2 학기
JAVA Class

'이것이 자바다'

2025.09

Cover Description

made by Lewis

복습

- 두 수를 입력 받아 더하는 프로그램을 코딩 하시오.

```
<terminated> PlusCalc [Java Ap
1. input Val : 22
2. input Val : 33
iVal_1 + iVal_2 = 55
```

01 연산자란?

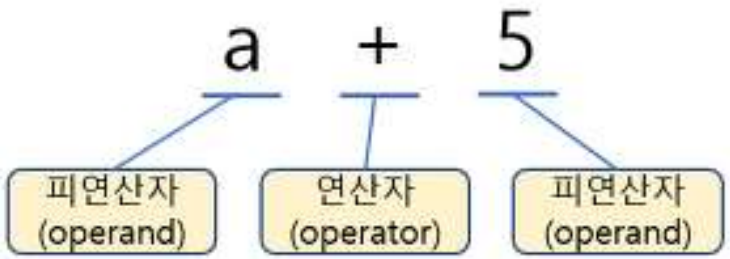
02 연산자 종류

03 연산 방향과 우선 순위

컴퓨터가 이해하는 코드는
누구라도 작성할 수 있습니다.
뛰어난 프로그래머는 사람이
이해하는 코드를 작성합니다.

연산자란 ?

- 주어진 식을 계산하여 결과를 얻어내는 과정을 연산이라고 하며,
  연산을 수행하는 기호를 연산자라고 한다.



연산자(operator) : 연산을 수행하는 기호(+,-,*,/ 등)
피연산자(operand) : 연산자의 작업 대상(변수, 상수, 수식)

연산자 종류

1. 부호/증감 연산자
2. 산술 연산자
3. 비교 연산자
4. 논리 연산자
5. 비트 논리/이동 연산자
6. 대입 연산자
7. 삼항(조건) 연산자

부호/증감 연산자

- * 부호 : + , -

  ➔ 변수의 부호 설정(음수, 양수)

SignOperatorExample.java

```
1  package ch03.sec01;
2
3  public class SignOperatorExample {
4      public static void main(String[] args) {
5          int x = -100;
6          x = -x;
7          System.out.println("x: " + x);
8
9          byte b = 100;
10         int y = -b;
11         System.out.println("y: " + y);
12     }
13 }
14
```

Console | Problems | Debug Shell

```
<terminated> SignOperatorExample [Java Application] C:₩Users₩Lewis
x: 100
y: -100
```

p.80 SignOperatorExample.java

부호/증감 연산자

```
int iFirst = 23;
int iSecond = -23;
```

다음중 false 인 것은?

1. +iFirst == iFirst;
2. -iFirst == iSecond;
3. -iSecond == iFirst;
4. + iSecond == iFirst;

부호/증감 연산자

- 증감 연산자 : ++ , --

  ➔ 변수의 값을 1 증가, 1 감소

```
a = 1;
b = 0 ;
b = ++a;        // b == ?
c = a++ + b;    // c == ?
                // a == ??
```

* a = a + 1;

  ++a; ➔ 차이점은?

  a++;

p.82 IncreaseDecreaseOperatorExample.java

IncreaseDecreaseOperatorExample.java

```
1  package ch03.sec01;
2
3  public class IncreaseDecreaseOperatorExample {
4      public static void main(String[] args) {
5          int x = 10;
6          int y = 10;
7          int z;
8
9          x++;
10         ++x;
11         System.out.println("x=" + x);
12
13         System.out.println("-----------------------");
14         y--;
15         --y;
16         System.out.println("y=" + y);
17
18         System.out.println("-----------------------");
19         z = x++;
20         System.out.println("z=" + z);
21         System.out.println("x=" + x);
22
23         System.out.println("-----------------------");
24         z = ++x;
25         System.out.println("z=" + z);
26         System.out.println("x=" + x);
27
28         System.out.println("-----------------------");
29         z = ++x + y++;
30         System.out.println("z=" + z);
31         System.out.println("x=" + x);
32         System.out.println("y=" + y);
33     }
34 }
```

Console × Problems Debug Shell

<terminated> IncreaseDecreaseOperatorExample [Java Application] C:₩Users₩Lewis₩

```
x=12
-----------------------
y=8
-----------------------
z=12
x=13
-----------------------
z=14
x=14
-----------------------
z=23
x=15
y=9
```

부호/증감 연산자

```
int iFirst = 3;
int iSecond = -5;
int iThird = 2;
int iResult = ++iFirst + iSecond-- * iThird++;
iResult += 3;
```

iFirst ??

iSecond ??

iThird ??

iResult ??

