

10주 3강

기본 자바 소개 3



송실사이버대학교

송실사이버대학교의 강의콘텐츠는
저작권법에 의하여 보호를 받는바, 무단
전재, 배포, 전송, 대여 등을 금합니다.

* 사용서체 : 나눔글꼴

이번 주차에는...

기본 자바 소개 3

- 자바의 연산자
- 산술 연산자
- 증감 연산자
- 관계 연산자
- 논리 연산자
- 비트 연산자

1. 산술 연산자(1)

■ 기본 연산자

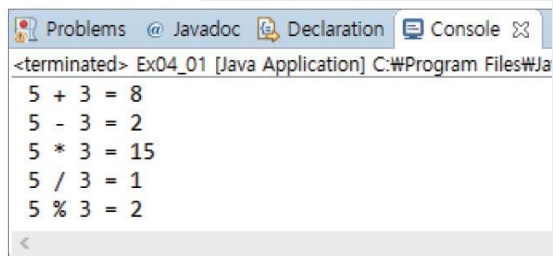
표 4-1 산술 연산자의 종류

| 산술 연산자 | 설명 | 사용 예 | |
|--------|-------|----------|-----------------------------|
| = | 대입 | $a=3$ | 정수 3을 a에 대입한다. |
| + | 더하기 | $a=5+3$ | 5와 3을 더한 값을 a에 대입한다. |
| - | 빼기 | $a=5-3$ | 5에서 3을 뺀 값을 a에 대입한다. |
| * | 곱하기 | $a=5*3$ | 5와 3을 곱한 값을 a에 대입한다. |
| / | 나누기 | $a=5/3$ | 5를 3으로 나눈 값을 a에 대입한다. |
| % | 나머지 값 | $a=5\%3$ | 5를 3으로 나눈 뒤 나머지 값을 a에 대입한다. |

2. 산술 연산자(2)

실습 4-1 산술 연산자 사용 예

```
01 public class Ex04_01 {  
02     public static void main(String[] args) {  
03         int a, b = 5, c = 3;  
04  
05         a = b + c; ----- b와 c를 더하기 연산 하여 a에 대입한다.  
06         System.out.printf(" %d + %d = %d  \n", b, c, a);  
07  
08         a = b - c; ----- b와 c를 빼기 연산 하여 a에 대입한다.  
09         System.out.printf(" %d - %d = %d  \n", b, c, a);  
10  
11         a = b * c; ----- b와 c를 곱하기 연산 하여 a에 대입한다.  
12         System.out.printf(" %d * %d = %d  \n", b, c, a);  
13  
14         a = b / c; ----- b와 c를 나누기 연산 하여 a에 대입한다.  
15         System.out.printf(" %d / %d = %d  \n", b, c, a);  
16  
17         a = b % c; ----- b와 c를 나누기 연산 하여 나머지 값을 a에 대입한다.  
18         System.out.printf(" %d %% %d = %d  \n", b, c, a);  
19     }  
20 }
```



```
<terminated> Ex04_01 [Java Application] C:\Program Files\Ja  
5 + 3 = 8  
5 - 3 = 2  
5 * 3 = 15  
5 / 3 = 1  
5 % 3 = 2  
<
```

그림 4-1 실행 결과

3. 산술 연산자(3)

- 우선순위와 강제 형 변환

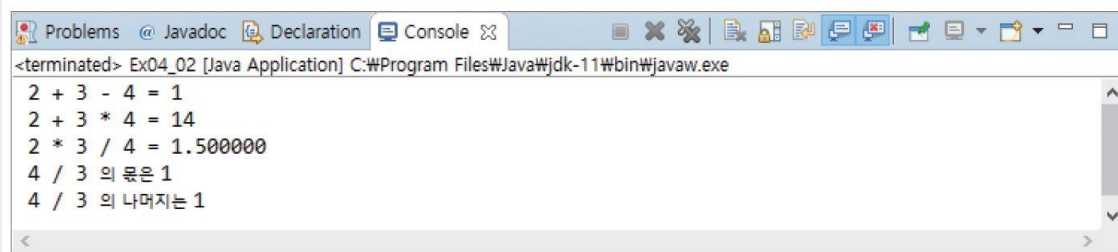
실습 4-2 우선순위와 강제 형 변환의 예

```
01 public class Ex04_02 {
02     public static void main(String[] args) {
03         int a = 2, b = 3, c = 4; ----- 정수형 변수를 선언한다.
04         int result1, mok, namugi; ----- 정수형 변수를 선언한다.
05         float result2; ----- 실수형 변수를 선언한다.
06
07         result1 = a + b - c; ----- 더하기와 빼기 연산을 동시에 수행한다.
08         System.out.printf(" %d + %d - %d = %d  \n", a, b, c, result1);
09
10         result1 = a + b * c; ----- 더하기와 곱하기 연산을 동시에 수행한다.
11         System.out.printf(" %d + %d * %d = %d  \n", a, b, c, result1);
12     }
```

4. 산술 연산자(4)

```
13    result2 = a * b / (float) c; ----- 정수 c를 실수로 강제 형 변환한 다음 연산한다.
14    System.out.printf(" %d * %d / %d = %f %n", a, b, c, result2);
15
16    1 = c / b; ----- 몫을 구한다.
17    System.out.printf(" %d / %d 의 몫은 %d %n", c, b, mok);
18
19    2 = c % b; ----- 나머지를 구한다.
20    System.out.printf(" %d / %d 의 나머지는 %d %n", c, b, namugi);
21 }
22 }
```

namugi 2 mok 1



```
<terminated> Ex04_02 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe
2 + 3 - 4 = 1
2 + 3 * 4 = 14
2 * 3 / 4 = 1.500000
4 / 3 의 몫은 1
4 / 3 의 나머지는 1
```

그림 4-2 실행 결과

5. 산술 연산자(5)

- 7행 : 연산자 우선순위

① `result1 = (a + b) - c;`

② `result1 = a + (b - c);`

- 덧셈과 뺄셈은 계산되는 순서(연산자 우선순위)가 동일하므로 어떤 것을 먼저 계산하든 결과가 동일. 괄호가 없으면 왼쪽에서 오른쪽 방향으로 계산

