산자

•• 사용 예

$1+2$

$0.5-1.2$

$2.5*32$

$7/2$

$7\%2$

02. 여러 가지 연산자들

사칙 연산자

.. 정수와 소수의 나눗셈 연산

7 / 2	// 결과는 3
7.0 / 2.0	// 결과는 3.5
7 / 2.0	// 결과는 3.5
7.0 / 2	// 결과는 3.5

.. 정수와 소수의 나눗셈 나머지 연산

7 % 2	// 결과는 1
7.5 % 2.0	// 결과는 1.5

02. 여러 가지 연산자들

사칙 연산자

• 사칙 연산자의 우선순위

$1 / 2 - 3$

// $1 / 2$ 가 먼저 계산됨

$2.0 + 1.5 * 2.0$

// $1.5 * 2.0$ 이 먼저 계산됨

• 우선순위가 같은 연산자의 처리 순서

$1.2 + 2.3 + 3.4$

// $1.2 + 2.3$ 이 먼저 계산됨

$10 / 3 / 2$

// $10 / 3$ 이 먼저 계산됨

$2 * 5 / 2$

// $2 * 5$ 가 먼저 계산됨

02. 여러 가지 연산자들

사칙 연산자

- [예제 4-1] 사칙 연산자의 우선 순위를 테스트하는 프로그램

```
1  class FourRulesExample1 {
2      public static void main(String args[]) {
3          int    num1 = 1 / 2 - 3;           // / 연산자가 먼저 처리됨
4          double num2 = 2.0 + 1.5 * 2.0;     // * 연산자가 먼저 처리됨
5          int    num3 = 10 / 3 / 2;         // 왼쪽 연산자부터 순서대로 처리됨
6          int    num4 = 2 * 5 / 2;         // 왼쪽 연산자부터 순서대로 처리됨
7          System.out.println(num1);
8          System.out.println(num2);
9          System.out.println(num3);
10         System.out.println(num4);
11     }
12 }
```



```
E:\work\chap4\4-2-1>java FourRulesExample1
-3
5.0
1
5
E:\work\chap4\4-2-1>
```

02. 여러 가지 연산자들

사칙 연산자

• 사칙 연산자의 자동 타입 변환 (1)

<code>100 + 200L</code>	<code>// 결과는 long 타입의 300</code>
<code>3.0 - 2</code>	<code>// 결과는 double 타입의 1.0</code>
<code>10.0f / 2L</code>	<code>// 결과는 float 타입의 5.0</code>

• 사칙 연산자의 자동 타입 변환 (2)

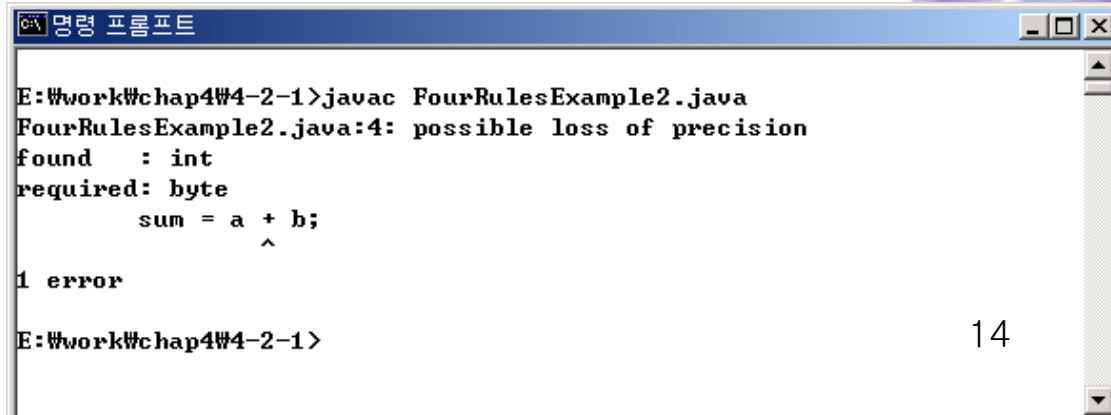
<code>byte a = 2, b = 3;</code>	
<code>short c = 4;</code>	
<code>sum = a + b;</code>	<code>// a + b의 결과는 byte 타입이 아니라 int 타입이 됨</code>
<code>diff = c - b;</code>	<code>// c - b의 결과는 short 타입이 아니라 int 타입이 됨</code>

02. 여러 가지 연산자들

사칙 연산자

- [예제 4-2] 사칙 연산자의 자동 타입 변환으로 문제가 발생하는 프로그램

```
1  class FourRulesExample2 {  
2      public static void main(String args[]) {  
3          byte a = 2, b = 3, sum;  
4          sum = a + b;  
5          System.out.println(sum);  
6      }  
7  }
```



```
명령 프롬프트  
E:\work\chap4\4-2-1>javac FourRulesExample2.java  
FourRulesExample2.java:4: possible loss of precision  
found   : int  
required: byte  
    sum = a + b;  
              ^  
1 error  
E:\work\chap4\4-2-1>
```

02. 여러 가지 연산자들

부호 연산자

• 부호 연산자 : 부호를 나타내는 연산자

+ 피연산자



피연산자의 부호를 그대로 둡니다.

- 피연산자



피연산자의 부호가 +이면 -로
-이면 +로 바꿉니다.

▶▶ 피연산자는 수치 타입이어야 합니다.

• 사용 예 (1)