산술 연산자

- 더하기, 빼기, 곱하기, 나누기, 나머지

  ➔ + , -, *, /, %

- 연산 중 타입변환은 피연산자 중 가장 큰 타입으로 변환 된다.
- int iFstVal = 14;

  int iRslt = 12 + iFstVal / 2;

  iRslt = ??

✓ 사과의 수가 123개이고 하나의 바구니에는 10개의 사과를 담을 수 있다면 몇 개의 바구니가 필요한지 코딩 하시오.

ArithmeticOperatorExample.java

```
1  package ch03.sec02;
2
3  public class ArithmeticOperatorExample {
4      public static void main(String[] args) {
5          byte v1 = 10;
6          byte v2 = 4;
7          int v3 = 5;
8          long v4 = 10L;
9
10         int result1 = v1 + v2;          //모든 피연산자는 int 타입으로 자동 변환 후 연산
11         System.out.println("result1: " + result1);
12
13         long result2 = v1 + v2 - v4;    //모든 피연산자는 long 타입으로 자동 변환 후 연산
14         System.out.println("result2: " + result2);
15
16         double result3 = (double) v1 / v2;  //double 타입으로 강제 변환 후 연산
17         System.out.println("result3: " + result3);
18
19         int result4 = v1 % v2;
20         System.out.println("result4: " + result4);
21     }
22 }
```

Console | Problems | Debug Shell

<terminated> ArithmeticOperatorExample [Java Application] C:₩Users₩Lewis₩.p2₩pool₩plugins₩org.eclipse.jus

```
result1: 14
result2: 4
result3: 2.5
result4: 2
```

p.84 ArithmeticOperatorExample.java

오버 / 언더플로우

- 오버플로우 : 해당 변수 타입의 최대 허용치를 벗어나는 것
- 언더플로우 : 해당 변수 타입의 최소 허용치를 벗어나는 것

* 변수 최대/최소 값을 벗어나면 발생되는 값 확인 -> 최대, 최소값으로

* 최대 / 최소 값 확인 방법:
  Byte.MIN_VALUE,
  Byte.MAX_VALUE
  Integer.MIN_VALUE
  Integer. MAX _VALUE
  Float. MIN_VALUE, MAX _VALUE
  Double. MIN_VALUE, MAX _VALUE

OverflowUnderflowExample.java

```java
package ch03.sec03;

public class OverflowUnderflowExample {
    public static void main(String[] args) {
        byte var1 = 125;
        for(int i=0; i<5; i++) {        //{ }를 5번 반복 실행
            var1++;                     //++ 연산은 var1의 값을 1 증가시킨다.
            System.out.println("var1: " + var1);
        }

        System.out.println("----------------------");

        byte var2 = -125;
        for(int i=0; i<5; i++) {            //{ }를 5번 반복 실행
            var2--;                         //-- 연산은 var2의 값을 1 감소시킨다.
            System.out.println("var2: " + var2);
        }
    }
}
```

Console | Problems | Debug Shell

```
<terminated> OverflowUnderflowExample [Java Application] C:₩Users₩Lewis₩.p2₩pool₩plugins₩org.ecli
var1: 126
var1: 127
var1: -128
var1: -127
var1: -126
----------------------
var2: -126
var2: -127
var2: -128
var2: 127
var2: 126
```

p.84 OverflowUnderflowExample.java

NaN, Infinity

- 나눗셈, 나머지 연산에서 좌측 피연산자가 정수이고 우측 피연산자가 0인경우

  5 / 0.0 -> Infinity (무한대, 0으로 나누기)

  5 % 0.0 -> Nan (Not a Number)

- Double.isInfinite(변수);
- Double.isNaN(변수);

InfinityAndNaNCheckExample.java

```
1  package ch03.sec05;
2
3  public class InfinityAndNaNCheckExample {
4      public static void main(String[] args) {
5          int x = 5;
6          double y = 0.0;
7          double z = x / y;
8          //double z = x % y;
9
10         //잘못된 코드
11         System.out.println(z + 2);
12
13         //알맞은 코드
14         if(Double.isInfinite(z) || Double.isNaN(z)) {
15             System.out.println("값 산출 불가");
16         } else {
17             System.out.println(z + 2);
18         }
19     }
20 }
21
```

Console | Problems | Debug Shell

<terminated> InfinityAndNaNCheckExample [Java Application] C:₩Users₩Lew

```
Infinity
값 산출 불가
```

p.89 InfinityAndNaNCheckExample.java

비교 연산자

- ==, !=, >, >=, <. <=

- 피연산자의 타입이 다를 경우 연산 전 타입을 일치시킨다.