- 10행

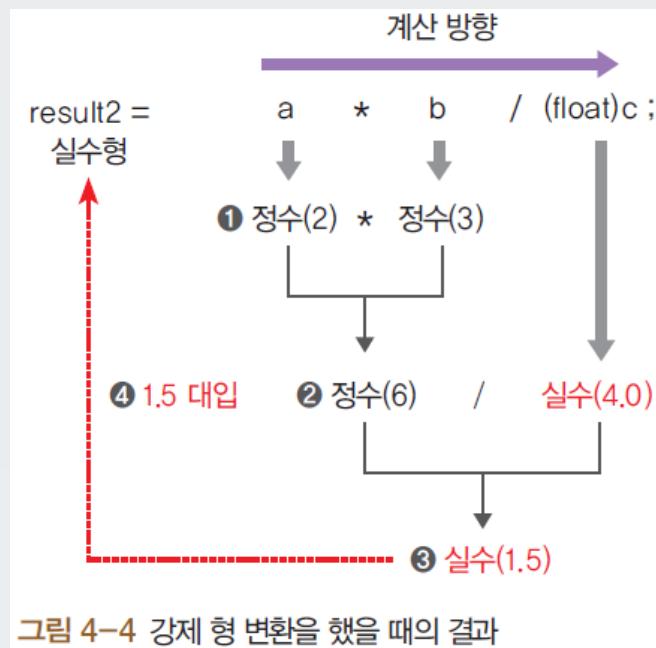
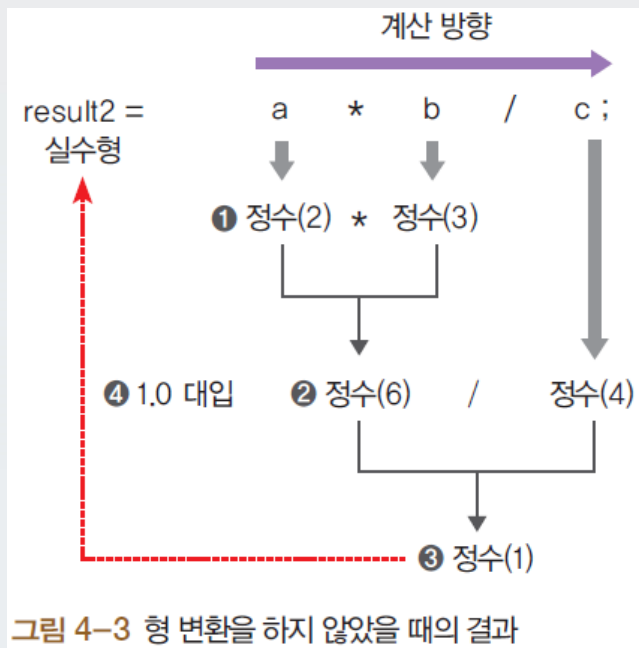
① `result1 = (a + b) * c;` → $(2 + 3) * 4 \rightarrow 5 * 4 \rightarrow 20$

② `result1 = a + (b * c);` → $2 + (3 * 4) \rightarrow 2 + 12 \rightarrow 14$

- 덧셈(또는 뺄셈)과 곱셈(또는 나눗셈)이 같이 나오는 경우에는 곱셈(또는 나눗셈)을 먼저 계산한 다음 덧셈(또는 뺄셈)을 계산

6. 산술 연산자(6)

- 데이터형의 강제 형 변환
 - [실습 4-2]의 13행



7. 산술 연산자(7)

▪ 대입 연산자와 증감 연산자

표 4-2 대입 연산자와 증감 연산자의 종류

| 연산자 | 설명 | 사용 예 | |
|-----------------|--------|--------------------------------------|---|
| <code>+=</code> | 대입 연산자 | <code>a+=3</code> | <code>a=a+3</code> 과 동일하다. |
| <code>-=</code> | 대입 연산자 | <code>a-=3</code> | <code>a=a-3</code> 과 동일하다. |
| <code>*=</code> | 대입 연산자 | <code>a*=3</code> | <code>a=a*3</code> 과 동일하다. |
| <code>/=</code> | 대입 연산자 | <code>a/=3</code> | <code>a=a/3</code> 과 동일하다. |
| <code>%=</code> | 대입 연산자 | <code>a%=3</code> | <code>a=a%3</code> 과 동일하다. |
| <code>++</code> | 증가 연산자 | <code>a++</code> 또는 <code>++a</code> | <code>a+=1</code> 또는 <code>a=a+1</code> 과 동일하다. |
| <code>--</code> | 감소 연산자 | <code>a--</code> 또는 <code>--a</code> | <code>a-=1</code> 또는 <code>a=a-1</code> 과 동일하다. |

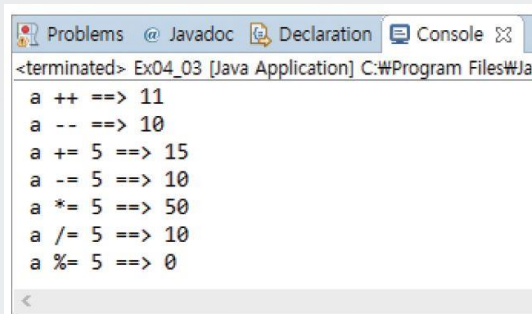
8. 산술 연산자(8)

실습 4-3 증감 연산자와 대입 연산자

```
01 public class Ex04_03 {  
02     public static void main(String[] args) {  
03         int a = 10;  
04  
05         a++; ----- a=a+1과 동일하다.  
06         System.out.printf(" a ++ ==> %d \n", a);  
07  
08         a--; ----- a=a-1과 동일하다.  
09         System.out.printf(" a -- ==> %d \n", a);  
10  
11         a += 5; ----- a=a+5와 동일하다.  
12         System.out.printf(" a += 5 ==> %d \n", a);
```

9. 산술 연산자(9)

```
13
14     a -= 5; ----- a=a-5와 동일하다.
15     System.out.printf(" a -= 5 ==> %d \n", a);
16
17     a *= 5; ----- a=a*5와 동일하다.
18     System.out.printf(" a *= 5 ==> %d \n", a);
19
20     a /= 5; ----- a=a/5와 동일하다.
21     System.out.printf(" a /= 5 ==> %d \n", a);
22
23     a %= 5; ----- a=a%5와 동일하다.
24     System.out.printf(" a %= 5 ==> %d \n", a);
25 }
26 }
```



```
Problems @ Javadoc Declaration Console
<terminated> Ex04_03 [Java Application] C:\Program Files\Ja
a += ==> 11
a -= ==> 10
a += 5 ==> 15
a -= 5 ==> 10
a *= 5 ==> 50
a /= 5 ==> 10
a %= 5 ==> 0
```

그림 4-5 실행 결과

■ a++와 ++a의 차이점

- a++(후치 증가 연산자) : a가 있고, a 값을 1 증가시킴
- ++a(전치 증가 연산자) : a 값을 1 증가시키고, a가 있음

10. 산술 연산자(10)

