Binary Classification

프로그램언어란?

① 언어란 정보의 매개체

숫자/문자 : Primitive 언어 x

한글/English : 자연어 O 나(개발자) —언어→ 대상(Human) ⇒ Word, Grammer (뜻이우선)

JAVA/J.S/C..: 프로그램언어 O 나(개발자) —언어→ 대상(JVM) ⇒ Term, Sytax (조금만 틀리면 Error)

② Bolean (true vs false)

- George Bull

기호 논리 학자 / 수학자 / 철학자.

'사고의 불의 사상 (참 vs 거짓) → 컴퓨터 과학 발전을 위한 기초.

⇒ 프로그래밍을 공부할때 이것을 Base 로 한다.

ex) 디렉토리 vs 파일

확장자 (.exe ...) 가 있냐 없냐

Object ?

존재 or Not ?

존재하는 것은 다~객체 이다.

① 공간을 점유? or Not.

- 여동/쥬/주다사 : Real_world_Object (공유라는 기능수행을) (메소드) 기능 수행을 위한 설계도 : Real World Object → 가 실체화되면 Class
- 우기 : Software_Object (〃) → 실체화 하는것 = Instance. ← Unit/copy of Class to Software.
- 공간 : Context (Bit502)

⇒ 공간을 이동 한다는 것 : Context switching.

종류	설치	저장공간	가변성
Real_world_Object	hard(외계O)	디스크	X
Software_Object	Virtual(외계X)	메모리	O

변동 O vs 변동 X

- 변동 O : Variable → Console (load 해야함) / 비슷한 개념 : Dynamic (동적) ex) 자바스크립트
- 변동 X : Constract → Code (save 해야함) / 〃 : Stactic (정적) ex) HTML → UI 개발

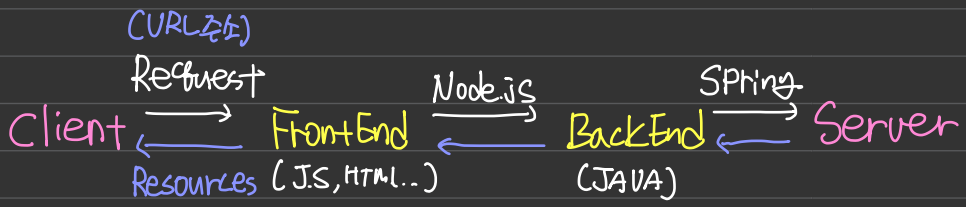
상속/변수는 항상 같이 존재 ex)

Property
속성 : 컵 → 담는공간
Value : 커피.

Property = " ";
Value O or X
attribute / property
라고 외워.

Data vs Value
디스크에 Save 메모리에 load
변화 X 변화 O
↓ ↓
Constract Valuable

Client Server Model



Ex) WWW, Email

객체지향 4대원칙

객체 : 디스크에 있는 Real_world_Object (클래스객체)

객체는 기능, 속성 의 집합이다 (집합론)

속성만 있으면? ex) class Person{ int age, String name }

⇒ Entity (개체) 이다.

: 데이터를 저장만하두있지만, 의미는 모름 (컴퓨터 입장)

마네킹과 같음, Entity산재명은 1995 DB 시절부터 존재

속성에 기능을 추가방법 → Object → Collection
- Method

Object의 집합

클래스에는 Object가 가져야할 구성멤버가 선언되어야함

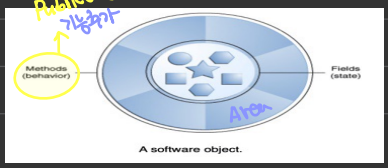
구성멤버 : 필드/생성자/메소드

*Field vs Area

변화O : Area

변화X : Field

가상머신으로 들어가면



→Method로 은닉 : Capsulation!

Ex) DOC vs DOM

DOC : 속성만존재

DOM : JS 추가로 기능 add ⇒ 객체로 변환.

- 상속 (extends / implement)

객체 지향에서 상속은 상위클래스의 특성을

하위클래스에서 상속하고 더 필요한 속성을 확장해서 사용

(계층관점 x 분류도 관점)

*구현 : is able to 인터페이스

* 상위 클래스 : 물려줄 특성 적음 Good

인터페이스 : 구현강제 메서드 적음 Good.

- 추상화

구체적인 것을 관심영역 에 있는 특성만을 가지고 재정의 하는것

ex) 병원에서 Class Person 은 환자 관점 도메인 (Entity)

상점 〃 고객 〃

- 다형성

Overriding / Overloading

↓ ↓

재정의 메소드이름만 동일

프로그램의 5대원칙 Relative 추상/구상

SOLID → Robert C. Martin "Design Principle and Design patterns"

① Single Responsibility Principle (SRP)

- 여기서 Responsibility 란?