  예외

  0.1f == 0.1 => false ➔ float != double

  0.1f = (float)0.1 → true ➔ 소수점 정밀도 차이

- 문자열 비교 : 원본.equals(비교);

p.92 CompareOperatorExample.java

CompareOperatorExample.java

```
1  package ch03.sec06;
2
3  public class CompareOperatorExample {
4      public static void main(String[] args) {
5          int num1 = 10;
6          int num2 = 10;
7          boolean result1 = (num1 == num2);
8          boolean result2 = (num1 != num2);
9          boolean result3 = (num1 <= num2);
10         System.out.println("result1: " + result1);
11         System.out.println("result2: " + result2);
12         System.out.println("result3: " + result3);
13
14         char char1 = 'A';
15         char char2 = 'B';
16         boolean result4 = (char1 < char2);    //65 < 66
17         System.out.println("result4: " + result4);
18
19         int num3 = 1;
20         double num4 = 1.0;
21         boolean result5 = (num3 == num4);
22         System.out.println("result5: " + result5);
23
24         float num5 = 0.1f;
25         double num6 = 0.1;
26         boolean result6 = (num5 == num6);
27         boolean result7 = (num5 == (float)num6);
28         System.out.println("result6: " + result6);
29         System.out.println("result7: " + result7);
30
31         String str1 = "자바";
32         String str2 = "Java";
33         boolean result8 = (str1.equals(str2));
34         boolean result9 = (! str1.equals(str2));
35         System.out.println("result8: " + result8);
36         System.out.println("result9: " + result9);
37     }
38 }
```

Console | Problems | Debug Shell

<terminated> CompareOperatorExample [Java Application] C:₩Users₩Lewis₩

```
result1: true
result2: false
result3: true
result4: true
result5: true
result6: false
result7: true
result8: false
result9: true
```

논리 연산자

- && : 피연산자 모두 True 인 경우에 True 리턴
- || : 피연산자 중 하나만 True 인 경우에 True 리턴
- ^ : 피연산자가 하나는 True, 다른 하나는 False 인 경우에 True 리턴
- ! : 피 연산자의 값 반대 리턴
- &, | : 논리연산자로 사용 시 &&, || 과 같은 결과를 리턴 하지만 양쪽 모두 무조건 평가

- 조건문, 반복문에서 주로 사용

p.92 CompareOperatorExample.java

LogicalOperatorExample.java

```
1  package ch03.sec07;
2
3  public class LogicalOperatorExample {
4      public static void main(String[] args) {
5          int charCode = 'A';
6          //int charCode = 'a';
7          //int charCode = '5';
8
9          if( (65<=charCode) & (charCode<=90) ) {
10             System.out.println("대문자 이군요");
11         }
12
13         if( (97<=charCode) && (charCode<=122) ) {
14             System.out.println("소문자 이군요");
15         }
16
17         if( (48<=charCode) && (charCode<=57) ) {
18             System.out.println("0~9 숫자 이군요");
19         }
20
21         //-----------------------------------------------
22
23         int value = 6;
24         //int value = 7;
25
26         if( (value%2==0) | (value%3==0) ) {
27             System.out.println("2 또는 3의 배수 이군요");
28         }
29
30         boolean result = (value%2==0) || (value%3==0);
31         if( !result ) {
32             System.out.println("2 또는 3의 배수가 아니군요.");
33         }
34     }
35 }
36
```

Console | Problems | Debug Shell

```
<terminated> LogicalOperatorExample [Java Application] C:₩Users₩Lewis₩.p2₩po
대문자 이군요
2 또는 3의 배수 이군요
```

비트 논리 연산자

- 비트 연산을 위해 2진 수로 변환 후 연산

A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

A	~A
0	1
1	0

```
int num = 13;
String binary = Integer.toBinaryString(num);
System.out.println(binary); // "1101"
```

비트 논리 연산자

136의 2진수 (32비트 int) 변환
00000000 00000000 00000000 10001000
8비트만 저장 → 하위 8비트만 남김
10001000
8비트를 byte (부호 있는 정수)로 해석
맨 앞자리(1)는 부호 비트
→ 음수라서 2의 보수(비트반전 0→1)
10001000 (원래 값)
01110111 (비트 반전)
+ 1

