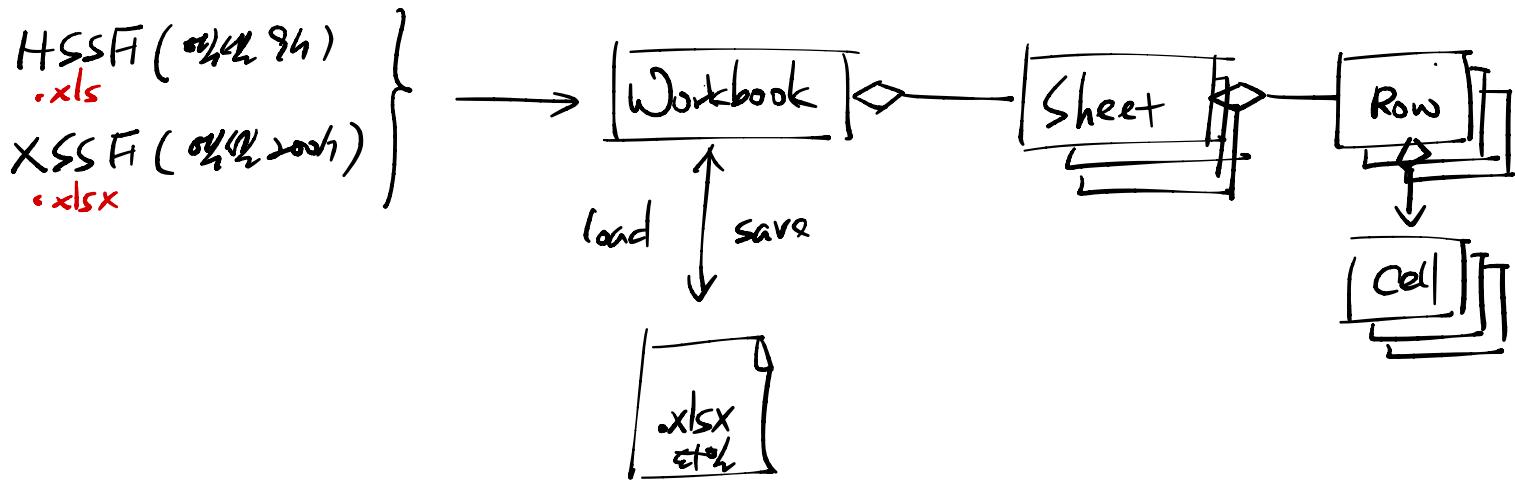


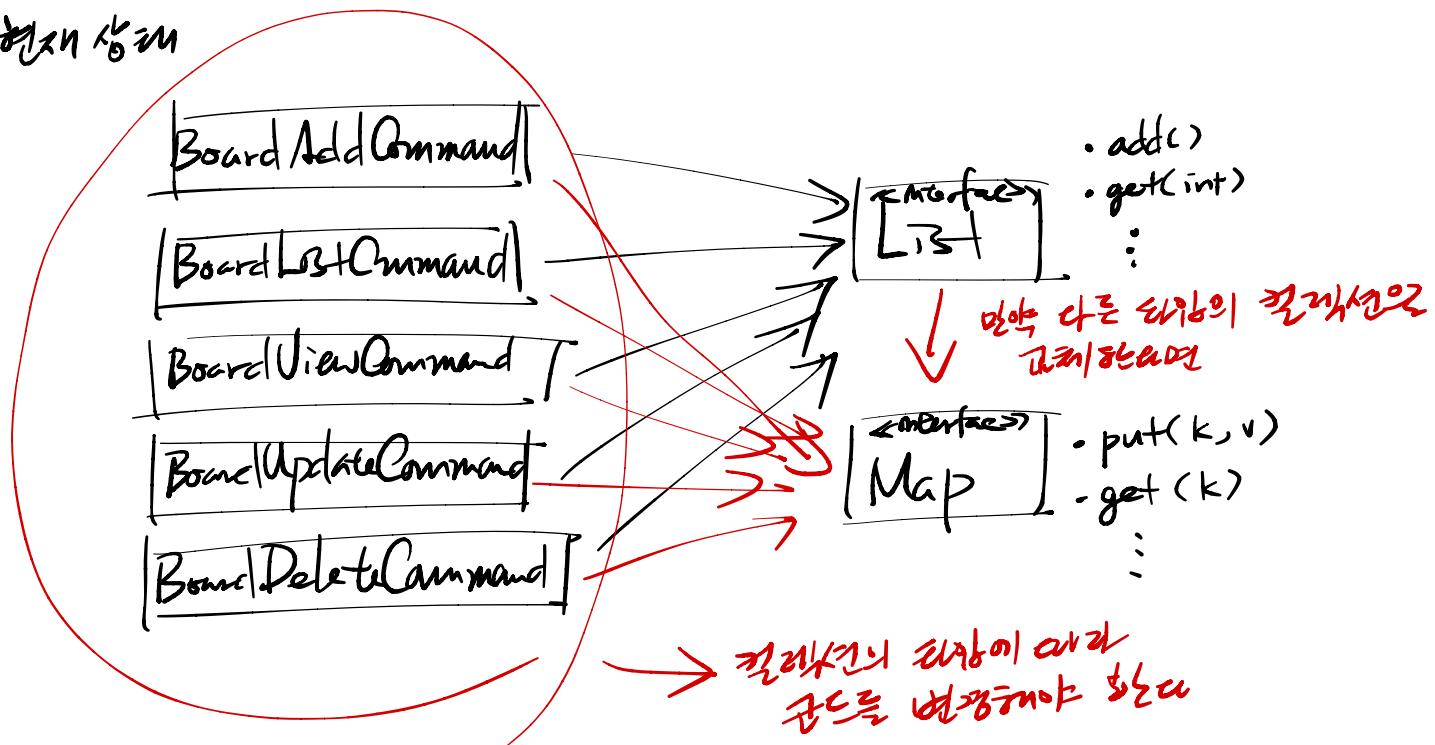
33. 콜렉터는 어떤 원칙을 따른다? : Apache POI 소개



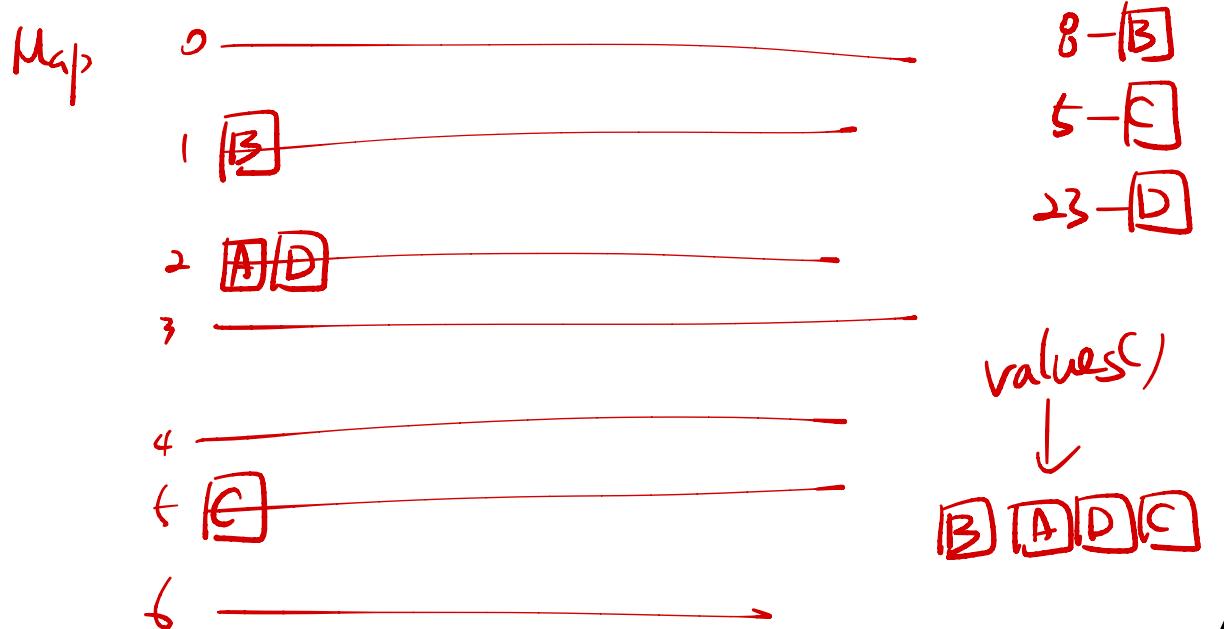
34. DAO 가 필요한 이유

↳ 데이터 베이스를 관리하기

① 테이블



* Map에 데이터를 만들면 그걸로 예측이 가능!

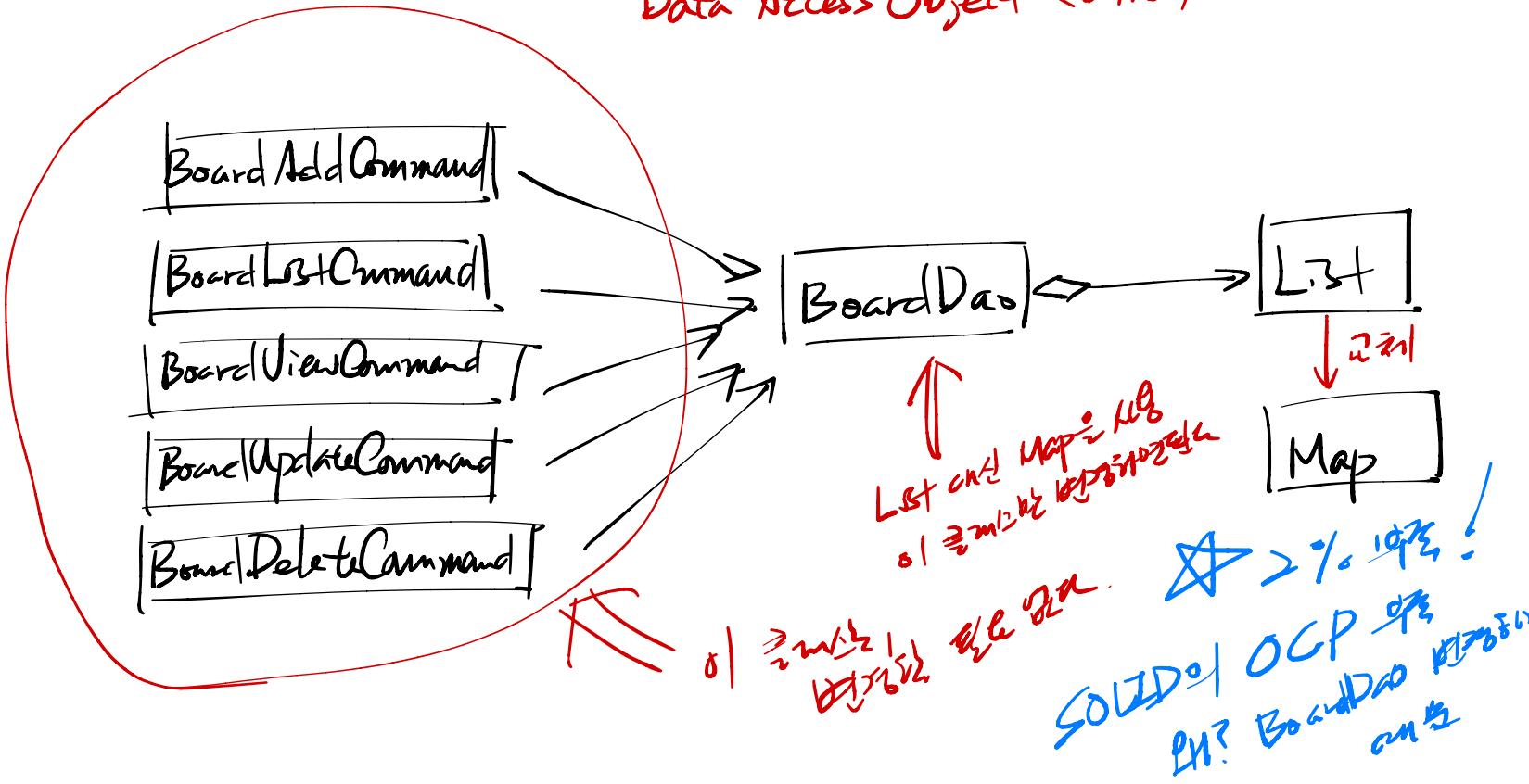


(1) 키를 갖는 HashTable을 구현하는
key의 Hash값을 기준으로 삽입하는
방법을 구현하는 방법이 있다.

35. 글로벌 키워드를 제거해보자

↳ 제거한 키워드는 뭘까? → 함수가 기반이다.

