

Lecture 1: 철학

- What is Programming?

input, Output, math, decisions, repetition // Believe it or not. That's pretty much all there is to it.

- high-level language?

Java, C, C++, Python ... Before they can run, high level languages have to be translated into a **low-level language**, also called machine language

- ① high-level language dis advantage

- translation takes some time

- ② high-level language Advantage

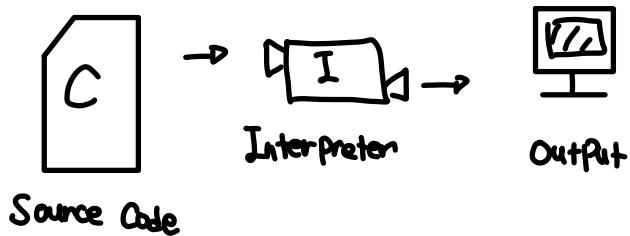
- much easier to program in high-level language. (less time to write, easier to read, more likely to be correct)

- Portable. meaning they can run on different kinds of computers with few or no modifications. → low-level programs can only run on one kind of computer and have to be rewritten to run on another.

Lecture 2 : 컴퓨터

- High level → low level
Translate by

Compiler & interpreters
 {
 Read line by line
 and performing Computations



[C 자세히]

Compiler: Code → 7기기어로 변환

전체 프로그램 일정 시작 전 완전히 번역.

소스 → 번역된 프로그램
 (high-level program)
 (Object code)

컴파일하는 번역이 번역하지 않아요

중간 interpreted Program 를 번역 시작

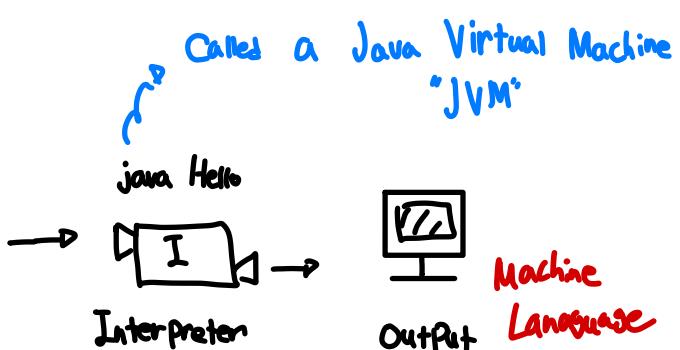
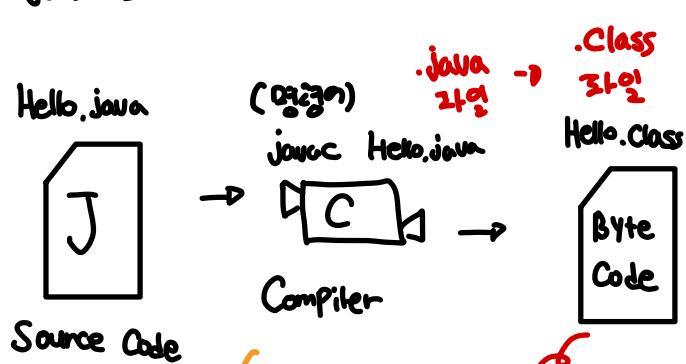
→ Interpreter: Code → 중간코드 → 실행마다
 실행마다

• 01정답 27. 번역방법(이설불리언)

7기기상으로 번역하는데
 사용장.

JAVA?

- Java is both Compiled and interpreted



Similar to machine language,
 byte Code is easy and fast to interpret

7기기에는 번역시
 HW설정.
 JVM은 SW설정

But, it is also portable!!

장점?

1. Platform 종속 X

2. Java 씨름 쓰는
 노안주제 장점이 컴퓨터 - 프로그램 사용
 될 수 있음 네트워크 상에서 연결

- 신뢰 X 프로그램 다운 설치 불가 -

Lecture 4 : Statement → Method → Class

- Java programs are made up of class and method definitions
methods are made up of statement.

Expression (식) : 값의 표현 가능, 조건문, if...else...
Statement은 Statement을 기반으로 Expression과
연결되는 구조다

- ①
- Statement : line of code that performs a basic operation.

ex) System.out.println("Hello world!"); // Print Statement

- Java 주의점 : Java는 구별 Case Sensitive입니다

- ②
- Method : Sequence of Statement. main()은 특별한 Method. 예 Class, Array - 2.

- ③
- Class : Collection of methods.

Java는 //가 주석

Class 디자인 : File Name = Class Name

클래스 이름은 파일 이름과 동일합니다

Lecture 5

- Escape Sequences :
 - \n : new line
 - \t : tab
 - \" : double quote
 - \\" : back slash

HTML 은 4종

324명의 Get.

Get Candidate Your Get

자신

6. 2 - Chap 2 Variables and Operators

변수

연산자

• Declaring Variables (변수 선언)

make named storage location

Update Value

Declare → Initialize
선언 초기화

• 변수명 규칙 (이름이 뭘인 약속)

① 2단어형?: firstName → 첫 단어 대문자하고 나머지 단어 대문자
player FirstName

② int x, y, .. ; 가능

③ Public, Class, Static, int .. 예약어는 불가능

Lecture 8.

print, println: Can display the value of a Variable using this

ex) System.out.println("first line is " + firstLine);

산술 연산자 조합연산자

• Arithmetic Operators (Operands)

int 옹, float 옹 연산자 7가지 있다.

정수 나누기는 항상 0으로 Rounds toward zero 된다.

Ex) 98이든 0.001이든 소수는 0이된다

정수 정수적 형 변환, 실수적 형 변환. → 어떤 게 이상하게 잘되 보니 잘 관리

• Rounding Errors: 원하는 숫자와 같은 부동 소수점 사이 차이 : 반올림 오류

① Most floating-point numbers are only Approximately Correct.

