

컴퓨터 프로그래밍

values, variables, Data Types

## 자료의 표현

- Program이 하는 일
  - data를 저장하고
  - 저장된 data를 읽어서 계산하고
  - 계산된 결과를 다시 저장
- Data의 표현 및 처리 방법이 중요

# value

- 두 가지 종류로 표현
  - constant & variable
  - 모든 값은 메모리에 저장
- constant (상수)
  - 주어진 값 (코드 상에 직접 작성)
  - 메모리에 저장되어 연산 후 즉시 소멸
  - e.g. 15, 193.45, 'a'
- variable (변수)
  - 변할 수 있는 값
  - 변수의 life cycle 동안 메모리에 상주

# constant

- 정수
  - 8진수, 10진수, 16진수
    - e.g. 021, 17, 0x11
- 실수
  - 3.14, 314e-2
- 문자
  - 'a', '+'

# variable

• `c = a + 27;`

상수: 코드 상에 직접 주어진 값

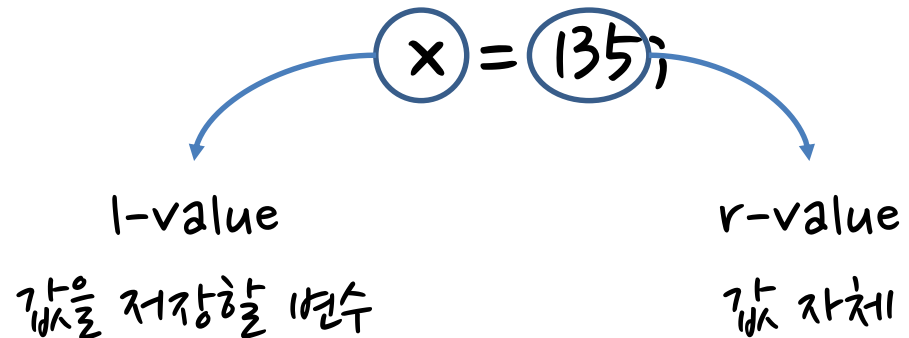
변수: 변할 수 있는 값, 변수 이름 필요

## • 변수 이름 규칙

- 영문, 숫자, `_`, `$` 로만 구성
- 첫 글자로 숫자 불가
- 대소문자 구별

# Assignment

- 변수에 값을 대입하려면, = 사용
  - $x = 135;$
- l-value vs. r-value
  - variable이 l-value이냐 r-value이냐에 따라 역할이 다름
  - l-value일 때는 저장하기
  - r-value일 때는 불러오기