-3

+12.5

+4

-0.5

02. 여러 가지 연산자들

부호 연산자

- 부호 연산자와 덧셈 연산자의 차이

```
num2 = - num1;
```

```
sum = + a + b;
```

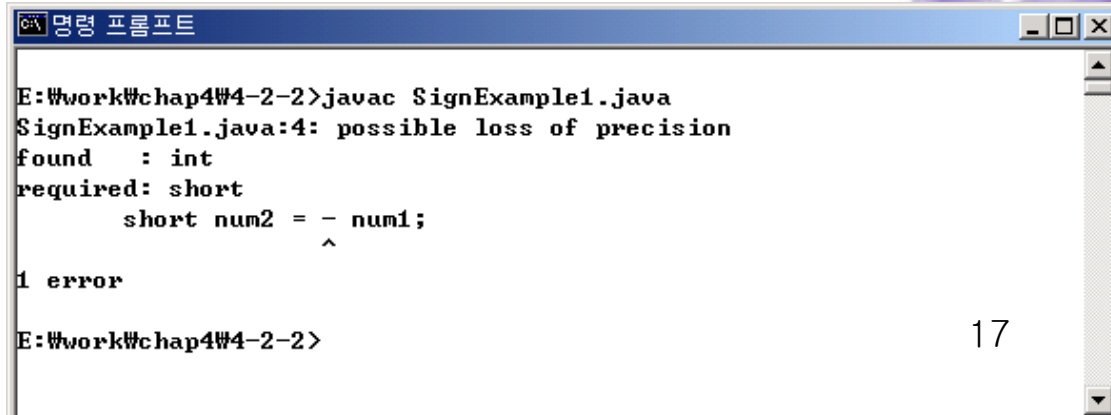
// a앞의 +는 부호 연산자, a와 b사이의 +는 덧셈 연산자

02. 여러 가지 연산자들

부호 연산자

- [예제 4-3] 부호 연산자의 작동 방식을 잘 모르고 작성한 프로그램

```
1    class SignExample1 {  
2        public static void main(String args[]) {  
3            short num1 = 100;  
4            short num2 = - num1;  
5            System.out.println(num2);  
6        }  
7    }
```



```
명령 프롬프트  
E:\work\chap4\4-2-2>javac SignExample1.java  
SignExample1.java:4: possible loss of precision  
found   : int  
required: short  
        short num2 = - num1;  
                      ^  
1 error  
E:\work\chap4\4-2-2>
```

연산자

02. 여러 가지 연산자들

문자열 연결 연산자

.. 문자열 연결 연산자 : 두 개의 문자열을 연결해서 새로운 문자열을 만드는 연산자

문자열1 + 문자열2

↑
문자열1 뒤에 문자열2를
연결한 문자열을 만듭니다.

문자열 + 비문자열데이터

↖
문자열이 아닌 데이터를
문자열로 바꾼 후에 연결합니다.

비문자열데이터 + 문자열

.. 사용 예

"Hello, " + "Java"

"num = " + 30

365 + "일"

02. 여러 가지 연산자들

문자열 연결 연산자

• 두 번째 +는 덧셈 연산자일까요? 문자열 연결 연산자일까요?

`"num=" + 3 + 4`

● 02. 여러 가지 연산자들

● 문자열 연결 연산자

• 다음 연산식의 결과는 무엇일까요?

$3 + 4 + \text{"=num"}$

연산자

02. 여러 가지 연산자들

문자열 연결 연산자

- [예제 4-4] +의 처리 순서를 테스트하는 프로그램

```
1    class ConcatExample1 {  
2        public static void main(String args[]) {  
3            String str1 = "num=" + 3 + 4;  
4            String str2 = 3 + 4 + "=num";  
5            System.out.println(str1);  
6            System.out.println(str2);  
7        }  
8    }
```



```
명령 프롬프트  
E:\work\chap4\4-2-3>java ConcatExample1  
num=34  
?=num  
E:\work\chap4\4-2-3>
```

02. 여러 가지 연산자들

단순 대입 연산자

•• 단순 대입 연산자 : 오른쪽 식의 결과를 왼쪽 변수에 대입하는 연산자

변수 = 식



오른쪽 식의 결과를 왼쪽 변수에 대입합니다.

•• 사육 예

```
num = 3    sum = a + b    ch = 'A'    total = total + 1
```

02. 여러 가지 연산자들

단순 대입 연산자

- 단순 대입 연산자를 사용할 때는 좌변과 우변의 타입에 주의해야 합니다.

```
int num1 = 2 + 3;    // 올바른 대입  
int num2 = 1.2;      // 잘못된 대입
```

- 좁은 범위 타입의 값을 넓은 범위 타입의 변수에 대입하는 것은 가능합니다.

```
long num3 = 100;      // 올바른 대입
```

02. 여러 가지 연산자들

단순 대입 연산자

- [예제 4-5] 수치 타입의 값을 그보다 넓은 범위 타입의 변수에 대입하는 예

```
1    class AssignmentExample1 {  
2        public static void main(String args[]) {  
3            byte    num1 = 9;  
4            short   num2 = num1;  
5            int     num3 = num2;  
6            long    num4 = num3;  
7            float   num5 = num4;  
8            double  num6 = num5;  
9            System.out.println(num6);  
10        }  
11    }
```



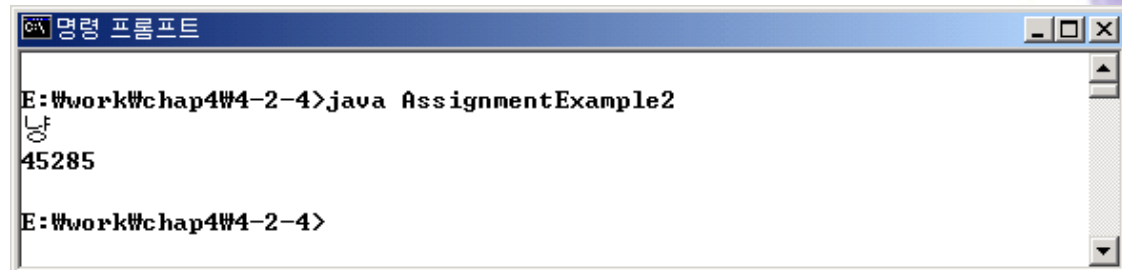
```
C:\>명령 프롬프트  
E:\work\chap4\4-2-4>java AssignmentExample1  
9.0  
E:\work\chap4\4-2-4>
```


02. 여러 가지 연산자들

단순 대입 연산자

- [예제 4-6] `char` 타입의 값을 `int` 타입의 변수에 대입하는 예

```
1    class AssignmentExample2 {  
2        public static void main(String args[]) {  
3            char ch = '냥';  
4            System.out.println(ch);  
5            int num = ch;  
6            System.out.println(num);  
7        }  
8    }
```



```
명령 프롬프트  
E:\work\chap4\4-2-4>java AssignmentExample2  
냥  
45285  
E:\work\chap4\4-2-4>
```

02. 여러 가지 연산자들

단순 대입 연산자

- 단순 대입 연산자는 `boolean` 타입과 레퍼런스 타입에도 사용할 수 있습니다.