실습 4-4 증감 연산자 사용 예

```
01 public class Ex04_04 {  
02     public static void main(String[] args) {  
03         int a = 10, b;  
04  
05         b = a++; ----- b=a를 수행한 다음 a를 1 증가시킨다.  
06         System.out.printf(" %d %n", b);  
07  
08          ----- a를 1 증가시킨 다음 b=a를 수행한다.  
09         System.out.printf(" %d %n", b);  
10     }  
11 }
```

b++ = 11

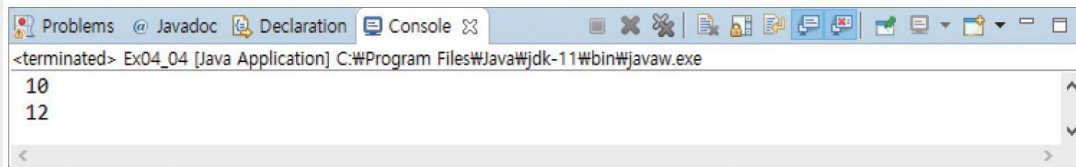
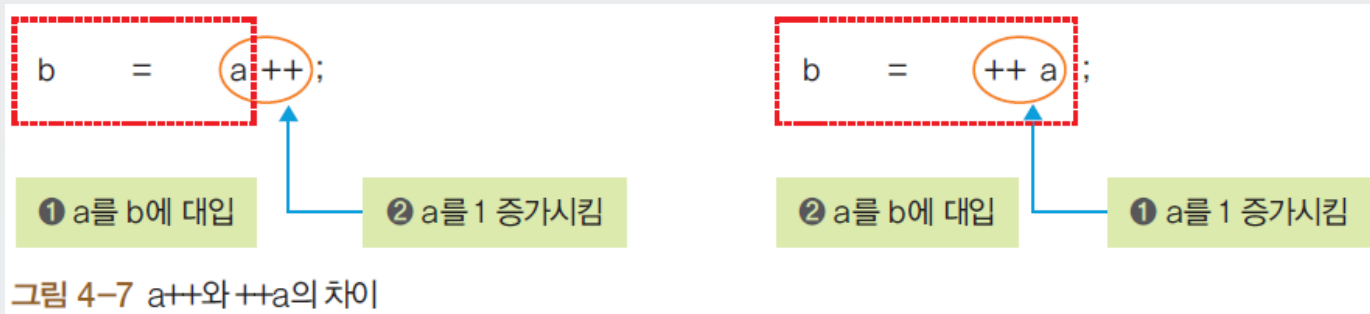


그림 4-6 실행 결과

11. 산술 연산자(11)

- 5행, 8행



- TIP : `++a`를 전치 증가 연산자, `--a`를 전치 감소 연산자, `a++`를 후치 증가 연산자, `a--`를 후치 감소 연산자라고 함

12. 관계 연산자(1)

■ 관계 연산자

- 두 값을 비교하는 관계 연산자의 결과는 항상 참(true)이나 거짓(false)으로 표현

$$a < b = \begin{cases} \text{참 : true} \\ \text{거짓 : false} \end{cases}$$

그림 4-8 관계 연산자의 기본 개념

표 4-3 관계 연산자의 종류

| 관계 연산자 | 의미 | 설명 |
|--------|---------|------------------|
| == | 같다. | 두 값이 동일하면 참이다. |
| != | 같지 않다. | 두 값이 다르면 참이다. |
| > | 크다. | 왼쪽이 크면 참이다. |
| < | 작다. | 왼쪽이 작으면 참이다. |
| >= | 크거나 같다. | 왼쪽이 크거나 같으면 참이다. |
| <= | 작거나 같다. | 왼쪽이 작거나 같으면 참이다. |

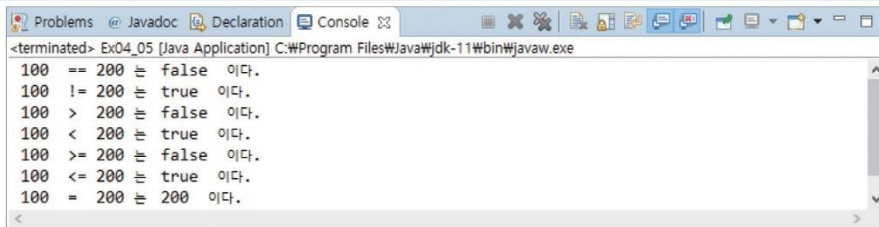
13. 관계 연산자(2)

실습 4-5 관계 연산자 사용 예

```
01 public class Ex04_05 {
02     public static void main(String[] args) {
03         int a = 100, b = 200;
04
05         System.out.printf(" %d == %d 는 %s 이다.\n", a, b, a == b);
06         System.out.printf(" %d != %d 는 %s 이다.\n", a, b, a != b);
07         System.out.printf(" %d > %d 는 %s 이다.\n", a, b, a > b);
08         System.out.printf(" %d < %d 는 %s 이다.\n", a, b, a < b);
09         System.out.printf(" %d >= %d 는 %s 이다.\n", a, b, a >= b);
10         System.out.printf(" %d <= %d 는 %s 이다.\n", a, b, a <= b);
11
12         System.out.printf(" %d = %d 는 %s 이다.\n", a, b, a = b);
13     }
14 }
```

같다, 같지 않다, 크다, 작다, 크거나 같다, 작거나 같다는 관계 연산자를 실행한다.

대입 연산자를 실행한다.

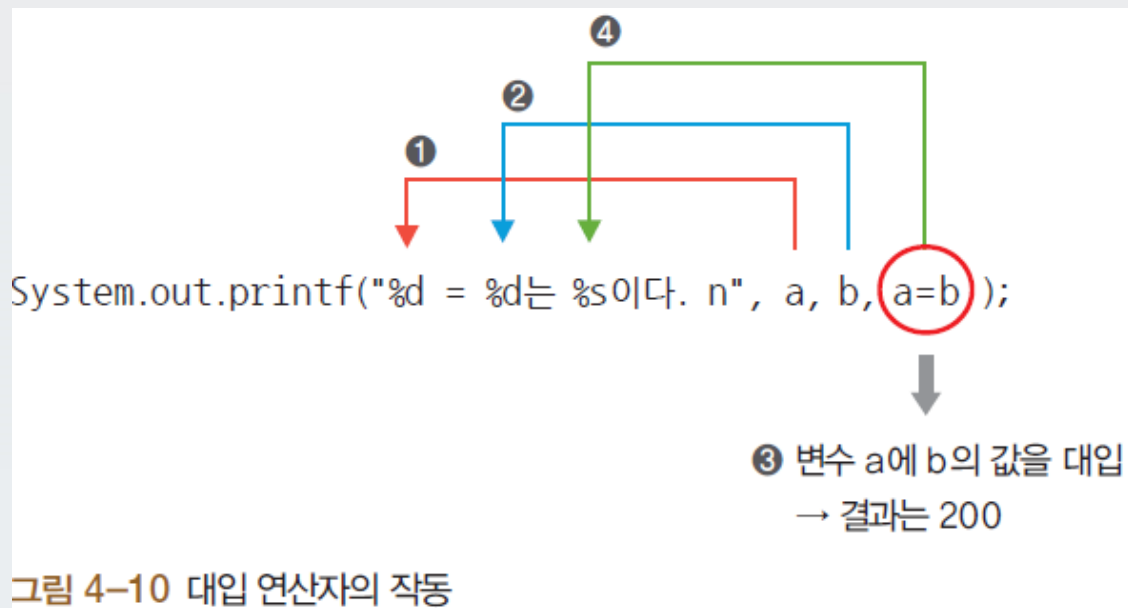


```
<terminated> Ex04_05 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe
100 == 200 는 false 이다.
100 != 200 는 true 이다.
100 > 200 는 false 이다.
100 < 200 는 true 이다.
100 >= 200 는 false 이다.
100 <= 200 는 true 이다.
100 = 200 는 200 이다.
```