부동 소수점은

근사적으로 정확한

ex) Print ln(0.1 * 10) → 1.0

Print ln(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1)

Why? 10의 중간 부분인 0.1은 2의 반올림

{ 근사치 합산하여 반올림 오류 누적됨.

② 컴퓨터 그래픽, 암호화, 딜레이미디어 라이브리 등에서는 부동소수점 활용 빛과 큰 이점 있음.

↳ 다른 언어가 손쉬운 int 사용

문자열간 연산 가능 Java 원 → 문자 흑인

Print ln("Hello" + 1 + 2); 는 Hello 12
문자열로 생각 하면.
문자열은 가로() 된다!

1 + 2 + "Hello"
= 3Hello
"Hello" + 1 + 2
= Hello12

• About Error

• Compile time error, Runtime error, logic error

△ Compile-time error: Violate the Syntax Rules

- 1. Parsing : 번역기 전 프로세스를 일컫는 과정
(파싱, 구문 분석)
- 2. 조사 중 공정 탈의하거나 파일 끝이나 조밀하게 놓인 경우 Omitted
(忽略)
- { 정의하는 프로그램 중 일부 부분에 오류 발견 but 뉴락 부분은 그전에 풀어

▷ 예외적상황. 예외적인 일이 난거라
"exception"

- Run time error

: 프로그램 실행이 시작될 때까지 나온다거나 혹은.

Java에서 byte code를 실행하는 동안 발생

Ex) Exception in thread "main" java.lang.ArithmaticException: / by zero at Hello.main()

(ω) 예상치 못한 예외.

- logic error : 예상치 X, Compiled하는 데서 발견한 대로 험.

3. Input Output - Lecture 12 ~

Java의 println?

System.out의 System은 프로그램이 실행되는 환경과
관련된 방법을 제공하는 Class.

if System.out.println(System.out);?
→ result: java.io.PrintStream @685d72cd

↳ System.out은 Java.io라는 패키지의 정적인 PrintStream.

제작자는 관련 Class의 모음. Java.io는 입력/출력을 나타내는 "I/O" 클래스를 포함함

● 기본드입력: System.in이 있지만 사용하기 어렵고 다른 Class 사용함.

* Scanner Class : 단어, 숫자 및 기타 데이터 입력 방법을 제공하는 Class

Java.util에 의해 제공, 유동적인 클래스라고 불릴 정도로 유용한 클래스를 포함하는 패키지

△ Import Statement

① import java.util.Scanner; → Compile할 때 Scanner는 다른 패키지의 스캐너라는
정보를 알게됨. 다른 Class가 있을 수 있으니
구별해야하기 때문.

Class만기 있을 수 있음. 관례상 그림 사용

알보님에 존재

스캐너 만들기

o System Class는 왜
import X?

② Scanner in = new Scanner(System.in);
(declare variable name in) (System.in에서 입력을 가져오는 새 스캐너 생성)

java.lang Package가
자동포함. Java 프로그램
에서 사용하기 편리한 클래스
제공

String line;

line = in.nextLine(); 기본드 입력을 문자열로 받으려면 NextLine을 사용

System.out.println(line);」

• Program Structure

(de Sire)

Package : java.util - Collection of Classes . Method 정의
 Class
 Method
 Statement
 Expression
 Token

```

graph TD
    Package --> Class
    Class --> Method
    Method --> Statement
    Statement --> Expression
    Expression --> Token
  
```

Class : Scanner
 Method : nextInt - Contain Statements, Some of which Contain Expression
 Statement : hour = in.nextInt();
 Expression : hour * 60 - Made up of tokens.
 Token : hour - Basic elements of a program.

숫자, 문자 이름, 연산자, 키워드, 괄호, 시마줄론 ...

new? : 인스턴스(객체) 생성할 때 사용하는 코드.
+ 객체는가 실제 data가 아닌 참조값을 가진다는 내용 포함.

클래스	기본형 변수	= new 클래스();
자료형	참조값 저장 (인스턴스 변수)	DIagram 인스턴스 생성 참조값 2곳 (->기본형) 생성자 호출 C++, Java는 헐
Vector3	V = new Vector3(1,2,3);	Unit7 Vector3는 스택 자주 활용되는 경우 명시변수로 선언 사용. 별개로 가비지로 상성

Class: 12M12

Instance : 제1종

New Vector3()는
Class Vector3를
new3 메소드로 만들기

$H_A = H_E$ (State)

$\partial \chi^c = \partial \chi^I$ (behave)

```
Vector3 V = new Vector3(12,3);
```

$V \in \text{Vector}^3$ by def. \rightarrow Class의 인스턴스인 'Vector'라는 이름(객체지향 프로그래밍 - OOP)

Lecture - 13

- final variables - ex) final double CM_PER_INCH = 2.54;
Cannot be reassigned
once it has been initialized
Constants: 상수 가정
3
다 데려온거 _
놓아줄거 final로

. formatting output. - printf는 Formatter의 자유로운 활용 가능

(C랑 같은, System.out.printf(" %.2f %d, %f", 71, 22, 179.2);

Lecture - 14 : type cast - inch=(int)(3.24)=3, %(4%12)

Lecture - 15: Scanner bug

- Scanner.nextInt()는 개행문자(\n) 처리하지 않음.

개행문자 전까지만 숫자로 입력

↓

nextInt(); old nextLine(); 하면 빈문자를 return.

빈문자 "/n"가 있어?

답: 1. 초기 '/n' 처리를 빈 Scanner.nextLine();

추가

반환

값이 빈거나

2. 숫자를 Integer.parseInt(Scanner.nextLine());

으로 넣기

Lecture 16

ecture 16
java.lang package - Math Class π (sin, PI, log,
 $\text{Math.Pow}(\text{double } x, \text{double } y)$ round, cos, exp...)
 $= x^y$

이중 관계: Class: 각 단어 시작 대문자

Method: 칭찬의 수준, 이득 단어 사용 체계화 (Cancel case)

① Public ② Static ③ Void ④ main (String [] args)

०

public: it can be invoked from other classes

(Class가 public이면 다른 Package에서 사용 가능 Class임)

Private : 오직 해당 클래스의 山부 메서드만 접근 가능
[접근권한] (상속된 하위클래스도 불가. 제외 강력한 접근제어)

Class의 인스턴스는 항상 Private 여야 함.

