변수와 타입

Contents

3절. 타입 변환

- 자동 타입 변환(Promotion)
- 강제 타입 변환(Casting)

3절. 타입 변환

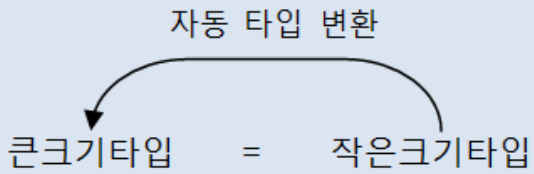
❖ 타입 변환

- 데이터 타입을 다른 타입으로 변환하는 것
  - byte ↔ int, int ↔ double
- 종류
  - 자동(묵시적) 타입 변환: Promotion
  - 강제(명시적) 타입 변환: Casting

3절. 타입 변환

❖ 자동 타입 변환

- 프로그램 실행 도중 작은 타입은 큰 타입으로 자동 타입 변환 가능

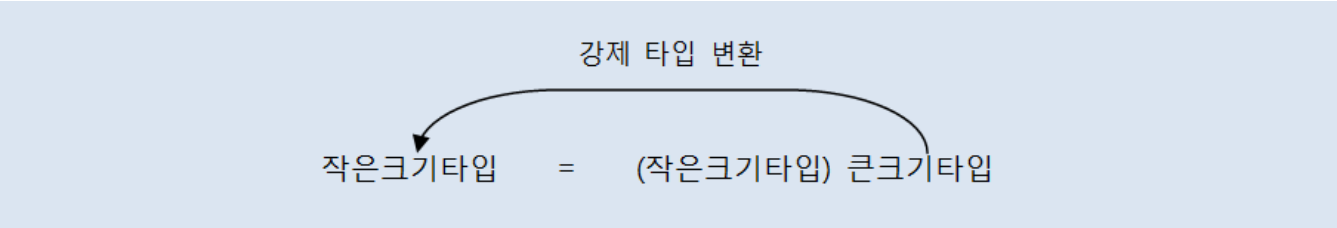



```
byte(1) < short(2) < int(4) < long(8) < float(4) < double(8)
```

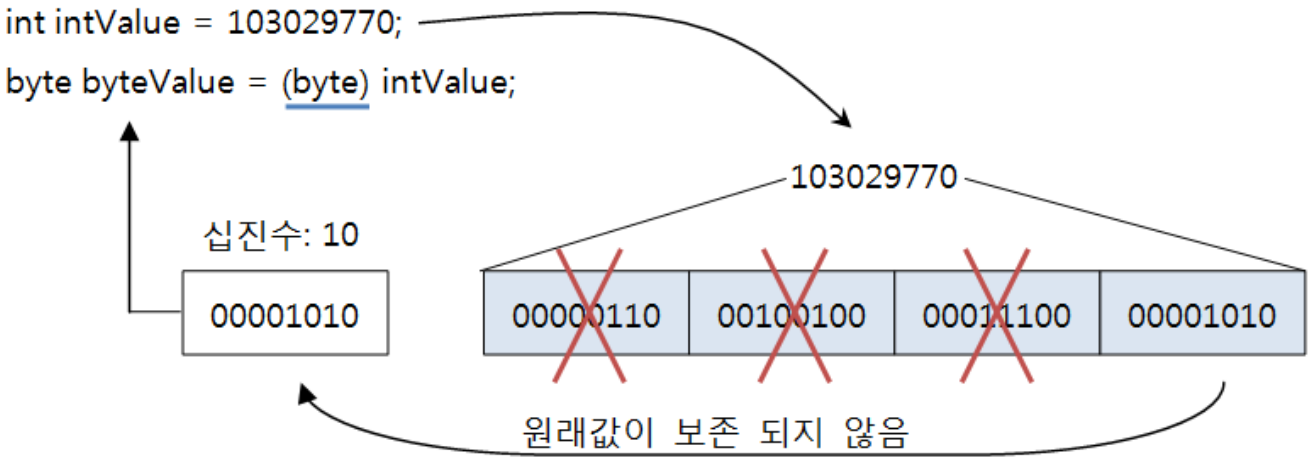
3절. 타입 변환

❖ 강제 타입 변환

- 큰 타입을 작은 타입 단위로 쪼개기
- 끝의 한 부분만 작은 타입으로 강제적 변환



- Ex) int 를 byte에 담기



3절. 타입 변환

❖ 연산식에서 자동 타입 변환

- 연산은 같은 타입의 피연산자(operand)간에만 수행
  - 서로 다른 타입의 피연산자는 같은 타입으로 변환
  - 두 피연산자 중 크기가 큰 타입으로 자동 변환

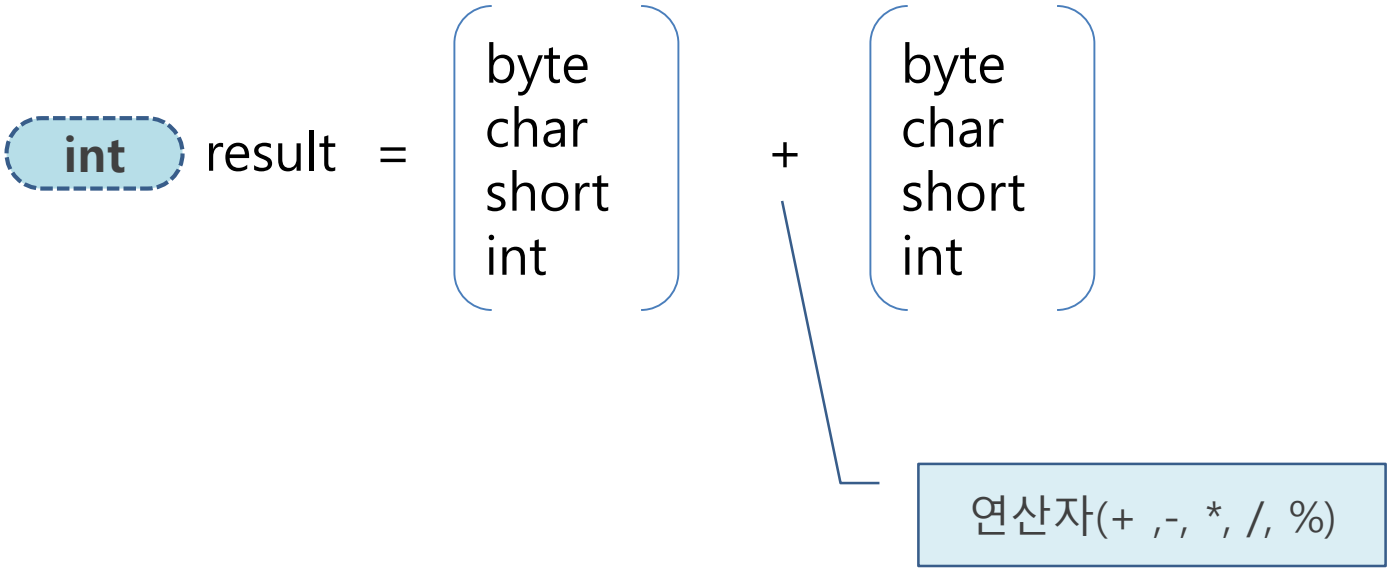
```
int intValue = 10;
double doubleValue = 5.5;

                double 타입으로 자동 변환
double result = intValue + doubleValue;       //result 에 15.5 가 저장
```