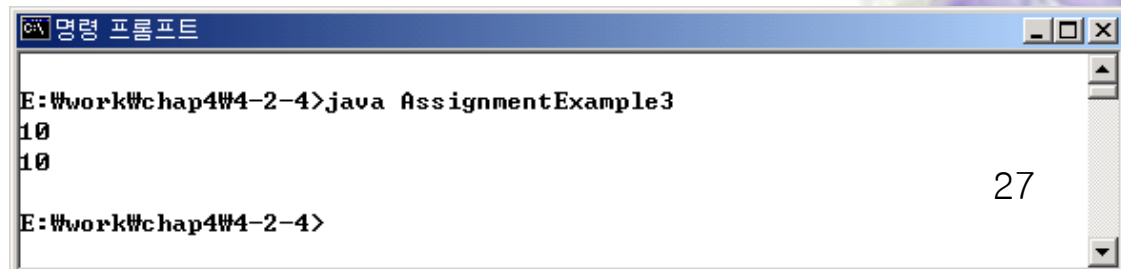
```
boolean truth = 10 > 20;    // 10 > 20의 결과는 boolean 타입 변수에 대입할 수 있음  
String str = "Hello, Java"; // 문자열은 String 타입 변수에 대입할 수 있음
```

02. 여러 가지 연산자들

단순 대입 연산자

- [예제 4-7] 대입 연산자가 산출하는 값을 이용하는 프로그램

```
1    class AssignmentExample3 {  
2        public static void main(String args[]) {  
3            int num1, num2;  
4            num2 = (num1 = 10);  
5            System.out.println(num1);  
6            System.out.println(num2);  
7        }  
8    }
```



```
명령 프롬프트  
E:\work\chap4\4-2-4>java AssignmentExample3  
10  
10  
E:\work\chap4\4-2-4>
```

02. 여러 가지 연산자들

단순 대입 연산자

• 다음 대입문에서는 어느 = 연산자가 먼저 처리될까요?

```
num2 = num1 = 10;
```

= 연산자는 오른쪽부터 처리되기 때문에 위의 대입문은
num2 = (num1 = 10); 과 똑같은 일을 합니다.

02. 여러 가지 연산자들

증가 연산자와 감소 연산자

- 증가 연산자 ++ : 변수의 값에 1을 더한 결과를 다시 변수에 담는 연산자
- 감소 연산자 -- : 변수의 값에서 1을 뺀 결과를 다시 변수에 담는 연산자

변수 ++ ++ 변수

↑ ↑
변수의 값에 1을 더한 결과를
다시 변수에 담습니다.

변수 -- -- 변수

↑ ↑
변수의 값에서 1을 뺀 결과를
다시 변수에 담습니다.

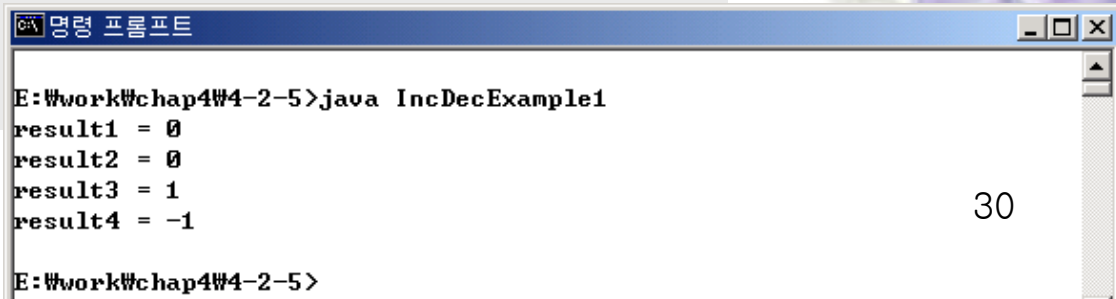
연산자

02. 여러 가지 연산자들

증가 연산자와 감소 연산자

- [예제 4-8] 증가 연산자와 감소 연산자가 산출하는 값을 출력하는 프로그램

```
1  class IncDecExample1 {
2      public static void main(String args[]) {
3          int num1 = 0, num2 = 0, num3 = 0, num4 = 0;
4          int result1 = num1++;          // ++ 연산의 결과는 num1의 기존 값
5          int result2 = num2--;          // -- 연산의 결과는 num2의 기존 값
6          int result3 = ++num3;          // ++ 연산의 결과는 num3의 새로운 값
7          int result4 = --num4;          // -- 연산의 결과는 num4의 새로운 값
8          System.out.println("result1 = " + result1);
9          System.out.println("result2 = " + result2);
10         System.out.println("result3 = " + result3);
11         System.out.println("result4 = " + result4);
12     }
13 }
```



```
명령 프롬프트
E:\work\chap4\4-2-5>java IncDecExample1
result1 = 0
result2 = 0
result3 = 1
result4 = -1
E:\work\chap4\4-2-5>
```

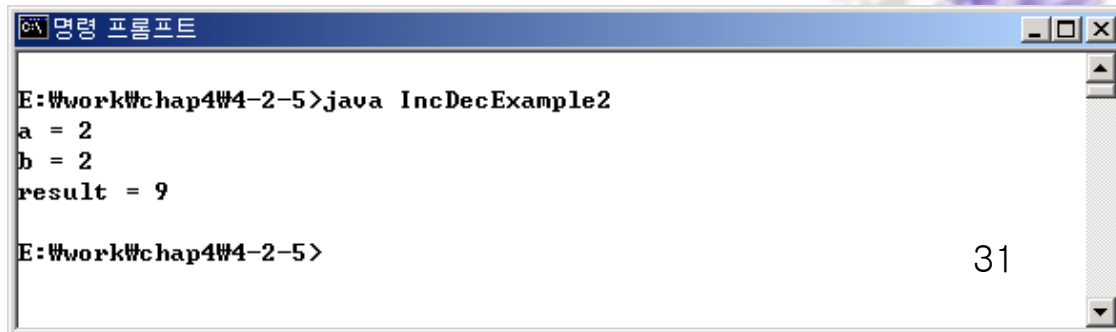
연산자

02. 여러 가지 연산자들

증가 연산자와 감소 연산자

- [예제 4-9] 복잡한 연산식에 ++, -- 연산자를 사용한 예 – 바람직하지 못한 예

```
1    class IncDecExample2 {  
2        public static void main(String args[]) {  
3            int a = 2, b = 3, result;  
4            result = ++a + --b * a--;  
5            System.out.println("a = " + a);  
6            System.out.println("b = " + b);  
7            System.out.println("result = " + result);  
8        }  
9    }
```



```
명령 프롬프트  
E:\work\chap4\4-2-5>java IncDecExample2  
a = 2  
b = 2  
result = 9  
E:\work\chap4\4-2-5>
```

02. 여러 가지 연산자들

수치 비교 연산자

• 수치 비교 연산자 : 두 수의 크기를 비교하는 연산자

피연산자1 < 피연산자2

이 값이 이 값보다 작으면 true,
그렇지 않으면 false입니다.

피연산자1 > 피연산자2

이 값이 이 값보다 크면 true,
그렇지 않으면 false입니다.