그림 4-9 실행 결과

14. 관계 연산자(3)

- 12행



15. 관계 연산자(4)

■ 논리 연산자

- 두 가지 이상의 조건을 표현하는 경우에는 논리 연산자를 사용

표 4-4 논리 연산자의 종류

| 논리 연산자 | 의미 | 설명 | 사용 예 |
|--------|---------------|---------------------|----------------------|
| && | ~이고, 그리고(AND) | 둘 다 참이어야 참이다. | (a>100) && (a<200) |
| | ~이거나, 또는(OR) | 둘 중 하나만 참이어도 참이다. | (a==100) (a==200) |
| ! | ~아니다, 부정(NOT) | 참이면 거짓이고, 거짓이면 참이다. | !(a<100) |

표 4-5 true, false 표

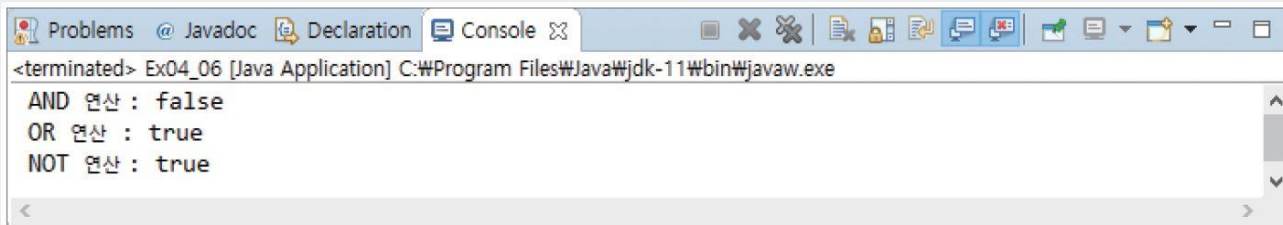
| A | B | A && B | A B | !A |
|-------|-------|--------|--------|-------|
| true | true | true | true | false |
| true | false | false | true | false |
| false | true | false | true | true |
| false | false | false | false | true |

16. 관계 연산자(5)

실습 4-6 논리 연산자 사용 예 1

```
01 public class Ex04_06 {  
02     public static void main(String[] args) {  
03         int a = 99;  
04  
05         System.out.printf(" AND 연산 : %s \n", (a >= 100) && (a <= 200));  
06         System.out.printf(" OR 연산 : %s \n", (a >= 100) || (a <= 200));  
07         System.out.printf(" NOT 연산 : %s \n", !(a == 100));  
08     }  
09 }
```

각각 AND, OR,
NOT 연산이다.



```
<terminated> Ex04_06 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe  
AND 연산 : false  
OR 연산 : true  
NOT 연산 : true
```

그림 4-11 실행 결과

17. 관계 연산자(6)

실습 4-7 논리 연산자 사용 예 2

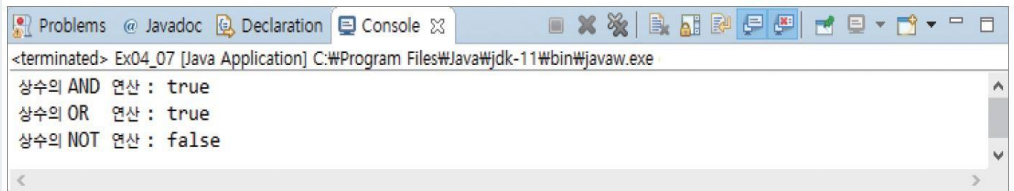
```
01 public class Ex04_07 {
02     public static void main(String[] args) {
03         int num1 = 100, num2 = -200;
04
05         boolean a = (num1 != 0);
06         boolean b = ☐ ;
07
08         System.out.printf(" 상수의 AND 연산 : %s %n", a && b);
09         System.out.printf(" 상수의 OR 연산 : %s %n", ☐ );
10         System.out.printf(" 상수의 NOT 연산 : %s %n", !a);
11     }
12 }
```

num1 값이 0이 아닌지를 확인하여 논리형 변수 a에 true/false를 저장한다.

num2 값이 0이 아닌지를 확인하여 논리형 변수 b에 true/false를 저장한다.

각각 AND, OR, NOT 연산이다.

q || e 2 (0=i 2wnu) 1 ㅁ ㅁ ㅁ



```
<terminated> Ex04_07 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe
상수의 AND 연산 : true
상수의 OR 연산 : true
상수의 NOT 연산 : false
```

그림 4-12 실행 결과

18. 비트 연산자(1)

■ 비트 연산자

- 정수나 문자 등을 2진수로 변환한 다음 각 자리의 비트끼리 연산을 수행

표 4-6 비트 연산자의 종류

| 비트 연산자 | 설명 | 의미 |
|--------|---------------------|-------------------------|
| & | 비트 논리곱 연산자(AND) | 둘 다 1이면 1이다. |
| | 비트 논리합 연산자(OR) | 둘 중 하나만 1이면 1이다. |
| ^ | 비트 배타적 논리합 연산자(XOR) | 둘이 같으면 0이고, 둘이 다르면 1이다. |
| ~ | 비트 부정 연산자 | 1은 0으로 바꾸고, 0은 1로 바꾼다. |
| << | 왼쪽 시프트 연산자 | 비트를 왼쪽으로 시프트한다. |
| >> | 오른쪽 시프트 연산자 | 비트를 오른쪽으로 시프트한다. |