Protected : 상속 관계에 있는 자식 클래스만 사용 가능.

같은 Package의 다른 Class와 SubClass
에서도 접근 가능

default : 디폴트 default, 흥미로운 클래스가 있는 클래스의
모든 디폴트가 접근 가능

② Static은 나중에

③ Return 값, Void, C강 등원

④ 디버깅, 미세조는 인수를 저장.

main은 args이름의 String[] 타입을 디버깅.

→ 문자열 배열 형태로 (주석 설명)

Lecture 17. 그냥 디버깅 얘기 ~~ 단순한거.

영어와 이상해 보일뿐

• 전역, 지역 변수 얘기

Lecture 18. ① 디버깅 예제가 7개 ~~ 디버깅 예제

② 스택 그림 (Stack diagram) ~~

별도 이름
준아도 실수하는
경우 다른거

Javadoc: 내가 쓴 설명서

* 2개는 JavaDoc으로
설명할 수 있는
주석이라는 거.

/ * Start

여러줄 주석:
한줄 주석 : //

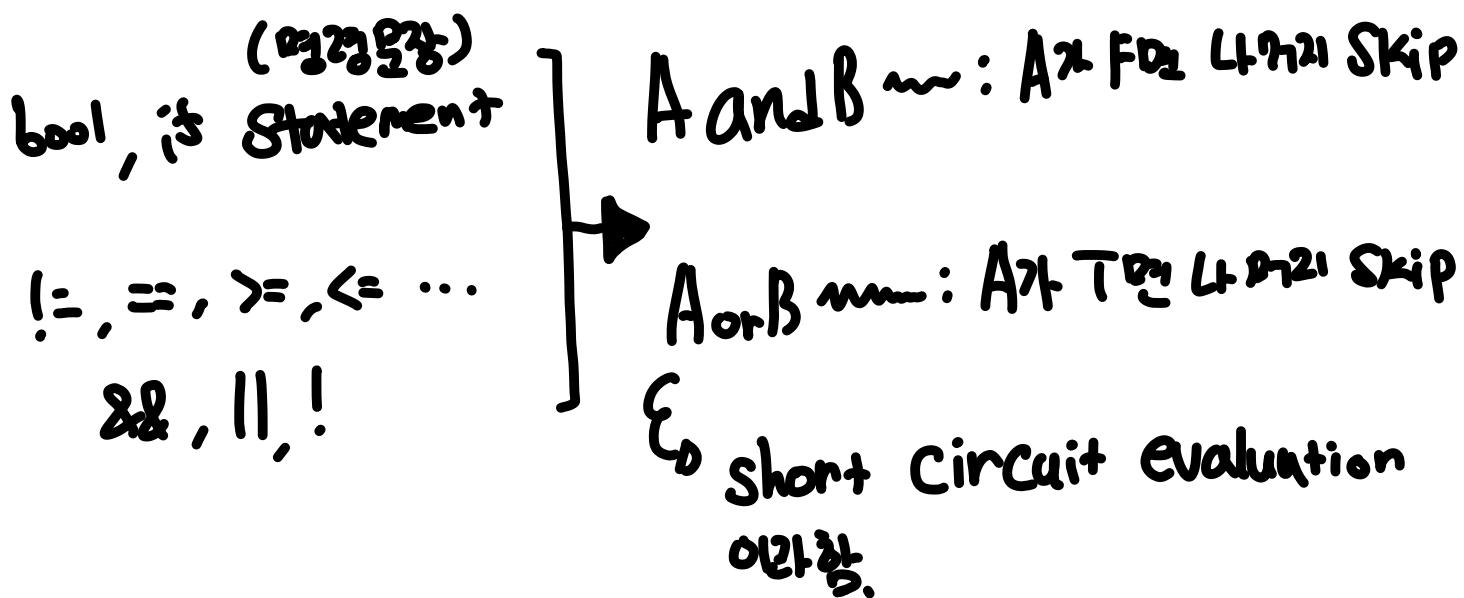
/ End

↑ 주석하고 Javadoc을 만들면
그때마다 주석이 다 정리된 표로
나온다.

Lecture 19 - [Ch.5 - Conditionals and Logic]

(조건)

(논리)



• 드로잉 논리

$$!(A \&\& B) = !A \parallel !B$$

) 단순. 주제상수

$$!(A \parallel B) = !A \&\& !B$$

if는 if, else, elseif, else ..

☞ String의 ==, != 는 같은 대상이 의도대로 안된다.

Why?

$A == B$, A, B는 String은 A, B가 참조번호와

참조값이 비교됨. String은 아니야.

- Lecture 20 -

Chaining: if, elseif, else 한줄인가

Nesting : if 안에 if 또 있는 것

if ()
{ -- }
if ()
{ - : }

Flag Variables : 조건이 true가 될 때까지 하나의 값을 가지

도록 정의하는 변수

예시?) boolean evenFlag = (n%2 == 0);

boolean positiveFlag = (x > 0);

↳ 이 변수는 조건을 끝나는 Flag.

