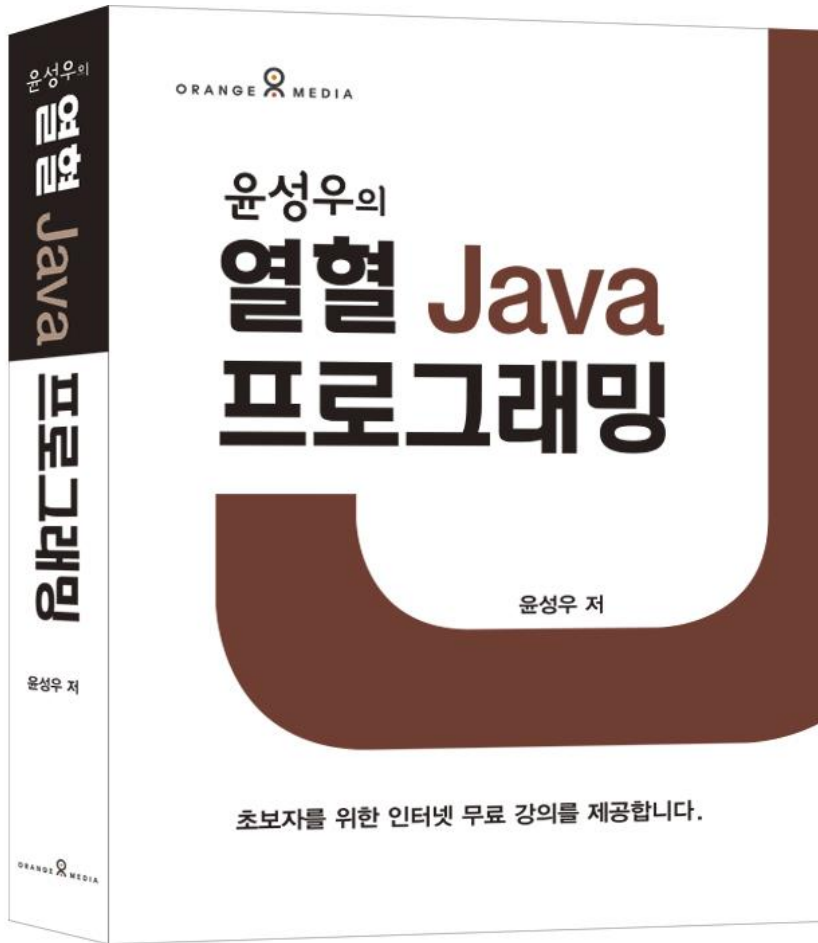


2022년 봄학기

---

JAVA

나사렛대학교  
IT인공지능학부  
김광기



# 열혈 Java 프로그래밍

---

## Chapter 02. 변수와 자료형

02-1.

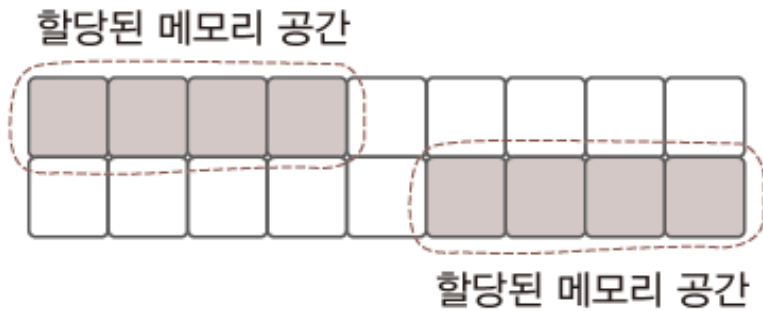
변수의 이해와 활용

# 메모리 공간의 활용을 위해 필요한 변수

---

## ▶ 변수(Variable)

- 메모리 공간의 활용을 위한 도구
- 메모리 공간의 **할당**과 **접근**을 위해 필요한 도구
- 변수의 선언은 '메모리 공간의 할당'으로 이어진다.