피연산자1 <= 피연산자2

이 값이 이 값보다 작거나 같으면 true,
그렇지 않으면 false입니다.

피연산자1 >= 피연산자2

이 값이 이 값보다 크거나 같으면 true,
그렇지 않으면 false입니다.

▶▶ 피연산자는 모두 수치 타입이어야 합니다.

02. 여러 가지 연산자들

수치 비교 연산자

•• 사용 예

`3 < 4`

`10 > 20.0`

`12.5f <= 11`

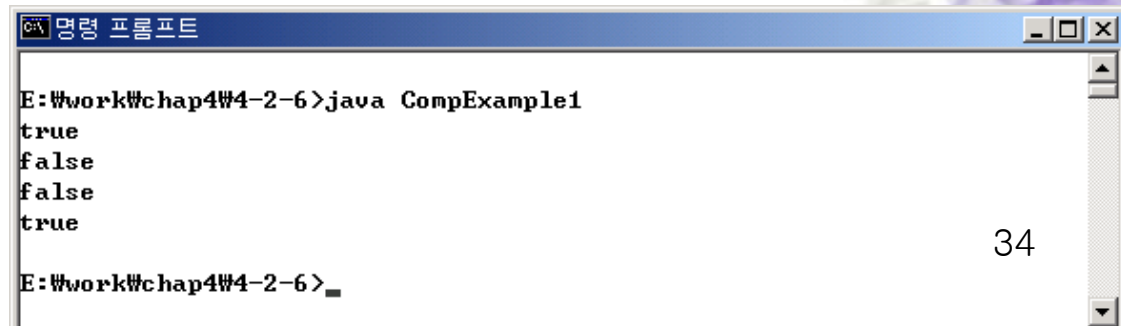
`7 >= 7`

02. 여러 가지 연산자들

수치 비교 연산자

- [예제 4-10] 수치 비교 연산자의 사용 예

```
1    class CompExample1 {  
2        public static void main(String args[]) {  
3            System.out.println(3 < 4);  
4            System.out.println(10 > 20.0);  
5            System.out.println(12.5f <= 11);  
6            System.out.println(7 >= 7);  
7        }  
8    }
```



```
명령 프롬프트  
E:\work\chap4\4-2-6>java CompExample1  
true  
false  
false  
true  
E:\work\chap4\4-2-6>
```

02. 여러 가지 연산자들

동등 연산자

• 동등 연산자 : 두 데이터의 값이 같은지 다른지 판단하는 연산자

피연산자1 == 피연산자2

↑ ↑
이 값과 이 값이 같으면 true,
다르면 false입니다.

피연산자1 != 피연산자2

↑ ↑
이 값과 이 값이 다르면 true,
같으면 false입니다.

• 사용 예

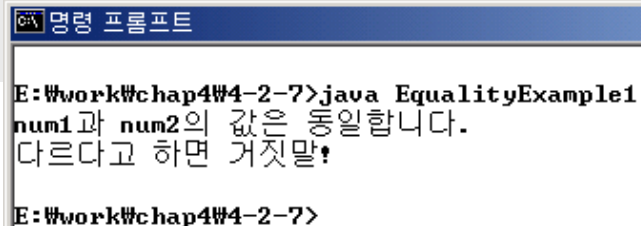
```
1 == 10      'a' == 'a'      0.5 != 0.5      true != false
```

02. 여러 가지 연산자들

동등 연산자

- [예제 4-11] 동등 연산자의 사용 예

```
1  class EqualityExample1 {
2      public static void main(String args[]) {
3          int num1 = 5;
4          int num2 = 2 + 3;
5          if (num1 == num2)
6              System.out.println("num1 과 num2 의 값 은
7  동일합니다.");
8          if ((num1 != num2) == false)
9              System.out.println("다르다고 하면 거짓말!");
10     }
}
```



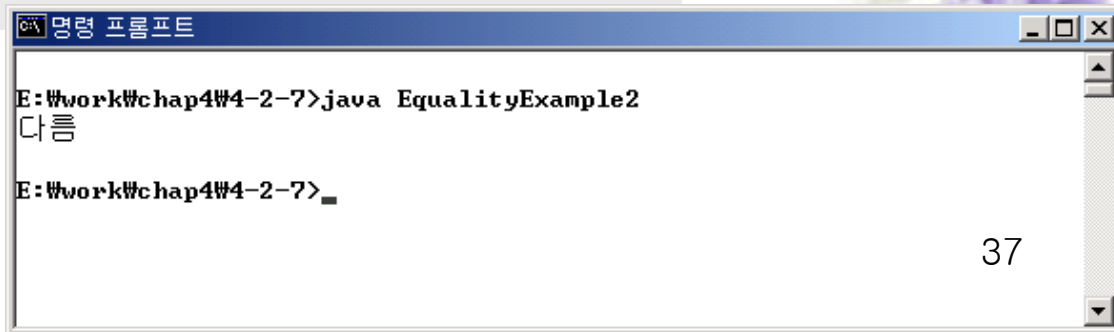
```
C:\> 명령 프롬프트
E:\work\chap4\4-2-7> java EqualityExample1
num1과 num2의 값은 동일합니다.
다르다고 하면 거짓말!
E:\work\chap4\4-2-7>
```

02. 여러 가지 연산자들

동등 연산자

- [예제 4-12] 부동소수점 수에 대해 동등 연산자를 사용한 프로그램

```
1  class EqualityExample2 {  
2      public static void main(String args[]) {  
3          double num1 = 1.1 + 2.2;  
4          double num2 = 3.3;  
5          if (num1 == num2)  
6              System.out.println("같음");  
7          if (num1 != num2)  
8              System.out.println("다름");  
9      }  
10 }
```



```
명령 프롬프트  
E:\work\chap4\4-2-7>java EqualityExample2  
다름  
E:\work\chap4\4-2-7>
```

02. 여러 가지 연산자들

논리 연산자

• 논리 연산자 : `boolean` 값들을 가지고 논리식을 만드는 연산자

피연산자1 & 피연산자2



두 값이 모두 true면 true,
그렇지 않으면 false입니다.

피연산자1 | 피연산자2



두 값이 모두 false면 false,
그렇지 않으면 true입니다.

피연산자1 ^ 피연산자2



하나가 true, 하나가 false면 true,
그렇지 않으면 false입니다.

!피연산자1



이 값이 true이면 false,
false이면 true입니다.

▶▶ 피연산자는 모두 불리언 타입이어야 합니다. 피연산자가 정수 타입인 & | ^는 논리 연산자가 아니라 비트 연산자입니다.