Why? 어떤 조건은 유무에 대한 신호를 보내고 싶기 때문

(terminate)

• Return : 함수의 끝나는 지점에 조건을 중료할 수 있도록 한다

예시) 예제 7-7

public static void printLogarithm(double x){

if (x < 0.0){

(System.out.println("ERROR: x must be positive")

return; } OutputStream original + warning

double result = Math.log(x);

으로 자주 쓰이거나,

System.out.println("The log of x is " + result);

,

Validating Input (입력 검증)

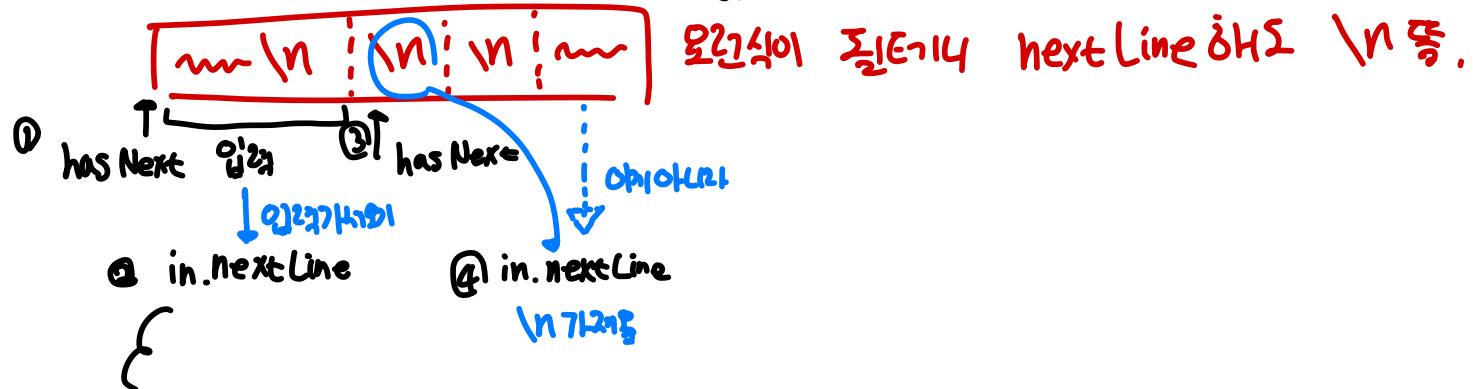
```
Public static void scanDouble(){  
    Scanner in = new Scanner(System.in);  
    System.out.print ("Enter a Number");  
    if (!in.hasNextDouble()) {  
        String word = in.next();  
        System.err.println(word + " is not a number");  
        return;  
    }  
    double x = in.nextDouble(); // Double을 문자열 입력과 가나눠 사용  
    Print Logarithm (x);  
}
```

① hasNext() : 입력된 토큰이 있으면 true
또는 새로운 입력이 있을 때까지 무한정 기다리고 입력되면
true.

hasNextDouble : 입력된 토큰이 Double이면 true
또는 새로운 입력까지 무한정 기다리고 입력하면 true.

* 신기한점 : "\n"은 Null이라 입력으로 인식하지 않는가 같음.
but nextLine은 \n를 포함한 라인을 옮겨 값을 내보내니
새로운 입력 기다리는 중에 그냥 Enter 키면
나중에 in.nextLine() 쪽으로 보내 "\n"만 드.

즉 hasNext + @() 조이 인터넷으로
Double, Int, Long, String, 문자 받아오는거



nextLine() 등의 입력과 같은,

nextInt()

:

이미 입력된 문자가 있으면

인식 된다는 거지.

(다른 예시)

```
package com.tutorialspoint;

import java.util.*;
import java.util.regex.Pattern;

public class ScannerDemo {

    public static void main(String[] args) {
        String s = "Hello World! 3+3.0=6";

        // create a new scanner with the specified String Object
        Scanner scanner = new Scanner(s);

        // check if the scanner has a token
        System.out.println("+" + scanner.hasNext());

        // print the rest of the string
        System.out.println("+" + scanner.nextLine());

        // check if the scanner has a token after printing the line
        System.out.println("+" + scanner.hasNext());

        // close the scanner
        scanner.close();
    }
}
```

Try it

↳ 더 이상 토큰이 남은가 X

- 출력 -

true

Hello World! 3+3.0=6

false

- Lecture 21 - Recursive Method (재귀)

- Stack Over flow Error: 자료형의 용량이 아니라 손상의 소작률과

- Binary Numbers (이진수) 흘리기

$$\begin{array}{l} 23/2 = 11 \dots 1 \\ 11/2 = 5 \dots 1 \\ 5/2 = 2 \dots 1 \\ 2/2 = 1 \dots 0 \\ 1/2 = 0 \dots 1 \end{array}$$

이진수 $11101 = 23$

Recursively 흘리기 가능

- Lecture 22 - [Ch6. Value Methods]

그냥 return val 타입. 사용은 ~~byVal~~ = Value Method
on

$\log(\text{height})$
↳ 흘리기.

- dead Code : 이미 알았던 혹은 return이 된거나
정교실방법이 없는 미지로는 코드.

- Incremental Development : 몇 줄마다 이미 X. 중간중간
(7장전체) 개발) 확인하라는 듯

-Lecture 23. Method 작성법

~~7(k)32, 알겠지만 네트 ~

- **Self scaffolding:** 코드의 작은 Stub(도약)을 Print으로 놓아서 확인하고자 하는 코드. 초기작업 틀이나는 살지.

~항상 가능 잘하기 잘하기 좋음. 낮은 error, 높은 유지보수. ~

2

-Lecture 24-

• Overloading (실습간이소설정)

- Boolean Method: 그냥 return 값 boolean

- Javadoc Tags: 주석중에 **@Param** 으로 Tag 설정
/*
 *
 * @Param width 흐

* @Param height 흐이

* @Return 계산한 폴더

*/

실습(week 5) 정리.

• 연산자 · if, else, · Switch · for, while - do-while

→ 이 실습은 필요한 것만 정리

• Switch

ex)

```
int a, b, c; // C=a Operator B  
Char Operator = '+'
```

Switch (Operator)

{

Case '+':

```
C=a+b;
```

```
break;
```

Case '-':

```
C=a-b;
```

```
break;
```

default :

```
System.out.println (" +,- 연산자 아님.");
```

```
System.exit(0);
```

}

```
System.out.println("a+b는 "+C+"이다");
```

for, while, do while, Switch 등등 ... C++ 중일

실습 Week 6.