## 메모리 공간 할당의 예

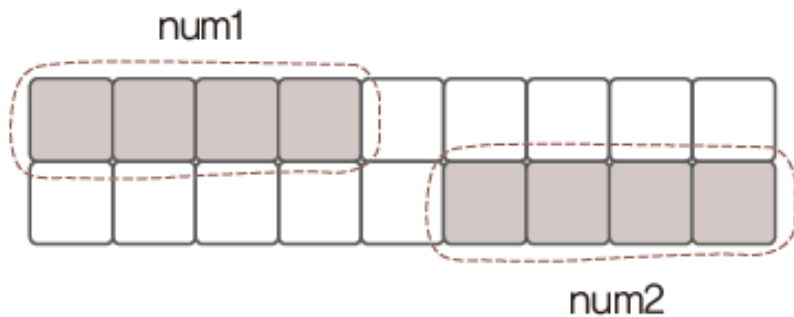
---

▶ 변수의 선언을 통해 결정하는 것 두 가지!

- 변수의 이름
- 변수의 용도

int num1;    ◁ 변수 num1의 선언

int num2;    ◁ 변수 num2의 선언



## ◆ UseVariable.java

```
1. class UseVariable {  
2.     public static void main(String[] args) {  
3.         int num1;    // 변수 num1의 선언  
4.         num1 = 10;   // 변수 num1에 10을 저장  
5.  
6.         int num2 = 20; // 변수 num2 선언과 동시에 20으로 초기화  
7.         int num3 = num1 + num2; // 두 변수 값을 대상으로 덧셈  
8.         System.out.println(num1 + " + " + num2 + " = " + num3);  
9.     }  
10. }
```

자바에서 = 은 '같다'가 아닌 '대입'의 의미이다.

대입은 오른쪽에 있는 값을 왼쪽으로!



```
C:\JavaStudy>java UseVariable  
10 + 20 = 30  
C:\JavaStudy>
```

## 변수 활용의 예

자료형	데이터	크 기	표현 가능 범위
boolean	참과 거짓	1 바이트	true, false
char	문자	2 바이트	유니코드 문자
byte	정수	1 바이트	-128 ~ 127
short		2 바이트	-32,768 ~ 32,767
int		4 바이트	-2,147,483,648 ~ 2,147,483,647
long		8 바이트	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
float	실수	4 바이트	$\pm(1.40 \times 10^{-45} \sim 3.40 \times 10^{38})$
double		8 바이트	$\pm(4.94 \times 10^{-324} \sim 1.79 \times 10^{308})$

자바에서 기본적으로 제공하는 자료형이라 하여 '기본 자료형(Primitive Data Type)'이라 한다.

## 자료형의 종류와 구분

---

## ◆ VariableDecl.java

```
1. class VariableDecl {  
2.     public static void main(String[] args) {  
3.         double num1, num2;    // 두 개의 변수 동시 선언  
4.         double result;  
5.         num1 = 1.0000001;  
6.         num2 = 2.0000001;  
7.         result = num1 + num2;  
8.         System.out.println(result);  
9.     }  
10. }
```



```
C:\JavaStudy>java VariableDecl  
3.0000001999999997  
  
C:\JavaStudy>
```

## 다양한 자료형 활용의 예

기대하는 값 3.00000002 가 출력되지 않았다.

이유는 실수 표현에 오차가 존재하기 때문이다.



01

자바는 대소문자를  
구분한다.

02

변수의 이름은  
숫자로 시작할 수  
없다.

03

\$과 \_ 이외의  
특수문자는 변수의  
이름에 사용할 수  
없다.

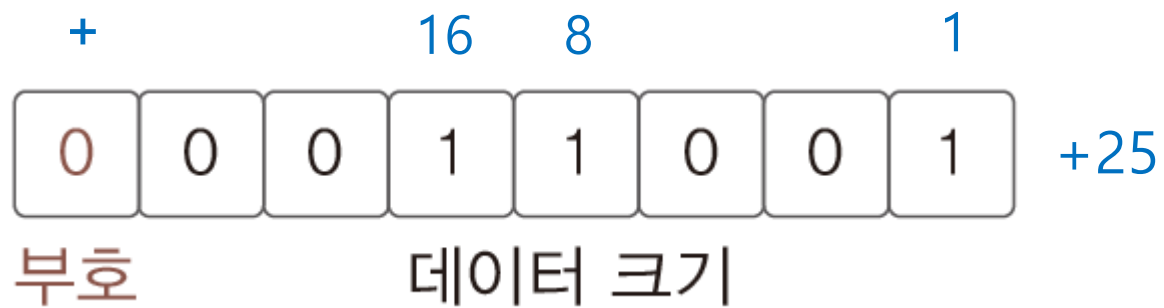
04

키워드는 변수의  
이름으로 사용할 수  
없다.