02. 여러 가지 연산자들

논리 연산자

•• 사용 예

```
true & false    false | true    true ^ false    !(3 == 3)
```

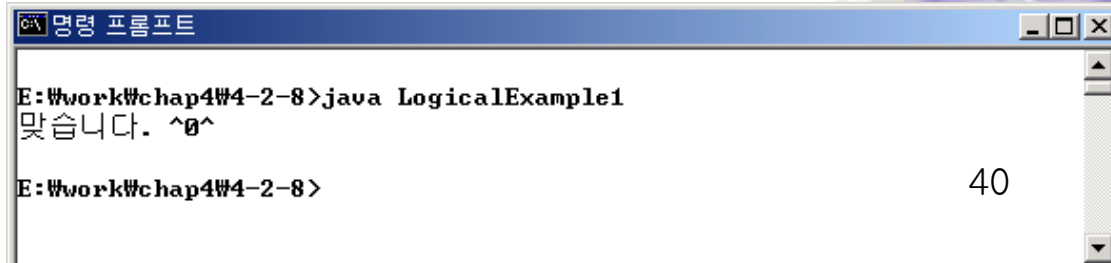
연산자

02. 여러 가지 연산자들

논리 연산자

- [예제 4-13] 논리 연산자의 사용 예

```
1    class LogicalExample1 {  
2        public static void main(String args[]) {  
3            int a = 3, b =4, c =3, d =5;  
4            if ((a == 2 | a == c) & !(c > d) & (1 == b ^ c != d))  
5                System.out.println("맞습니다. ^0^");  
6            else  
7                System.out.println("아닌데요. 0TL");  
8        }  
9    }
```



```
명령 프롬프트  
E:\work\chap4\4-2-8>java LogicalExample1  
맞습니다. ^0^  
E:\work\chap4\4-2-8>
```


02. 여러 가지 연산자들

조건 AND/OR 연산자

• 조건 AND/OR 연산자 : 최적화된 AND/OR 연산자

피연산자1 && 피연산자2

두 값이 모두 true면 true,
그렇지 않으면 false입니다.

피연산자1 || 피연산자2

두 값이 모두 false면 false,
그렇지 않으면 true입니다.

▶▶ 피연산자는 모두 불리언 타입이어야 합니다.

• 사용 예

```
true && false
```

```
false || true
```

02. 여러 가지 연산자들

조건 AND/OR 연산자

• 컴퓨터는 다음과 같은 식을 어떤 방식으로 계산할까요?

$$(1 > 2) \ \& \ (3 < 4)$$

02. 여러 가지 연산자들

조건 AND/OR 연산자

• 컴퓨터는 다음과 같은 식을 어떤 방식으로 계산할까요?

$(1 > 2) \ \&\& \ (3 < 4)$

02. 여러 가지 연산자들

조건 AND/OR 연산자

•• | 연산자와 || 연산자의 작동 방식 차이

$(2 == 2) \mid (3 != 3)$

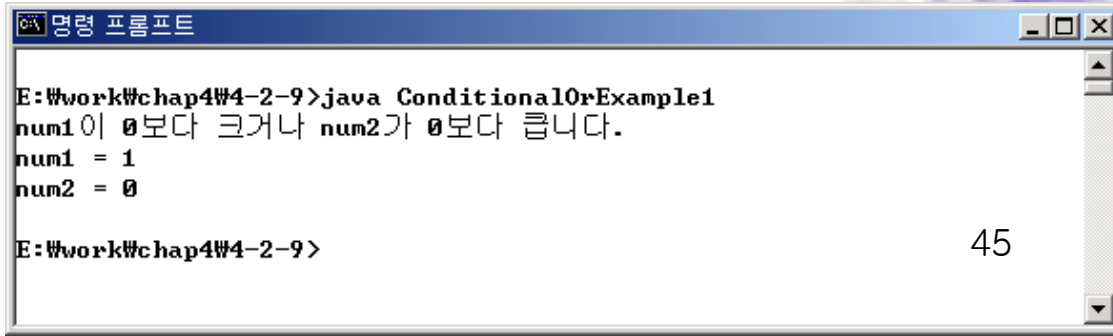
$(2 == 2) \parallel (3 != 3)$

02. 여러 가지 연산자들

조건 AND/OR 연산자

- [예제 4-14] || 연산자의 사용 예

```
1    class ConditionalOrExample1 {
2        public static void main(String args[]) {
3            int num1 = 0, num2 = 0;
4            if (++num1 > 0 || ++num2 > 0)
5                System.out.println("num1이 0보다 크거나 num2가 0보다 큼니다.");
6            System.out.println("num1 = " + num1);
7            System.out.println("num2 = " + num2);
8        }
9    }
```



```
E:\work\chap4\4-2-9>java ConditionalOrExample1
num1이 0보다 크거나 num2가 0보다 큼니다.
num1 = 1
num2 = 0

E:\work\chap4\4-2-9>
```

02. 여러 가지 연산자들

조건 연산자

• 조건 연산자 : 조건식의 결과에 따라 두 피연산자 중 하나를 취하는 연산자

조건식 ? 식1 : 식2

↑ ↑ ↑
이 조건식이 true이면 식1을 계산하고 그렇지 않으면 식2를 계산합니다.

• 사용 예

```
a < b ? a + 1 : b * 2
```

02. 여러 가지 연산자들

조건 연산자

• 조건 연산자는 다음과 같은 대입문 형태로 많이 사용됩니다.

```
max = a < b ? a : b;    // a < b 이면 a 값, 그렇지 않으면 b 값이 max에 대입됨
```

02. 여러 가지 연산자들

조건 연산자

- [예제 4-15] 조건 연산자의 사용 예

```
1  class ConditionalOpExample1 {  
2      public static void main(String args[]) {  
3          int a = 20, b = 30, max;  
4          max = a < b ? a : b;  
5          System.out.println(max);  
6      }  
7  }
```



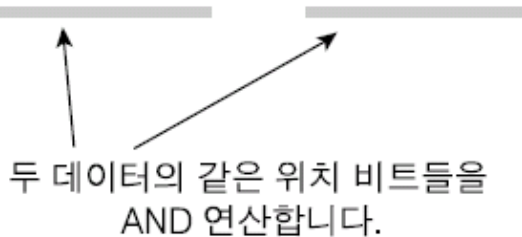
```
명령 프롬프트  
E:\work\chap4\4-2-10>java ConditionalOpExample1  
20  
E:\work\chap4\4-2-10>
```