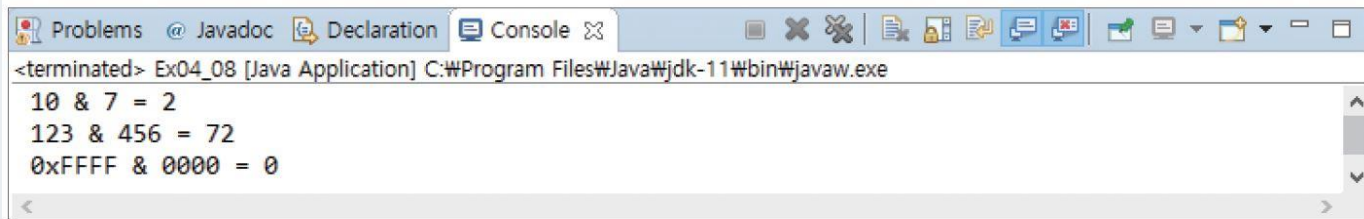
19. 비트 연산자(2)

20. 비트 연산자(3)

실습 4-8 비트 논리곱 연산자 사용 예

```
01 public class Ex04_08 {  
02     public static void main(String[] args) {  
03         System.out.printf(" 10 & 7 = %d \n", 10 & 7);  
04         System.out.printf(" 123 & 456 = %d \n", 123 & 456);  
05         System.out.printf(" 0xFFFF & 0000 = %d \n ", 0xFFFF & 0000);  
06     }  
07 }
```

----- 10과 7의 비트 논리곱을 수행한다.
----- 123과 456의 비트 논리곱을 수행한다.
----- 16진수 FFFF와 0의 비트 논리곱을 수행한다.



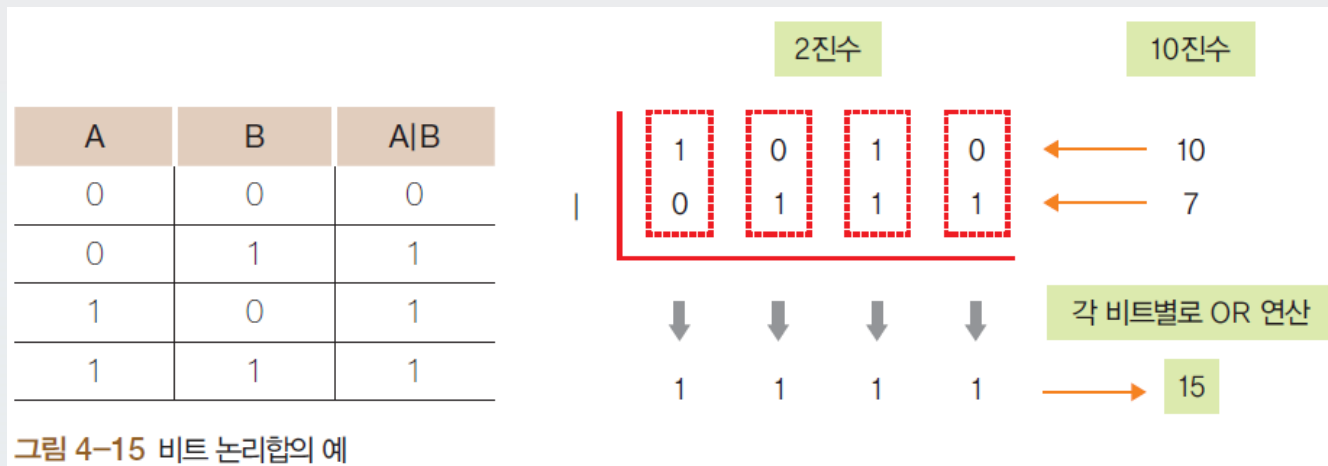
The screenshot shows a Java IDE window with the 'Console' tab selected. The title bar indicates the application is 'Ex04_08 [Java Application]' running at 'C:\Program Files\Java\jdk-11\bin\javaw.exe'. The console output displays the results of the bitwise AND operations performed in the code: '10 & 7 = 2', '123 & 456 = 72', and '0xFFFF & 0000 = 0'.

```
<terminated> Ex04_08 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe  
10 & 7 = 2  
123 & 456 = 72  
0xFFFF & 0000 = 0
```

그림 4-14 실행 결과

21. 비트 연산자(4)

- 비트 논리합 연산자 |
 - '10 | 7'

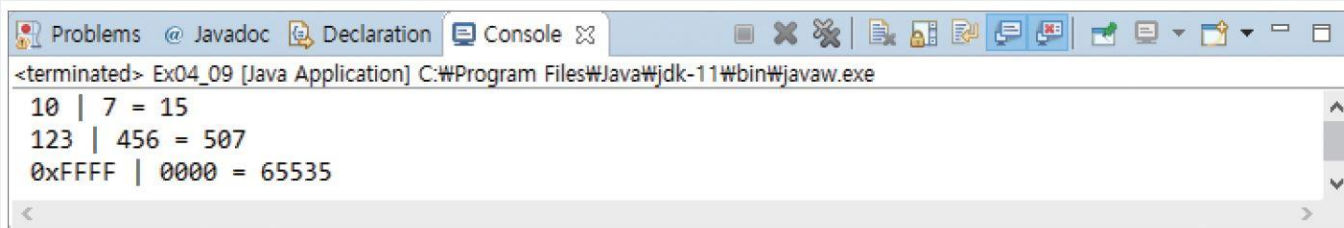


- 비트 논리합의 결과는 1111_2 이고, 이는 10진수로 15

22. 비트 연산자(5)

실습 4-9 비트 논리합 연산자 사용 예

```
01 public class Ex04_09 {
02     public static void main(String[] args) {
03         System.out.printf(" 10 | 7 = %d \n", 10 | 7); ----- 10과 7의 비트 논리합을 수행한다.
04         System.out.printf(" 123 | 456 = %d \n", 123 | 456); ----- 123과 456의 비트 논리합을
                                                                수행한다.
05         System.out.printf(" 0xFFFF | 0000 = %d \n ", 0xFFFF | 0000); -----
06     }                                                                16진수 FFFF와 0의 비트 논리합을 수행한다.
07 }
```