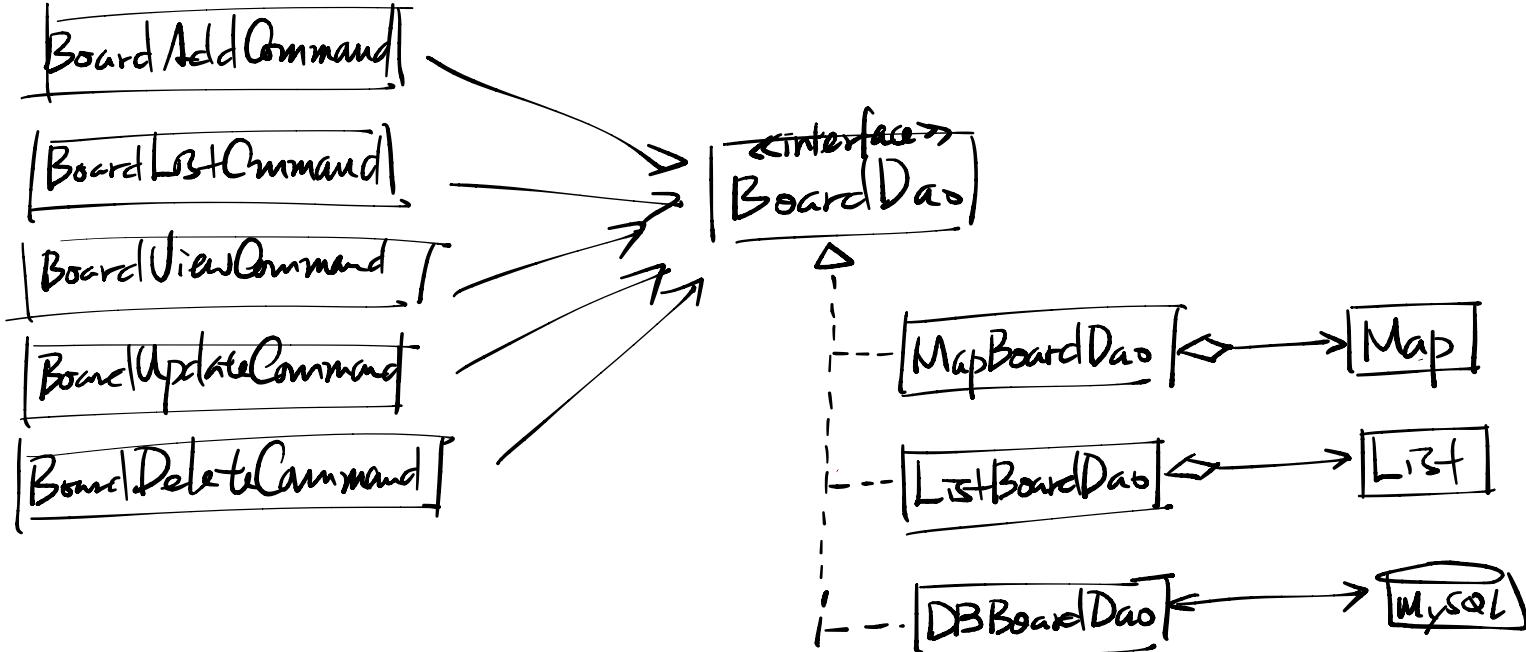
Data Access Object (DAO)



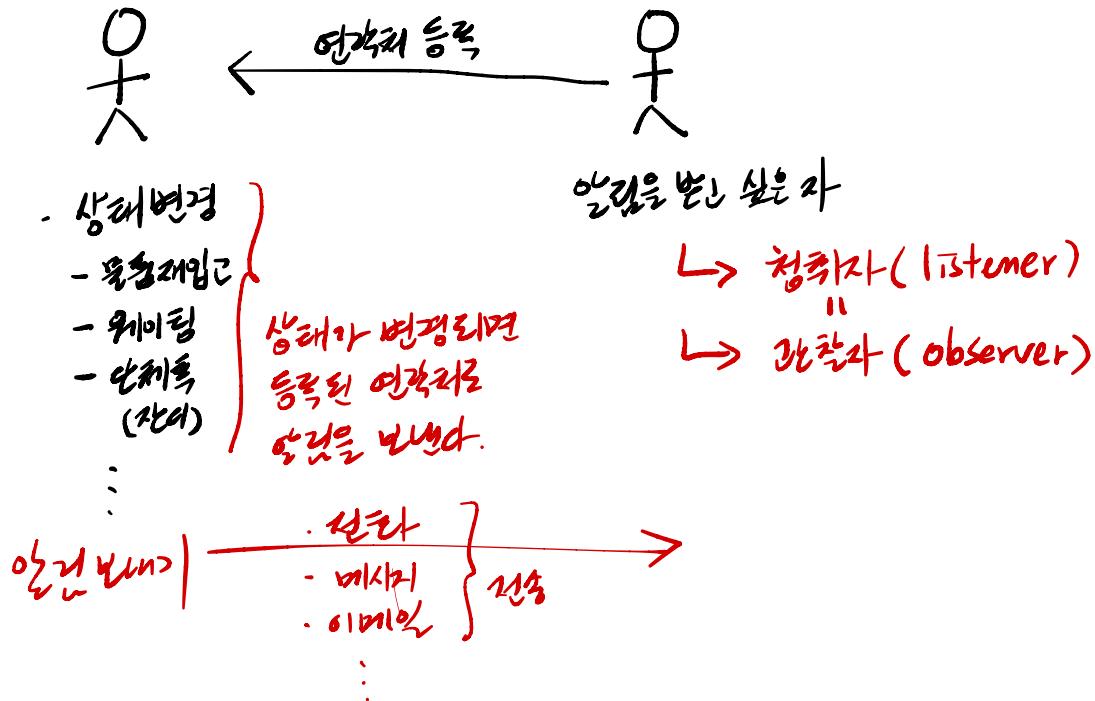
35. 글로벌 키워드는 final이면 됨

↳ 제한된 개수의 경우 → 제한된 개수의 제한된 개수.

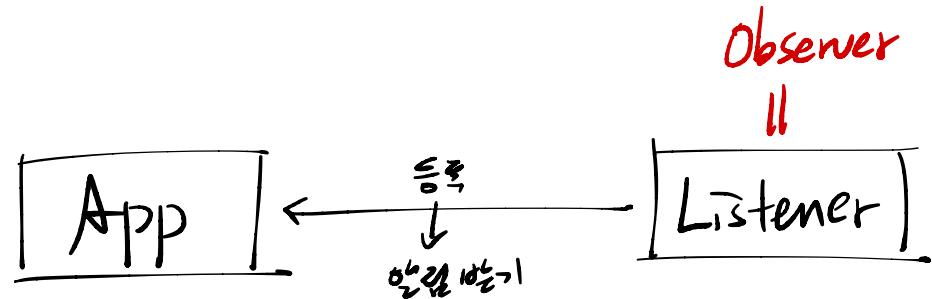
Data Access Object (DAO)



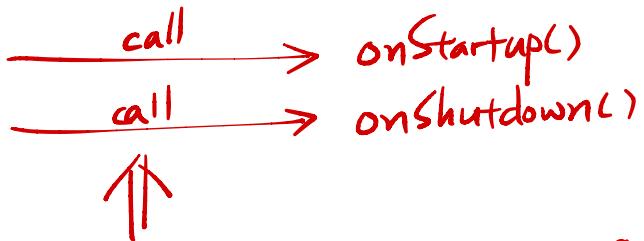
* 36. 알림 패턴 : GOF의 Observer 패턴
(설명문서)



$\exists \subseteq \text{Invert}()$

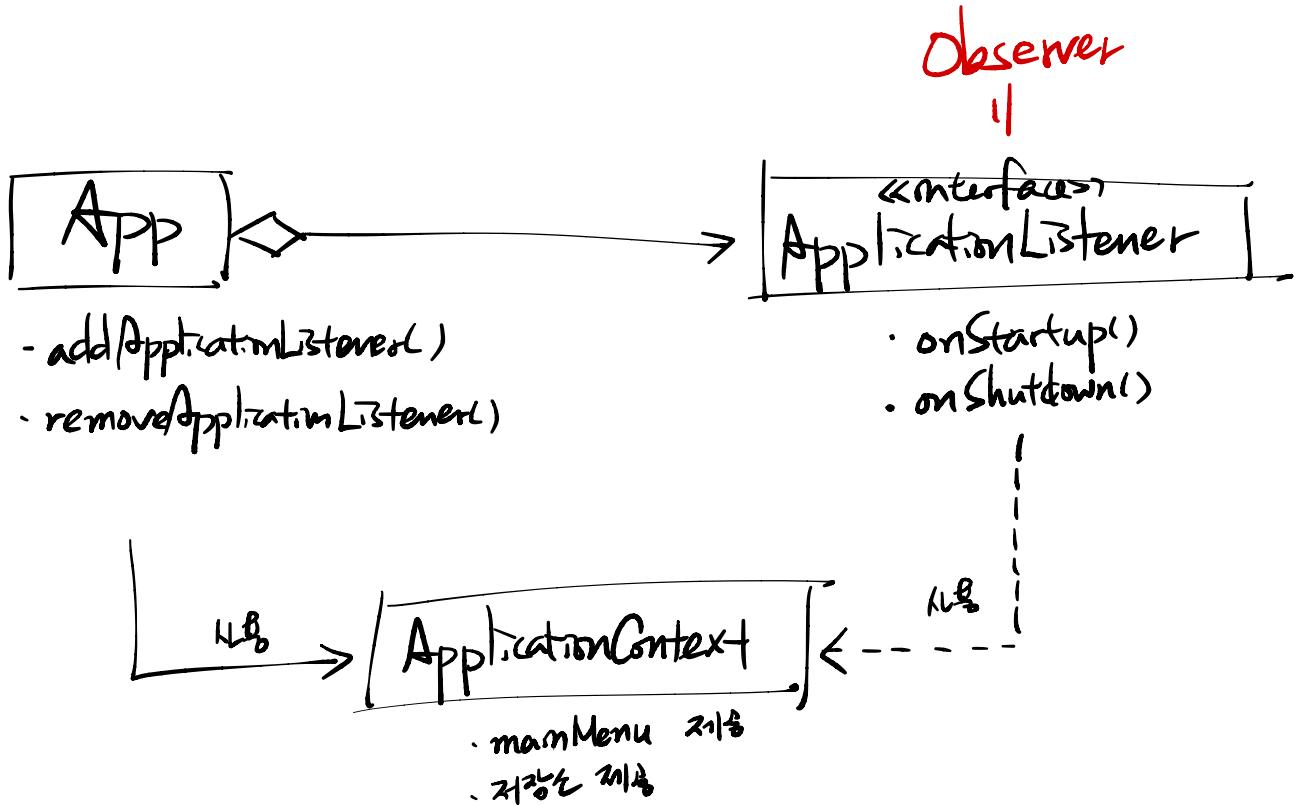


- 설정에 따라
- 사용자
- 풍경



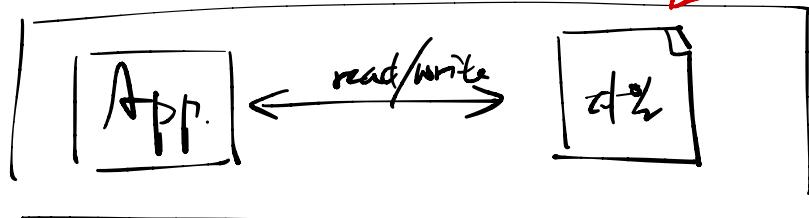
설정에 따라 설정에 따라 설정에 따라
설정에 따라 설정에 따라 설정에 따라

* GOF의 Observer 패턴 대처



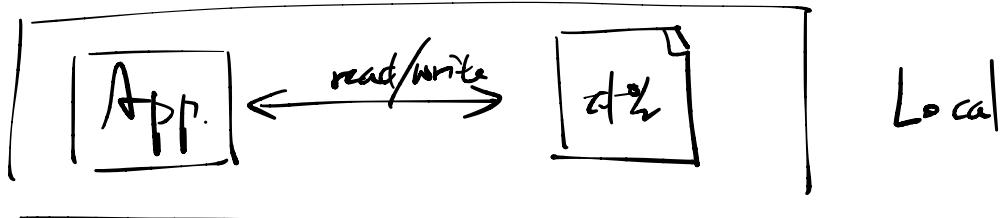
31. Application 간에 데이터 공유

현재 상태



App.의
데이터로 데이터를
Local

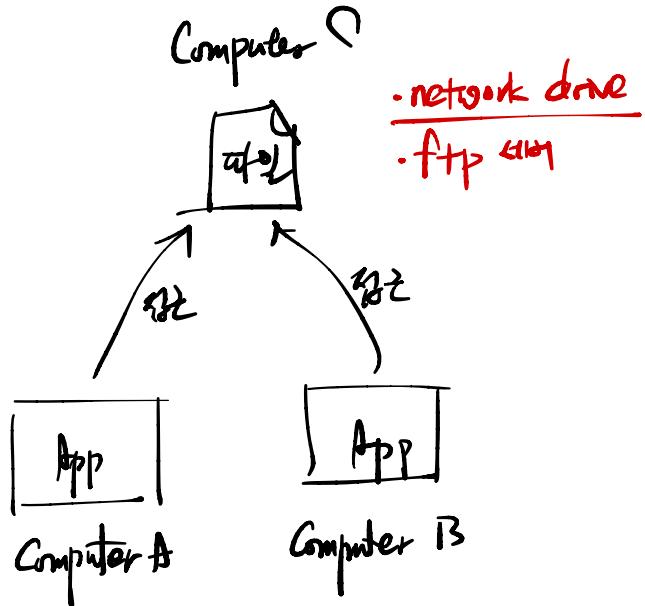
App.-간의
데이터 공유는!