1. 객체지향의 정의와 주요개념

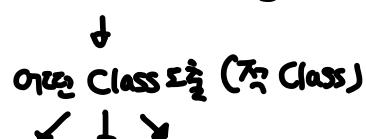
① 객체지향의 정의

□ **객체지향 프로그램** : 객체와 객체 간의 상호작용을 이용해 어려운 계이션을 개발하는 프로그램 paradigm.(틀, 체계)

□ **객체?**

- 클래스를 이용해 구체화된 모델
- 속성이 구체적으로 현실화된 모델

ex) 클래스와 객체의 관계에서 객체의 공통 속성과 기능을 정점 추상화



적 Class의 속성과 기능을 물려받아 가로, 슬라이드 등 좀 더 구체화된 Class 만들 수 있음.

□ **객체지향언어 기술적 특성** : 갑수화, 다형성, 상속성, .. etc...

□ **객체를 생성하기 위해 Class를 사용한다.** (형식, 전달)

- Class는 추상적 데이터 타입으로 객체를 정의하는 템플릿(Template)이 됨.

▣ 초기지향의 정의

* Tip

Class = 설계도

변수 = 상태 (state)

Instance: 제품

메소드 = 행동 (behave)

▣ 인스턴스화 (Instance)

Class가 구현되었을 때면 초기화 됨.
(객체지향 OOP)

- 프로그램에서 Class를 이용해 객체를 생성하는 행위



- New 키워드 사용 : 객체 변수가 실제 data가 아닌 참조 값을 가진다는 내용 포함

[New 키워드를 이용한 Class 인스턴스 생성방법]

C++, Java : heap

Class 명 참조변수명 = new Class 생성자 Unity와 Vector3는 Stack.
클래스 초기값 메모리 할당 생성자 호출 보통함 but 같은 Vector3 같은 예외
 (인스턴스 만들기) 인스턴스 생성 참조값 리턴 (→ 객체)

예시) Vector3 v = new Vector3(1,1,1);

[자전거 객체를 생성하는 초기지향 프로그래밍 개발]

[클래스정] 자전거	
[속성]	
이름, ID	비주얼
종류	etc ...
[메서드]	
종류 설정();	
이름 설정();	
:	

차일드 200D, 부자금 300 ...

다른 제작스튜디오, ...

객체생성

•
•
•

상속

상속

[클래스정] 상속용 자전거	
[속성]	
:	
:	
[메서드]	
:	
:	
:	

[클래스정] 하이브리드 자전거	
[속성]	
:	
:	
[메서드]	
:	
:	
:	

▣ 필드 (Field)

□ 개체를 구분하는 특성을 저장하는 변수. Data Member라고도 한다.

□ 인스턴스 변수

- 클래스를 객체화 시킬 때 객체의 고유특성을 저장하는 변수
- 객체 생성 시 독립적인 메모리 공간을 확보한다.
- 자신이 포함된 객체에서만 값이 유효하다.

□ 클래스 변수

- 각 클래스에서 오직 하나의 메모리 공간만 갖는 변수
- 클래스에서 인스턴스는 모든 개체는 클래스 변수를 공유할 수 있다.
- Static 키워드로 선언됨.
- 모든 인스턴스가 값을 공유해 한쪽에서 값을 바꾸면 모든 개체에서 바뀐 값을 참조함.

- 추가 설명 + 예시 -

Public Class test { **생성자 (메모리 할당자)**

선언위치 int iv; //인스턴스 변수 : 인스턴스가 생성될 때, 인스턴스별로 다른 값 가능.

클래스 정의 Static int CV; // 클래스 변수 클래스가 메모리에 올라갈 때 (메모리에 딱 한 번만 올라감)
한 클래스의 모든 인스턴스가 공유된 값 공유

Void method() {

 int lv; // 지역 변수

}

선언위치 : 클래스 멤버영역

(메소드, 생성자, 초기화 블록)

Public을 넣으면 같은 프로그램 내 어디서든 접근 가능한 '전역 변수'가 됨.

+ 인스턴스 변수와 접근법과 다르게 인스턴스를 생성하지 않고

클래스 이름, 클래스 변수명을 통해서 접근 가능

▣ 필드 (Field)

□ 접근 한정자

Static (접근 한정자) (데이터 타입) (변수명) // 클래스 내부

- 외부에서 필드에 접근을 허용할 것인지 결정하는 키워드

- 클래스, 클래스를 구성하는 메서드에서도 동일하게 적용

Public: it can invoked from other classes

(Class가 public이면 다른 Package 안 사용 가능 Class는)

Private: 오직 해당 클래스의 일부 메서드만 접근 가능

Protected: (상속된 하위클래스도 불가. 상위 간격은 접근제어)

Class의 인스턴스는 항상 Private 여야 함.

Protected: 상속관계에 있는 자식 클래스만 사용 가능.

같은 Package의 클래스와 다른 Package의 SubClass
에서도 접근 가능

default: DRR과 default, 동일한 package에 있는 클래스의

모든 다른 메서드 접근 가능

」

■ 메서드 (Method)

□ 클래스 내부 속성을 이용해 특정한 일을 수행. 외부 객체와 인터페이스를 제공

□ 인스턴스 메서드: 항상 객체에 포함되어 있으며, 객체를 통하여 호출할 수 있음.

□ 클래스 메서드 or 정적 메서드 (Static Method)

- 클래스의 객체가 없어도 실행할 수 있다.