  - Ex) int type으로 계산 결과를 얻고 싶다면?
    - Double type 변수를 먼저 int로 변환 후 계산

3절. 타입 변환

❖ 연산식에서 자동 타입 변환

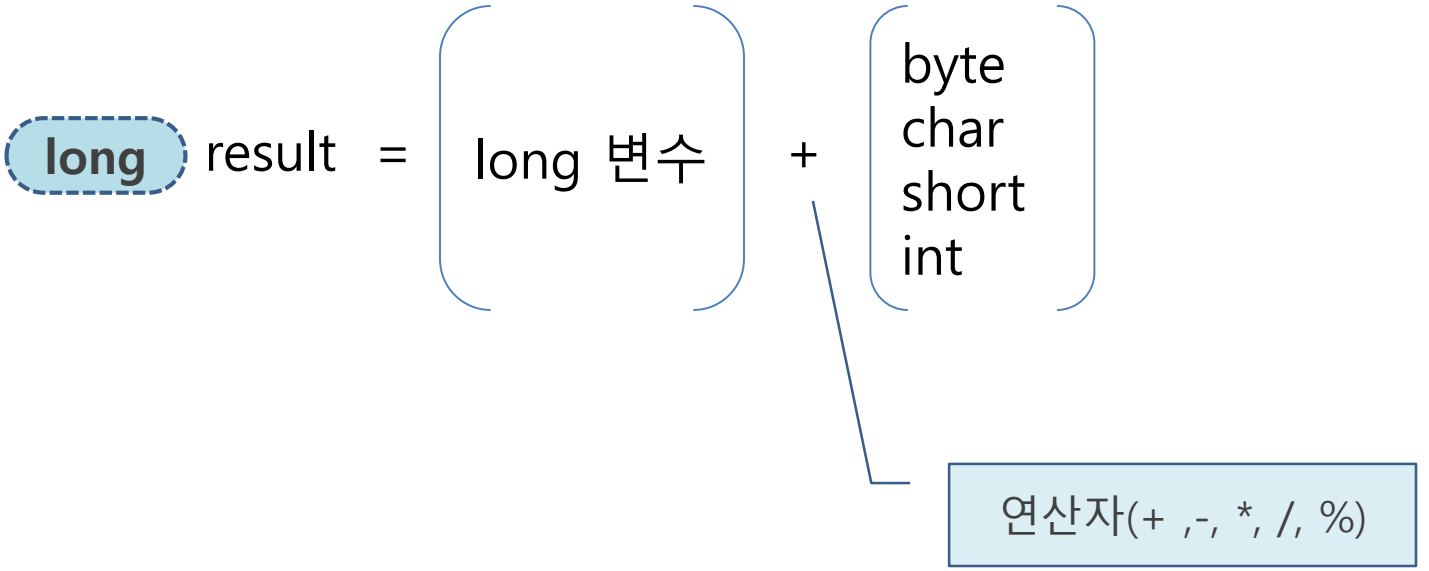
- 자바는 정수 연산일 경우 int 타입을 기본으로 한다
  - 그 이유는 피연산자를 4byte 단위로 저장하기 때문이다
  - 크기가 4byte 보다 작은 타입(byte, char, short)은 4byte인 int로 변환된 후에 연산이 수행된다



3절. 타입 변환

❖ 연산식에서 자동 타입 변환

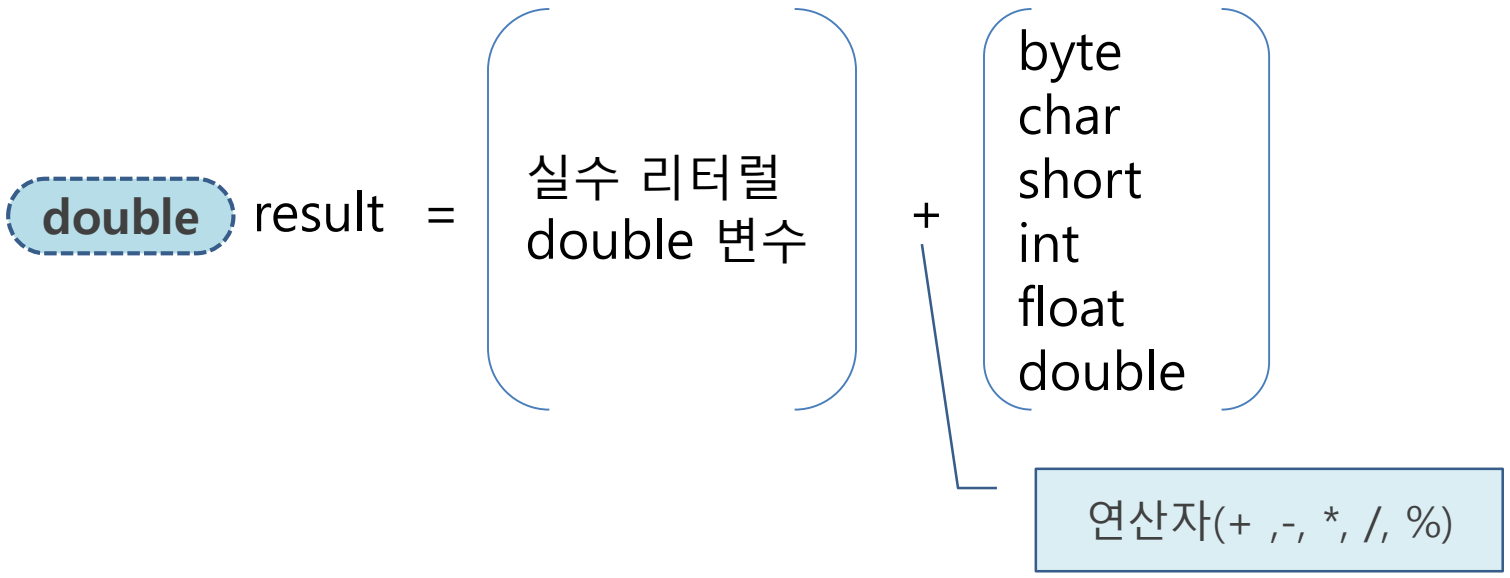
- 피연산자 중 하나가 long 타입이라면 다른 피연산자도 long 타입으로 자동 타입 변환되고 연산의 결과는 long 타입이 된다



3절. 타입 변환

❖ 연산식에서 자동 타입 변환

- 피연산자 중에 실수 리터럴이나 double 타입이 있다면 다른 피연산자도 double 타입으로 자동 타입 변환되고 결과는 double 타입으로 저장된다



실습

PromotionType.java

```java
package week3;

public class PromotionType {
    public static void main(String[] args) {
        byte bValue = 25;
        int  iValue = bValue;
        System.out.println(iValue);

        char cValue = '가';
        iValue      = cValue;
        System.out.println(iValue);

        iValue      = 375;
        long lValue = iValue;
        System.out.println(lValue);

        double dValue = iValue;
        System.out.println(dValue);
    }
}
```

실습

PromotionType.java

```
1  package week3;
2
3  public class PromotionType {
4      public static void main(Stri
5          byte bValue = 25;
6          int  iValue = bValue;
7          System.out.println(iValue);
8
9          char cValue = '가';
10         iValue      = cValue;
11         System.out.println(iValue);
12
13         iValue      = 375;
14         long lValue = iValue;
15         System.out.println(lValue);
16
17         double dValue = iValue;
18         System.out.println(dValue);
19     }
20 }
```

Console

<terminated> PromotionType [Java Application] C:₩Program

```
25
44032
375
375.0
```

실습

CastingType.java

```java
package week3;

public class CastingType {
    public static void main(String[] args) {
        int  iData = 65;
        char cData = (char)iData;
        System.out.printf("cData = %c\n", cData);

        long lData = 500;
        iData      = (int)lData;
        System.out.printf("iData = %d\n", iData);

        double dData = 3.14;
        iData        = (int)dData;
        System.out.printf("iData = %d\n", iData);
    }
}
```