•
•
•

31. Application 간에 파일 교환

① 파일로드된 파일 공유



※ 문제점

- 동시에 여러 App. 실행
문제점이 있다



파일을 쉽게 쓸 수 있다.

31. Application 간에 데이터 교환

② 멀티프로세스 애플리케이션

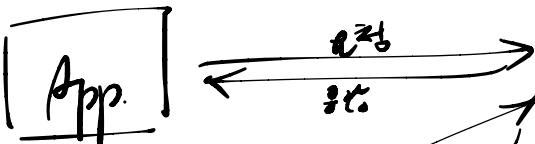
- networking prog.
- multi-tasking prog.
 - ↳ multi-threading
 - ↳ synchronous

애플리케이션
Data는 쓰고 읽을 때
Data를 주고 받음

Computer A



Computer B

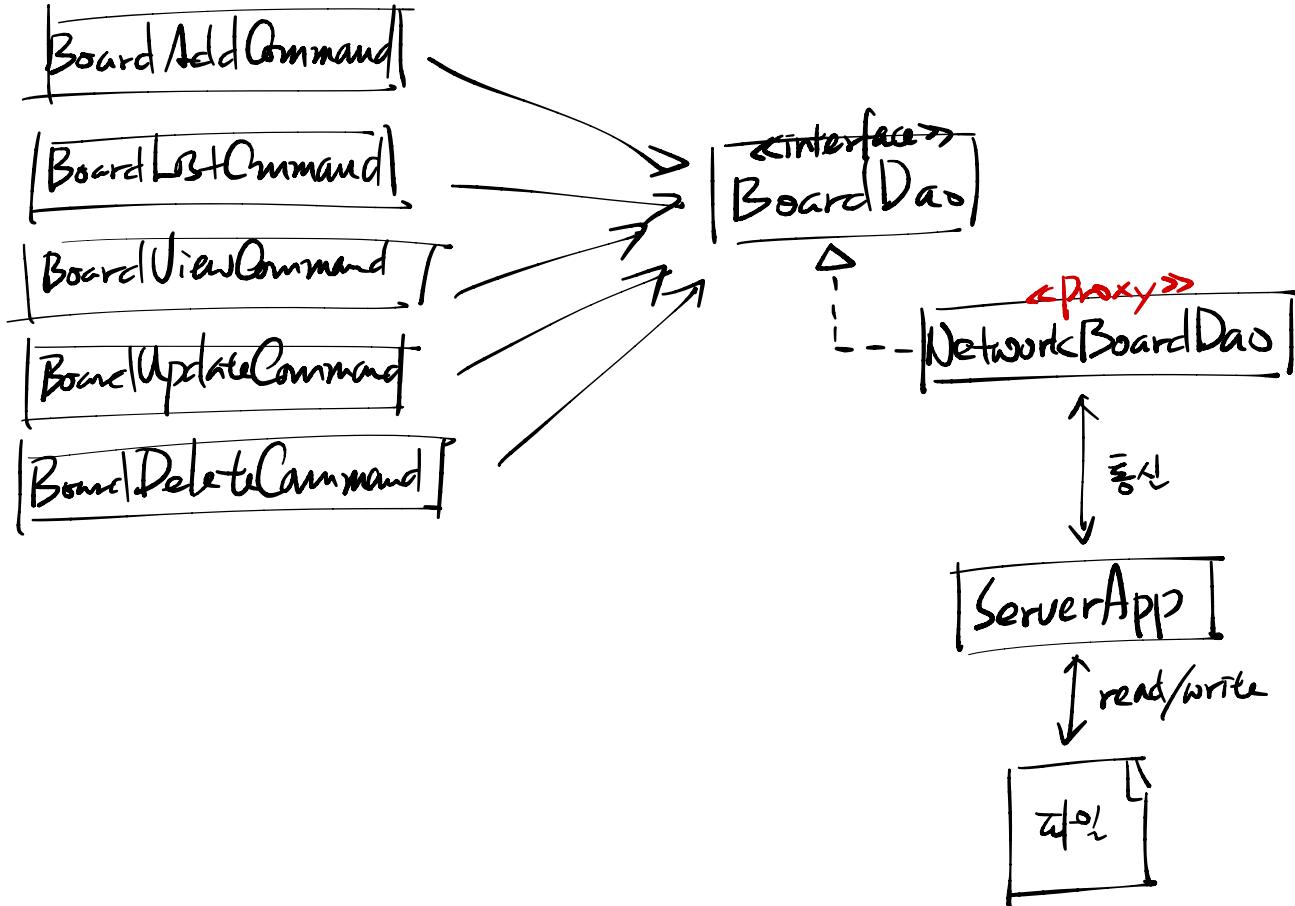


Computer C

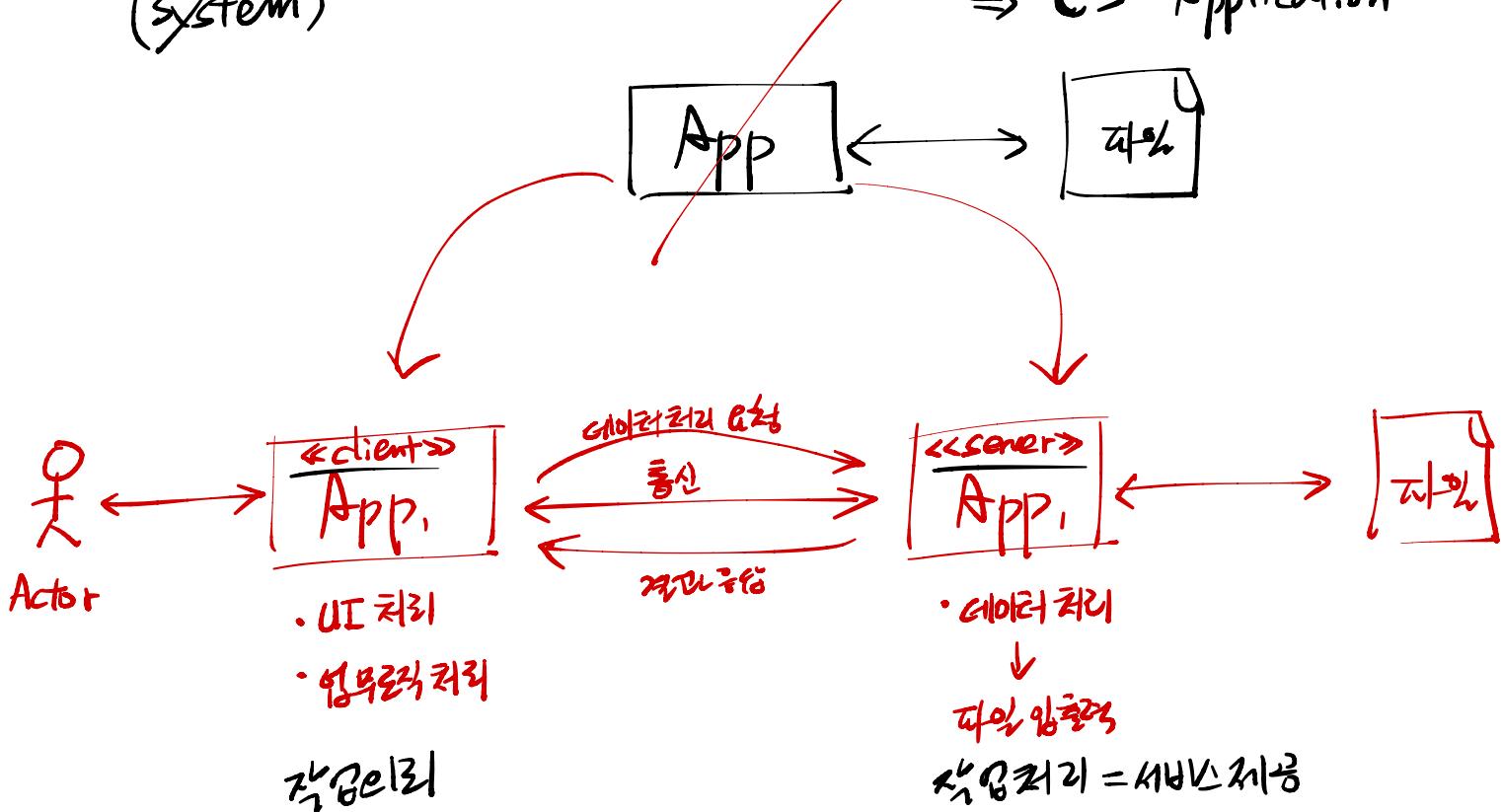


networking

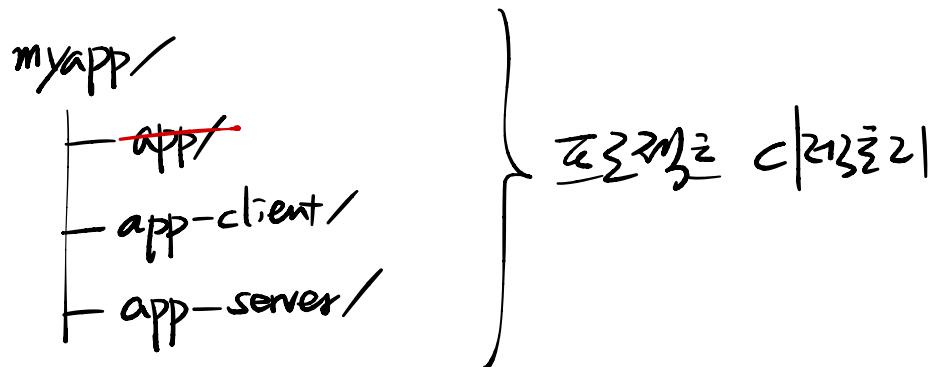
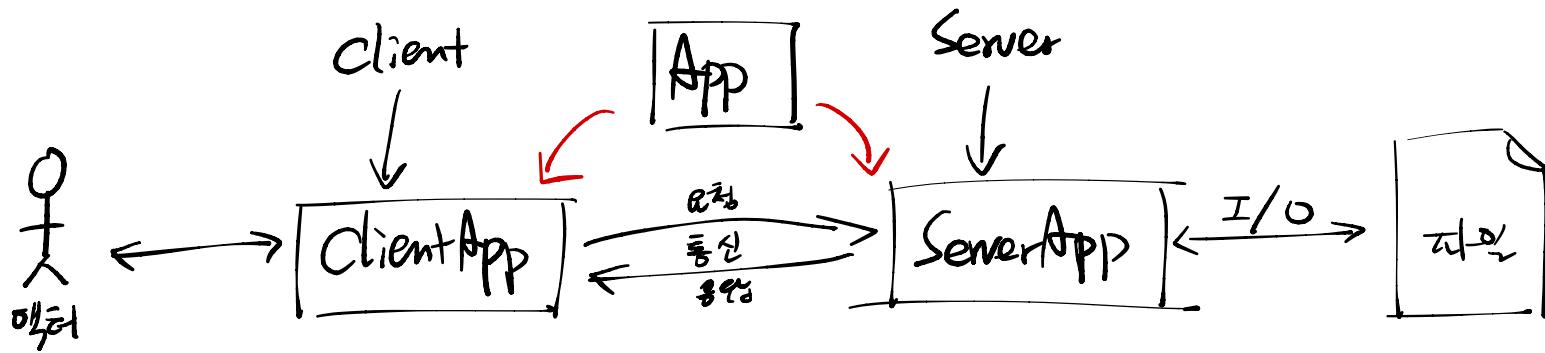
단계 32번의 단계 32번 일정 시스템
(process) (process) 단계 32번
단계 32번