01111000 (십진수 120) ➔ -120

(8비트 기준)
128 64 32 16 8 4 2 1
ex: 255 의 2진수 값 -> 11111111

BitLogicExample.java

```
1  package ch03.sec08;
2
3  public class BitLogicExample {
4      public static void main(String[] args) {
5          System.out.println("45 & 25 = " + (45 & 25));
6          System.out.println("45 | 25 = " + (45 | 25));
7          System.out.println("45 ^ 25 = " + (45 ^ 25));
8          System.out.println("~45 = " + (~45));
9          System.out.println("------------------------------");
10
11         byte receiveData = -120;
12
13         //방법1: 비트 논리곱 연산으로 Unsigned 정수 얻기
14         int unsignedInt1 =  receiveData & 255;
15         System.out.println(unsignedInt1);
16
17         //방법2: 자바 API를 이용해서 Unsigned 정수 얻기
18         int unsignedInt2 = Byte.toUnsignedInt(receiveData);
19         System.out.println(unsignedInt2);
20
21
22         int test = 136;
23         byte btest = (byte) test;
24         System.out.println(btest);
25     }
26 }
27
```

Console | Problems | Debug Shell

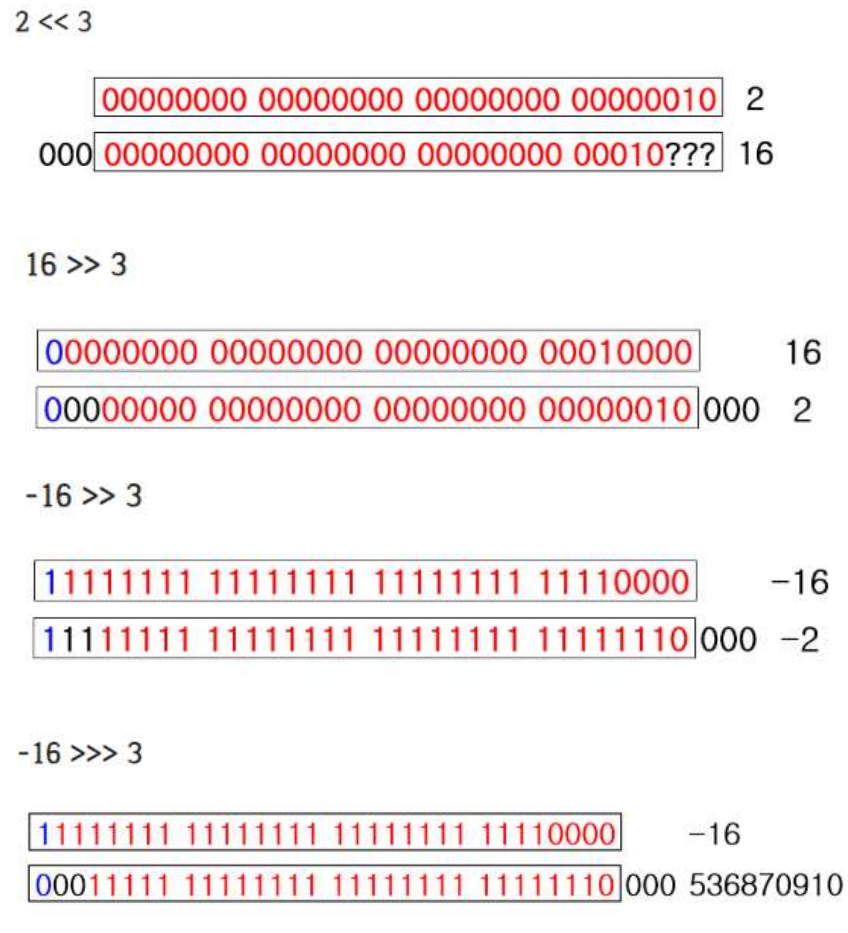
<terminated> BitLogicExample [Java Application] C:₩Users₩Lewis₩.p2₩pool₩plugins₩org.ec

```
45 & 25 = 9
45 | 25 = 61
45 ^ 25 = 52
~45 = -46
------------------------------
136
136
-120
```

