Literal

리터럴이란 소스코드의 고정된 값을 대표하는 용어입니다. 무슨말인지 잘 모르시겠죠? 예제를 통해서 보겠습니다.

```
int myInt = 123;
byte myByte = 38;
short myShort = 2;
```

여기서 **123**, **38**, **2** 는 '정수 리터럴'입니다. 기본적으로는 **int** 자료형이지만 **byte** 나 **short** 변수의 초기값으로 설정하면 아무 문제 없이 값이 들어갑니다.

영상 강의에서 다룬 적 있는 **long** 에 대해서도 봅시다.

```
long myLong = 12345678910; // 오류: 정수 값이 너무 크다
```

여기서도 오른쪽 **12345678910** 은 **int** 리터럴인데, **int** 가 담을 수 있는 범위를 넘어섰기 때문에 오류가 나는 것입니다. 이 문제를 해결하기 위해서는 뒤에 **L** 을 붙여주면 됩니다.

```
long myLong = 12345678910L;
```

위에서 **12345678910L** 은 정수 리터럴이 아니라 '롱 리터럴'입니다.

여러 예시를 통해 숫자형 리터럴에 익숙해져 보세요~

```
long e = 123;
long f = 123L;   // long의 리터럴 표현방법

float g = 3.14;  // 오류: 형이 맞지 않음. 필요한 값: float, 발견된 값: double
float h = 3.14f; // float의 리터럴 표현방법
float i = 314f;  // float의 리터럴 표현방법

double j = 3.14;
double k = 314d; // double의 리터럴 표현 방법
```

아무런 표기가 없는 소수형은 **double** 이라는 것을 알겠죠? 물론 **double** 을 명시하는 리터럴로 **d** 를 붙여주는 방법도 있습니다.

랭크

Type	Size	Range
byte	1 byte	-128 ... 127

short	2 byte	-32,768 ... 32,767
int	4 byte	-2,147,483,648 ... 2,147,483,647

Type	Size	Range
long	8 byte	-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807
float	4 byte	1.4023985 x 10^-45 ... 3.4028235 x 10^38
double	8 byte	4.940656458412465 x 10^-324 ... 1.797693134862316 x 10^308

예전에 소수형이 정수형보다 강하기 때문에(랭크가 높기 때문에) 소수형과 정수형 간의 연산에는 소수형 결과값이 나온다고 했었죠? 표의 위에서 부터 아래로 갈수록 랭크가 높은 자료형입니다. 기준은 각 자료형이 가질 수 있는 값의 범위입니다.

형 변환

자바의 숫자 자료형들은 위에서 배운 랭크에 따라 형 변환이 가능합니다.

to ₩ from	byte	short	int	long	float	double
byte	-	X	X	X	X	X
short	O	-	X	X	X	X
int	O	O	-	X	X	X
long	O	O	O	-	X	X
float	O	O	O	O	-	X
double	O	O	O	O	O	-

바꾸고자 하는 형(to)이 기존의 형(from)보다 넓은 데이터를 담을 수 있는 자료형일 경우 특별한 처리 없이 형을 변환할 수 있습니다.

```
int a = 36;
double b = a;     // int to double

short c = 17;
long d = c;       // short to long

float e = 3.14f;
double f = e;     // float to double
```

타입 캐스팅 (Type Casting)

값(혹은 변수) 앞에 **(자료형)** (예: **(int) x**)을 적어주면 강제적으로 형을 변환시킬 수 있습니다. 물론 형 변환이 가능한 경우에 대해서만 가능하겠죠? 숫자 자료형들 사이에서는 모두 가능합니다.

```
int a = 3;
double b = (double) a;
long c = (long) a;
```

```
System.out.println(b);
System.out.println(c);
```

```
3.0
3
```

더 큰 랭크의 값을 더 작을 랭크의 변수에 담는 것도 가능하지만, 데이터의 손실이 있다는 걸 주의해주세요!

```
double pi = 3.14;
int myInt = (int) pi; // 데이터 손실 (소수 부분)
System.out.println(myInt);
```

```
3
```

또 다른 예시를 봅시다.

```
int a = 9, b = 5;
System.out.println(a / b);
```

```
1
```

이렇게 정수값 **a** 와 **b** 가 주어졌을 때, **9 / 5** 를 하면 **1** 이 나오죠? 여기서 **1.8** 이 나오게 하고 싶으면 타입 캐스팅을 해서 **9** 를 **9.0** 으로 바꿔주고, **9.0 / 5** 를 계산하면 됩니다.

```
int a = 9, b = 5;
System.out.println((double) a / b);
```

```
1.8
```

형 변환은 나중에 배울 클래스, 상속 등의 개념에서도 등장합니다. 그 때도 숫자형의 형 변환 처럼 방향성이 중요한 역할을 하니, 형 변환은 잘 기억해두세요!

☑ 수업을 완료하셨으면 체크해주세요.

수강생 Q&A 보기

(/questions?assignment_id=263&sort_by=popular)

질문하기

(/questions/new?assignment_id=263&op1=%EA%B0%9D%EC%B2%B4+%EC%A7

이전 강의 다음 강의

이전 형식
형 변환 (/assignments/253)

다음 형식
연산자와 형 변환 (/assignments/270)