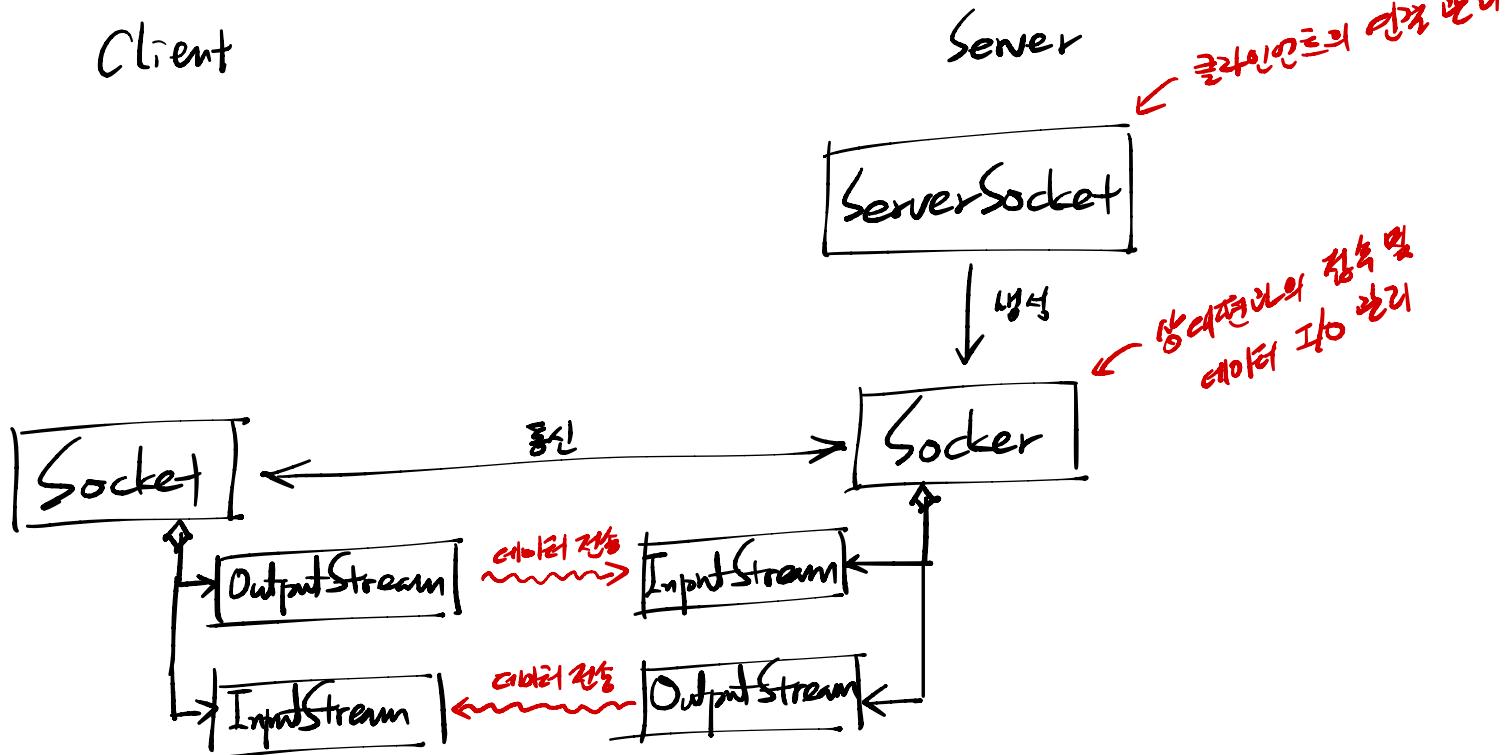
* Software Architecture : Client / Server Architecture
(System) ⇒ CS Application



* CS Architecture 구조

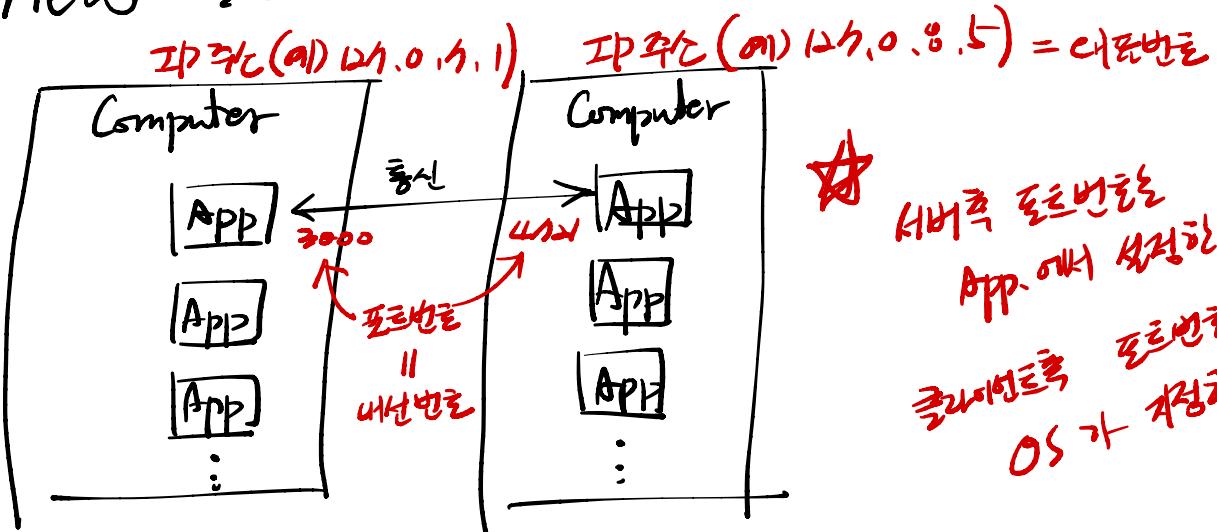


* Networking API API



* ServerSocket

new ServerSocket(포트번호, 대기열크기)



☆
내비쪽 포트번호는
App.에서 설정하고
클라이언트쪽 포트번호
OS가 지정한다.

통신내선번호
포트번호
클라이언트포트번호

* Socket

Client IP 주소



Client IP 주소



new Socket (IP 주소, 포트번호)

* 나의 포트번호?
(로컬호스트)

OS가 제공합니다.

* 특별한 IP 주소

127.0.0.1

Local
주소
||
로컬 호스트 주소

||
"localhost"

* Protocol : 데이터 송수신 규칙

client

컬렉션 - "users"

작동 명령 - "insert" | "list" | "get" |
"update" | "delete"

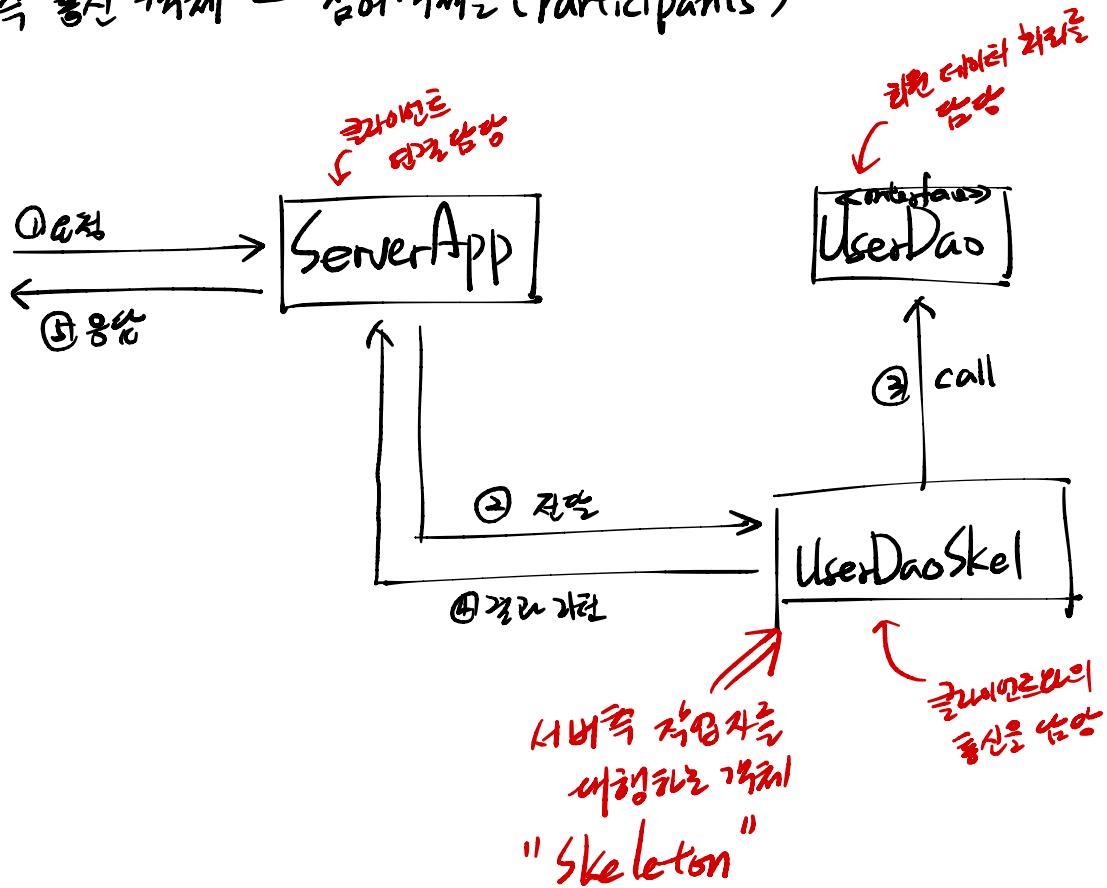
전송 데이터 - *

Server

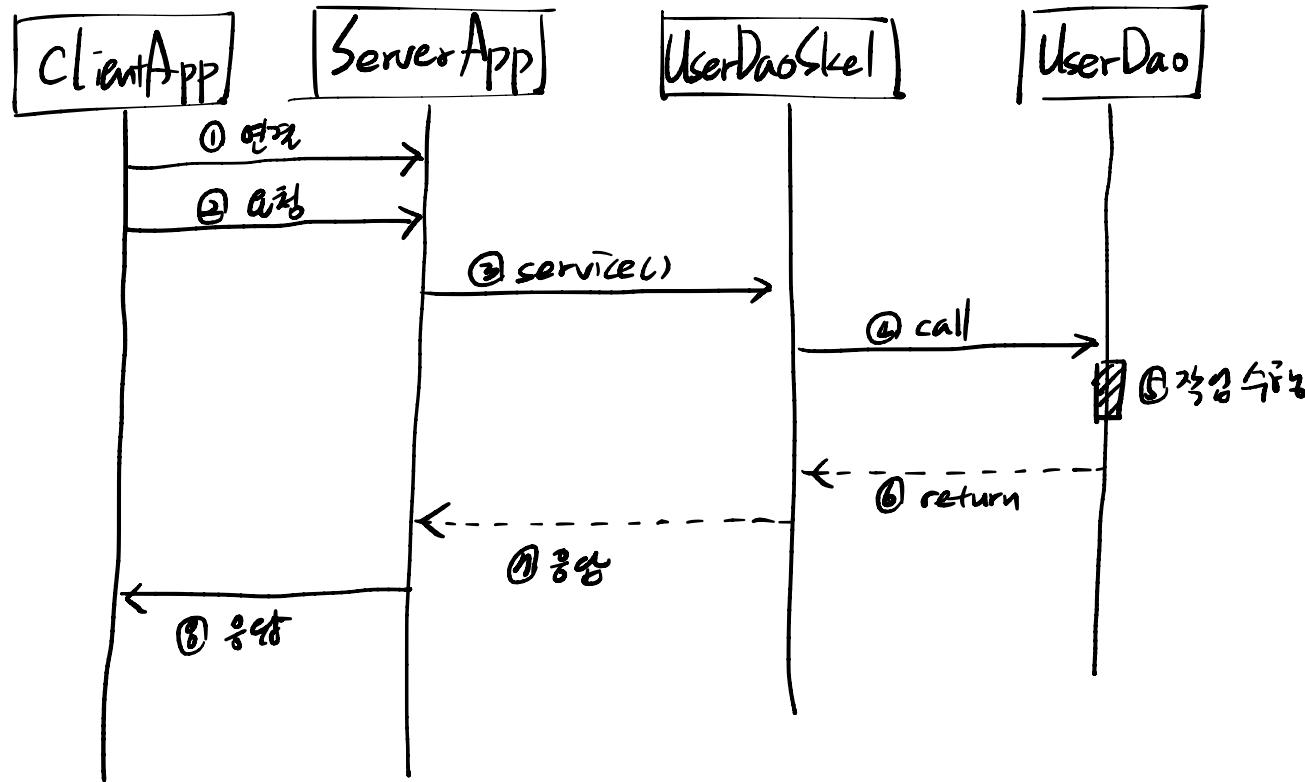
응답 상태 - "success" | "failure"

