Java

Hello! Object-Oriented Programming

DAY 1 객체지향 준비

Java

Java History

1995년 자바의 아버지라고 불리는 제임스 고슬링과 그의 동료들에 의해서 시작된 프로젝트다.

Java는 원래 소형 디바이스를 제어하기 위한 언어로 고안되었지만 웹의 등장으로 엄청난 성공을 거두면서 주류 언어가 되었다.

Characteristic of Java

- 객체지향언어 (Object - Oriented Programming)
- 바이트코드언어 (Bytecode Language)
- **Write Once, Run Anywhere!**

Java Keyword

Java Series

- Java ME
- Java SE
- Java EE

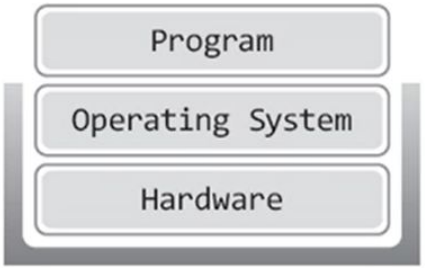
JDK

Compiler + Dev Tools + JRE..

JRE

JVM + Library + javaw..

Write Once, Run Anywhere



일반적인 프로그램의 실행구조

자바 프로그램의 실행구조

JVM (Java Virtual Machine)

JVM은 자바가 실제로 구동하는 환경이다. Java Application은 JVM이라는 가상화된 환경에서 구동된다.
Java Application과 OS 사이에서 중계자 역할을 한다.

Java 프로그램의 실행과정

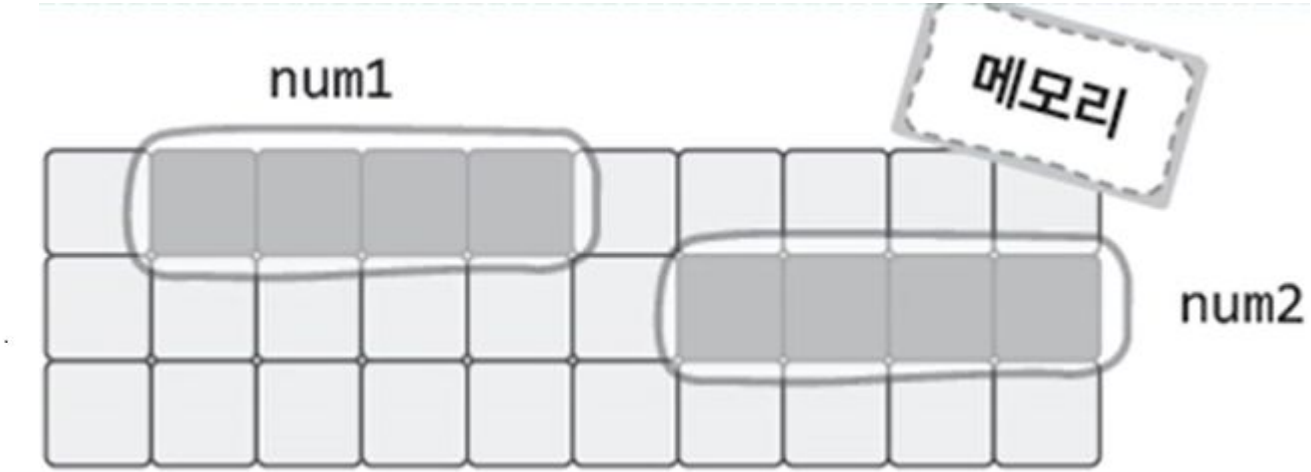
Hello World!

```java

public class HelloWorld {

    // Main Method (Entry Point)
    public static void main(String[] args){
        System.out.println("Hello Wolrd!");
    }

}

```

Entry Point

Statement (명령문)

Variable



int num1;

int num2;

Data Type (자료형)

Data Type

Java가 데이터를 표현하는 방법

정수 4의 int형 표현 : 00000000 00000000 00000000 00000100

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	$\pm(1.40\times10^{-45} \sim 3.40\times10^{38})$
double		8 바이트	$\pm(4.94\times10^{-324} \sim 1.79\times10^{308})$