비트 이동 연산자

- 비트 연산을 위해 32bit로 변환 후 연산
- 피 연산자가 음수라면 빈공간은 1로
  양수라면 0 으로

- 이동 연산자 : <<, >>, >>>
  a << b : 정수 a의 각 비트를 왼쪽으로 b 만큼 이동(빈칸은 0) $a * 2^b$
  a >> b : 정수 a의 각 비트를 오른쪽으로 b 만큼 이동 (빈칸은 최상위 부호비트로) $a / 2^b$
  a >>> b :정수 a의 각 비트를 오른쪽으로 b 만큼 이동 자바에만 있는 연산, >>과 동일 하게 변환 후 빈칸은 0으로



비트 이동 연산자

- Math.pow(2, 3) => 2 * 2 * 2 = 8

  double 값을 반환 하기에 int 로 형변환
- 같은 결과를 나타내는 다양한 방법

BitShiftExample1.java

```
1  package ch03.sec09;
2
3  public class BitShiftExample1 {
4      public static void main(String[] args) {
5          int num1 = 1;
6          int result1 = num1 << 3;
7          int result2 = num1 * (int) Math.pow(2, 3);
8          System.out.println("result1: " + result1);
9          System.out.println("result2: " + result2);
10
11         int num2 = -8;
12         int result3 = num2 >> 3;
13         int result4 = num2 / (int) Math.pow(2, 3);
14         System.out.println("result3: " + result3);
15         System.out.println("result4: " + result4);
16     }
17 }
```

Console | Problems | Debug Shell

<terminated> BitShiftExample1 [Java Application] C:₩Users₩Lewis₩.p2₩pool₩

```
result1: 8
result2: 8
result3: -1
result4: -1
```

p.101 BitShiftExample1.java

비트 이동 연산자

- >>> 연산 실습

BitShiftExample2.java

```
1  package ch03.sec09;
2
3  public class BitShiftExample2 {
4      public static void main(String[] args) {
5          int value = 772; //[00000000] [00000000] [00000011] [00000100]
6
7          //우측 3byte(24bit) 이동하고 끝 1바이트만 읽음: [00000000]
8          byte byte1 = (byte) (value >>> 24);
9          int int1 = byte1 & 255;
10         System.out.println("첫 번째 바이트 부호없는 값: " +  int1);
11
12         //우측 2byte(16bit) 이동하고 끝 1바이트만 읽음: [00000000]
13         byte byte2 = (byte) (value >>> 16);
14         int int2 = Byte.toUnsignedInt(byte2);
15         System.out.println("두 번째 바이트 부호없는 값: " +  int2);
16
17         //우측 1byte(8bit) 이동하고 끝 1바이트만 읽음: [00000011]
18         byte byte3 = (byte) (value >>> 8);
19         int int3 = byte3 & 255;
20         System.out.println("세 번째 바이트 부호없는 값: " +  int3);
21
22         //끝 1바이트만 읽음: [00000100]
23         byte byte4 = (byte) value;
24         int int4 = Byte.toUnsignedInt(byte4);
25         System.out.println("네 번째 바이트 부호없는 값: " +  int4);
26     }
27 }
28
```