실습

CastingType.java

```
1  package week3;
2
3  public class CastingType {
4      public static void main(String[]
5          int  iData = 65;
6          char cData = (char)iData;
7          System.out.printf("cData = %c\n", cData);
8
9          long lData = 500;
10         iData      = (int)lData;
11         System.out.printf("iData = %d\n", iData);
12
13         double dData = 3.14;
14         iData        = (int)dData;
15         System.out.printf("iData = %d\n", iData);
16     }
17 }
```

Console

<terminated> CastingType [Java Application] C

```
cData = A
iData = 500
iData = 3
```

실습

```
16
17          int  iValue = 128;
18          byte bValue = (byte)iValue;
19          System.out.printf("bValue = %d\n", bValue);
20
21          if (iValue >= Byte.MIN_VALUE && iValue <= Byte.MAX_VALUE) {
22              bValue = (byte)iValue;
23              System.out.printf("bValue = %d\n", bValue);
24          } else {
25              System.out.printf("Casting 하고자하는 변수의 값을 확인하세요\n");
26              System.out.printf("범위를 벗어납니다");
27          }
28      }
29  }
```

실습

```
16
17          int  iValue = 128;
18          byte bValue = (byte)iValue;
19          System.out.printf("bValue = %
20
21          if (iValue >= Byte.MIN_VALUE
22              bValue = (byte)iValue;
23              System.out.printf("bValue = %d\n", bValue);
24          } else {
25              System.out.printf("Casting 하고자하는 변수의 값을 확인하세요\n");
26              System.out.printf("범위를 벗어납니다");
27          }
28      }
29  }
```

Console

<terminated> CastingType2 [Java Application] C:₩Progr

```
cData = A
iData = 500
iData = 3
bValue = -128
Casting 하고자하는 변수의 값을 확인하세요
범위를 벗어납니다
```

연산자

Contents

❖ 목차

1절. 연산자와 연산식

❖ 연산이란?

- 데이터를 처리하여 결과를 산출하는 것

- 연산자(Operations)
  - 연산에 사용되는 표시나 기호(+, -, *, /, %, =, …)

- 피연산자(Operand): 연산 대상이 되는 데이터(리터럴, 변수)

- 연산식(Expressions)
  - 연산자와 피연산자를 이용하여 연산의 과정을 기술한 것

1절. 연산자와 연산식

❖ 연산자의 종류

연산자 종류	연산자	피연산자 수	산출값 타입	기능 설명
산술	+, -, *, /, %	이항	숫자	사칙연산 및 나머지 계산
부호	+, -	단항	숫자	음수와 양수의 부호
문자열	+	이항	문자열	두 문자열을 연결
대입	=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=	이항	다양	우변의 값을 좌변의 변수에 대입
증감	++, --	단항	숫자	1 만큼 증가/감소
비교	==, !=, >, <, >=, <=, instanceof	이항	boolean	값의 비교
논리	!, &, |, &&, ||	단항 이항	boolean	논리적 NOT, AND, OR 연산
조건	(조건식) ? A : B	삼항	다양	조건식에 따라 A 또는 B 중 하나를 선택
비트	~, &, |, ^	단항 이항	숫자 blooean	비트 NOT, AND, OR, XOR 연산
쉬프트	>>, <<, >>>	이항	숫자	비트를 좌측/우측으로 밀어서 이동

2절. 연산의 방향과 우선 순위

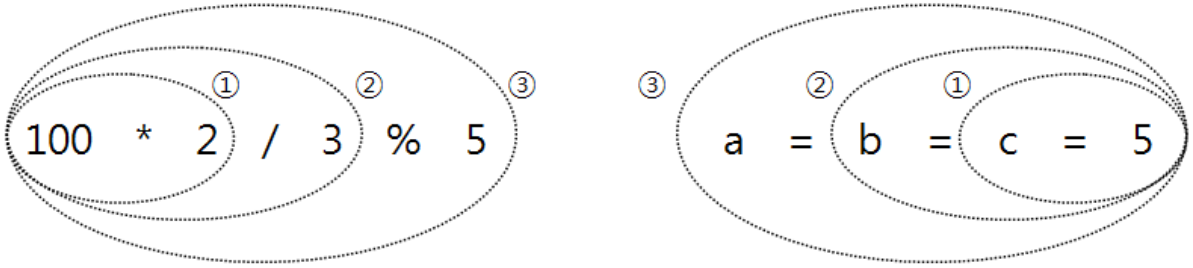
❖ 연산의 방향과 우선 순위

- **연산자의 우선 순위에 따라 연산된다.**

```
x > 0 && y < 0
```

- **동일한 우선 순위의 연산자는 연산의 방향 존재**

*, /, %는 같은 우선 순위를 갖고 있다. 이들 연산자는 연산 방향이 왼쪽에서 오른쪽으로 수행된다. 100 * 2 가 제일 먼저 연산되어 200 이 산출되고, 그 다음 200 / 3 이 연산되어 66 이 산출된다. 그 다음으로 66 % 5 가 연산되어 1 이 나온다.



하지만 단항 연산자(++, --, ~, !), 부호 연산자(+, -), 대입 연산자(=, +=, -=, ...)는 오른쪽에서 왼쪽(←)으로 연산된다.

2절. 연산의 방향과 우선 순위

❖ 연산의 방향과 우선 순위

연산자	연산 방향	우선 순위
증감(++, --), 부호(+, -), 비트(~), 논리(!)	←	높음
산술(*, /, %)	→	
산술(+, -)	→	
쉬프트(<<, >>, >>>)	→	
비교(<, >, <=, >=, instanceof)	→	
비교(==, !=)	→	
논리(&)	→	
논리(^)	→	
논리(|)	→	
논리(&&)	→	
논리(||)	→	
조건(?:)	→	
대입(=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=)	←	낮음

3절. 단항 연산자

❖ 단항연산자란?

- 피연산자가 1개인 연산자

❖ 단항 연산자의 종류

- 부호 연산자: +, -
  - boolean 타입과 char 타입을 제외한 기본 타입에 사용 가능
  - 부호 연산자의 산출 타입은 int
- 증감 연산자: ++, --
  - 변수의 값을 1증가 시키거나 (++) 1 감소 (--) 시키는 연산자
  - 증감 연산자가 변수 뒤에 있으면 다른 연산자 먼저 처리 후 증감 연산자 처리

3절. 단항 연산자

❖ 단항 연산자의 종류

▪ 논리 부정 연산자: !

- Boolean type 에만 사용가능

연산식		설명
!	피연산자	피연산자가 true 이면 false 값을 산출 피연산자가 false 이면 true 값을 산출

▪ 비트 반전 연산자: ~

- byte, short, int, long 타입만 피연산자가 될 수 있다.
- 비트값을 반전(0 -> 1, 1 -> 0)시킨다.