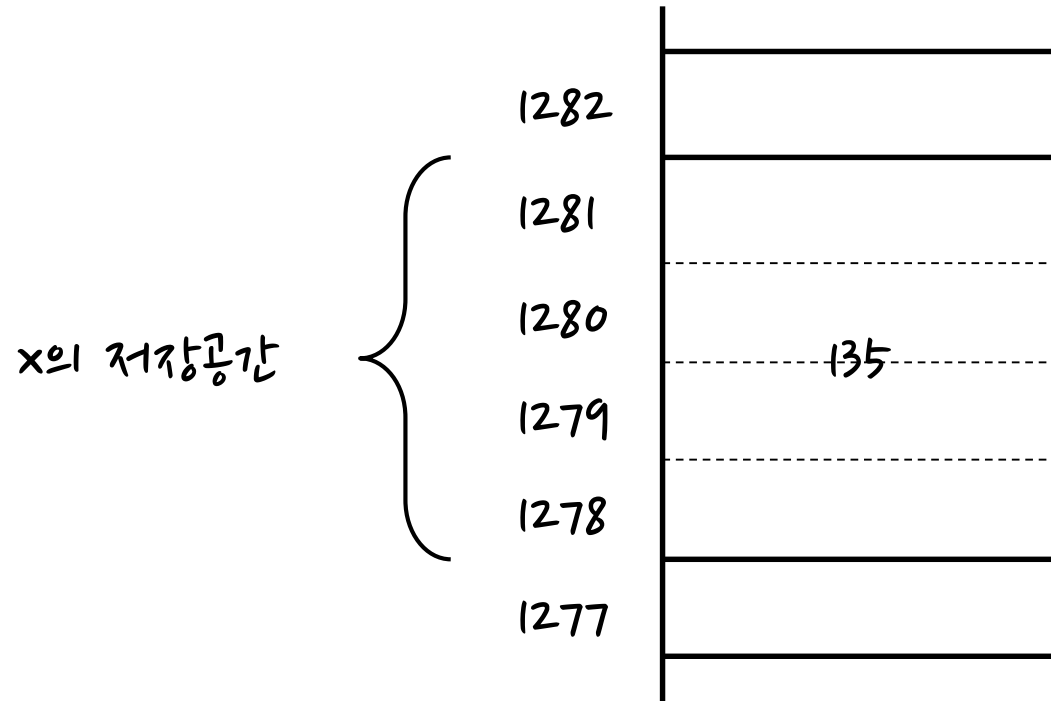


$a = x;$

$x$ 의 값을 읽어서,  
 $a$ 에 저장

## variable과 저장공간

- $x = 135;$



r-value에  $x$ 를 쓰면  $x$ 에 저장된 값을 불러온다

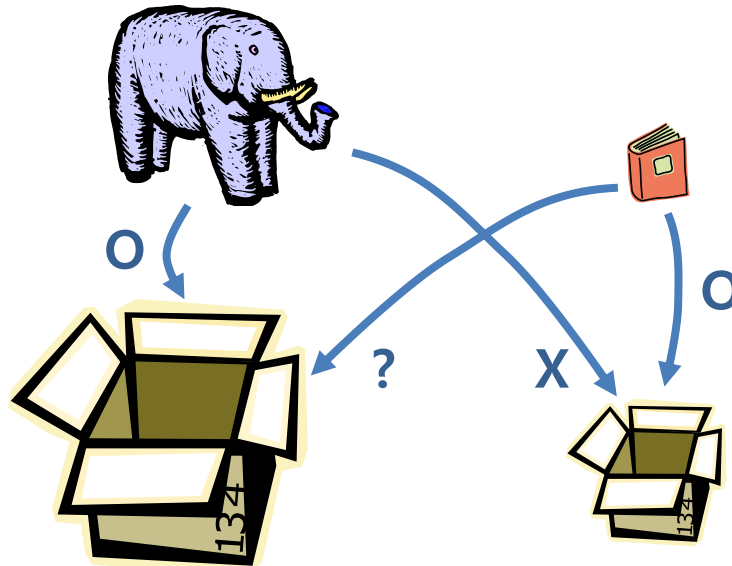
# Variable Declaration

- 코드에서 변수를 사용하려면 변수 선언(declaration) 필요
- 변수 선언을 통해 저장공간을 할당
- 변수 선언은
  - `Data_Type`[variable\_name\_list];
  - e.g. `int x, y, z;`



# Data Type

- 변수에 저장될 값의 표현 및 저장 방법을 명시
- 연산 방법 명시
- 종류
  - Primitive: 내부적으로 미리 정의되어 있는 type
  - User-defined: 사용자 정의 type



# Primitive Types in Java Language

자료형	데이터	메모리 크기	표현 가능 범위
boolean	참과 거짓	1 바이트	true, false
char	문자	2 바이트	모든 유니코드 문자
byte	정수	1 바이트	-128 ~ 127
short		2 바이트	-32768 ~ 32767
int		4 바이트	-2147483648 ~ 2147483647
long		8 바이트	-9223372036854775808 ~ 9223372036854775807
float	실수	4 바이트	$\pm(1.40 \times 10^{-45} \sim 3.40 \times 10^{38})$
double		8 바이트	$\pm(4.94 \times 10^{-324} \sim 1.79 \times 10^{308})$

# Number Expression

- Integer

- 2 or 4 bytes로 나타낼 수 있는 수의 개수는?

- 음수는 어떻게 표현?