응답 데이터 - *

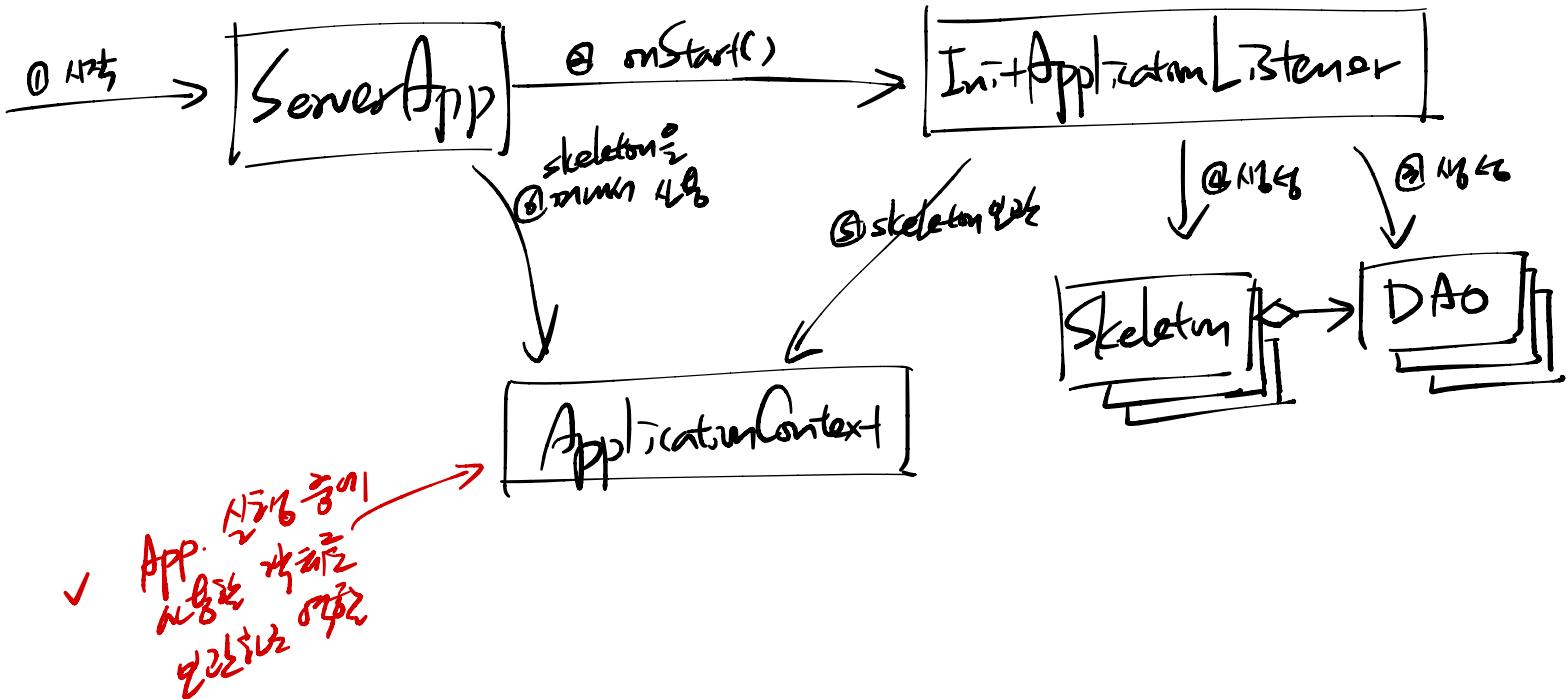
* 서버측 통신 구조 - 참여자들 (Participants)



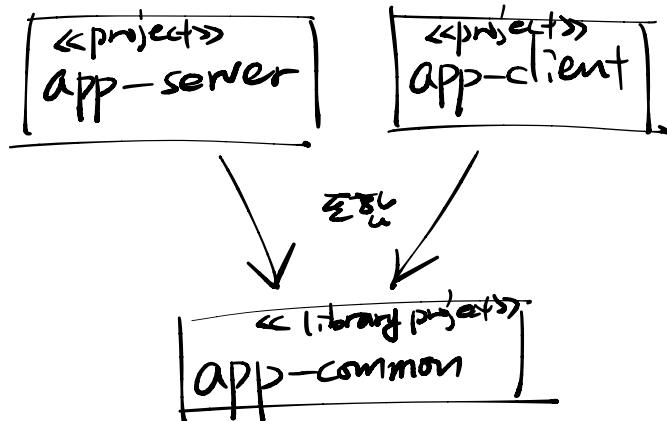
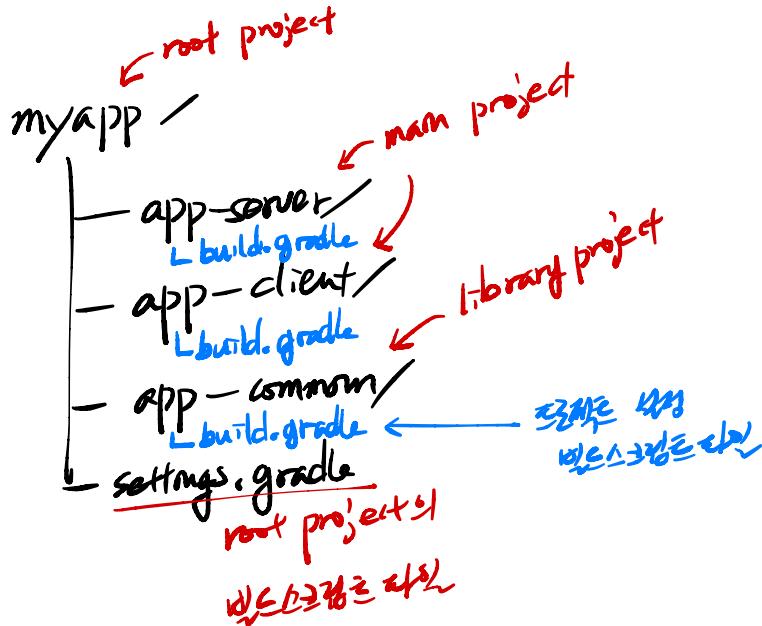
* 서버측 통신 구조 - Sequence Diagram (클라이언트가 서버의 서비스를 호출하는 과정)