연산식		설명
~	10 (0 0 … 0 1 0 1 0)	산출결과: -11 (1 1 … 1 0 1 0 1)

단항 연산자 실습(OneOperand1.java)

```java
public class OneOperand1 {
    public static void main(String[] args) {
        int    iValue1 = +100;
        int    iValue2 = -100;
        double dValue1 = +3.14;
        double dValue2 = -10.5;

        int result1 = +iValue1;
        int result2 = -iValue1;
        System.out.printf("result1 = %d\n", result1);
        System.out.printf("result2 = %d\n", result2);

        short sValue  = 100;
        //int보다 크기가 작은 경우 부호 연산자의 결과는 int 타입이 된다
        //short sResult = -sValue;       //에러발생
        int sResult = -sValue;
        System.out.println("sResult = " + sResult);

        byte  bValue  = -100;
        int   bResult = -bValue;
        System.out.println("bResult = " + bResult);

        long  lValue  = 100;
        long  lResult = -lValue;
        System.out.println("lResult = " + lResult);

        double dResult = -dValue1;
        System.out.println("dResult = " + dResult);
    }
}
```

단항 연산자 실습(OneOperand1.java)

```
3  public class OneOperand1 {
4      public static void main(String[] args) {
5          int    iValue1 = +100;
6          int    iValue2 = -100;
7          double dValue1 = +3.14;
8          double dValue2 = -10.5;
9
10         int result1 = +iValue1;
11         int result2 = -iValue1;
12         System.out.printf("result1 = %d\n",
13         System.out.printf("result2 = %d\n",
14
15         short sValue  = 100;
16         //int보다 크기가 작은 경우 부호 연산자의 결과는 int
17         //short sResult = -sValue;       //에러발생
18         int sResult = -sValue;
19         System.out.println("sResult = " + sResult);
20
21         byte  bValue  = -100;
22         int   bResult = -bValue;
23         System.out.println("bResult = " + bResult);
24
25         long  lValue  = 100;
26         long  lResult = -lValue;
27         System.out.println("lResult = " + lResult);
28
29         double dResult = -dValue1;
30         System.out.println("dResult = " + dResult);
31     }
32 }
```

Console

<terminated> OneOperand1 [Java Application] C:₩Program Files₩Java₩jdk1.8

```
result1 = 100
result2 = -100
sResult = -100
bResult = 100
lResult = -100
dResult = -3.14
```

단항 연산자 실습 (OneOperand2.java)

```
3  public class OneOperand2 {
4      public static void main(String[] args) {
5          int x = 10;
6          int y = 10;
7          int z;
8
9          //++연산자는 피연산자의 기존 값에 1을 더해서 그 결과를 다시 피연산자에 저장한다
10         //++ 기호가 피연산자의 뒤에 있으면 문장을 수행한 뒤에 1을 더한다
11         System.out.println("x++ = " + x++);
12         System.out.println("------------------");
13
14         //++ 기호가 피연산자의 앞에 있으면 문장을 수행하기 전에 1을 먼저 더한다
15         System.out.println("++x = " + ++x);
16         System.out.println("------------------");
17
18         z = x++;
19         System.out.println("z = " + z + ", x = " + x);
20         System.out.println("------------------");
21
22         //--연산자는 피연산자의 기존 값에 1을 빼서 그 결과를 다시 피연산자에 저장한다
23         z = ++x + y--;
24         System.out.println("z = " + z);
25         System.out.println("x = " + x);
26         System.out.println("y = " + y);
27     }
28 }
```

단항 연산자 실습 (OneOperand2.java)

```java
public class OneOperand2 {
    public static void main(String[] args) {
        int x = 10;
        int y = 10;
        int z;

        //++연산자는 피연산자의 기존 값에 1을 더해서 그 결과를 다시 피
        //++ 기호가 피연산자의 뒤에 있으면 문장을 수행한 뒤에 1을 더한
        System.out.println("x++ = " + x++);
        System.out.println("------------------");

        //++ 기호가 피연산자의 앞에 있으면 문장을 수행하기 전에 1을 먼
        System.out.println("++x = " + ++x);
        System.out.println("------------------");

        z = x++;
        System.out.println("z = " + z + ", x = " + x);
        System.out.println("------------------");

        //--연산자는 피연산자의 기존 값에 1을 빼서 그 결과를 다시 피연산자에 저장한다
        z = ++x + y--;
        System.out.println("z = " + z);
        System.out.println("x = " + x);
        System.out.println("y = " + y);
    }
}
```

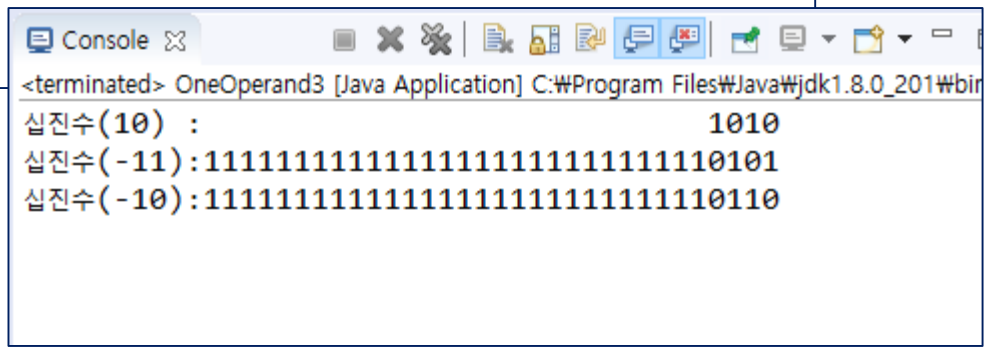
```
Console
<terminated> OneOperand2 [Java Application] C:₩Program Files₩Java₩
x++ = 10
------------------
++x = 12
------------------
z = 12, x = 13
------------------
z = 24
x = 14
y = 9
```

단항 연산자 실습 (OneOperand3.java)

```
2
3  public class OneOperand3 {
4      public static void main(String[] args) {
5          int var1 = 10;
6          int var2 = ~var1;
7          int var3 = ~var1 + 1;
8
9          System.out.printf("십진수(%d) :%32s\n",var1, Integer.toBinaryString(var1));
10         System.out.printf("십진수(%d):%32s\n", var2, Integer.toBinaryString(var2));
11         System.out.printf("십진수(%d):%32s\n", var3, Integer.toBinaryString(var3));
12     }
13 }
14
```

단항 연산자 실습 (OneOperand3.java)

```
2
3  public class OneOperand3 {
4      public static void main(String[] args) {
5          int var1 = 10;
6          int var2 = ~var1;
7          int var3 = ~var1 + 1;
8
9          System.out.printf("십진수(%d) :%32s\n",var1, Integer.toBinaryString(var1));
10         System.out.printf("십진수(%d):%32s\n", var2, Integer.toBinaryString(var2));
11         System.out.printf("십진수(%d):%32s\n", var3, Integer.toBinaryString(var3));
12     }
13 }
14
```



단항 연산자 실습 (OneOperand3.java)

```
2
3  public class OneOperand3 {
4      public static void main(String[] args) {
5          int var1 = 10;
6          int var2 = ~var1;
7          int var3 = ~var1 + 1;
8
9          System.out.printf("십진수(%d) :%32s\n",var1, Integer.toBinaryString(var1));
10         System.out.printf("십진수(%d):%32s\n", var2, Integer.toBinaryString(var2));
11         System.out.printf("십진수(%d):%32s\n", var3, Integer.toBinaryString(var3));
12     }
13 }
14
```

정수 타입의 변수값에 비트 반전 연산자(~)를 적용한 후 1을 더해주면 원래 정수값의 부호가 반대인 값을 구할 수 있다

```
Console
<terminated> OneOperand3 [Java Application] C:₩Program Files₩Java₩jdk1.8.0_201₩bin
십진수(10) :                                                1010
십진수(-11):11111111111111111111111111110101
십진수(-10):11111111111111111111111111110110
```

4절. 이항 연산자

❖ 이항 연산자란?