- 클래스 변수와 마찬가지로 객체의 선언 없이 클래스 명을 이용해 접근이 용이

[클래스 메서드 선언방법]

Static (타입) (메서드명) (매개변수 ...) {

DongJung ~

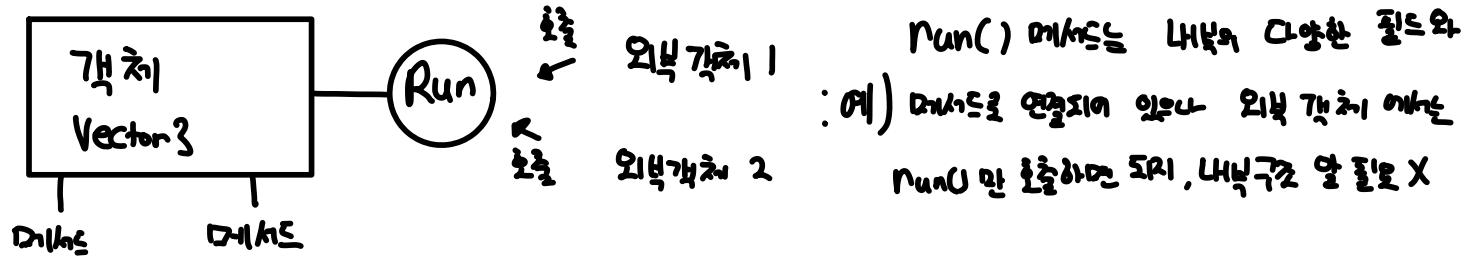
return (반환변수)

2. 개체지향의 특징

▣ 캡슐화

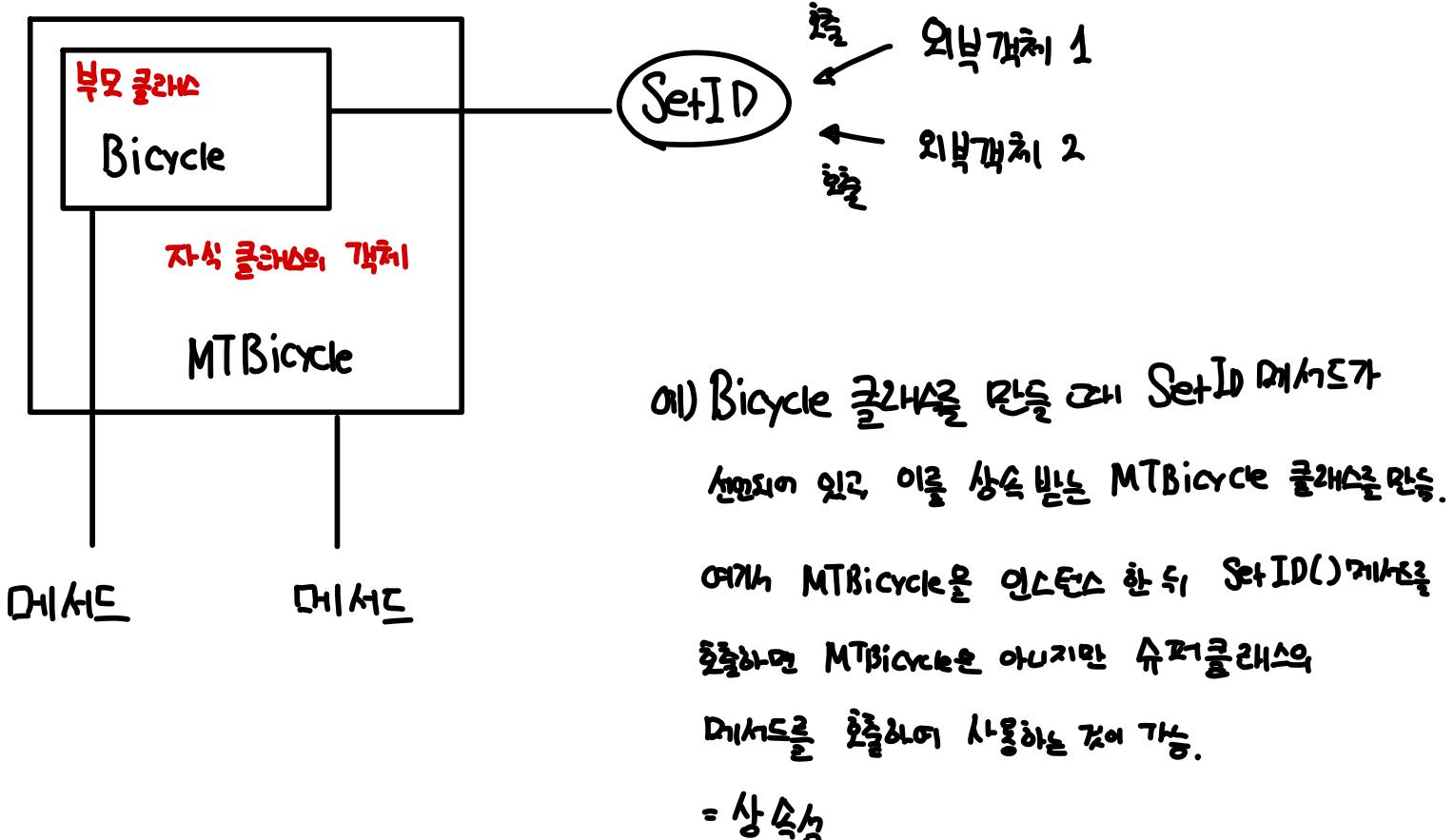
□ 생성한 개체를 어떤 메서드와 풀드로 어떻게 일을 수행할지 외부에 숨기는 기능

□ 외부에 노출한 인터페이스만 이용해 자신을 접근할 수 있도록 한다.