: 책임 X, Classes should have only one Job (Single Purpose) todo

ex) 도메인 in spring framework

User Domain Class 에서 @Component @Service @Controller @Repository 가 가능한가?

Answer : Syntax error 안생김 X but 잘못됐다!

어노테이션 → It's NOT SINGLE Purpose !!

↳ Spring Framework 에서 클래스에게 Responsibility 수는 3가지.

위 4가지가 Default 이다!

↓ 좀더 나아가 추상의 개념을 알아보자!

binary classification : 변화할 수 있나? 없나?

있다 : 추상 (Abstract) 없다 : 구상 (Concrete)

(Load / Valuable) ↳ Default (Save / Constact)

↳ 선언만 (사용가능)

class : Interface (추상메소드만가능) Class < abstract / concrete → Default

method : Curl brace X Curl brace O

↳ method 란? State (문) + Expression (식) 의 결합. / 조건을 + 완벽한 조건.

① hierarchy

```
Class {
    method {
        State {
            Expression ( )
        }
    }
}
```

Default Class 에선 추상메소드, 구상메소드 사용가능

Interface 는 그럼 객체인가?

- 객체란? 속성 + 기능 의 집합체.
  - 변화 O : Valuable
  - 변화 X : Constant

⇒ abstract Method (기능) 존재 속성 (Property) 존재 X

↳ Responsibility 주면안됨 Why? Class 가 아니기 때문

- 속성 (Property) 란? Value를 담을 공간 이다.

ex) 물 (Value) - 컵 (Property) - 마신다 (Method)

Public Static String Str = " ";
↳ Type

만약 OLj 이것만 있을시 Type 은 무엇인가?

Answer: long Why? OLj ← 이것때를 알수 있다.

⇒ 이것을 Literal 이라고 함.

↳ 상수풀: 한 영역에 저장될 공간

↳ 이곳 위해선 new 선언 해야함

Instance

Literal = fixed Value

ex) Member m = new Member(); 객체의 리터럴

String Str = " "; 속성의 리터럴

Literal 은 Value에 대한 표기법.

Why 사용? 컴퓨터가 Value를 인지하기 위해

ex) int a = 0; / long l = 0L; / float = 0.0f;

⇒ 사람들은 그냥 0을 써도 다르다는 것을 인지

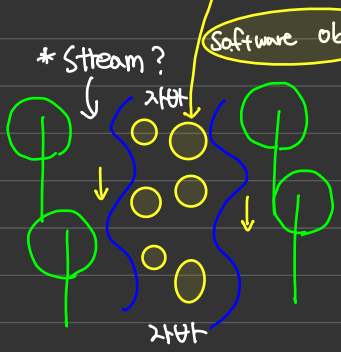
컴퓨터는 인지 X → Literal 로 표시.

★ Instance

Serialisation vs Deserialisation

객체 들의 데이터를 연속적인 Data로 변환하여 Stream을 통해 Data를 이동을 해주며
주로 객체들을 통해로 파일을 저장, 전송하고 싶을 때 주로 사용 (반대개념이 역직렬화)

java.io.Serializable 인터페이스 구현



인스턴스

↳ getter → 은닉성 확보

ex) User. password 은닉성 X

User. getPassword() 은닉성 O

*Library vs frame work

①Library

단순활용가능한 도구들의 집합

개발자가 만든 클래스에서 호출하여 사용, 클래스의 나열로 필요한 클래스를 불러서 사용하는 방식

② Frame work

뼈대 / 기반구조 라는 뜻, 제어의 역전 개념 적용된 대표적 기술

In Software, 프레임워크는 소프트웨어의 뼈대 (자바에서는 객체지향으로 통합성↓/연관성↓ 문제)

해결하기 위해서 상호 협력하는 클래스/인터페이스의 집합.

두개의 차이점

Frame work : 우리집

Library : 남의집.

제어의 흐름에 대한 주도성이 누구에게 있나?

즉, 어플리케이션의 flow 를 누가 쥐고 있냐의 차이.

프레임워크는 전체적 흐름을 스스로 가짐 / 라이브러리는 사용자가 전체적인 흐름을 통제

⇒프레임워크는 사용자가 프레임워크(맥락)에 내부로 들어가서 사용한다.

라이브러리는 사용자가 직접 라이브러리를 사용, 호출해서 흐름을 만드는 것.

스프링 4대원칙

Defendency Injection