- 피연산자가 2개인 연산자

- 종류
  - 산술 연산자: +, -, *, /, %
  - 문자열 연결 연산자: +
  - 대입 연산자: =, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=
  - 비교 연산자: <, <=, >, >=, ==, !=
  - 논리 연산자: &&, ||, &, |, ^, !
  - 비트 논리 연산자: &, |, ^
  - 비트 이동 연산자: <<, >>, >>>

4절. 이항 연산자

❖ 산술 연산자

- boolean 타입을 제외한 모든 기본 타입에 사용 가능
- 결과값 산출할 때 Overflow 주의
- 정확한 계산은 정수를 사용
- NaN과 Infinity 연산은 주의할 것

연산식			설명
피연산자	+	피연산자	덧셈 연산
피연산자	-	피연산자	뺄셈 연산
피연산자	*	피연산자	곱셈 연산
피연산자	/	피연산자	좌측 피연산자를 우측 피연산자로 나눗셈 연산
피연산자	%	피연산자	좌측 피연산자를 우측 피연산자로 나눈 나머지를 구하는 연산

```
int result = num % 3;
```

0, 1, 2 중의 한 값

num 을 3 으로 나눈 나머지

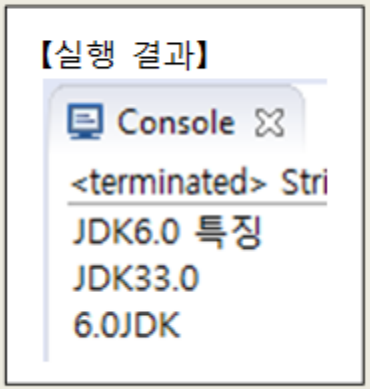
4절. 이항 연산자

❖ 문자열 연산자

- 피연산자 중 문자열이 있으면 문자열로 결합

```
String str1 = "JDK" + 6.0;
String str2 = str1 + " 특징";
System.out.println(str2);


String str3 = "JDK" + 3 + 3.0;
String str4 = 3 + 3.0 + "JDK";
System.out.println(str3);
System.out.println(str4);
```

【실행 결과】



4절. 이항 연산자

❖ 비교 연산자(==, !=, <, >, <=, >=)

- 대소(<, <=, >, >=) 또는 동등(==, !=) 비교해 boolean 타입인 true/false 산출

구분	연산식			설명
동등 비교	피연산자	==	피연산자	두 피 연산자의 값이 같은지를 검사
	피연산자	!=	피연산자	두 피 연산자의 값이 다른지를 검사
크기 비교	피연산자	>	피연산자	피 연산자 1 이 큰지를 검사
	피연산자	>=	피연산자	피 연산자 1 이 크거나 같은지를 검사
	피연산자	<	피연산자	피 연산자 1 이 작은지를 검사
	피연산자	<=	피연산자	피 연산자 1 이 작거나 같은지를 검사

- 동등 비교 연산자는 모든 타입에 사용
- 크기 비교 연산자는 boolean 타입 제외한 모든 기본 타입에 사용
- 흐름 제어문인 조건문(if), 반복문(for, while)에서 주로 이용
  - 실행 흐름을 제어할 때 사용

4절. 이항 연산자

❖ 논리 연산자 (&&, ||, &, |, ^, !)

- 논리곱(&&), 논리합(||), 배타적 논리합(^) ,논리 부정(!) 연산 수행
- 피연산자는 boolean 타입만 사용 가능

구분	연산식			결과	설명
AND (논리곱)	true	&& 또는 &	true	true	피 연사자 모두가 true 일 경우에만 연산 결과는 true
	true		false	false	
	false		true	false	
	false		false	false	
OR (논리합)	true	|| 또는 |	true	true	피 연산자 중 하나만 true 이면 연산 결과는 true
	true		false	true	
	false		true	ture	
	false		false	false	
XOR (배타적 논리합)	true	^	true	false	피 연산자가 하나는 ture 이고 다른 하나가 false 일 경우에만 연산 결과는 true
	true		false	true	
	false		true	ture	
	false		false	false	
NOT (논리부정)		!	true	false	피 연산자의 논리값을 바꿈
			false	true	

논리 연산자 실습 (LogicalOperator.java)

```
2
3  public class LogicalOperator {
4      public static void main(String[] args) {
5          int charCode1 = 'A';
6          if (charCode1 >= 65 & charCode1 <= 90) {
7              System.out.println((char)charCode1 + "는 알파벳 대문자입니다");
8          }
9
10         int charCode2 = 'b';
11         if (charCode2 >= 97 && charCode2 <= 122) {
12             System.out.println((char)charCode2 + "는 알파벳 소문자입니다");
13         }
14
15         int charCode3 = '9';
16         if ( !(charCode3 < 48) && !(charCode3 > 57) ) {
17             System.out.println((char)charCode3 + "는 0~9 사이의 숫자입니다");
18         }
19
20         int iValue = 4;
21         if ( (iValue%2 == 0) | (iValue%3 == 0) ) {
22             System.out.println(iValue + "는 2 또는 3의 배수입니다");
23         }
24     }
25 }
```

논리 연산자 실습 (LogicalOperator.java)

```
2
3  public class LogicalOperator {
4      public static void main(String[] args) {
5          int charCode1 = 'A';
6          if (charCode1 >= 65 & charCode1 <= 90) {
7              System.out.println((char)charCode1 +
8          }
9
10         int charCode2 = 'b';
11         if (charCode2 >= 97 && charCode2 <= 122)
12             System.out.println((char)charCode2 + "는 알파벳 소문자입니다");
13         }
14
15         int charCode3 = '9';
16         if ( !(charCode3 < 48) && !(charCode3 > 57) ) {
17             System.out.println((char)charCode3 + "는 0~9 사이의 숫자입니다");
18         }
19
20         int iValue = 4;
21         if ( (iValue%2 == 0) | (iValue%3 == 0) ) {
22             System.out.println(iValue + "는 2 또는 3의 배수입니다");
23         }
24     }
25 }
```

```
Console
<terminated> LogicalOperator [Java Application] C:₩Program Files₩Java₩jd
A는 알파벳 대문자입니다
b는 알파벳 소문자입니다
9는 0~9 사이의 숫자입니다
4는 2 또는 3의 배수입니다
```

4절. 이항 연산자

❖ 비트 연산자(&, |, ^, ~, <<, >>, >>>)

- 비트(bit) 단위로 연산 하므로 0과 1이 피연산자
  - 0과 1로 표현이 가능한 정수 타입만 비트 연산 가능
  - 실수 타입인 float과 double은 비트 연산 불가
- 종류
  - 비트 논리 연산자(&, |, ^, ~)
  - 비트 이동 연산자(<<, >>, >>>)

4절. 이항 연산자

❖ 비트 논리 연산자(&, |, ^, ~)

- 피 연산자가 boolean타입일 경우 일반 논리 연산자
- 피연산자가 정수 타입일 경우 비트 논리 연산자로 사용
- 비트 연산자는 피연산자를 int타입으로 자동 타입 변환 후 연산 수행

구분	연산식			결과	설명
AND (논리곱)	1	&	1	1	두 비트가 모두가 1 일 경우에만 연산 결과는 1
	1		0	0	
	0		1	0	
	0		0	0	
OR (논리합)	1	|	1	1	두 비트 중 하나만 1 이면 연산 결과는 1
	1		0	1	
	0		1	1	
	0		0	0	
XOR (배타적 논리합)	1	^	1	0	두 비트 중 하나는 1 이고 다른 하나가 0 일 경우 연산 결과는 1
	1		0	1	
	0		1	1	
	0		0	0	
NOT (논리부정)		~	1	0	보수
			0	1	

4절. 이항 연산자

❖ 비트 이동 연산자(<<, >>, >>>)