02. 여러 가지 연산자들

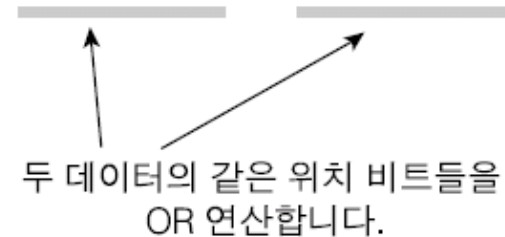
비트 연산자

•• 비트 연산자 : 데이터 구성 비트를 가지고 AND, OR, XOR, NOT 연산을 수행하는 연산자

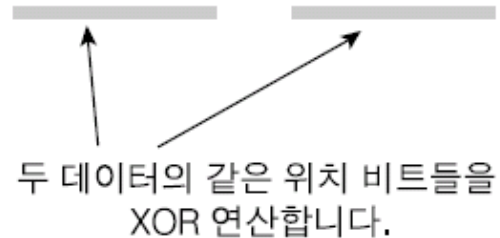
피연산자1 & 피연산자2



피연산자1 | 피연산자2



피연산자1 ^ 피연산자2



~피연산자



02. 여러 가지 연산자들

비트 연산자

• 비트 연산의 규칙

[비트 AND 연산]

피연산자2 피연산자1	1	0
1	1	0
0	0	0

[비트 OR 연산]

피연산자2 피연산자1	1	0
1	1	1
0	1	0

[비트 XOR 연산]

피연산자2 피연산자1	1	0
1	1	1
0	1	0

[비트 NOT 연산]

피연산자	결과
1	0
0	1

연산자

02. 여러 가지 C 비트 연산자

```
명령 프롬프트
E:\work\chap4\4-2-11>java BitsExample1
FF000000
FFFFFF00
00FFFF00
0000FFFF

E:\work\chap4\4-2-11>
```

• [예제 4-16] 비트 연산자의 사용 예

```
1  class BitsExample1 {
2      public static void main(String args[]) {
3          int num1 = 0xFFFF0000; }
4          int num2 = 0xFF00FF00; }
5          int result1 = num1 & num2;
6          int result2 = num1 | num2;
7          int result3 = num1 ^ num2;
8          int result4 = ~num1;
9          System.out.printf("%08X %n", result1); }
10         System.out.printf("%08X %n", result2); }
11         System.out.printf("%08X %n", result3); }
12         System.out.printf("%08X %n", result4); }
13     }
14 }
```

비트 연산에 사용할 데이터는
16진수로 표기하는 것이 편리합니다.

결과도 보기 편하게
8자리의 16진수로 출력합니다

02. 여러 가지 연산자들

비트 연산자

- 비트 연산자는 피연산자의 타입이 서로 다르면 넓은 타입 쪽으로 자동 변환을 수행합니다.

```
1 & 2           // 결과는 int 타입
3L | 4          // 결과는 long 타입
0xff ^ 0x0fL    // 결과는 long 타입
```

- 단, 두 피연산자가 모두 int보다 좁은 타입이면 둘 다 int 타입으로 자동 변환을 수행합니다

```
byte  num1 = 1, num2 = 2;
short num3 = 3;
char  ch = 'A';
byte  result1 = num1 & num2; // 결과는 int 타입이므로 잘못된 대입문
short result2 = num2 | num3; // 결과는 int 타입이므로 잘못된 대입문
int    result3 = num3 ^ ch;   // 결과는 int 타입이므로 올바른 대입문
byte  result4 = ~num1;        // 결과는 int 타입이므로 잘못된 대입문
```

02. 여러 가지 연산자들

쉬프트 연산자

.. 쉬프트 연산자 : 데이터 구성 비트를 오른쪽/왼쪽으로 밀어서 이동시키는 연산자

정수 << 비트수

↑ ↑

정수를 구성하는 주어진 비트수만큼
비트를 왼쪽으로 이동하고
 빈 공간은 0으로 채웁니다.

정수 >> 비트수

↑ ↑

정수를 구성하는 주어진 비트수만큼
비트를 오른쪽으로 이동하고
 빈 공간은 MSB와 똑같은
 비트로 채웁니다

정수 >>> 비트수

↑ ↑

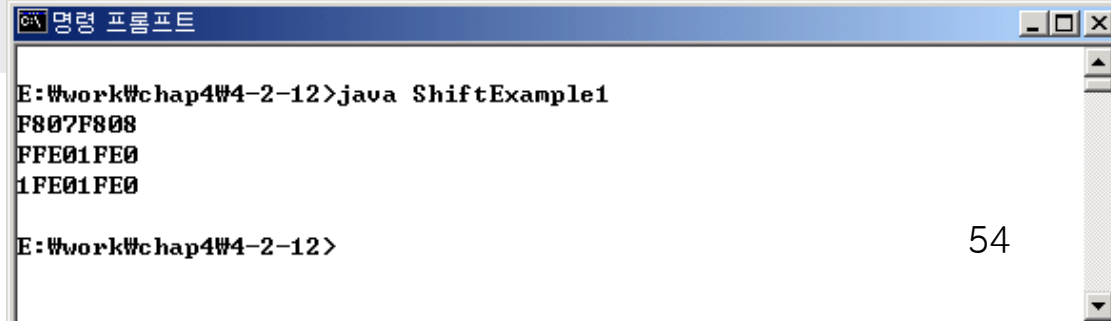
정수를 구성하는 주어진 비트수만큼
비트를 오른쪽으로 이동하고
 빈 공간은 0으로 채웁니다.

02. 여러 가지 연산자들

쉬프트 연산자

- [예제 4-17] 쉬프트 연산자의 사용 예

```
1  class ShiftExample1 {
2      public static void main(String args[]) {
3          int num = 0xFF00FF01;           // 11111111 00000000 11111111 00000001
4          int result1 = num << 3;         // 11111000 00000111 11111000 00001000
5          int result2 = num >> 3;         // 11111111 11100000 00011111 11100000
6          int result3 = num >>> 3;        // 00011111 11100000 00011111 11100000
7          System.out.printf("%08X %n", result1);
8          System.out.printf("%08X %n", result2);
9          System.out.printf("%08X %n", result3);
10     }
11 }
```



```
명령 프롬프트
E:\work\chap4\4-2-12>java ShiftExample1
F807F808
FFE01FE0
1FE01FE0
E:\work\chap4\4-2-12>
```