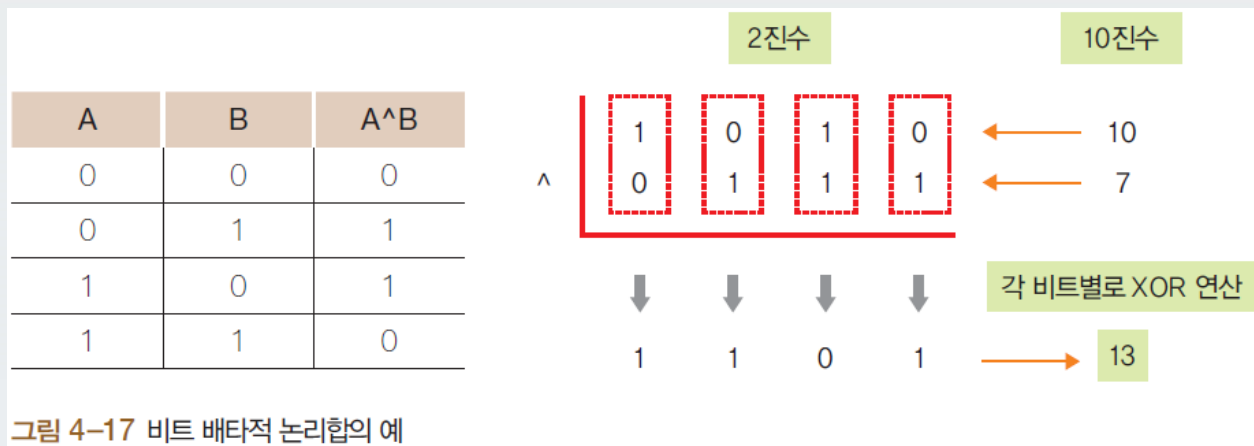
```
<terminated> Ex04_09 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe
10 | 7 = 15
123 | 456 = 507
0xFFFF | 0000 = 65535
```

그림 4-16 실행 결과

23. 비트 연산자(6)

■ 비트 배타적 논리합 연산자 ^

- 두 값이 다르면 1, 같으면 0이 됨. 즉 $1 \wedge 1$ 이나 $0 \wedge 0$ 이면 결과가 거짓(0)이고, $1 \wedge 0$ 이나 $0 \wedge 1$ 이면 결과가 참(1)
- $10 \wedge 7$



- 비트 배타적 논리합 결과는 1101_2 이고, 이는 10진수로 13

24. 비트 연산자(7)

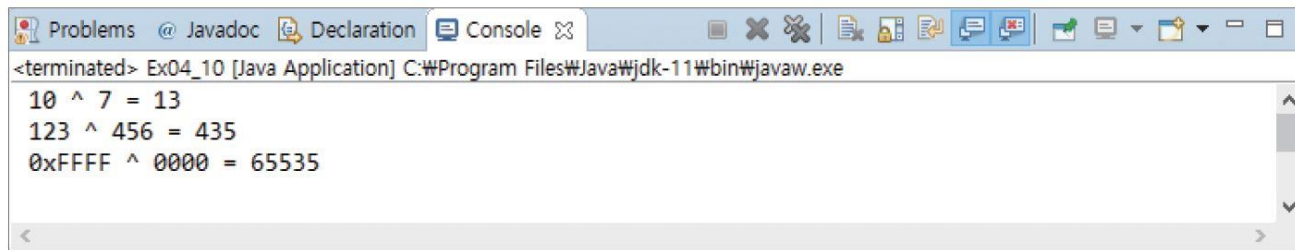
실습 4-10 비트 배타적 논리합 연산자 사용 예

```
01 public class Ex04_10 {
02     public static void main(String[] args) {
03         System.out.printf(" 10 ^ 7 = %d \n", 10 ^ 7);
04         System.out.printf(" 123 ^ 456 = %d \n", 123 ^ 456);
05         System.out.printf(" 0xFFFF ^ 0000 = %d \n ", 0xFFFF ^ 0000);
06     }
07 }
```

----- 10과 7의 비트 배타적 논리합을 수행한다.

----- 123과 456의 비트 배타적 논리합을 수행한다.

----- 16진수 FFFF와 0의 비트 배타적 논리합을 수행한다.



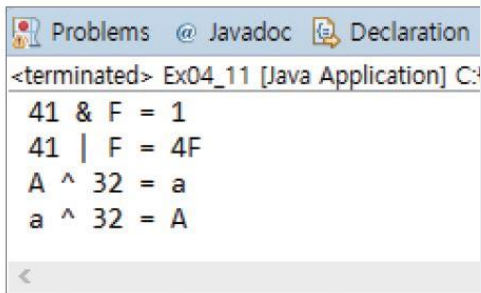
```
<terminated> Ex04_10 [Java Application] C:\Program Files\Java\jdk-11\bin\javaw.exe
10 ^ 7 = 13
123 ^ 456 = 435
0xFFFF ^ 0000 = 65535
```

그림 4-18 실행 결과

25. 비트 연산자(8)

실습 4-11 비트 연산에 마스크를 사용한 예

```
01 public class Ex04_11 {
02     public static void main(String[] args) {
03         byte a = 'A', b;
04         byte mask = 0x0F; ----- 마스크 값(0000 11112)을 설정한다.
05
06         System.out.printf(" %X & %X = %X \n", a, mask, a & mask);
07         System.out.printf(" %X | %X = %X \n", a, mask, a | mask); ----- 'A'와 0x0F의 논리곱
08                                     및 논리합을 수행한다.
09
09         mask = 'a' - 'A'; ----- 'a'와 'A'의 차이는 32이다.
10
11         b = (byte) (  ); ----- 'A'와 마스크(32)의 배타적 논리합을 수행한다.
12         System.out.printf(" %c ^ %d = %c \n", a, mask, b);
13         a = (byte) (  ); ----- 'a'와 마스크(32)의 배타적 논리합을 수행한다.
14         System.out.printf(" %c ^ %d = %c \n", b, mask, a);
15     }
16 }
```



```
<terminated> Ex04_11 [Java Application] C:\
41 & F = 1
41 | F = 4F
A ^ 32 = a
a ^ 32 = A
```

그림 4-19 실행 결과

mask: 0x0F a: 0x41 b: 0x01

26. 비트 연산자(9)