- 정수 데이터의 비트를 좌측 또는 우측으로 밀어 이동시키는 연산 수행

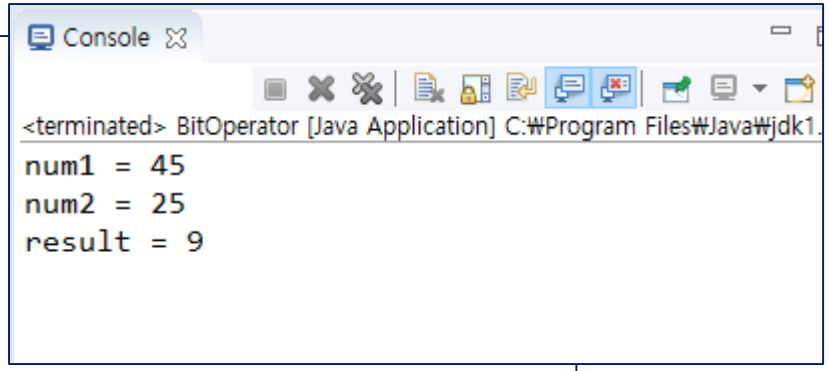
구분	연산식			설명
이동 (쉬프트)	a	<<	b	정수 a 의 각 비트를 b 만큼 왼쪽으로 이동 (빈자리는 0 으로 채워진다.)
	a	>>	b	정수 a 의 각 비트를 b 만큼 오른쪽으로 이동 (빈자리는 정수 a 의 최상위 부호 비트(MSB)와 같은 값으로 채워진다.)
	a	>>>	b	정수 a 의 각 비트를 오른쪽으로 이동 (빈자리는 0 으로 채워진다.)

비트 연산자 실습 (BitOperator.java)

```
 2
 3 public class BitOperator {
 4     public static void main(String[] args) {
 5         byte num1   = 45;
 6         byte num2   = 25;
 7         int  result = num1 & num2;
 8
 9         System.out.println("num1 = " + num1);
10         System.out.println("num2 = " + num2);
11         System.out.println("result = " + result);
12     }
13 }
14
```

비트 연산자 실습 (BitOperator.java)

```
2
3  public class BitOperator {
4      public static void main(String[] args) {
5          byte num1   = 45;
6          byte num2   = 25;
7          int  result = num1 & num2;
8
9          System.out.println("num1 = " + num1);
10         System.out.println("num2 = " + num2);
11         System.out.println("result = " + result);
12     }
13 }
14
```

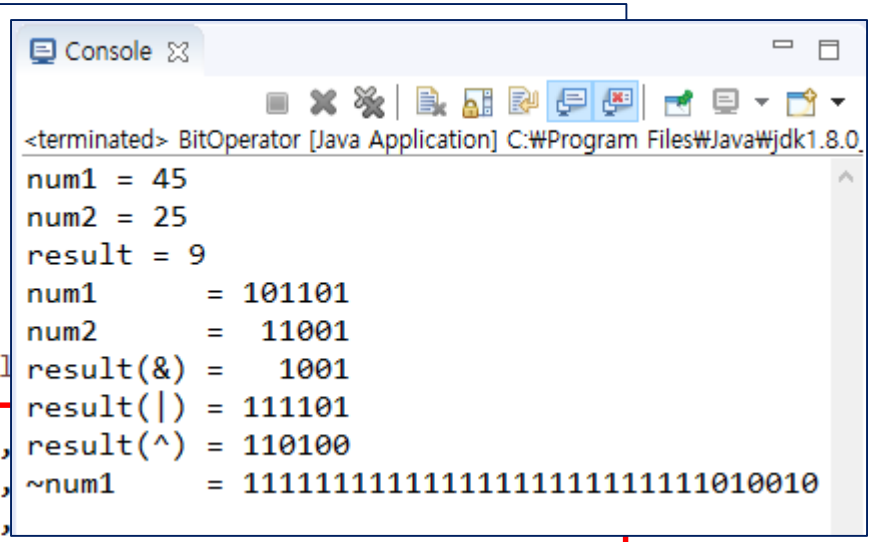


비트 연산자 실습 (BitOperator.java)

```
 2
 3 public class BitOperator {
 4     public static void main(String[] args) {
 5         byte num1   = 45;
 6         byte num2   = 25;
 7         int  result = num1 & num2;
 8
 9         System.out.println("num1 = " + num1);
10         System.out.println("num2 = " + num2);
11         System.out.println("result = " + result);
12
13         System.out.printf("num1       = %6s\n", Integer.toBinaryString(num1));
14         System.out.printf("num2       = %6s\n", Integer.toBinaryString(num2));
15         System.out.printf("result(&) = %6s\n", Integer.toBinaryString(result));
16
17         result = num1 | num2;
18         System.out.printf("result(|) = %6s\n", Integer.toBinaryString(result));
19
20         result = num1 ^ num2;
21         System.out.printf("result(^) = %6s\n", Integer.toBinaryString(result));
22
23         result = ~num1;
24         System.out.printf("~num1      = %6s\n", Integer.toBinaryString(result));
25     }
26 }
```

비트 연산자 실습 (BitOperator.java)

```
2
3  public class BitOperator {
4      public static void main(String[] args) {
5          byte num1   = 45;
6          byte num2   = 25;
7          int  result = num1 & num2;
8
9          System.out.println("num1 = " + num1);
10         System.out.println("num2 = " + num2);
11         System.out.println("result = " + resul
12
13         System.out.printf("num1      = %6s\n",
14         System.out.printf("num2      = %6s\n",
15         System.out.printf("result(&) = %6s\n",
16
17         result = num1 | num2;
18         System.out.printf("result(|) = %6s\n", Integer.toBinaryString(result));
19
20         result = num1 ^ num2;
21         System.out.printf("result(^) = %6s\n", Integer.toBinaryString(result));
22
23         result = ~num1;
24         System.out.printf("~num1     = %6s\n", Integer.toBinaryString(result));
25     }
26 }
```



비트 이동 연산자 실습 (BitShiftOperator.java)

```
2
3  public class BitShiftOperator {
4      public static void main(String[] args) {
5          System.out.println("1 << 3   = " + (1<<3));
6
7          System.out.println();
8          System.out.println("-8 >> 3  = " + (-8>>3));
9
10         System.out.println();
11         System.out.println("-9 >>> 3 = " + (-9>>>3));
12     }
13 }
14
15
```

비트 이동 연산자 실습 (BitShiftOperator.java)

```
2
3  public class BitShiftOperator {
4      public static void main(String[] args) {
5          System.out.println("1 << 3   = " + (1<<3));
6
7          System.out.println();
8          System.out.println("-8 >> 3  = " + (-8>>3));
9
10         System.out.println();
11         System.out.println("-9 >>> 3 = " + (-9>>>3));
12     }
13 }
14
15
```

Console

<terminated> BitShiftOperator [Java Application] C:₩Program Files₩Java₩jdk1.8.0_201

```
1 << 3   = 8

-8 >> 3  = -1

-9 >>> 3 = 536870910
```

비트 이동 연산자 실습 (BitShiftOperator.java)

```
2
3  public class BitShiftOperator {
4      public static void main(String[] args) {
5          System.out.println("1 << 3 = " + (1<<3));
6          System.out.printf ("1      = %8s\n", Integer.toBinaryString(1));
7          System.out.printf ("(1<<3) = %8s\n", Integer.toBinaryString(1<<3));
8
9          System.out.println();
10         System.out.println("-8 >> 3   = " + (-8>>3));
11         System.out.printf ("-8       = %32s\n", Integer.toBinaryString(-8));
12         System.out.printf ("(-8 >> 3) = %32s\n", Integer.toBinaryString(-8>>3));
13
14         System.out.println();
15         System.out.println("-9 >>> 3 = " + (-9>>>3));
16         System.out.printf ("-9       = %32s\n",Integer.toBinaryString(-9));
17         System.out.printf ("(-9>>>3) = %32s",  Integer.toBinaryString(-9>>>3));
18     }
19 }
20
```