- 1's complement

- 각 bit를 반전

- $3 - 2 = ?$

- 2's complement

- 1's complement + 1

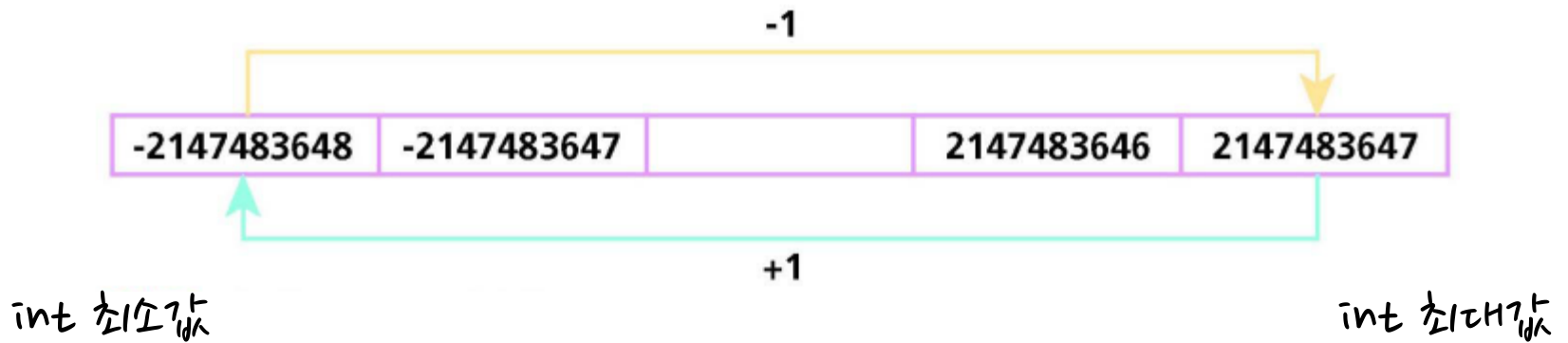
- 최상위 bit는 sign bit

숫자	1의 보수	2의 보수
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
0	0000	0000
-0	1111	-
-1	1110	1111
-2	1101	1110
-3	1100	1101
-4	1011	1100
-5	1010	1011
-6	1001	1010
-7	1000	1001
-8	-	1000

# Number Expression

- overflow & underflow

— 정수형 data type이 표현할 수 있는 범위를 초과하면?

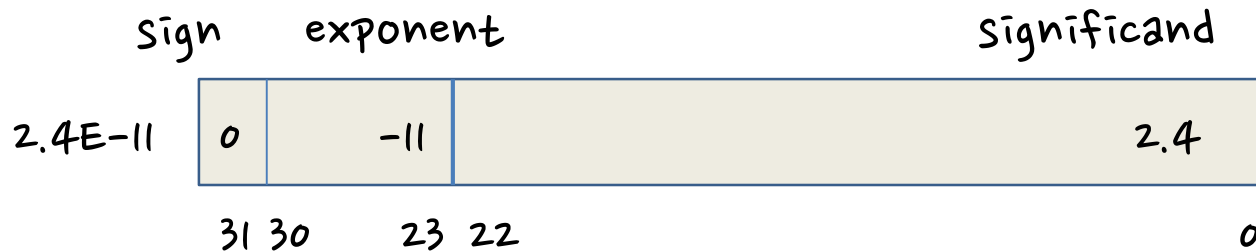


숫자	2의 보수
+7	0111
+6	0110
:	:
-7	1001
-8	1000

+1

# Number Expression

- Floating Point (float)



— e.g. 12.625

- Sign = 0
- $12.625 = 1100.101_2 = 1.100101 \times 2^3$
- Significand = 첫 1을 제외한 나머지 수 100101
- exponent = bias 127을 더한 130 =  $10000010_2$

# character Expression

- `char c;`  
`c = 'a';`
  - 문자형 변수 `c`에 소문자 `a`를 저장
- Unicode 사용
  - 문자를 2 bytes 정수에 맵핑한 코드
  - $2^{16}$ 개 문자 표현 가능

```
char ch1='A';
```

```
char ch2='한';
```

```
char ch1=65;           // 65는 16진수로 0x41
```

```
char ch2=54620;        // 54620은 16진수로 0xD55C
```

# character Expression

- Escape Sequence

- 특수문자 및 control 표현
- Backslash(\) + 특정 문자를 하나의 문자 혹은 control로 처리
- `c = '\n';`

이스케이프 시퀀스	의미
<code>\b</code>	backspace
<code>\t</code>	tab
<code>\n</code>	line feed
<code>\r</code>	carriage return
<code>\"</code>	double quotation
<code>'</code>	quotation
<code>\\</code>	backslash

# Logic Expression

- boolean b1, b2;

b1 = true;

b2 = false;

— true or false로 표현

System.out.println( 3 > 2 );

-> true 출력

- 조건문, 반복문의 수행 여부를 결정하기 위해 주로 사용

— e.g.

if (b1)

System.out.println("참");

while (b1)

System.out.println("무한 루프");



# variable Declaration / Initialization

- 변수 선언은

- `Data_Type [variable_Name_List];`

- e.g.