변수의 이름을 짓는데 있어서의 제약사항

02-2.

정수의 표현 방식 이해하기



- 부호 비트가 0이면 양의 정수, 1이면 음의 정수
- 부호 비트가 0이면, 나머지 비트들은 값의 크기를 결정

## 컴퓨터가 양의 정수를 표현하는 방식

---



-25가 맞을까?

$+$	0	0	0	1	1	0	0	1
	1	0	0	1	1	0	0	1

---

~ 0이 되어야 하는데 ~

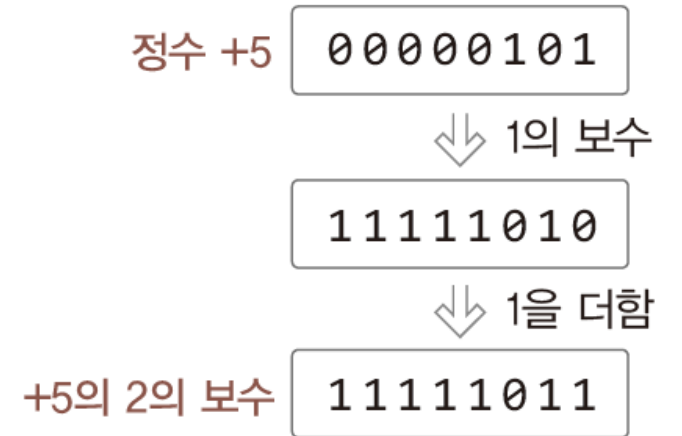
양의 정수 표현법 기반으로 음의 정수를 표현한다면?

---

# 음의 정수를 표현하는 방법

- 양의 정수의 이진수 표현에 2의 보수를 취한 결과를 음의 정수로 표현한다.

- 이 경우, 임의의 양의 정수가 있을 때,  
이와 절댓값이 같은 음의 정수의 합은 0이 된다.



$$\begin{array}{r} 00000101 \\ + 11111011 \\ \hline 1\ 00000000 \end{array}$$

올림 수 버림

02-3.

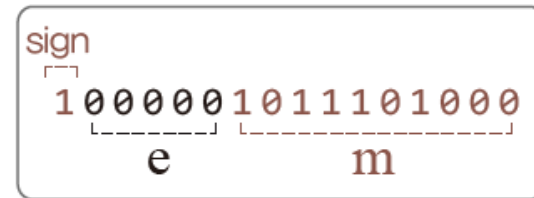
실수의 표현 방식 이해하기

# 실수의 표현 방식 이해

- 정수와 달리 실수는 오차 없이 표현이 불가능
- 따라서 정밀도를 낮추고 표현할 수 있는 값의 범위 넓힘
- 실수 표현 방법의 기준 『IEEE 754』

$$\pm (1.m) \times 2^{e-127}$$

↑ 이 수식에 반영



```
double num1 = 1.00000001;
```

```
double num2 = 2.00000001;
```

num1과 num2에는

최대한 가까운 실수의 표현이 저장된다.

02-4.

자바의 기본 자료형



byte

1 byte

short

2 byte

int

4 byte

long

8 byte

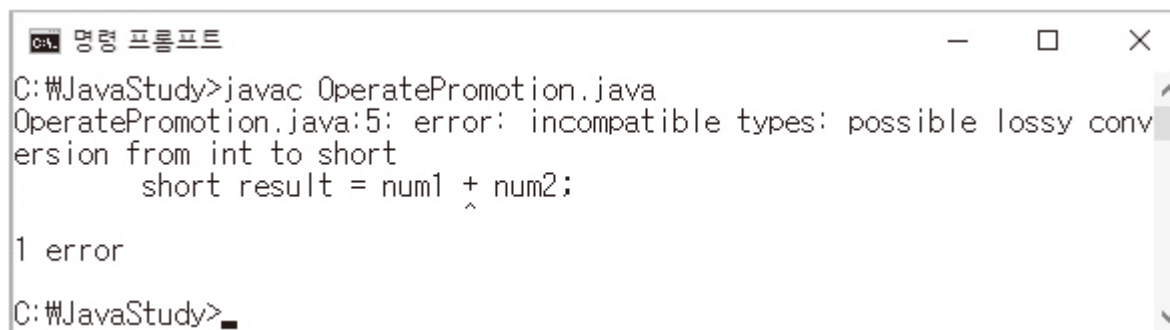
변수의 자료형 결정은 '해당 변수에 값을 저장 및  
참조하는 방식의 결정'을 의미한다.