비트 이동 연산자 실습 (BitShiftOperator.java)

```
2
3  public class BitShiftOperator {
4      public static void main(String[] ar
5          System.out.println("1 << 3 = " +
6          System.out.printf ("1       = %8
7          System.out.printf ("(1<<3) = %8
8
9          System.out.println();
10         System.out.println("-8 >> 3   =
11         System.out.printf ("-8         =
12         System.out.printf ("(-8 >> 3) =
13
14         System.out.println();
15         System.out.println("-9 >>> 3 =
16         System.out.printf ("-9         = %32s\n",Integer.toBinaryString(-9));
17         System.out.printf ("(-9>>>3) = %32s",  Integer.toBinaryString(-9>>>3));
18     }
19 }
20
```

```
Console
<terminated> BitShiftOperator [Java Application] C:₩Program Files₩Java₩jdk1.8.0_201₩b
1 << 3 = 8
1       =        1
(1<<3) =     1000

-8 >> 3   = -1
-8        = 11111111111111111111111111111000
(-8 >> 3) = 11111111111111111111111111111111

-9 >>> 3 = 536870910
-9       = 11111111111111111111111111110111
(-9>>>3) =    11111111111111111111111111110
```

비트 이동 연산자 실습 (BitShiftOperator.java)

BitShiftOperator.java

```java
package week4;

public class BitShiftOperator {
    public static void main(String[] args) {
        //좌측이동 연산자(<<)는 최종적으로 2의 3승을 곱한 결과를 얻을 수 있다.
        System.out.println("1 << 3 = " + (1<<3));
        System.out.printf ("1      = %8s\n", Integer.toBinaryString(1));
        System.out.printf ("(1<<3) = %8s\n", Integer.toBinaryString(1<<3));

        System.out.println();
        //우측이동 연산자(>>)는 최종적으로 2의 3승으로 나눈 결과를 얻을 수 있다.
        System.out.println("-8 >> 3   = " + (-8>>3));
        System.out.printf ("-8       = %32s\n", Integer.toBinaryString(-8));
        System.out.printf ("(-8 >> 3) = %32s\n", Integer.toBinaryString(-8>>3));

        System.out.println();
        System.out.println("-9 >>> 3 = " + (-9>>>3));
        System.out.printf ("-9       = %32s\n",Integer.toBinaryString(-9));
        System.out.printf ("(-9>>>3) = %32s",  Integer.toBinaryString(-9>>>3));
    }
}
```

Console

<terminated> BitShiftOperator [Java Application] C:₩Program Files₩Java₩jdk1.8.0_201₩b

```
1 << 3 = 8
1      =        1
(1<<3) =     1000

-8 >> 3   = -1
-8        = 11111111111111111111111111111000
(-8 >> 3) = 11111111111111111111111111111111

-9 >>> 3 = 536870910
-9       = 11111111111111111111111111110111
(-9>>>3) =    11111111111111111111111111110
```

4절. 이항 연산자

- **대입 연산자(=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=)**
  - 오른쪽 피연산자의 값을 좌측 피연산자인 변수에 저장
  - 모든 연산자들 중 가장 낮은 연산 순위 -> 제일 마지막에 수행
  - 종류
    - 단순 대입 연산자
    - 복합 대입 연산자
      - 정해진 연산을 수행한 후 결과를 변수에 저장

4절. 이항 연산자

❖ 대입 연산자의 종류

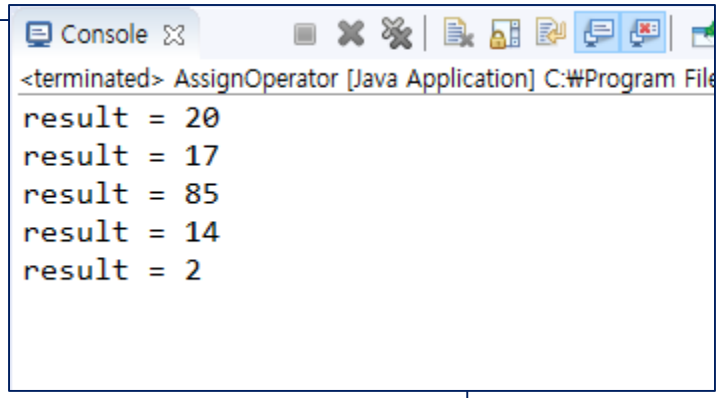
구분	연산식			설명
단순 대입 연산자	변수	=	피연산자	우측의 피연산자의 값을 변수에 저장
복합 대입 연산자	변수	+=	피연산자	우측의 피연산자의 값을 변수의 값과 더한 후에 다시 변수에 저장 (변수=변수+피연산자 와 동일)
	변수	-=	피연산자	우측의 피연산자의 값을 변수의 값에서 뺀 후에 다시 변수에 저장 (변수=변수-피연산자 와 동일)
	변수	*=	피연산자	우측의 피연산자의 값을 변수의 값과 곱한 후에 다시 변수에 저장 (변수=변수*피연산자 와 동일)
	변수	/=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 다시 변수에 저장 (변수=변수/피연산자 와 동일)
	변수	%=	피연산자	우측의 피연산자의 값으로 변수의 값을 나눈 후에 나머지를 변수에 저장 (변수=변수%피연산자 와 동일)
	변수	&=	피연산자	우측의 피연산자의 값과 변수의 값을 & 연산 후 결과를 변수에 저장 (변수=변수&피연산자 와 동일)
	변수	|=	피연산자	우측의 피연산자의 값과 변수의 값을 | 연산 후 결과를 변수에 저장 (변수=변수|피연산자 와 동일)
	변수	^=	피연산자	우측의 피연산자의 값과 변수의 값을 ^ 연산 후 결과를 변수에 저장 (변수=변수^피연산자 와 동일)
	변수	<<=	피연산자	우측의 피연산자의 값과 변수의 값을 << 연산 후 결과를 변수에 저장 (변수=변수<<피연산자 와 동일)
	변수	>>=	피연산자	우측의 피연산자의 값과 변수의 값을 >> 연산 후 결과를 변수에 저장 (변수=변수>>피연산자 와 동일)
	변수	>>>=	피연산자	우측의 피연산자의 값과 변수의 값을 >>> 연산 후 결과를 변수에 저장 (변수=변수>>>피연산자 와 동일)

대입 연산자 실습 (AssignOperator.java)

```
2
3  public class AssignOperator {
4      public static void main(String[] args) {
5          int result = 10;
6
7          result += 10;
8          System.out.println("result = " + result);
9
10         result -= 3;
11         System.out.println("result = " + result);
12
13         result *= 5;
14         System.out.println("result = " + result);
15
16         result /= 6;
17         System.out.println("result = " + result);
18
19         result %= 4;
20         System.out.println("result = " + result);
21     }
22 }
23
```

대입 연산자 실습 (AssignOperator.java)

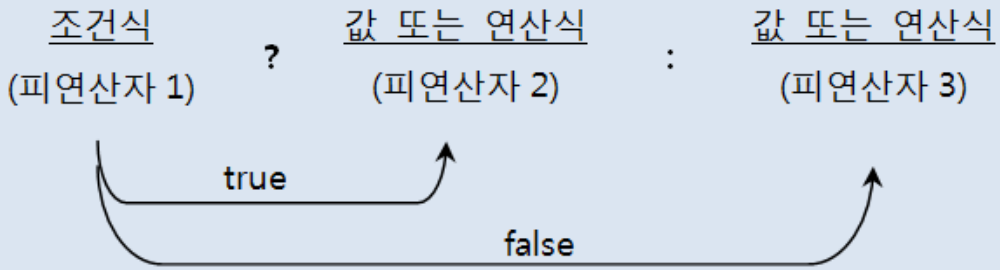
```
2
3  public class AssignOperator {
4      public static void main(String[] args) {
5          int result = 10;
6
7          result += 10;
8          System.out.println("result = " + result);
9
10         result -= 3;
11         System.out.println("result = " + result);
12
13         result *= 5;
14         System.out.println("result = " + result);
15
16         result /= 6;
17         System.out.println("result = " + result);
18
19         result %= 4;
20         System.out.println("result = " + result);
21     }
22 }
23
```



5절. 삼항 연산자

❖ 삼항 연산자란?