02. 여러 가지 연산자

쉬프트 연산자

- [예제 4-18] 쉬프트 연산자의 사용 예

```
명령 프롬프트
E:\work\chap4\4-2-12>java ShiftExample2
FFFFFFFFE
80000000
00000000
FFFFFFFFE
FFFFFFFC
FFFFFFF8
E:\work\chap4\4-2-12>
```

```
1  class ShiftExample2 {
2      public static void main(String args[]) {
3          int num1 = 0xFFFFFFFF;    // 11111111 11111111 11111111 11111110
4          int num2 = num1 << 30;
5          int num3 = num1 << 31;
6          int num4 = num1 << 32;
7          int num5 = num1 << 33;
8          int num6 = num1 << 34;
9          System.out.printf("%08X %n", num1);
10         System.out.printf("%08X %n", num2);
11         System.out.printf("%08X %n", num3);
12         System.out.printf("%08X %n", num4);
13         System.out.printf("%08X %n", num5);
14         System.out.printf("%08X %n", num6);
15     }
16 }
```

연산자

02. 여러 가지 연산자들

쉬프트 연산자

- 쉬프트 연산자는 피연산자가 int보다 좁은 타입이면 int 타입으로 자동 변환을 수행합니다.

```
byte  num1 = 1;  
short num2 = 3;  
char  ch = 'A';  
byte  result1 = num1 << 3;  
short result2 = num2 >> 2L;  
char  result3 = ch >>> 1;
```

```
// 결과는 int 타입이므로 잘못된 대입문  
// 결과는 int 타입이므로 잘못된 대입문  
// 결과는 int 타입이므로 잘못된 대입문
```


02. 여러 가지 연산자들

복합 대입 연산자

• 복합 대입 연산자 : $+$, $-$, $*$, $/$, $\%$, $\&$, $|$, $^$, \ll , \gg , \ggg 와 $=$ 연산자의 기능을 함께 수행하는 연산자

변수 $+=$ 식

(변수의 기존값 + 식의 결과)를
변수에 대입합니다.

변수 $-=$ 식

(변수의 기존값 - 식의 결과)를
변수에 대입합니다.

변수 $*=$ 식

(변수의 기존값 * 식의 결과)를
변수에 대입합니다.

변수 $/=$ 식

(변수의 기존값 / 식의 결과)를
변수에 대입합니다.

변수 $\%=$ 식

(변수의 기존값 % 식의 결과)를
변수에 대입합니다.

변수 $\&=$ 식

(변수의 기존값 & 식의 결과)를
변수에 대입합니다.

변수 $|=$ 식

(변수의 기존값 | 식의 결과)를
변수에 대입합니다.

변수 $\^{}=$ 식

(변수의 기존값 ^ 식의 결과)를
변수에 대입합니다.

변수 $\ll=$ 식

(변수의 기존값 \ll 식의 결과)를
변수에 대입합니다

변수 $\gg=$ 식

(변수의 기존값 \gg 식의 결과)를
변수에 대입합니다.
변수 $\ggg=$ 식

변수 $\ggg=$ 식

(변수의 기존값 \ggg 식의 결과)를
변수에 대입합니다.

02. 여러 가지 연산자들

복합 대입 연산자

•• 사용 예

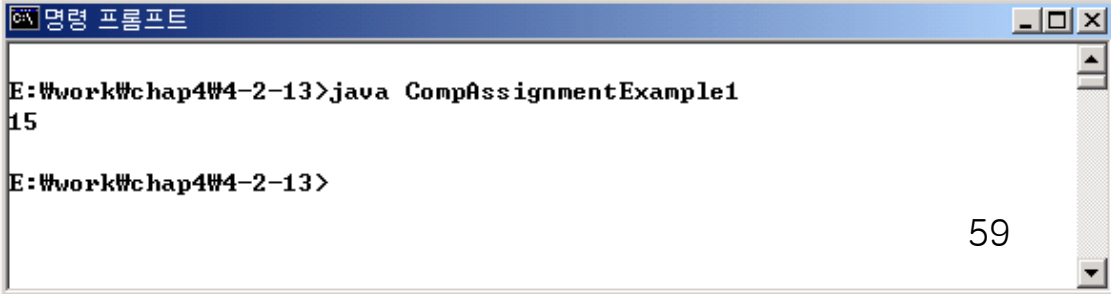
```
total += 2  
result *= 3.2 - 1.5  
num1 &= 4  
num2 <<= 2
```

02. 여러 가지 연산자들

복합 대입 연산자

- [예제 4-19] 복합 대입 연산자의 사용 예

```
1    class CompAssignmentExample1 {  
2        public static void main(String args[]) {  
3            int num = 29;  
4            num -= 2 + 3 * 4; ----- num = num - (2 + 3 * 4) 와 똑같은 효과를 갖는 대입문  
5            System.out.println(num);  
6        }  
7    }
```

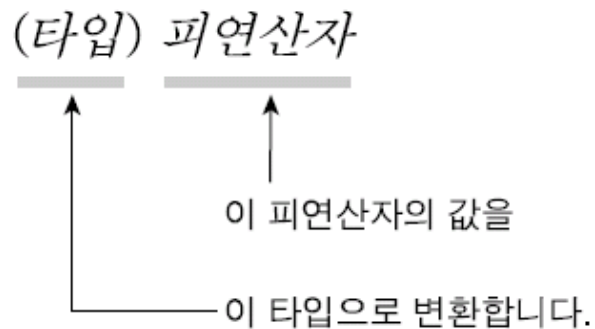


```
명령 프롬프트  
E:\work\chap4\4-2-13>java CompAssignmentExample1  
15  
E:\work\chap4\4-2-13>
```

02. 여러 가지 연산자들

캐스트 연산자

• 캐스트 연산자 : 타입의 변환을 수행하는 연산자



• 사용 예

```
(long) 15    (int) ch    (byte) 12.5    (float) 0.000725
```

02. 여러 가지 연산자들

캐스트 연산자

• 캐스트 연산자는 피연산자와 똑같은 값을 갖는 새로운 타입의 값을 만들어 냅니다.

```
int num1 = 3;  
double num2 = (double) num1;    // int 타입을 double 타입으로 변환
```

num2에는 새로 만들어진 double 타입의 3.0이라는 값이 대입됩니다.

● 02. 여러 가지 연산자들

● 캐스트 연산자

- 캐스트 연산자는 넓은 범위 수치 타입을 좁은 범위로 변환할 수도 있습니다.

```
int num4 = (int) 12.9;           // double 타입을 int 타입으로 변환
```

02. 여러 가지 연산자들

캐스트 연산자

• 캐스트 연산자가 모든 종류의 타입 변환을 다 수행할 수 있는 것은 아닙니다.