## 정수 자료형

---

## ◆ OperatePromotion.java

```
1. class OperatePromotion {  
2.     public static void main(String[] args) {  
3.         short num1 = 11;  
4.         short num2 = 22;  
5.         short result = num1 + num2;  
6.         System.out.println(result);  
7.     }  
8. }
```



```
C:\JavaStudy>javac OperatePromotion.java  
OperatePromotion.java:5: error: incompatible types: possible lossy conversion from int to short  
    short result = num1 + num2;  
                        ^  
1 error  
C:\JavaStudy>
```

컴파일 에러는 정수형 덧셈 시 자료형에 상관없이 int형 덧셈을 진행함을 의미한다.

short형 변수와 int형 변수 중 하나를 선택한다면?

float

4 byte

double

8 byte

float	실수	4 바이트	$\pm(1.40 \times 10^{-45} \sim 3.40 \times 10^{38})$
double		8 바이트	$\pm(4.94 \times 10^{-324} \sim 1.79 \times 10^{308})$

float와 double 사이의 자료형 선택 기준은 **정밀도!**

실수 자료형

---

	D5D	D5E	D5F	D60	D61	D62	D63	D64	D65
0	헐 D5D0	헝 D5E0	험 D5F0	혀 D600	헝 D610	헞 D620	헟 D630	홀 D640	혹 D650
1	헝 D5D1	헞 D5E1	헟 D5F1	헝 D601	헡 D611	헢 D621	헣 D631	홉 D641	홉 D651
2	헡 D5D2	헢 D5E2	헣 D5F2	헝 D602	헡 D612	헢 D622	헣 D632	홉 D642	홉 D652
3	헢 D5D3	헣 D5E3	헣 D5F3	헟 D603	헡 D613	헢 D623	헣 D633	홉 D643	홉 D653
4	헟 D5D4	헞 D5E4	헡 D5F4	헞 D604	헡 D614	헢 D624	헣 D634	홉 D644	화 D654
5	헡 D5D5	헢 D5E5	헣 D5F5	헡 D605	헣 D615	헣 D625	헢 D635	홉 D645	화 D655

한글 유니코드의 일부

## 문자 자료형

- 자바의 문자 자료형 char
- 자바는 문자를 2바이트 유니코드로 표현한다.
- 작은 따옴표로 묶어서 하나의 문자를 표시한다.
- 문자의 저장은 유니코드 값의 저장으로 이어진다.

```
char ch1 = '헐';
```