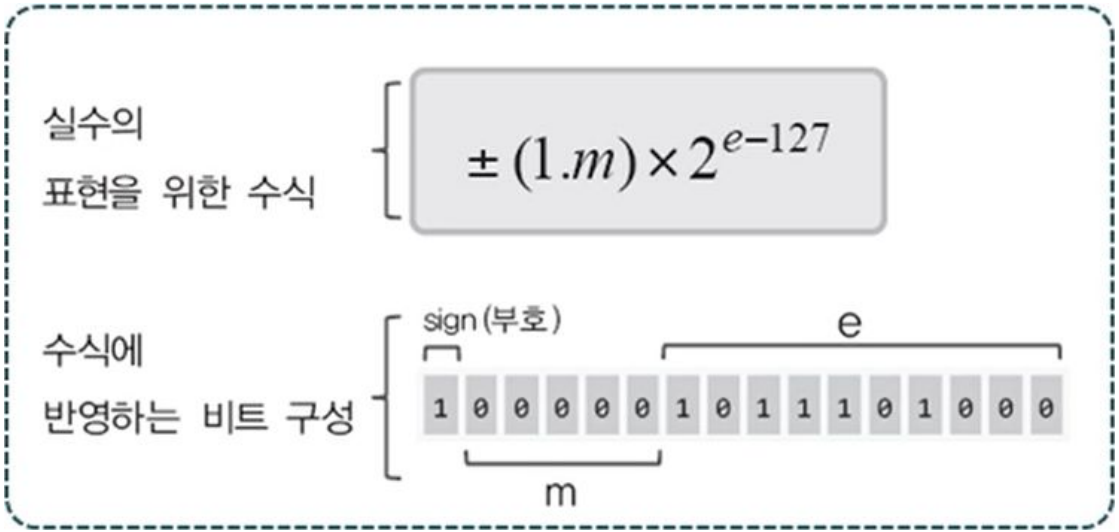
Expression of Integer And Real Number

Integer

- MSB
- 2의 보수 - 음의 정수

Real Number

- 부동소수점오차
- IEEE 754



Literal (Constant)

```
public static void main(String[] args) {

        int n1 = 10; // int 형을 근거로 표현

        int n2 = 5;

        double n3 = 3.14 + 2.0; // double 형을 근거로 표현

}
```

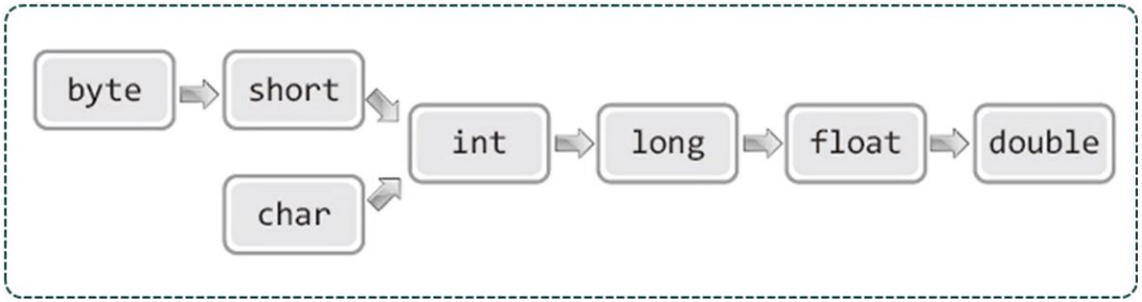
Literal
소스 코드내에서 직접 입력된 값, 변경이 불가능한 데이터
상수는 존재의미가 없어지면 다음행에서 바로 소멸

접미사 L, F

Data Type Casting (자료형 변환)

데이터의 표현을 새롭게 다시표현하는 것, 표현 방법을 완전히 뒤바꾸는 것

- **CPU의 연산 시 두 데이터의 자료형 일치**
- **Auto (implicit) Data Type Casting**
- **Auto (implicit) Data Type Casting Convention**



- **Explicit Data Type Casting - 형변환 연산자 (Type)**

Operator

- 단항, 이항 연산자
- 대입연산자 =
- 산술연산자 +, -, *, /, *
- 실수/실수, 정수/정수 나누기 연산의 차이
- 복합대입연산자
- 관계연산자 <, >, <=, >=, ==, !=
- 부호 연산자
- 증가, 감소 연산자 ++, --, Postfix, Prefix
- 논리연산자 &&, ||, !, SCE
- 비트연산자 &, |, ^, ~
- 비트 시프트 연산자 <<, >>, >>>

Flow Control - 조건문

- if, else
- if ~ else
- if ~ else if ~ else의 진실
- 삼항연산자 (조건문) ? data1 : data2
- switch~case + break, Label 묶기

Flow Control - 반복문

- while
- do~while
- for문의 분석
- continue
- break
- Infinite Loop
- 반복문의 중첩 - 구구단

Method

- 수학적 관점에서의 함수, 언어에 따라 메소드라고 부르기도 하고, 함수로 부르기도 한다.
  - f(x) = 3x + 2
  - e(x) = f(g(x)) + 1

- 메소드는 특정 기능의 코드 집합
- 메소드논 코드를 재사용 할 수 있게 해준다.
- 반환 (Return)
- 함수형언어 지향점은 함수(메소드)도 원자화 쪼갬
- 메소드 특징
  - 메소드 안에 메소드 정의 불가
  - 클래스 안에서 메소드 끼리 호출가능
  - Static 영역의 메소드는 클래스 자신 안에서 접근없이 함수명으로만 호출 가능

Method

기본 형식

ReturnType Name (Parameter) {

Body

}

- 메소드 중괄호 내에 존재하는 문장들은 순차적으로 실행
- 인자와 매개변수의 자료형은 일치해야 한다.
- 반환하는 데이터와 반환타입은 일치해야 한다.
- void, return;

Variable’s Scope (유효범위)

- 변수가 메모리 공간에 할당이 되었는데 절대 사라지지 않는다면 어떻게 될까?

- 중괄호 { }
- Local Variable
  - 중괄호 **{ }** 내에서 선언된 변수들은 **{ }** 영역이 종료되면 소멸된다. **(Method, if, while…..)**
  - 지역변수는 자기가 선언된 영역 내에서만 보인다**(Scope : 망원경)**
  - 지역이 다르면 지역변수에 접근을 할 수 없다.
  - 동일이름의 변수는 동일한 지역 내에서는 선언될 수 없으나,
    지역이 다르면 **Scope**가 다르기 때문에 이름이 중복되어도 상관없다.
- Global Variable - Member Variable