

상수와 형 변환

03-1. 상수(Constants)

자바의 일반적인 상수

▶ 자바에서 말하는 '상수'

- 변수에 값을 딱 한 번만 할당할 수 있으면 그것은 상수!
- 한 번 할당된 값은 변경이 불가능하다.
- 키워드 `final` 선언이 붙어있는 변수

상수 선언의 예 초기화 하지 않으면 딱 한번 초기화 가능하다!

▶ `final` 기반의 상수 선언의 예

`MAX_SIZE = 100;`


- 상수의 이름은 모두 대문자로 짓는 것이 관례
- 이름이 둘 이상의 단어로 이뤄질 경우 단어를 언더바로 연결하는 것이 관례

```
final int
```

final 상수 선언의 예

◆ Constants.java

```
1. class Constants {  
2.     public static void main(String[] args) {  
3.         final int MAX_SIZE = 100;  
4.         final char CONST_CHAR = '상';  
5.         final int CONST_ASSIGNED;  
6.  
7.         CONST_ASSIGNED = 12;    // 할당하지 않았던 상수의 값 할당  
8.         System.out.println("상수1 : " + MAX_SIZE);  
9.         System.out.println("상수2 : " + CONST_CHAR);  
10.        System.out.println("상수3 : " + CONST_ASSIGNED);  
11.    }  
12. }
```



```
C:\JavaStudy>java Constants  
상수1 : 100  
상수2 : 상  
상수3 : 12  
C:\JavaStudy>
```

리터럴(Literals)에 대한 이해

▶ 리터럴

- 자료형을 기반으로 표현이 되는 상수를 의미한다.

ex) `int num1 = 5 + 7;`

ex) `double num2 = 3.3 + 4.5;`


- 정수는 무조건 `int`형으로 인식하기로 약속되어 있음
- 따라서 5와 7은 '정수형 리터럴'이다.
- 그리고 3.3과 4.5는 '실수형 리터럴'이다.

'리터럴'이라는 표현은 '상수'라는 표현으로 대신하는 경우가 많다.

정수형 상수(리터럴)의 표현 방법

◆ IntegerLiterals.java

```
1. class IntegerLiterals {
2.     public static void main(String[] args) {
3.         int num1 = 123;    // 10진수 표현
4.         int num2 = 0123;   // 8진수 표현
5.         int num3 = 0x123;  // 16진수 표현
6.
7.         System.out.println("num1: " + num1);
8.         System.out.println("num2: " + num2);
9.         System.out.println("num3: " + num3);
10.
11.        System.out.println("11 + 22 + 33 = " + (11 + 22 + 33));
12.        System.out.println("011 + 022 + 033 = " + (011 + 022 + 033));
13.        System.out.println("0x11 + 0x22 + 0x33 = " + (0x11 + 0x22 + 0x33));
14.    }
15. }
```



```
C:\JavaStudy>java IntegerLiterals
num1: 123
num2: 83
num3: 291
11 + 22 + 33 = 66
011 + 022 + 033 = 54
0x11 + 0x22 + 0x33 = 102
C:\JavaStudy>
```

Long형 상수(리터럴)의 표현 방법

```
System.out.println(3147483647 + 3147483648);
```

컴파일시 Integer number too large 라는 오류 메시지를 전달한다.

```
System.out.println(3147483647L + 3147483648L);
```

| 또는 **L**을 붙여서 long형 상수로 표현해 달라는 요청을 해야 한다.

정수형 상수의 이진수 표현방법과 언더바 삽입

```
byte seven = 0B111;  
int num205 = 0B11001101;
```

0B 또는 0b를 붙여서 이진수 표현

```
int num = 100_000_000;  
int num = 12_34_56_78_90;
```

원하는 위치에 언더바 삽입 가능

실수형 상수(리터럴)

```
System.out.println(3.0004999 + 2.0004999);  
System.out.println(3.0004999D + 2.0004999D);
```

실수는 기본 double형 double형임을 명시하기 위해 d 또는 D 삽입 가능

```
System.out.println(3.0004999f + 2.0004999f);
```

실수형 상수를 float형으로 표현하려면 f 또는 F 삽입

실수형 상수의 e표기법

$$3.4\text{e}3 \quad \Rightarrow \quad 3.4 \times 10^3 = 3400.0$$

$$3.4\text{e}-3 \quad \Rightarrow \quad 3.4 \times 10^{-3} = 0.0034$$

부울형 상수와 문자형 상수

true false

부울형 상수

‘한’ ‘글’ ‘A’ ‘Z’

문자형 상수

이스케이프 시퀀스(escape sequences)

'\b'	백스페이스 문자
'\t'	탭 문자
'\\'	백슬래시 문자
'\''	작은따옴표 문자
'\"'	큰따옴표 문자
'\n'	개행 문자
'\r'	캐리지 리턴(carriage return) 문자

화면상의 어떠한 상황 또는 상태를 표현하기 위해 약속된 문자

이스케이프 시퀀스의 예

◆ EscapeSequences.java

```
1. class EscapeSequences {  
2.     public static void main(String[] args) {  
3.         System.out.println("AB" + '\b' + 'C');  
4.         System.out.println("AB" + '\t' + 'C');  
5.         System.out.println("AB" + '\n' + 'C');  
6.         System.out.println("AB" + '\r' + 'C');  
7.     }  
8. }
```



```
C:\JavaStudy>java EscapeSequences  
AC  
AB    C  
AB  
C  
AB\rC  
C:\JavaStudy>
```

03-2. 형 변환

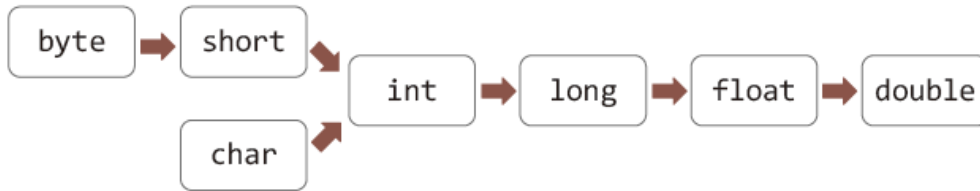
자료형 변환의 의미와 필요한 이유는?

```
int num1 = 50;  
  
long num2 = 3147483647L;  
  
System.out.println(num1 + num2);
```

num1에 저장된 값이 long형으로 형 변환 된다. (자동 형 변환)

- 두 피연산자의 자료형이 일치해야 동일한 방법을 적용하여 연산을 진행할 수 있다.
- 피연산자의 자료형이 일치하지 않을 때 형(Type)의 변환을 통해 일치를 시키야 한다.

자동형 변환(Implicit Conversion)



- 규칙 1. 자료형의 크기가 큰 방향으로 형 변환이 일어난다.
 - 규칙 2. 자료형의 크기에 상관없이 정수 자료정보보다 실수 자료형이 우선한다.
- ex) `double num1 = 30;`
- ex) `System.out.println(59L + 34.5);`

명시적 형 변환(Explicit Conversion)

자동 형 변환 규칙에 부합하지는 않지만, 형 변환이 필요한 상황이면 명시적 형 변환을 진행한다.

ex1)

```
double pi = 3.1415;  
int wholeNumber = (int)pi;
```

ex2)

```
long num1 = 3000000007L;  
int num2 = (int)num1;
```

ex3)

```
short num1 = 1;  
short num2 = 2;  
short num3 = (short)(num1 +  
num2);
```