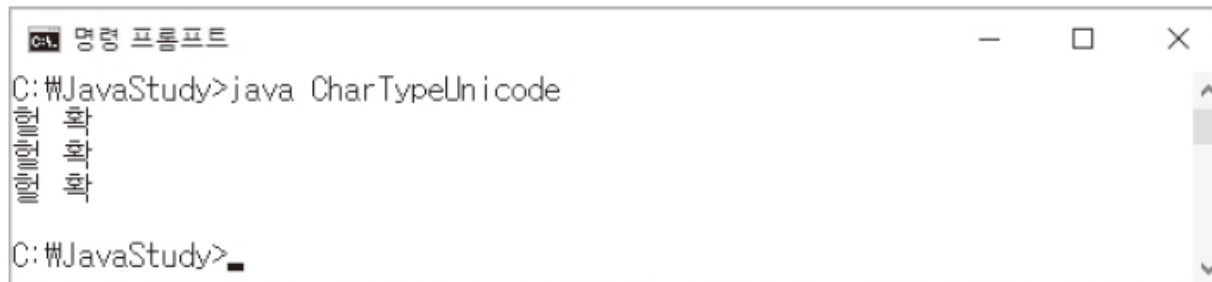
```
char ch2 = '혹';
```

문자 '헐'의 유니코드 값 D5D와 0의 조합 결과 D5D0

문자 '혹'의 유니코드 값 D65와 5의 조합 결과 D655

## ◆ CharTypeUnicode.java

```
1. class CharTypeUnicode {  
2.     public static void main(String[] args) {  
3.         char ch1 = '혈';  
4.         char ch2 = '확';  
5.         char ch3 = 54736;    // 문자 '혈'의 유니코드 값  
6.         char ch4 = 54869;    // 문자 '확'의 유니코드 값  
7.         char ch5 = 0xD5D0;  
8.         char ch6 = 0xD655;  
9.         System.out.println(ch1 + " " + ch2);  
10.        System.out.println(ch3 + " " + ch4);  
11.        System.out.println(ch5 + " " + ch6);  
12.    }  
13. }
```



```
C:\#JavaStudy>java CharTypeUnicode  
혈 확  
54736 54869  
D5D0 D655  
C:\#JavaStudy>
```

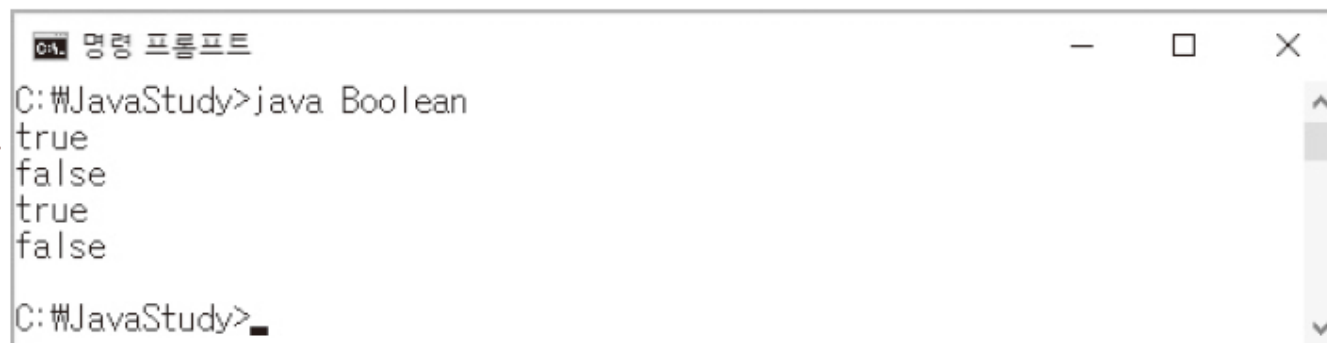
코드상에서 문자 표현의 예

## ◆ Boolean.java

```
1. class Boolean {  
2.     public static void main(String[] args) {  
3.         boolean b1 = true;  
4.         boolean b2 = false;  
5.         System.out.println(b1);    // b1이 지닌 값 출력  
6.         System.out.println(b2);  
7.  
8.         int num1 = 10;  
9.         int num2 = 20;  
10.        System.out.println(num1 < num2);  
11.        System.out.println(num1 > num2);  
12.    }  
13. }
```