Console | Problems | Debug Shell

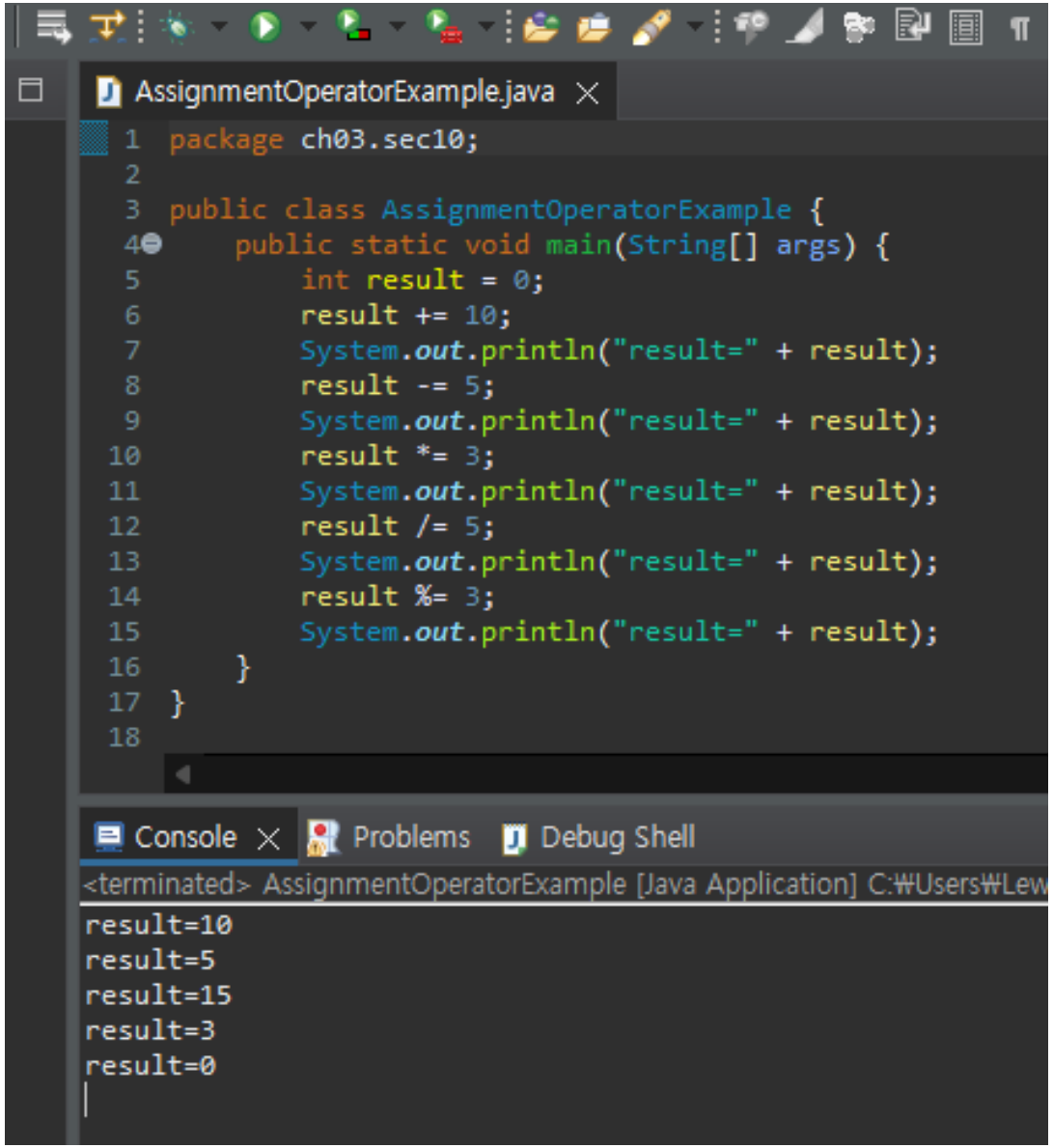
```
<terminated> BitShiftExample2 [Java Application] C:₩Users₩Lewis₩.p2₩pool₩plugins₩org.eclipse.just
첫 번째 바이트 부호없는 값: 0
두 번째 바이트 부호없는 값: 0
세 번째 바이트 부호없는 값: 3
네 번째 바이트 부호없는 값: 4
```

p.103 BitShiftExample2.java

대입 연산자

- = : 우측 피연산자의 값을 좌측 피연산자인 변수에 대입
- =, +=, -=, *=, /=, %, &=, !=, ^=, <<=, >>=, >>>=

- int iAge = 10;
- 10 = iAge;



p.105 AssignmentOperatorExample.java

삼항(조건) 연산자

- 조건식 ? 반환값1 : 반환값2;

ConditionalOperationExample.java

```
1  package ch03.sec11;
2
3  public class ConditionalOperationExample {
4      public static void main(String[] args) {
5          int score = 85;
6          char grade = (score > 90) ? 'A' : ( (score > 80) ? 'B' : 'C' );
7          System.out.println(score + "점은 " + grade + "등급입니다.");
8      }
9  }
10
11
```

Console | Problems | Debug Shell

<terminated> ConditionalOperationExample [Java Application] C:₩Users₩Lewis₩.p2₩pool₩plugins₩org.

85점은 B등급입니다.

p.106 ConditionalOperationExample.java

int num = 10;
System.out.println(??);
➔ num의 값에 따라 '양수', '음수', '0'을 출력

➔ num > 0 ? "양수" : (num < 0 ? "음수" : "0")

연산 방향과 우선 순위

1. 단항, 이항, 삼항 연산자 순.
2. 산술, 비교, 논리, 대입 연산자 순
3. 단항, 부호, 대입 연산자를 제외한 모든 연산의 방향은 왼쪽에서 오른쪽
4. 먼저 처리할 연산을 괄호 ()로 묶는다