■ 4행

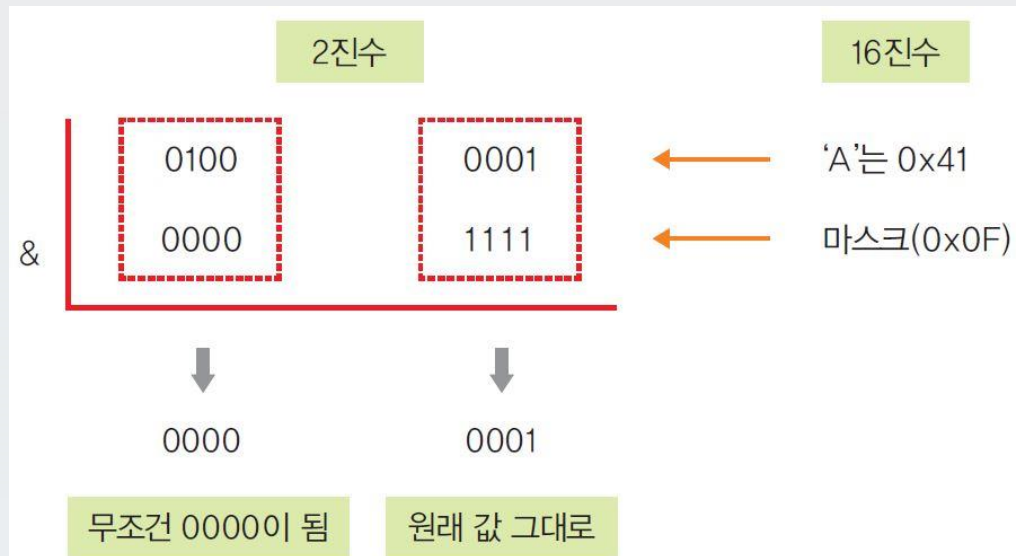
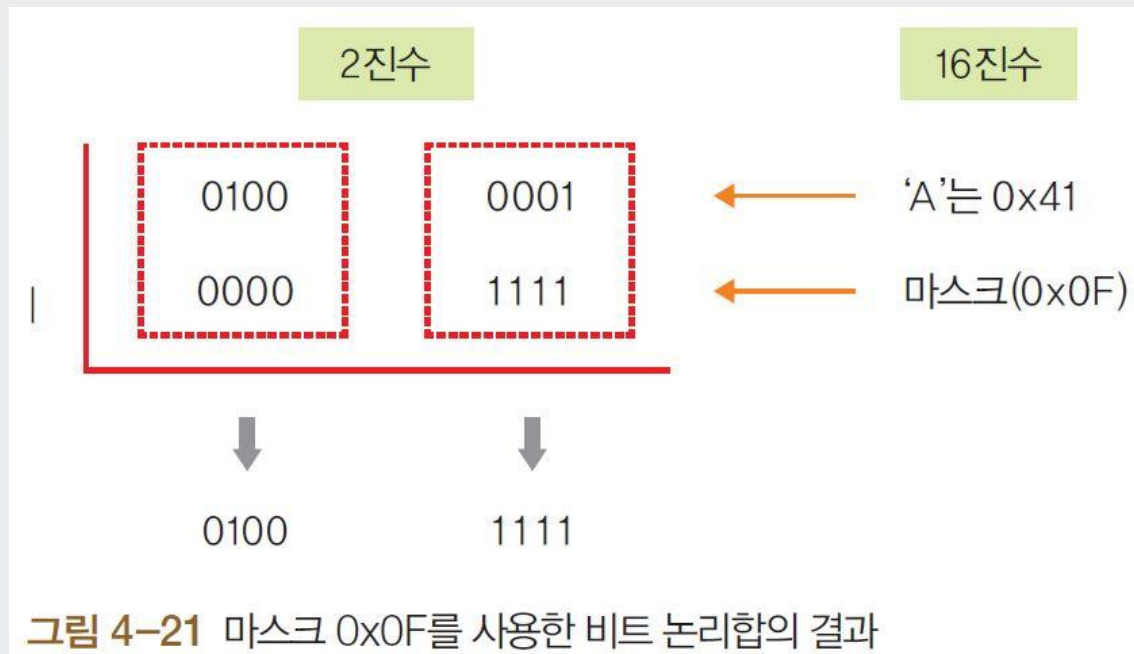


그림 4-20 마스크 0x0F를 사용한 비트 논리곱의 결과

27. 비트 연산자(10)

- 7행



28. 비트 연산자(11)

■ 비트 부정 연산자 ~

- 각 비트를 반대로 만드는 연산자. 즉 0은 1로 바꾸고, 1은 0으로 바꿈. 이렇게 반전된 값을 1의 보수라 하며, 그 값에 1을 더한 값을 2의 보수라 함. 비트 부정 연산자는 해당 값의 음수(-) 값을 찾고자 할 때 사용.

실습 4-12 비트 부정 연산자 사용 예

```
01 public class Ex04_12 {  
02     public static void main(String[] args) {  
03         int a = 12345;  
04  
05         System.out.printf(" %d %n", ~a + 1); ----- a 값의 2의 보수를 구한다.  
06     }  
07 }
```

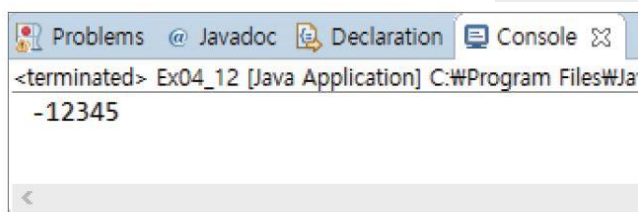


그림 4-22 실행 결과

29. 비트 연산자(12)

- 왼쪽 시프트 연산자 <<
 - 나열된 비트를 왼쪽으로 시프트(shift)
 - 26을 왼쪽으로 두 칸 시프트 연산



30. 비트 연산자(13)

실습 4-13 왼쪽 시프트 연산자 사용 예

```
01 public class Ex04_13 {  
02     public static void main(String[] args) {  
03         int a = 10;  
04         System.out.printf("%d 를 왼쪽 1회 시프트하면 %d 이다.\n", a, a<<1);  
05         System.out.printf("%d 를 왼쪽 2회 시프트하면 %d 이다.\n", a, a<<2);  
06         System.out.printf("%d 를 왼쪽 3회 시프트하면 %d 이다.\n", a, a<<3);  
07     }  
08 }
```

왼쪽
시프트한
결과를
출력한다.