▣ 상속성

□ 슈퍼 클래스의 코드를 서브 클래스가 사용할 수 있도록 해주는 개체지향의 대표적 특성

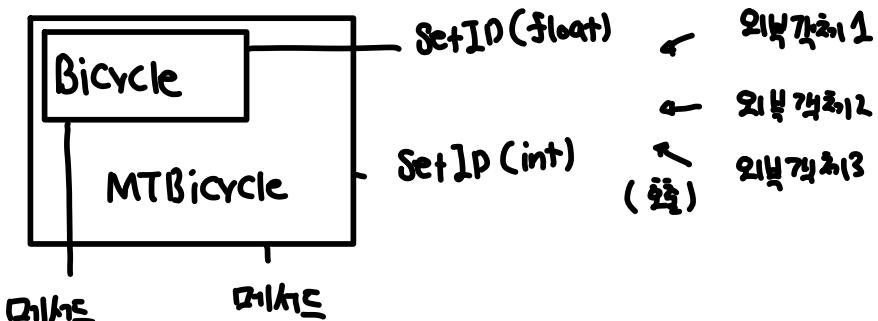


④ 다형성

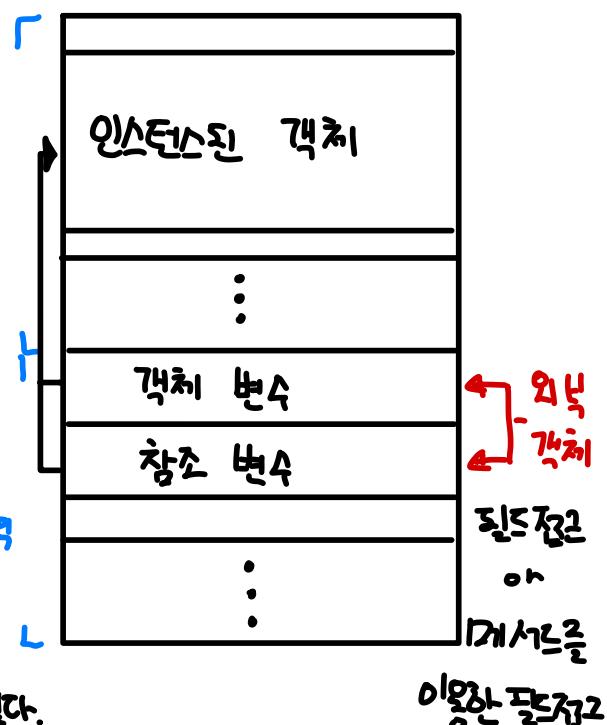
□ 클래스 내 메소드명과 슈퍼클래스와 서브클래스 간의 메소드명을 매개 변수의 형태에 따라 다르게 인지할 수 있도록 한다.

□ 동일한 이름으로 다른 데이터 타입의 메소드를 선언할 수 있도록 지원하는 것

□ 오버라이딩: 클래스를 상속할 때 슈퍼클래스에서 정의된 메소드를 재구현하는 기법 (독이 추가 설명)



[참조 변수를 이용한 객체 접근]



String name = "홍길동";

↓

문법적으로는 맞지만 Java에서 문자열은 일반 변수처럼 사용한 형태

String name = new String("홍길동");

↓

Class 관점에서 보면 위와 같이 선언되어야 함.

- 실제로는 왼쪽처럼 사용해도 오른쪽처럼 선언되어 사용됨.

④ 생성자 - 클래스 객체 변수 New 클래스(): 생성자를 호출하는 의미

- 클래스의 인스턴스를 생성할 때 반드시 호출되고 최초로 실행되는 일종의 메서드 (하지만 줄 나옴)
- 객체를 만들 때 자동으로 호출됨.
- 보통 인스턴스 변수를 초기화하는 목적으로 사용
- 반환 타입을 명시하지 못하고, 메서드명과 클래스명 동일함.
- 생성자의 한정자로 올 수 있는 키워드: abstract, final, synchronized

[선언 예시] 앞에 접근제어자만 올 수 있음

(줄고 한정자) (생성자명: 클래스명) (마지막) {
 명령문
}

3 Return 값이 있음. Void나 자료형 Return 값. 3
 메서드와 한정자.

```
Public Bicycle (int Color, int weight){  
    this.Color = Color;  
    this.Weight = weight;
```

⑤ 상속

- 부모클래스의 메서드나 필드를 서브클래스가 다른 선언없이 상속 받아 사용할 수 있도록 하는 기능

[상속을 위한 키워드]

상속 서브클래스를 지정하는 키워드

(줄고 한정자) Class (서브클래스명) extends (부모클래스명) {

필드

메서드

3

- 오버라이드 한 클래스 같은 이름 메서드 관계 -

메서드 이름: 동일

매개변수, 단입: 다른

Return 단입: 상관 없음

⑥ 메서드 오버로딩: 객체지향언어의 특징인 다형성의 대표적인 예

- 클래스에게 이름이 동일한 메서드를 사용할 수 있도록 지원하는 기능

SetColor (int Color)

SetColor (String Color)

} → 매개변수로 구별하기. 가능유사할 때 메서드 이름 짜고
 한동작하기 하려고

▣ 메서드 오버라이딩

□ 슈퍼클래스에 정의된 메서드를 서브 클래스에서 재정의 하는 것.

□ 원래 슈퍼 클래스에서 만들어진 메서드 대신 서브 클래스에 있는 메서드 호출하는 기능

예시)

```
Public Class Test {  
String name;  
  
Public Void prtName() {  
System.out.println("이름 : " + name);  
}  
}
```

- 오버라이딩 슈퍼클래스 메서드와 서브클래스 메서드 관계-

메서드 이름 · 동일
파라미터, 타입 : 동일
리턴 타입 : 동일

재정의

```
Public Class SubTest extends test {
```

```
Public Void prtName() {
```

```
System.out.println("Name: " + name + "입니다.");  
}
```

```
Public Static Void main (String[] args) {
```

```
Subtest S = new Subtest();
```

```
S.name = "홍길동";
```

```
S.prtName();
```

}

오버라이딩은 서브클래스에

기존 디렉트의 명령을만

이루어진다. Name : 홍길동입니다.

변경 가능할 것.

if, 서브 클래스의 오버라이딩으로 가려면

슈퍼클래스의 디렉트나 변수를 사용하고 싶다면?

Super라는 예약어를 앞에 붙이면 끝.

Super.prtName();

Super(변수 + a);

{ } 뒤 중괄호로 나중에나마 일단 잊다듬을 알아두자.