* ServerApp in Skeleton

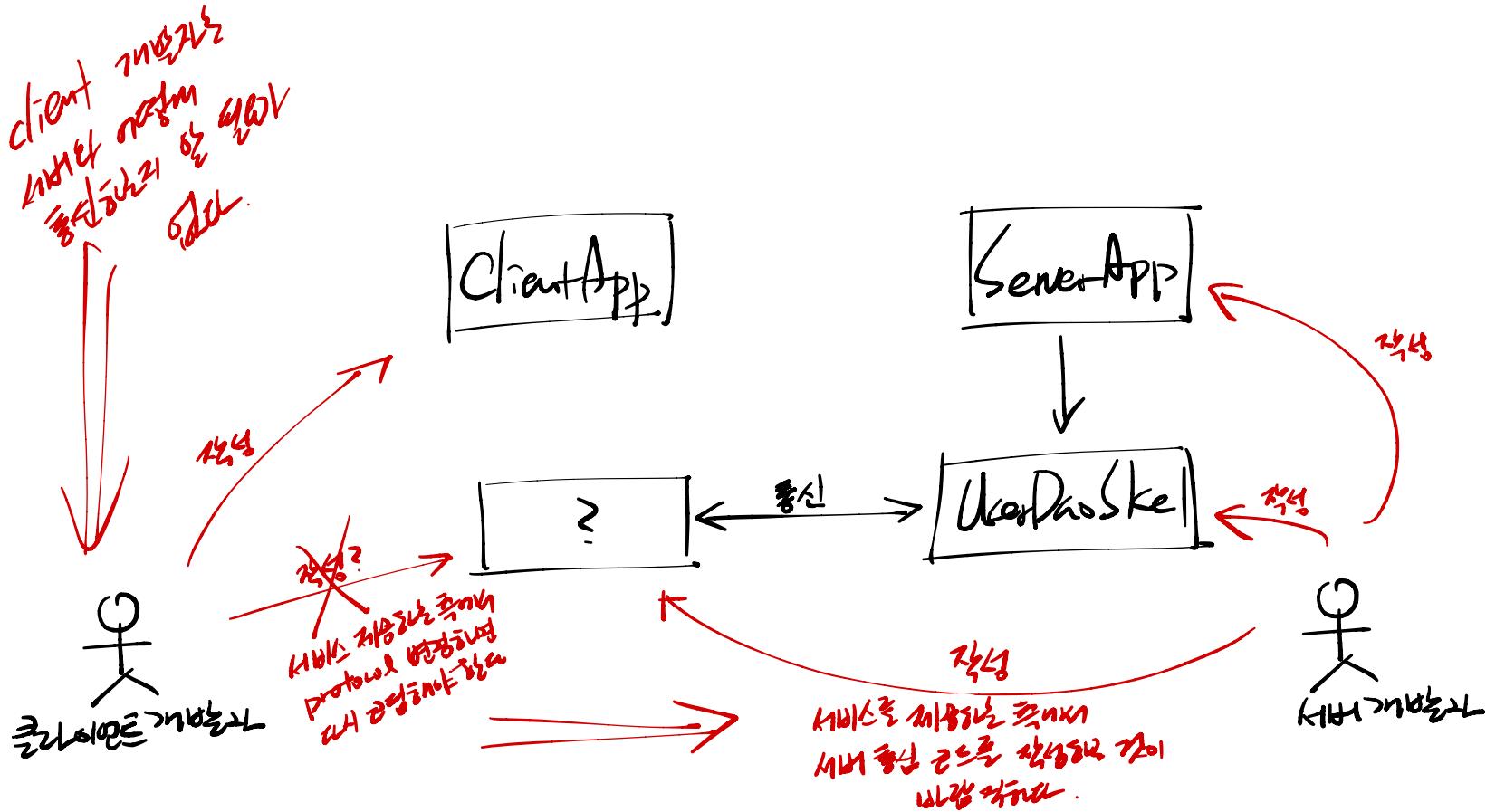


* 2019-2021 အချက် - အကျဉ်းချုပ်၏ ရေးဆွဲမှုပါမ်း

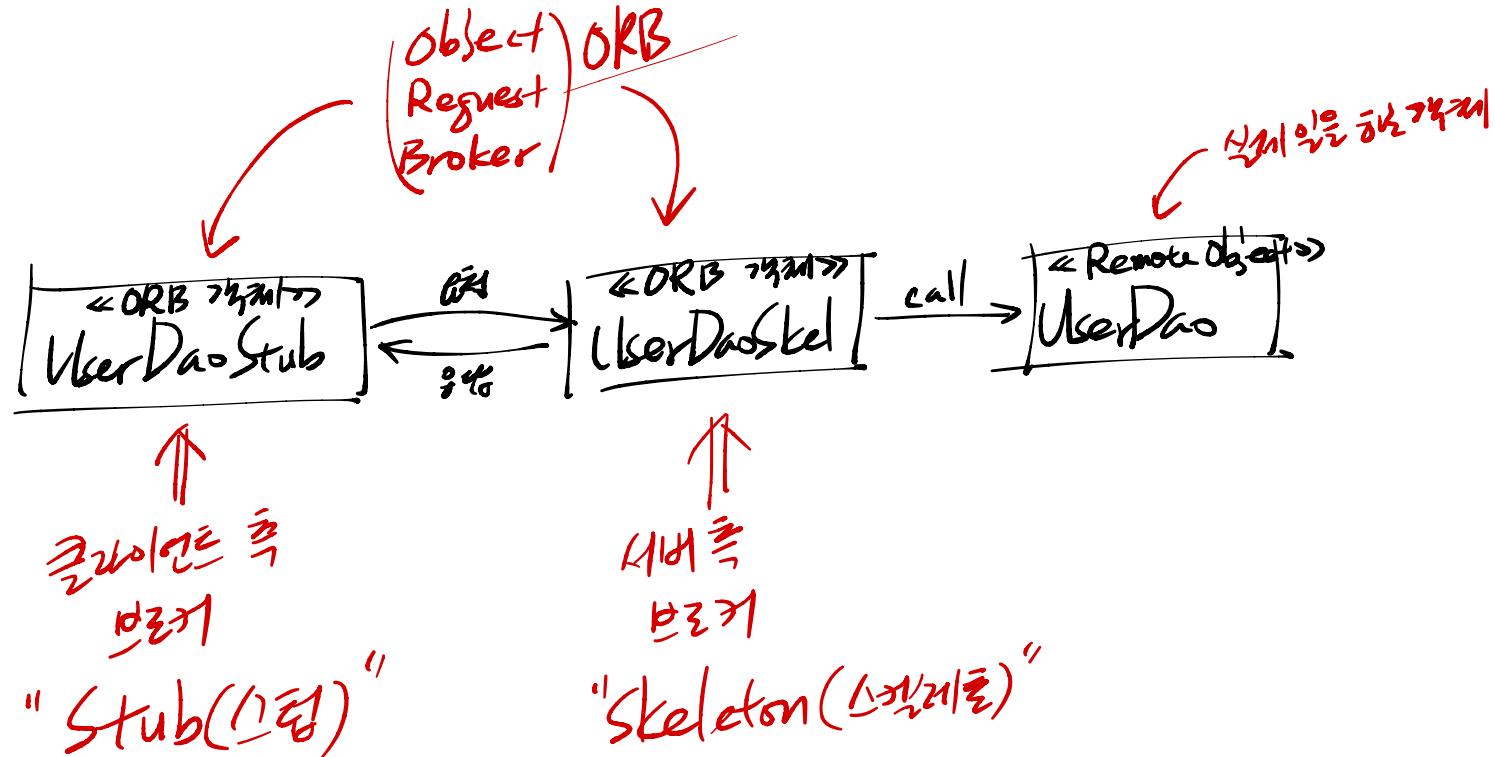


- VO ၁၃၈
- Net ၁၃၈

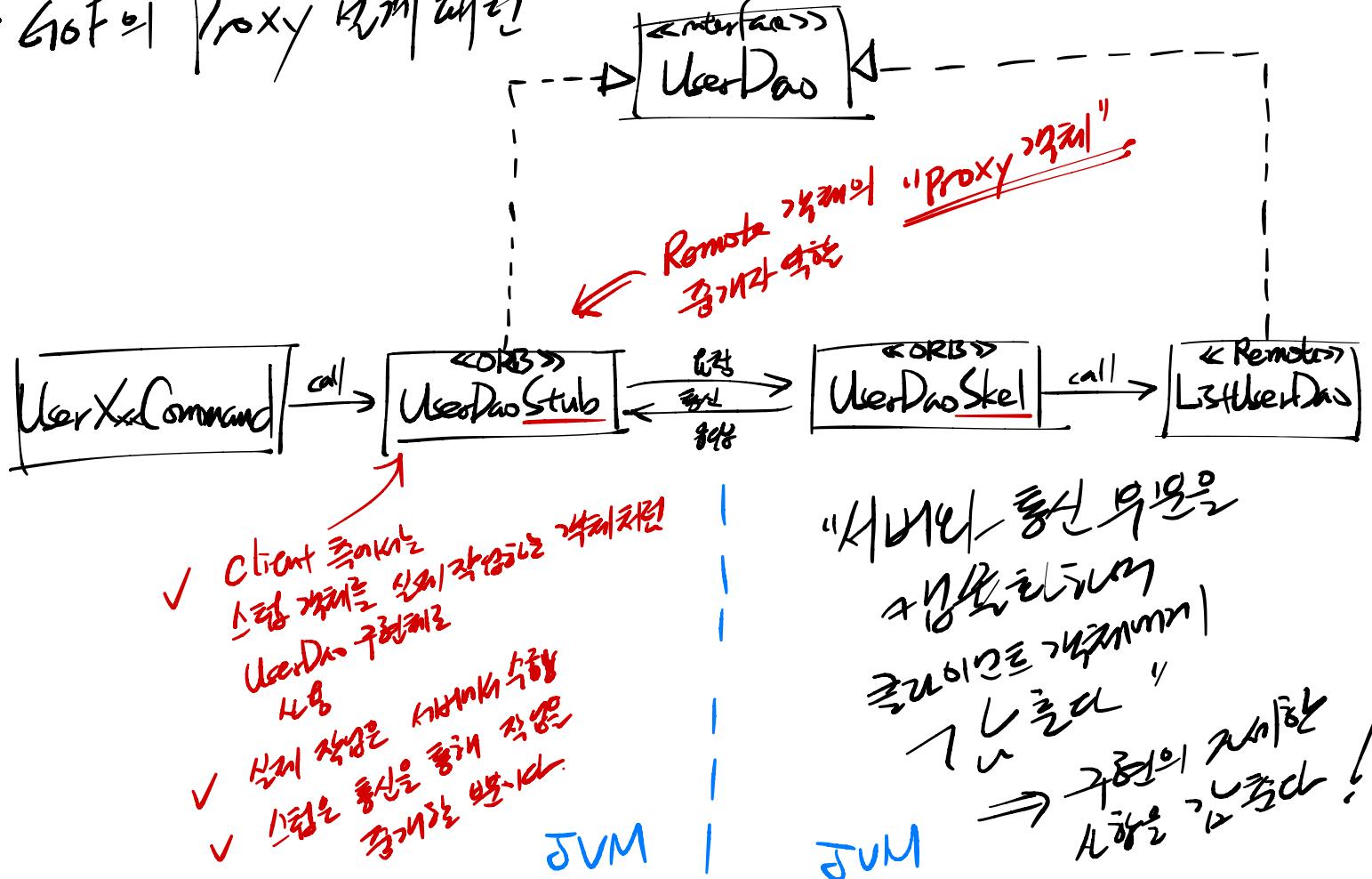
* Server 및 Client



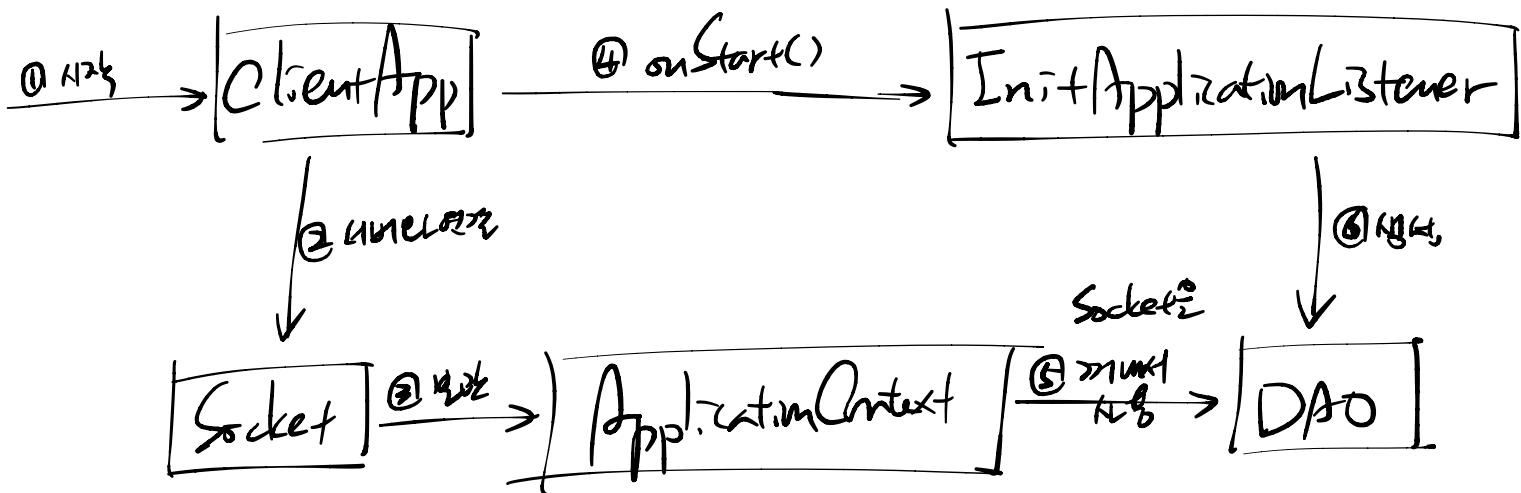
* Skeleton in Stub



* GOF의 Proxy 패턴 구현

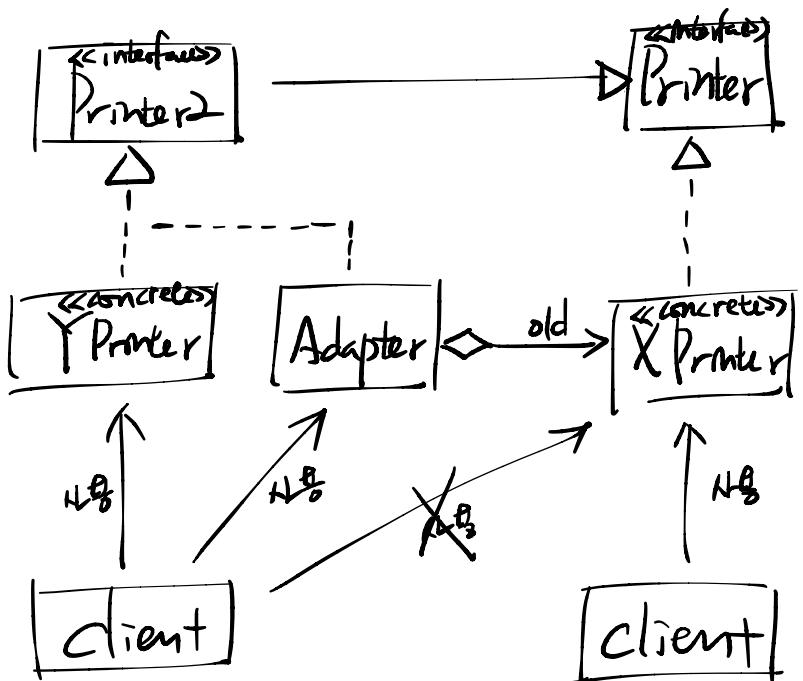


* Client Application & Stub

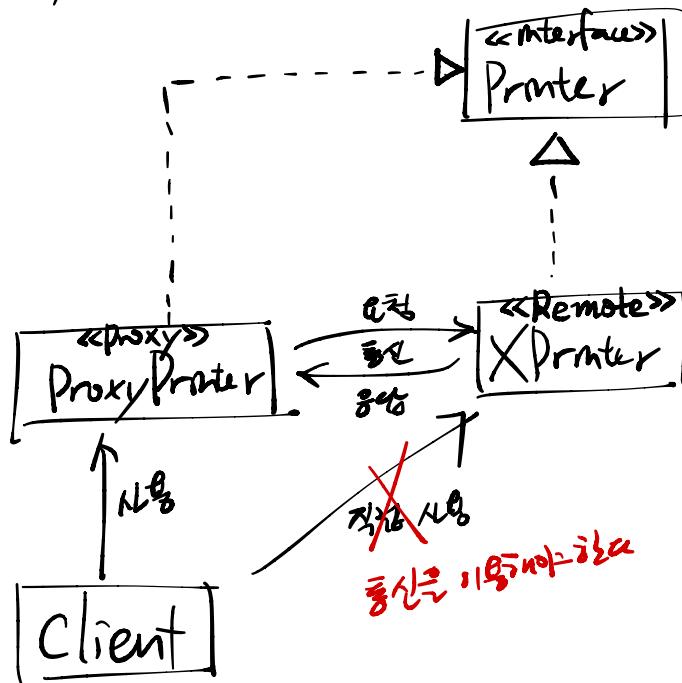


* Adapter 패턴 vs Proxy 패턴