- `int a;`

- `int a = 10;`

- `int a, b, c;`

- `int a = 10, b, c;`

- Java가 사용하는 keyword는 변수명으로 사용 불가

boolean	if	interface	class	true
char	else	package	volatile	false
byte	final	switch	while	throws
float	private	case	return	native
void	protected	break	throw	implements
short	public	default	try	import
double	static	for	catch	synchronized
int	new	continue	finally	const
long	this	do	transient	enum
abstract	super	extends	instanceof	null

# Simple Usage

- int type 변수 a, b, c를 선언
- 변수 a에 5 대입
- 변수 b에 10 대입
- 변수 a와 b를 더한 값을 c에 저장
- 변수 c에 저장된 결과값을 출력

# Output Method

- `System.out.println(expression);`
  - `expression` 계산 값을 출력
  - e.g. `a = 10;`  
`System.out.println(a);`

# Input Method

- 입력 관련 라이브러리
  - `import java.util.Scanner;`
- Scanner 형 인스턴스 생성
  - `Scanner input = new Scanner(System.in);`
- 입력 받은 내용을 내용에 따라 적절한 메소드 사용
  - `int num;`  
`num = input.nextInt();`
  - `nextByte()` / `nextShort()` / `nextInt()` / `nextLong()`  
`nextFloat()` / `nextDouble()`  
`nextBoolean()`  
`next()` / `nextLine()`

# Input Method

```
import java.util.Scanner;

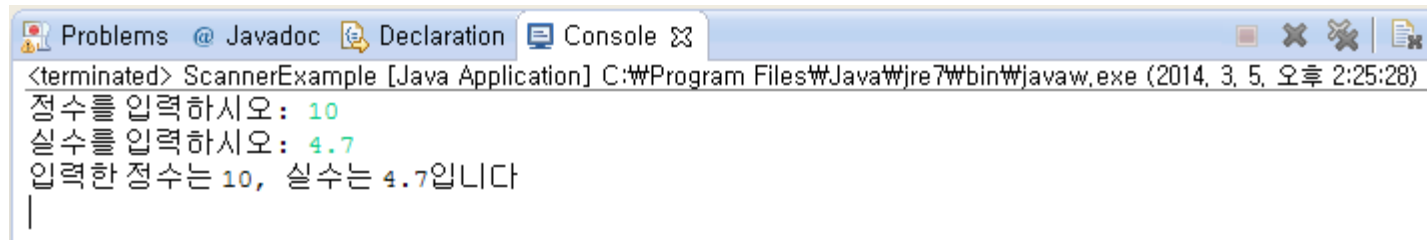
public class ScannerExample {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int numInt;
        float numFloat;

        Scanner input = new Scanner(System.in);

        System.out.print("정수를 입력하시오: ");
        numInt = input.nextInt();

        System.out.print("실수를 입력하시오: ");
        numFloat = input.nextFloat();

        System.out.println("입력한 정수는 "+numInt+", 실수는 "+numFloat+"입니다");
    }
}
```



The screenshot shows an IDE window with a tab labeled 'Console'. The console output is as follows:

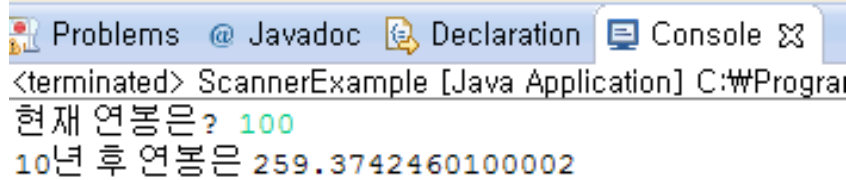
```
<terminated> ScannerExample [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (2014. 3. 5. 오후 2:25:28)
정수를 입력하시오: 10
실수를 입력하시오: 4.7
입력한 정수는 10, 실수는 4.7입니다
|
```

# Summary

- 값을 저장하기 위해서는 변수가 필요
- 변수를 사용하려면 변수 선언
  - Data type으로 변수의 크기 및 저장 방법 명시
- 대입 연산(=)을 통해 변수(l-value)에 원하는 값을 저장
- 변수 이름으로 해당 변수에 저장된 값 읽기 (r-value)
- 출력 메소드
  - `System.out.println()`
- 입력 메소드
  - Scanner 이용

시작

- 다음 내용을 프로그램으로 구현
  - 1년에 10% 연봉 인상을 해주는 회사를 다니고 있다면 10년 후 나의 연봉은?
    - 사용자로부터 현재 연봉을 입력 받고
    - 10년 후 연봉을 계산
    - `Math.pow(base, exp)` method 참고



The screenshot shows a Java IDE console window with the following text:

```
<terminated> ScannerExample [Java Application] C:\WProgra
현재 연봉은? 100
10년 후 연봉은 259.3742460100002
```

시작

- 영어 소문자 하나를 입력 받고, 입력한 영문자가 알파벳 순서상 몇 번째인지 출력하는 프로그램 구현

영어 소문자 하나를 입력하시오.

e  
입력한 문자 e는 5번째 알파벳입니다.

— 문자 입력 받기

- String s;  
char c;  
s = input.next();  
c = s.charAt(0);