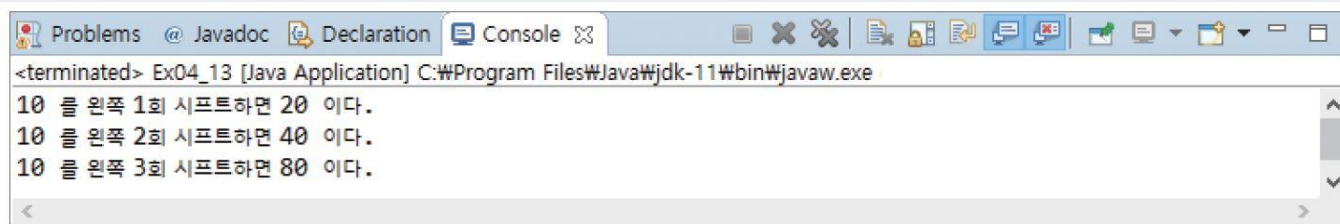
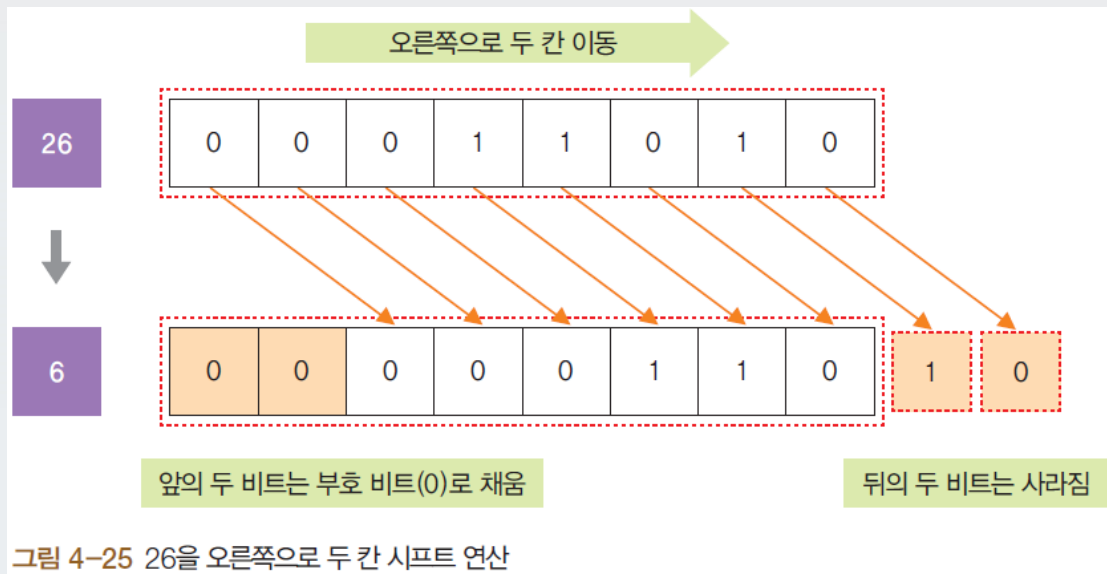


그림 4-24 실행 결과

31. 비트 연산자(14)

- 오른쪽 시프트 연산자 >>
 - 나열된 비트를 오른쪽으로 시프트하는 연산자
 - 26을 오른쪽으로 두 칸 시프트 연산



32. 비트 연산자(15)

실습 4-14 오른쪽 시프트 연산자 사용 예

```
01 public class Ex04_14 {  
02     public static void main(String[] args) {  
03         int a = 10;  
04         System.out.printf("%d 를 오른쪽 1회 시프트하면 %d 이다.\n", a, a >> 1);  
05         System.out.printf("%d 를 오른쪽 2회 시프트하면 %d 이다.\n", a, a >> 2);  
06         System.out.printf("%d 를 오른쪽 3회 시프트하면 %d 이다.\n", a, a >> 3);  
07         System.out.printf("%d 를 오른쪽 4회 시프트하면 %d 이다.\n", a, a >> 4);  
08     }  
09 }
```

오른쪽
시프트한
결과를
출력한다.

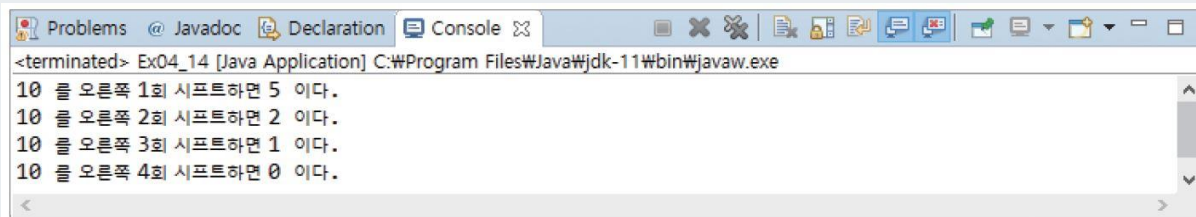


그림 4-26 실행 결과

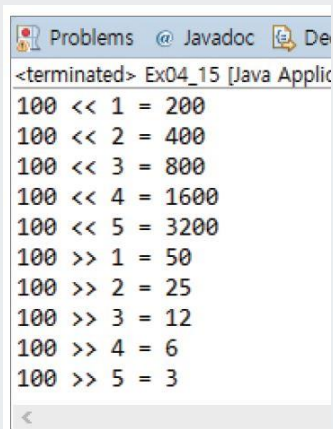
33. 비트 연산자(16)

실습 4-15 왼쪽, 오른쪽 시프트 연산자 사용 예

```
01 public class Ex04_15 {
02     public static void main(String[] args) {
03         int a = 100, result;
04         int i;
05
06         for (i = 1; i <= 5; i++) {
07             result = 
08             System.out.printf("%d << %d = %d\\n", a, i, result);
09         }
10
11         for (i = 1; i <= 5; i++) {
12             result = 
13             System.out.printf("%d >> %d = %d\\n", a, i, result);
14         }
15     }
16 }
```

왼쪽 시프트 연산을 다섯 번
반복해서 출력한다.

오른쪽 시프트 연산을 다섯
번 반복해서 출력한다.



```
Problems @ Javadoc De
<terminated> Ex04_15 [Java Applic
100 << 1 = 200
100 << 2 = 400
100 << 3 = 800
100 << 4 = 1600
100 << 5 = 3200
100 >> 1 = 50
100 >> 2 = 25
100 >> 3 = 12
100 >> 4 = 6
100 >> 5 = 3
```

그림 4-27 실행 결과

100 << 1 = 200 100 >> 1 = 50

34. 연산자 우선 순위

표 4-7 연산자 우선순위

| 우선순위 | 연산자 | 설명 | 순위가 같을 경우 진행 방향 |
|------|--------------------------------------|-------------------------|-----------------|
| 1 | () [] . | 1차 연산자 | → |
| 2 | + - ++ -- ~ ! (type) | 단항 연산자[변수(또는 상수) 앞에 붙음] | ← |
| 3 | * / % | 산술 연산자 | → |
| 4 | + - | 산술 연산자 | → |
| 5 | << >> >>> | 비트 시프트 연산자 | → |
| 6 | < <= > >= instanceof | 비교 연산자 | → |
| 7 | = != | 동등 연산자 | → |
| 8 | & | 비트 연산자 | → |
| 9 | ^ | 비트 연산자 | → |
| 10 | | 비트 연산자 | → |
| 11 | && | 논리 연산자 | → |
| 12 | | 논리 연산자 | → |
| 13 | ?: | 조건 삼항 연산자 | → |
| 14 | = += -= *= /= %= %= ^= = <<= >>= | 대입 연산자 | ← |



다음 시간

프로젝트 관리



송실사이버대학교

송실사이버대학교의 강의콘텐츠는
저작권법에 의하여 보호를 받는바, 무단
전재, 배포, 전송, 대여 등을 금합니다.

*사용서체 : 나눔글꼴