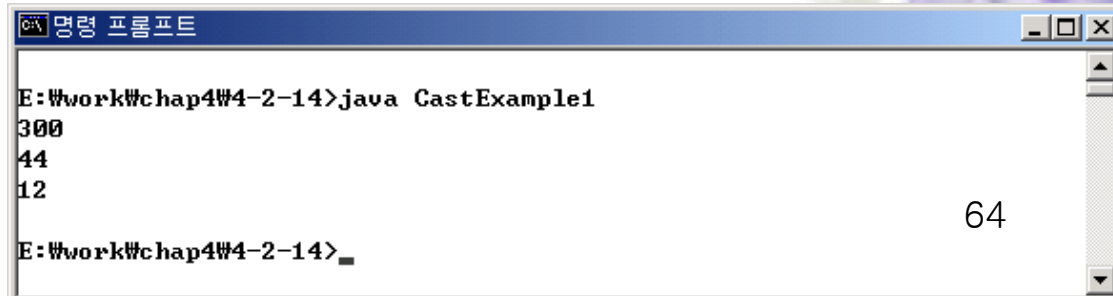
```
int num5 = (int) true;           // 불가능한 캐스트 연산
boolean truth = (boolean) 8;    // 불가능한 캐스트 연산
```

02. 여러 가지 연산자들

캐스트 연산자

- [예제 4-20] 캐스트 연산자의 사용 예

```
1  class CastExample1 {  
2      public static void main(String args[]) {  
3          short num1 = (short) 300;    // int 타입 상수 300을 short 타입으로 변환  
4          byte  num2 = (byte) 300;    // int 타입 상수 300을 byte 타입으로 변환  
5          int   num3 = (int) 12.9;    // double 타입 상수 12.9를 int 타입으로 변환  
6          System.out.println(num1);  
7          System.out.println(num2);  
8          System.out.println(num3);  
9      }  
10 }
```



```
명령 프롬프트  
E:\work\chap4\4-2-14>java CastExample1  
300  
44  
12  
E:\work\chap4\4-2-14>
```


연산자

03. 연산자의 우선 순위

연산자의 우선 순위

우선순위	연산자	처리 순서
↑ 높음	++ -- +(부호 연산자) -(부호 연산자) ~ ! 캐스트 연산자	←--
	* / %	--→
	+(덧셈 연산자, 문자열 연결 연산자) -(뺄셈 연산자)	--→
	<< >> >>>	--→
	< <= > >=	--→
	== !=	--→
	&	--→
	^	--→
		--→
	&&	--→
		--→
	? :	←--
↓ 낮음	= += -= *= /= %= &= = ^= <= >= >>=	←--

03. 연산자의 우선 순위

연산자의 우선 순위

• 우선순위가 같은 사칙 연산자들끼리는 어떤 순서로 처리될까요?

$$a > 0 \ \& \ b < 0$$

이런 연산식에서는
왼쪽 연산자부터 순서대로 처리됩니다.

03. 연산자의 우선 순위

연산자의 우선 순위

• 대입 연산자들끼리는 어떤 순서로 처리될까요?

$$a = b = c = d = 3;$$

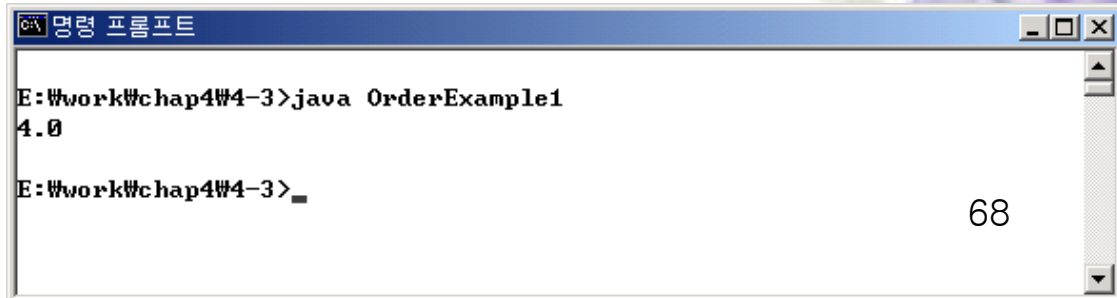
이런 연산식에서는
오른쪽 연산자부터 순서대로 처리됩니다.

03. 연산자의 우선 순위

연산자의 우선 순위

- [예제 4-21] 우선순위가 다른 여러 연산자 함께 사용한 연산식의 예

```
1    class OrderExample1 {  
2        public static void main(String args[]) {  
3            int a = 2, b = 3, c = 5;  
4            double d = 4.0, e = 0.5, f;  
5            f = a + b == c ? d : e;  
6            System.out.println(f);  
7        }  
8    }
```



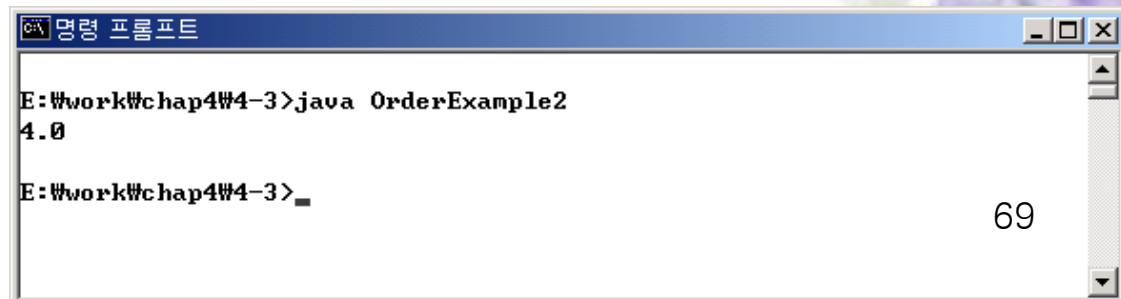
```
명령 프롬프트  
E:\work\chap4\4-3>java OrderExample1  
4.0  
E:\work\chap4\4-3>
```

03. 연산자의 우선 순위

연산자의 우선 순위

- [예제 4-22] 우선순위가 다른 여러 연산자 함께 사용한 연산식의 예 – 바람직한 예

```
1    class OrderExample2 {  
2        public static void main(String args[]) {  
3            int a = 2, b = 3, c = 5;  
4            double d = 4.0, e = 0.5, f;  
5            f = ((a + b) == c) ? d : e;  
6            System.out.println(f);  
7        }  
8    }
```



```
명령 프롬프트  
E:\work\chap4\4-3>java OrderExample2  
4.0  
E:\work\chap4\4-3>
```

어제

많은

일을

이루고

싶습니다~