형변환 (Casting)

#01. 암묵적 형변환

1) 정수의 경우

리터럴의 형변환

소스코드에 단독으로 명시되는 숫자나 문자열, 논리값 등을 **리터럴**이라고 한다.

💡 ex) `1`, `-100`, `true`, `"Hello World"`

정수 형태의 리터럴은 기본적으로 **int** 타입으로 식별된다. byte, short, long 타입의 변수에 리터럴을 대입하는 코드는 컴파일시에 **int** 타입에서 암묵적으로 형변환 된다.

아래의 코드는 모두 int 형의 리터럴을 다른 정수 타입으로 형변환 처리 한다.

```
// int 타입의 리터럴을 다른 타입으로 변환하는 것으로 간주
byte a = 1;
short b = 2;
long c = 3;
```

이미 선언된 변수를 다른 타입에 대입하는 경우

최초로 선언된 변수보다 *더 큰 크기의 메모리를 차지하는 데이터 타입의 변수*에는 특별한 처리 없이 대입이 가능하다.

예를 들어 int형의 변수를 long 타입의 변수에 대입하는 것은 4byte의 공간에 저장된 데이터를 8byte의 공간으로 옮기는 것이므로 아무런 문제가 없다.

```
int a = 10;     // 4byte
long b = a;     // 8byte
```

반대로 8byte의 메모리를 차지하는 long 타입의 변수를 4byte 공간의 int 형 변수에 대입하는 경우 메모리 공간의 부족으로 인한 데이터 손실이 발생하므로 컴파일시에 에러가 발생한다.

```
long a = 10;    // 8byte
int b = a;      // 4byte <-- 에러
```

위의 코드는 아래와 같이 **long에서 int로 변환할 수 없다(from long to int)**는 내용의 에러가 발생한다.

```
Test.java:4: error: incompatible types: possible lossy conversion **from long to
int**
```

```
        int b = a;
                ^
1 error
```

2) 실수의 경우

리터럴의 형변환

실수 형태의 리터럴 값은 기본적으로 **double**로 인식된다. 그렇기 때문에 **float** 타입의 변수에 실수를 대입하는 것은 8byte의 메모리 공간의 데이터를 4byte 공간으로 옮기는 것으로 간주되어 에러가 발생한다.

```
float a = 3.14;
```

아래와 같이 double 타입을 float 타입에 대입할 수 없다는 내용의 에러가 발생한다.

```
Test.java:3: error: incompatible types: possible lossy conversion **from double to
float**
        float a = 3.14;
                  ^
1 error
```

float 타입으로 선언된 변수에 리터럴을 대입하기 위해서는 Float 형임을 알려주기 위해 리터럴 뒤에 `F`를 붙여야 한다. (대소문자 가리지 않음)

```
float a = 3.14F;
float b = 12.345f;
```

이미 선언된 변수를 다른 타입에 대입하는 경우

float 타입의 변수를 **double**형에 대입하는 것은 성립되지만 반대의 경우는 성립되지 않는다.

```
float a = 10.5F;  // 4byte
double b = a;     // 8byte의 공간에 대한 대입이므로 성립함.
```

```
double a = 10.5;  // 8byte
float b = a;      // 4byte <-- 에러
```

3) char 타입의 경우

`char` 타입은 홑따옴표로 감싼 한 글자를 의미.

리터럴의 형변환

모든 숫자형 변수에 직접 대입 가능하다.

```
byte num1 = 'a';
short num2 = 'a';
int num3 = 'a';
long num4 = 'a';
float num5 = 'a';
double num6 = 'a';

System.out.println(num1);
System.out.println(num2);
System.out.println(num3);
System.out.println(num4);
System.out.println(num5);
System.out.println(num6);
```

- **출력결과**

```
97
97
97
97
97.0
97.0
```

아스키 코드

컴퓨터에서 사용되는 모든 글자에 적용된 일련번호

알파벳 **a**의 아스키 코드는 **97**이다.

`char` 타입의 리터럴을 숫자형 변수에 대입할 경우 자동으로 아스키 코드로 변환되어 진다.

97을 이진수로 변환하면 **1100001**인데 이는 알파벳 **a**의 바이너리 데이터인 **01100001**와 일치한다.

즉, 아스키 코드는 글자를 바이너리로 변환한 상태를 10진수로 표현한 값.

char 타입의 알파벳 `a` 와 `b` 를 더한 결과를 `int` 타입의 변수에 할당하면, `a`의 아스키코드 97과 `b`의 아스키 코드 98에 대한 합산 결과가 표시된다.

```
int k = 'a' + 'b';
System.out.println(k);
```

- **출력결과**

```
195
```

이미 선언된 변수를 다른 타입에 대입하는 경우

자바에서는 **char** 타입에게 2byte의 메모리 공간을 할당한다. 그러므로 1byte의 크기를 갖는 **byte** 타입과 같은 크기의 공간을 차지하는 **short** 타입을 제외한 모든 형태의 숫자형 변수에 대입 가능하다.

💡 암묵적 형 변환은 데이터 타입의 메모리 크기가 큰 타입으로만 적용 가능하다.

```
char word = 'a';
byte num1 = word;   // <-- 에러
```

```
char word = 'a';
short num2 = word;   // <-- 에러
```

그 외의 경우는 문제 없음

```
char word = 'a';
int num3 = word;
long num4 = word;
float num5 = word;
double num6 = word;
```

4) 그 밖의 경우

1. 모든 형태의 **정수형 변수**는 모든 형태의 **실수형 변수에 대입 가능**하다.
    - `long` 타입의 정수형 변수는 `float` 타입보다 메모리 크기가 더 크지만 `long`에서 `float`로의 암묵적 형변환은 허용된다.
2. 모든 형태의 **실수형 변수**는 모든 형태의 **정수형 변수에 대입 불가능**하다.
    - `float` 타입의 실수형 변수는 `long` 타입보다 메모리 크기가 더 작지만 `float`에서 `long`으로의 암묵적 형변환은 허용되지 않는다. (소수점 이하 자리에 대한 오차가 불가피하기 때문.)
3. `boolean` **타입은 어떠한 타입으로 변환 불가능.**

5) 암묵적 형변환에 대한 결론

1. 정수는 실수로 변환 가능. 실수는 정수로 변환 불가능.
2. 정수끼리 혹은 실수끼리는 메모리 크기가 더 큰 타입으로는 변환 가능. 반대는 불가능.
3. `char` 타입의 경우 `int`형 이상의 메모리 크기에는 대입가능.
4. 이론적으로는 메모리 크기에 대한 데이터 이동 제약이지만 자바에서는 단순히 **값에 대한 표현 범위로 이해**하는 것이 더 효율적이다.

#02. 명시적 형변환

암묵적 형변환에서 에러가 발생하는 경우는 메모리 크기의 차이로 인한 데이터의 손실이 발생하는 경우이다.

데이터의 손실을 감수하고서라도 강제로 타입을 변환하는 처리를 **명시적 형변환**이라고 한다.

적용방법

할당하는 리터럴이나 변수앞에 변환할 타입을 직접 명시한다.

실수형 값을 정수형으로 강제 변환 할 경우 소수점 아래자리는 모두 버려진다.

```
double k = 12.345;
int a = (int) k;
```