true '참'을 의미하는 값

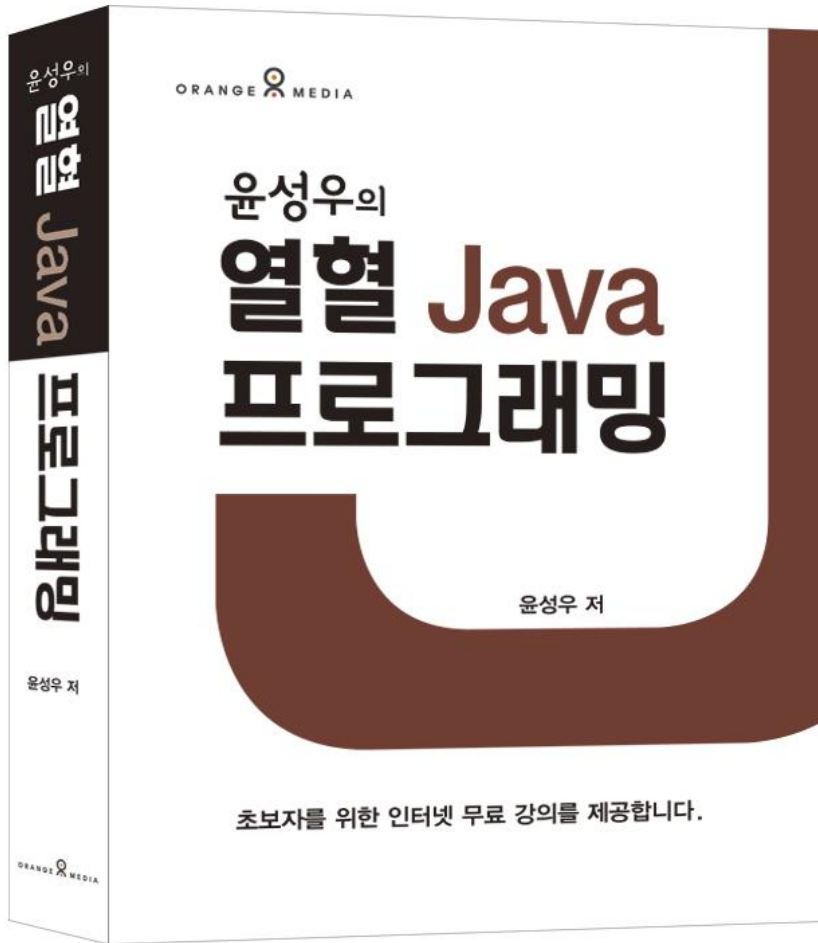
false '거짓'을 의미하는 값



명령 프롬프트

```
C:\JavaStudy>java Boolean  
true  
false  
true  
false  
  
C:\JavaStudy>_
```

논리 자료형



# 열혈 Java 프로그래밍

## Chapter 03. 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

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

```
final int MAX_SIZE = 100;
```

## ◆ 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>
```

## final 상수 선언의 예

---

# 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.

## ◆ 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. }
```

명령 프롬프트

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

## 정수형 상수(리터럴)의 표현 방법

## 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 L을 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 e 표기법

---

$$3.4e3 \quad \Rightarrow \quad 3.4 \times 10^3 = 3400.0$$

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



# 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'	개 행 문자
'\r'	캐리지 리턴(carriage return) 문자

화면상의 어떠한 상황 또는 상태를 표현하기 위해 약속된 문자

## ◆ 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  
CB  
C:\JavaStudy>
```

이스케이프 시퀀스의 예

## 03-2. 형 변환

## 자료형 변환의 의미와 필요한 이유는?

---

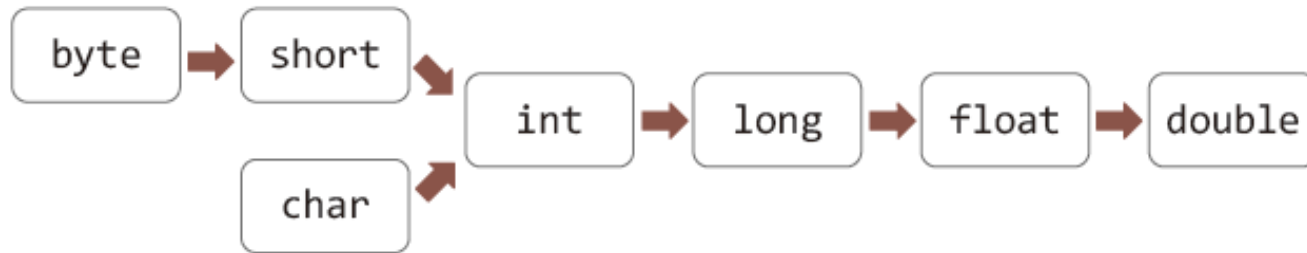
```
int num1 = 50;  
  
long num2 = 3147483647L;  
  
System.out.println(num1 + num2);
```

*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.

## 자동 형 변환(Implicit Conversion)

---



- 규칙 1. 자료형의 크기가 큰 방향으로 형 변환이 일어난다.
- 규칙 2. 자료형의 크기에 상관없이 정수 자료형보다 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);
```