- 세 개의 피연산자를 필요로 하는 연산자
- 앞의 조건식 결과에 따라 콜론 앞 뒤의 피연산자 선택 -> 조건 연산식




```
int score = 95;
char grade = (score>90) ? 'A' : 'B'
```

=

```
int score = 95;
char grade;
if(score>90) {
   grade = 'A';
} else {
   grade = 'B';
}
```

삼항 연산자 실습 (ConditionOperator.java)

```
2
3  public class ConditionOperator {
4      public static void main(String[] args) {
5
6          int num1 = 35;
7          int num2 = 47;
8          String result;
9
10         result = (num1 > num2)? "num1이 num2보다 큽니다" : "num1이 num2보다 작습니다";
11         System.out.printf("num1 = %d, num2 = %d\n", num1, num2);
12         System.out.printf("두 수를 비교한 결과는 %s\n\n", result);
13
14         System.out.printf("%d가 %d보다 큽니까? %b\n\n", num1, num2, (num1 > num2));
15
16         boolean bResult;
17         bResult = (num1 > num2)? true : false;
18         System.out.printf("num1 = %d, num2 = %d\n", num1, num2);
19         System.out.printf("num1이 num2보다 큽니까? %b\n\n", bResult);
20     }
21 }
22
23
```

삼항 연산자 실습 (ConditionOperator.java)

```
2
3  public class ConditionOperator {
4      public static void main(String[] args) {
5
6          int num1 = 35;
7          int num2 = 47;
8          String result;
9
10         result = (num1 > num2)? "num1이 num
11         System.out.printf("num1 = %d, num2
12         System.out.printf("두 수를 비교한 결과는 %
13
14         System.out.printf("%d가 %d보다 큽니까? %b\n\n", num1, num2, (num1 > num2));
15
16         boolean bResult;
17         bResult = (num1 > num2)? true : false;
18         System.out.printf("num1 = %d, num2 = %d\n", num1, num2);
19         System.out.printf("num1이 num2보다 큽니까? %b\n\n", bResult);
20     }
21 }
22
23
```

Console

<terminated> ConditionOperator [Java Application] C:₩Program Files₩Java₩jdk1.8.0_2

```
num1 = 35, num2 = 47
두 수를 비교한 결과는 num1이 num2보다 작습니다

35가 47보다 큽니까? false

num1 = 35, num2 = 47
num1이 num2보다 큽니까? false
```

삼항 연산자 실습 (ConditionOperator2.java)

```
2
3  public class ConditionOperator2 {
4      public static void main(String[] args) {
5          int    score = 85;
6          String result;
7
8          result = (score > 90)? "우수": ((score > 80)? "보통" : "미달");
9
10         System.out.println("점수는 = " + score);
11         System.out.println("점수 결과는 = " + result);
12
13     }
14 }
15
16
```

삼항 연산자 실습 (ConditionOperator2.java)

```
2
3  public class ConditionOperator2 {
4      public static void main(String[] args) {
5          int    score = 85;
6          String result;
7
8          result = (score > 90)? "우수": ((score > 80)? "보통" : "미달");
9
10         System.out.println("점수는 = " + score);
11         System.out.println("점수 결과는 = " + result);
12
13     }
14 }
15
16
```

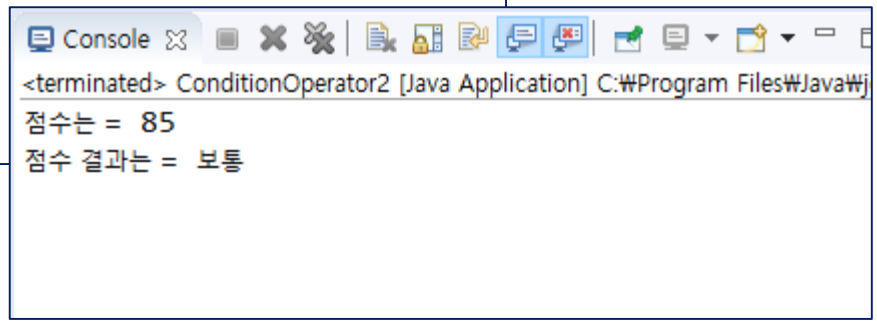
삼항 연산자 실습 (ConditionOperator2.java)

```
2
3  public class ConditionOperator2 {
4      public static void main(String[] args) {
5          int    score = 85;
6          String result;
7
8          result = (score > 90)? "우수": ((score > 80)? "보통" : "미달");
9
10         System.out.println("점수는 = " + score);
11         System.out.println("점수 결과는 = " + result);
12
13     }
14 }
15
16
```

```
((score > 80)? "보통" : "미달");
```

삼항 연산자 실습 (ConditionOperator2.java)

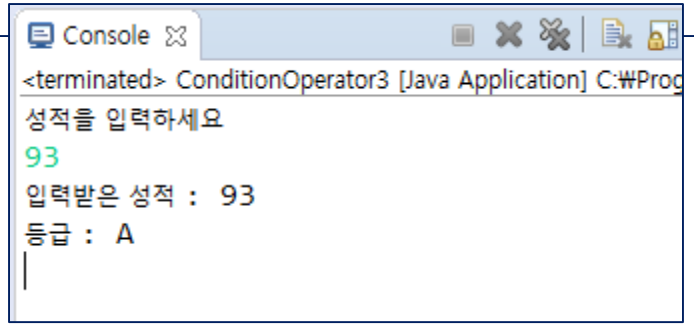
```
2
3  public class ConditionOperator2 {
4      public static void main(String[] args) {
5          int    score = 85;
6          String result;
7
8          result = (score > 90)? "우수": ((score > 80)? "보통" : "미달");
9
10         System.out.println("점수는 = " + score);
11         System.out.println("점수 결과는 = " + result);
12
13     }
14 }
15
16
```



과제물 (ConditionOperator3.java)

- Scanner 객체를 이용하여 성적을 입력받아 등급을 출력하는 프로그램을 완성하시오.
  - 90 ~ : A, 80 ~ 89 : B, 70 ~ 79 : C, 60 ~ 69 : D, ~ 59 : F
  - 3항 연산자를 사용할 것

< 실행결과 >




```
2
3  import java.util.Scanner;
4
5  public class ConditionOperator3 {
6      public static void main(String[] args) {
7
8          int  score;
9          char grade;
10
11         Scanner scanData = new Scanner(System.in);
12
13         System.out.println("성적을 입력하세요");
14         score = scanData.nextInt();
15
16
17
18
19
20
21         scanData.close();
22     }
23 }
24
```

소스 코드와 실행 결과를 포함하여 화면을 저장한 후 제출