· 07/22

① 회원관리 program → (새로운 회원을 등록 / 기존회원을 검색) → ★ (Users) Table부터 생성해 보아요

→ (Main.java → main method
UserDAO.java → DAO
User.java → DTO, VO, Entity, Bean) → package lecture0722.step1

★ Java

- language spec
- Class, Instance (객체지향개념)
- Class library (~) (Thread, IO, Network, collection)
- Database → JDBC

★ Java 기본

기술적 측면

- (Web (servlet → Spring Framework)
- IoT (android))

- ★ Refactoring
- ★ Design pattern ★

Class를 작성 (설계)할때 주의를 기울여야 하는건 → 유지보수

↳ 변화가 있을때 빠르고, 쉽고, 안정적으로 수정할 수 있어야 해요!

→ 코드의 수정이 다른 부분에 영향을 주는것을 최소화!

SoC (Separation of Concern)

* Refactoring → Method extraction 기법을 이용해서 우리코드를 재구성해 보아요!!

"쉽게 해결" → "유지보수성이 좋은 코드를 작성"

* 재사용성이 있는 Class인가요? → (가정) 우리 DAO를 판매

* Naver
* Kakao

◎ naver, kakao는 나름대로의 DBMS의 connection 연결 방식 사용

① 소스코드를 제공 X

② 상속을 이용해서 해결

↓ 우리 코드를 살펴보면.

[상위 class (UserDAO)에 기본적인 logic 흐름이 구현
이안에 protected method, abstract method가 있어요

하위 class (NUserDAO)에서 특정 method를 overriding 해서
하위 class에 맞도록 기능을 완성]

→ Design pattern에서 "Template method pattern"

↳ class를 확장하는 가장 기본적인 방식
(^ 상속)

→ 변경되지 않는 기능(logic) 상위 class, 변경되어야 하는 부분은 하위 class에서 구현

● 조금 다른 관점에서 살펴보면

UserDAO에는 getConnection이라는 추상 method가 존재

NUserDAO에서 구체적으로 객체를 어떻게 생성할 것인지를 기술

→ 하위 class에서 객체의 구체적인 생성방법을 결정해서 사용하는

Design pattern → Factory Method pattern

• 객체를 재 사용한다는 측면에서 좋은데 → Inheritance (?)

↓

상속은 다중상속의 제약이 있기때문에

"다른 방식으로 구현"

☆ • Class를 분리해서 만들어 보아요..!!

→ 수행은 잘 되지만 소스코드를 공개하지 않는 이상 재사용이 힘들어요.!!

↳ class가 tightly coupled 현상 발생 → 더 심해져요.!!!

(class내에 다른 class명 나오는 건 좋지 않아요)

↳ 어떻게 하면 해결이 될까요? → Interface

UserDAO → 다른 class에 종속되지 않아요.!!

DI (Dependency Injection)

"Review" → javaFx (Jdbc code)

⇒ [(Layered Architecture + interface]