▣ 추상화 클래스 : 기존 클래스 기능 유지하면서 기능 확장, 인터페이스 빠짐 때 사용하는 클래스.

기본기능을 각각으로 다룰 수 있도록 하거나, 해당 자료형에게 필요한 기능 등 손쉽게 사용하거나 다양한 방법으로 클래스 사용

예시) Integer.parseInt("3") = 3. 모드가?

실습 7- 추상 클래스 · 메서드, 인터페이스

1. 추상 클래스와 메서드

- ▣ 다양한 프레임워크에 기반을 둔 프로그램을 개발하는 대규모 시스템에서 주로 사용
- ▣ 정해진 둘이 따라 프로그램을 개발하는 기법적 방법
- ▣ 추상 클래스 : 추상 메서드를 한 개 이상 포함하는 클래스

▣ 추상 메서드

- 메서드의 이름만 있고 실행코드가 없는 메소드
 - 실행코드 부분 작성 X

[추상 클래스와 추상 메서드를 선언하는 방법]

Abstract 란?

개체지향 언어에서 클래스를 이용해 객체를 만드는 일은 인스턴스화라 함.
자身 추상화는 인스턴스화가 되면 안되는 듯
클래스 앞에 abstract 키워드를 붙이면 된다.
Abstract = 인스턴스화 금지 키워드

```
Abstract Class [클래스명]{  
    Abstract public void SetID(byte by ID);  
}
```

↳ 전자 () ; 끝나.
메서드가.
실행코드 부분 작성 X

□ 추상클래스의 구조적인 특징

- abstract 키워드를 추가
- 하나 이상의 추상 메서드가 존재
- 추상 메서드 외에 구현된 일반 메서드나 변수도 존재
- 추상 메서드 구현 내용은 존재하지 않음
- 서브 클래스를 만들어 추상 메서드를 오버라이딩 후 사용
- Private 접근 한정자 사용 불가

2. 추상클래스와 상속

- 사용 목적 : 상속과 실체화 사이에서 발생하는 논리적인 문제를 해결하기 위함.

- 수퍼클래스에서 필요한 메서드만 정의한 채 구현만 따로 하는 것이 클래스를 설계하고 프로그램을 개발하는데 훨씬 도움이 되는 접근방법.
- 수퍼 클래스에 구현을 다해 놓으면 서브 클래스가 그대로 쓰는 문제가 있음.

3. 인터페이스

■ 인터페이스란?

□ 의미상으로 떨어져 있는 객체를 서로 연결해주는 구조를 말함

□ 인터페이스는 개념이나 구조적으로 추상클래스와 유사함.

□ 특징

• 구현된 메소드가 없다.

• 초기값이 할당된 상수형 변수만 사용가능 (public, static, final 키워드를 자동으로 갖는 효과)

• 모든 메소드는 메소드명과 매개변수, 반환 타입만 있는 추상 메서드로 정의

• 메소드의 접근 한정자는 항상 public 이어야 함.

[인터페이스 선언하고 구현하는 방법]

인터페이스 선언 키워드

Public interface [인터페이스명] {

상수형 변수

Abstract Public [반환타입] [메소드명] (매개변수);

}

인터페이스 사용시 키워드

Public Class [자식클래스명] implements [인터페이스명] {

필드

메소드

}

모든 메소드는 Abstract Public. 생략 가능

□ 구조적 특징

• 인터페이스를 선언시 interface 키워드 사용

• 인터페이스명 뒤에 I로 시작

• public static final로 선언된 변수에는 상수 가능
(public static final 키워드 없어도 자동으로 컴파일됨)

• 반드시 추상 메소드만 정의 해야 함

• 인터페이스 사용하여면 구현(implements)
이라는 키워드 사용

4. 인터페이스의 구현 - 다중상속

■ Java에서 다중상속을 지원하지 않음.

- 다중상속은 프로그램을 설계할 때 많은 난이도
- Java에서는 다중상속의 해결책은 상속과 인터페이스 구현을 동시에 이용하여 해결한다.
- 하다의 implements 키워드를 사용후 각각의 인터페이스를 ';'를 이용해 나열

[다중인터페이스 구현방법]

```
public class [자식클래스명] extends [부모클래스] implements [인터페이스1], [인터페이스2], ...  
{  
    필드  
    메서드  
}
```

■ 상속의 개념의 존재

- 수록한 인터페이스를 상속하는 새로운 인터페이스를 정의할 수 있음
- 인터페이스의 계층구조 활용 : 더욱 정밀한 인터페이스와 클래스 설계 가능
- 인터페이스의 상속은 수록한 인터페이스의 구현이 아닌 상속의 개념

interface X

↑상속

interface Y extends X

↑상속

Class Z extends Y

■ 인터페이스 추가 설명

- 사용이유 : ① 완벽한 추상화

② 인터페이스로 다중상속 지원

③ 인터페이스의 사용은 약한 결합을 만들 수 있음

④ 다형성 사용가능

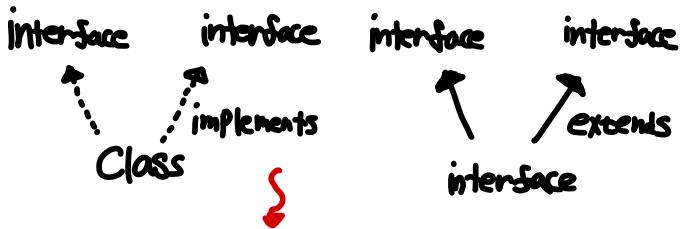
[Class · interface 관계]

Class interface interface

extends ↑ implements ↑ extends

Class Class interface

[다중상속 (Multiple Inheritance) in Java]



왜 Class는 지원 X?

it is not possible because 구현 클래스에 또한 구현을 할 때

Class는 구현이 이미 된 모양이 있음.

but interface는 빈다 추상 클래스. 흔히 X

오버로딩이 있음