① Adapter

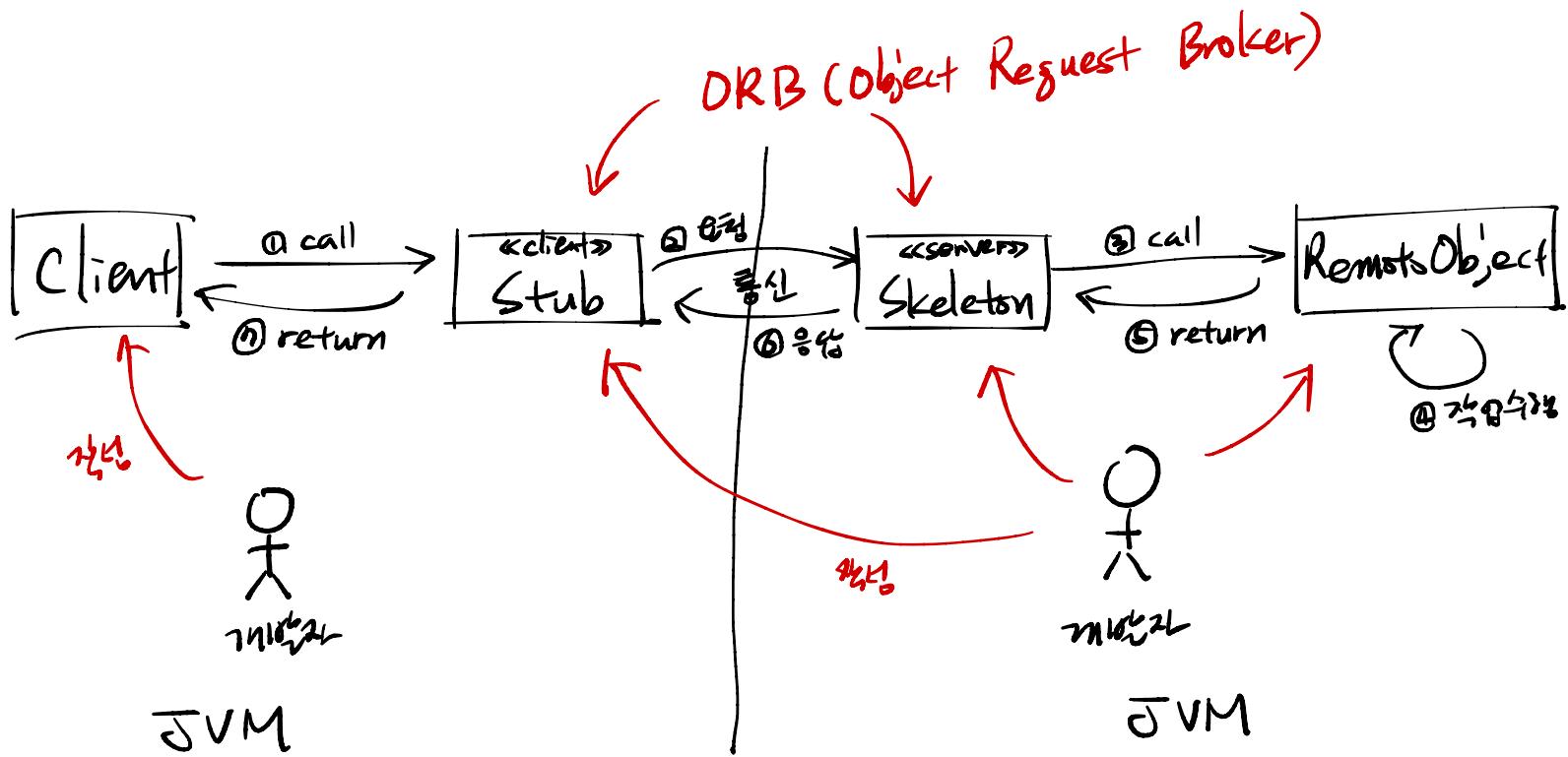


③ Proxy



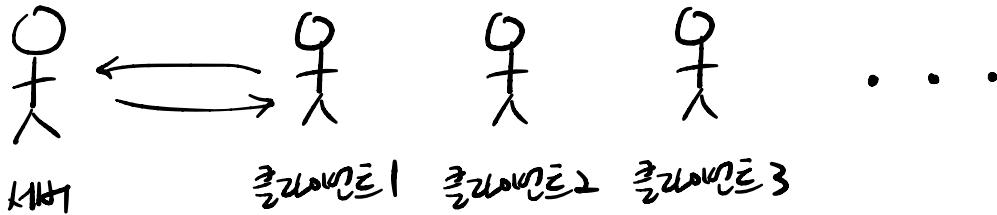
* Remote Object 개념

↳ 다른 JVM에서 만든 객체
(프록시)

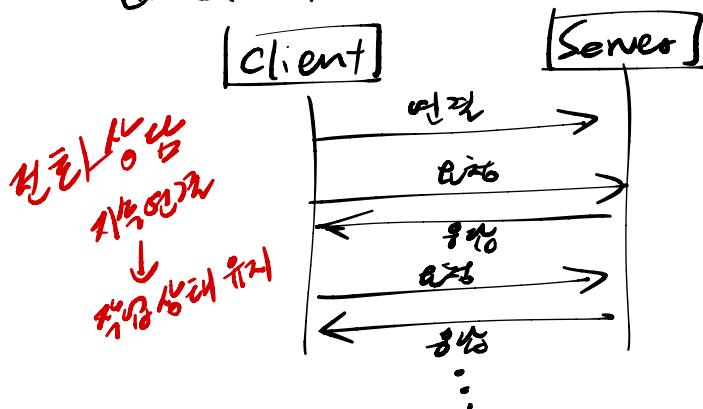


38. Stateful vs Stateless

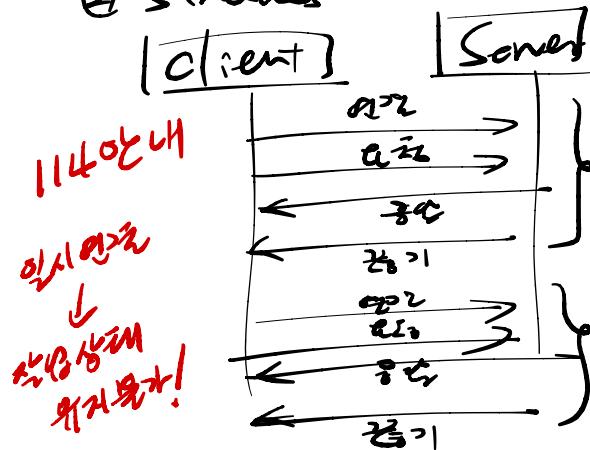
- * Stateful 웹서버 - 한번 연결한 후 여러번 요청/응답 처리
- * Stateless 웹서버 - 한번 연결한 후 한 번 요청/응답 처리



① Statefull

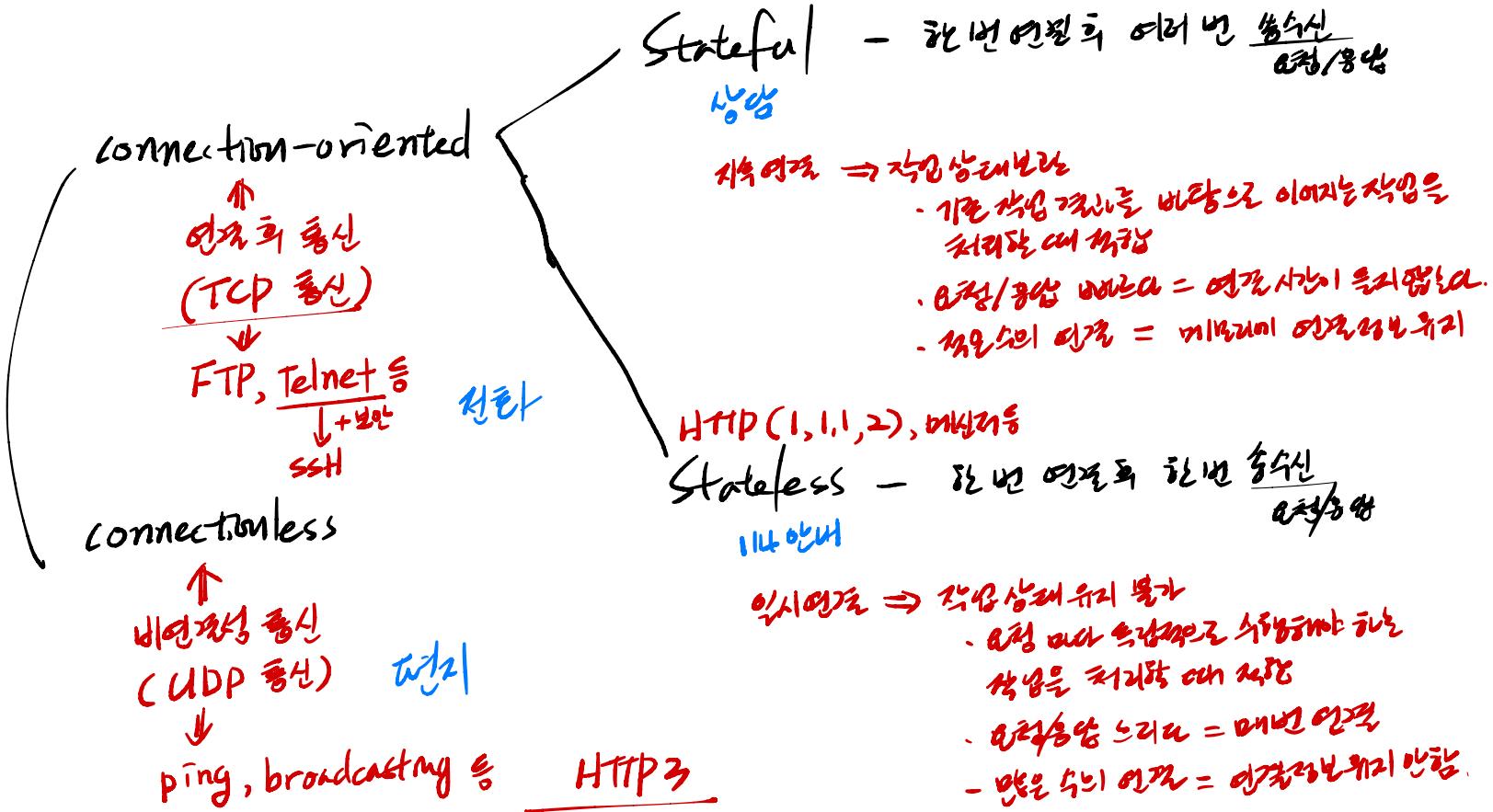


② Stateless

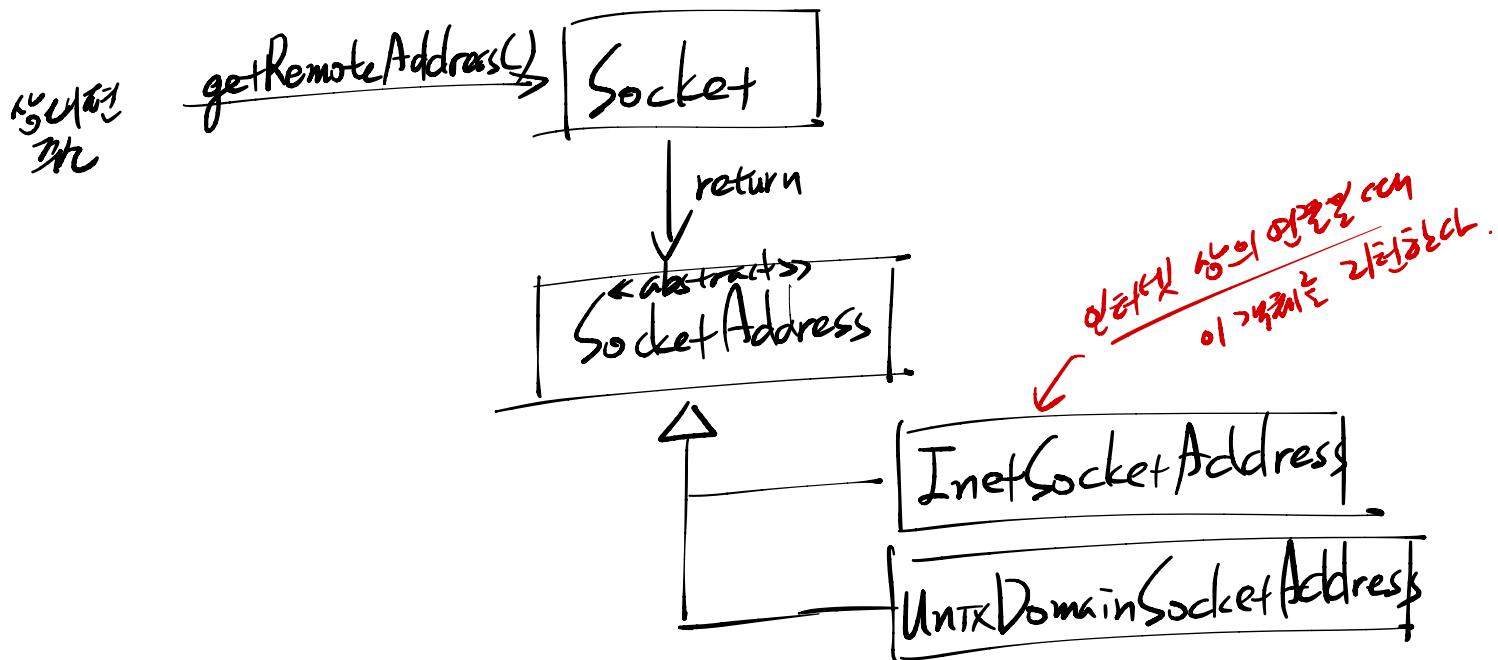


* 통신 18'시'

FTP, Telnet, SSH, 스트리밍, 채팅 등

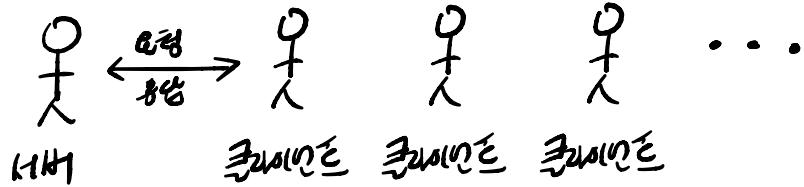


* Socket API

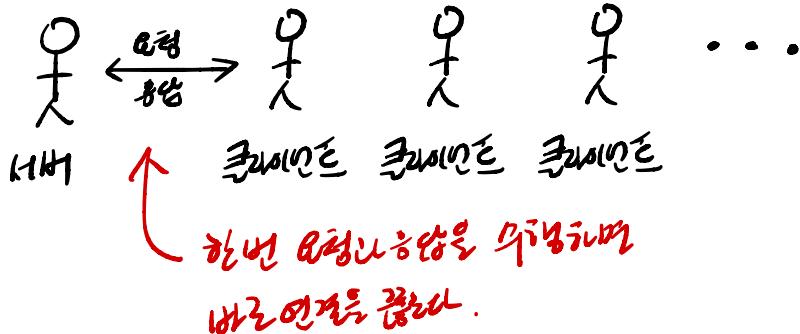


39. 미리보기드 306

① 대중문화의 관점 - Stateful



② 각각의 상태를 갖지 않는 것 - Stateless

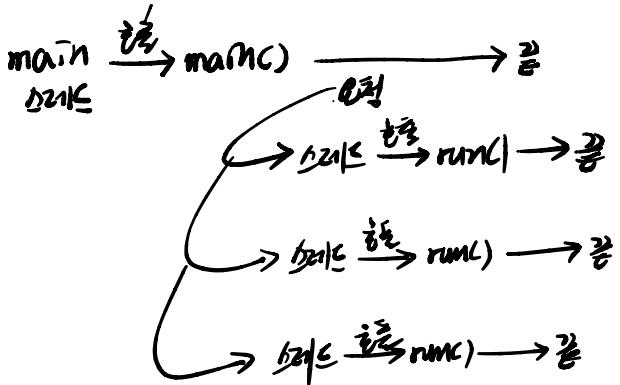
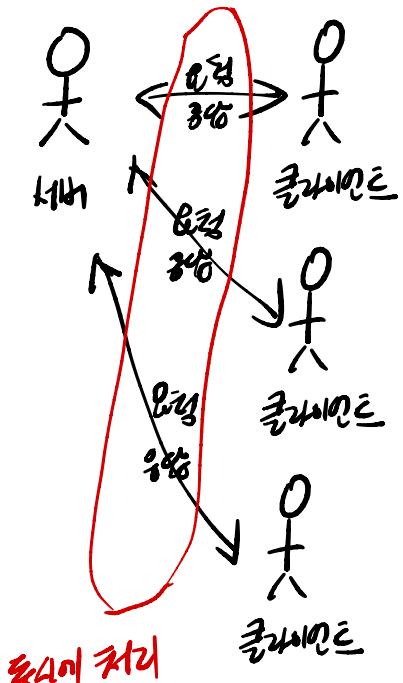


- ✓ 시비와 연료된 쿨리어먼트가
연료를 금속을 대체해
다음 쿨리어먼트는 가능해야 한다.
 - ✓ 16%의 수소로 쿨리 된다.

- ✓ Stateful vs Stateless \rightarrow Stateless / Stateful
↓
Stateful vs Stateless comparison

- ✓ 자판기 솔루션 대로 처리

③ 다중 쓰레드링크 예제 - Multi-threading \Leftarrow stateful & stateless 1회용의 고정적인 문제점 해결



고정화되어 사용되는 고려사항
고정화되는 대상은 고정화된 대상

클라이언트 고정을 고려해보자!

개인?

복잡화로 처리!

클라이언트 고정 처리는

main의 실행 도중에
처리된다면 좋다!

* 스레드를 만드는 예제 - main 스레드에서 다른 스레드를 만들 때의 예제

class 씨닝 extends Thread {

void run() {

여기 내용

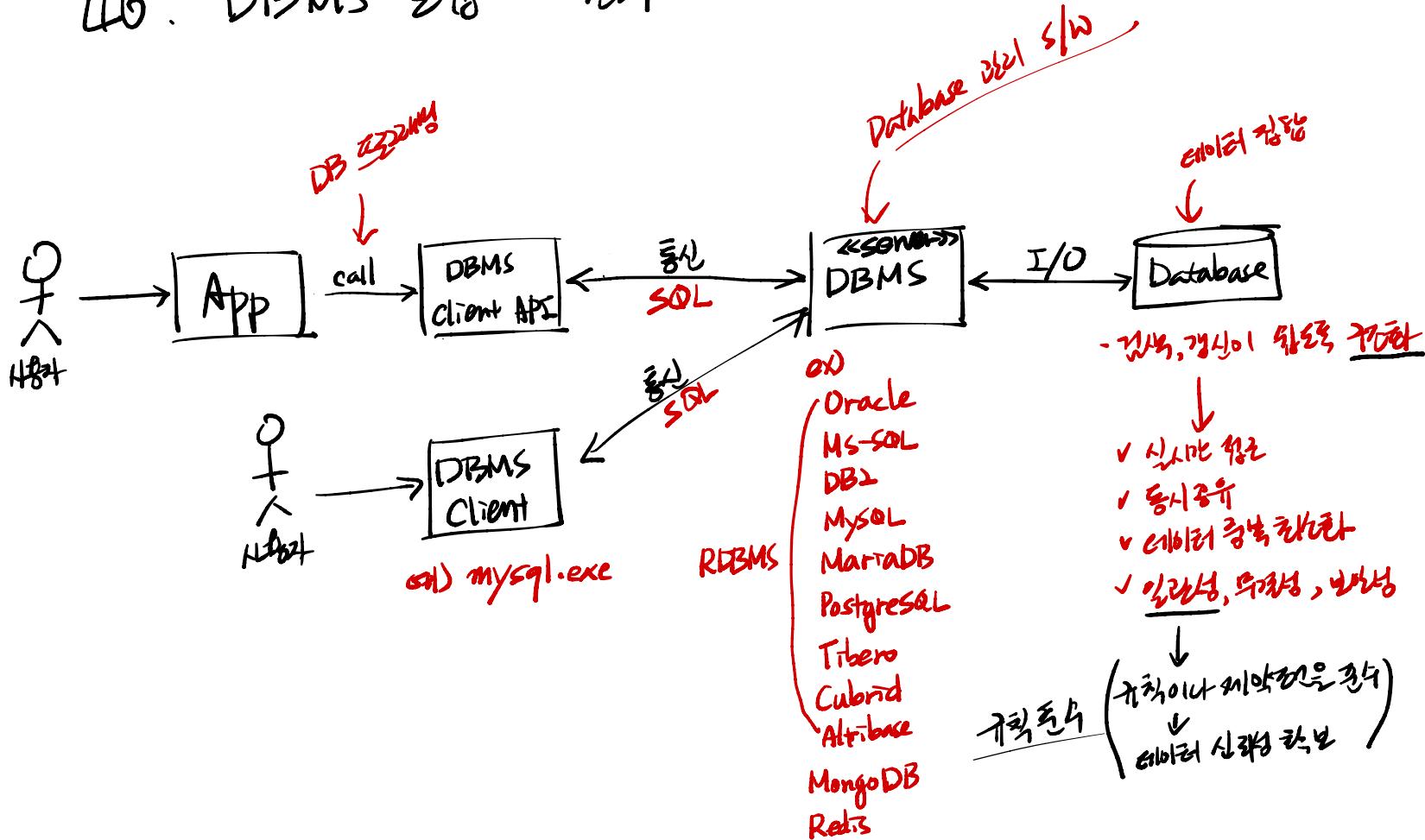
}

}

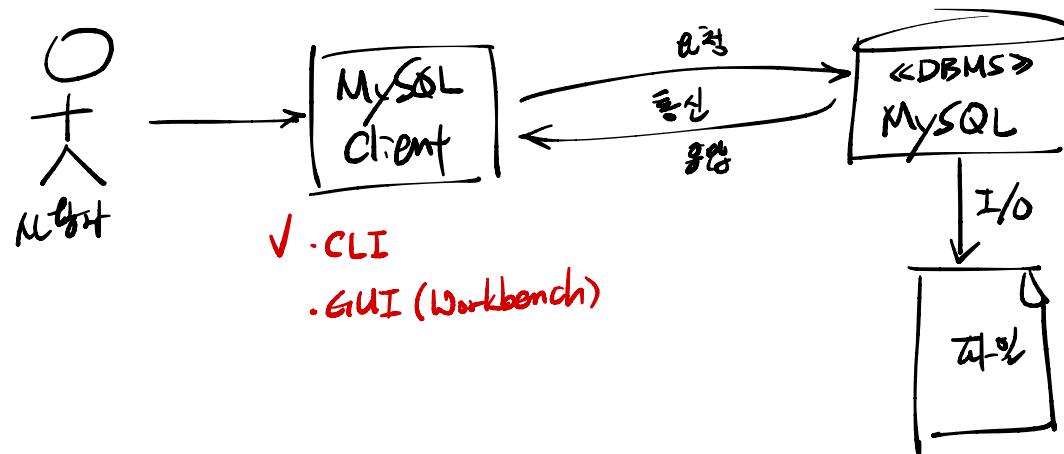
쓰레드.start();

여기서 쓰레드가 시작된다.

40. DBMS 솔루션 - 토론



40. DBMS 속성 - MySQL 설치



40. DBMS ဆို - DBMS အသေ

① 320J

≠ mysql -u root -P
DBMS 클라이언트 user id 암호입력

```
$ mysql -h 127.0.0.1 -u root -p
```

④ DBMS 사용자 관리

③ 01012181011 484

create database globe11010101 default character set utf8 default collate utf8_general_ci

~~도시락을 사면서~~
~~이제는 막수~~
"드디어 왔어!"

40. DBMS ဆို - DBMS ဘာ

④ 2019년 11월 12일 18:35

(4) 2010년 10월 10일 그 끝을 알리다

show databases

⑥ *entolichnion* ab K_2Si

use until 2018/10/15/2018

② 테이블 목록 조회

show tables

⑨ 경마복 상마복

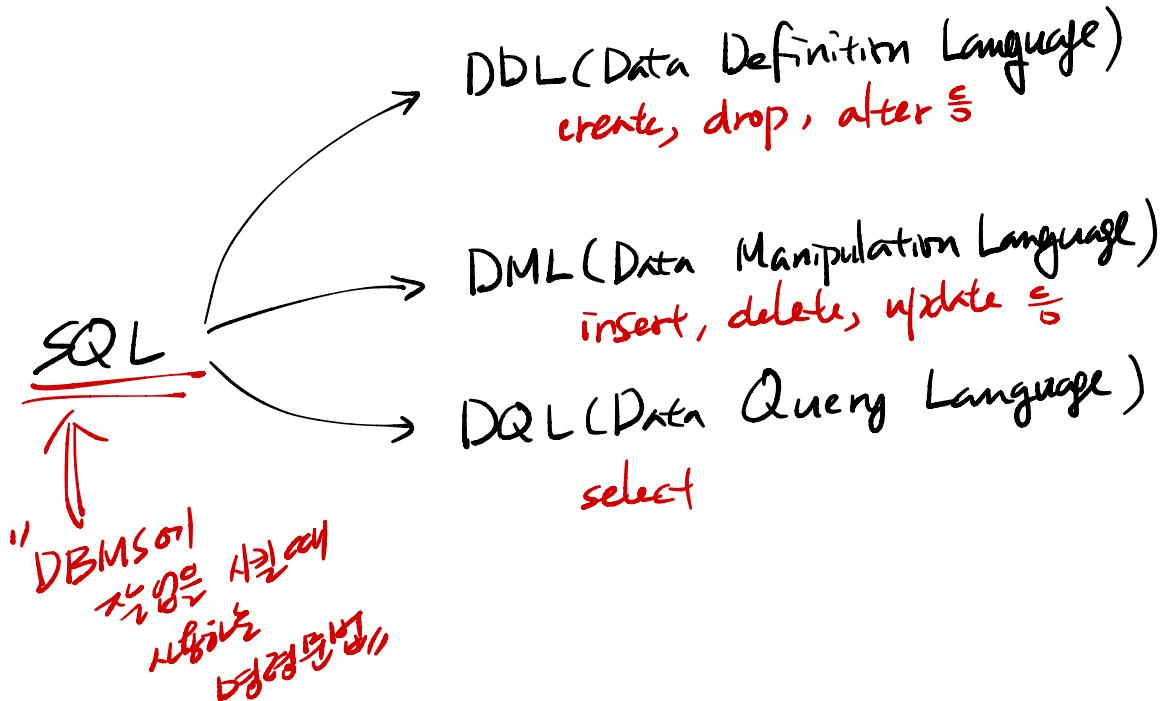
describe 仔이를 묘사

desc ကြပ်လျှော့များ

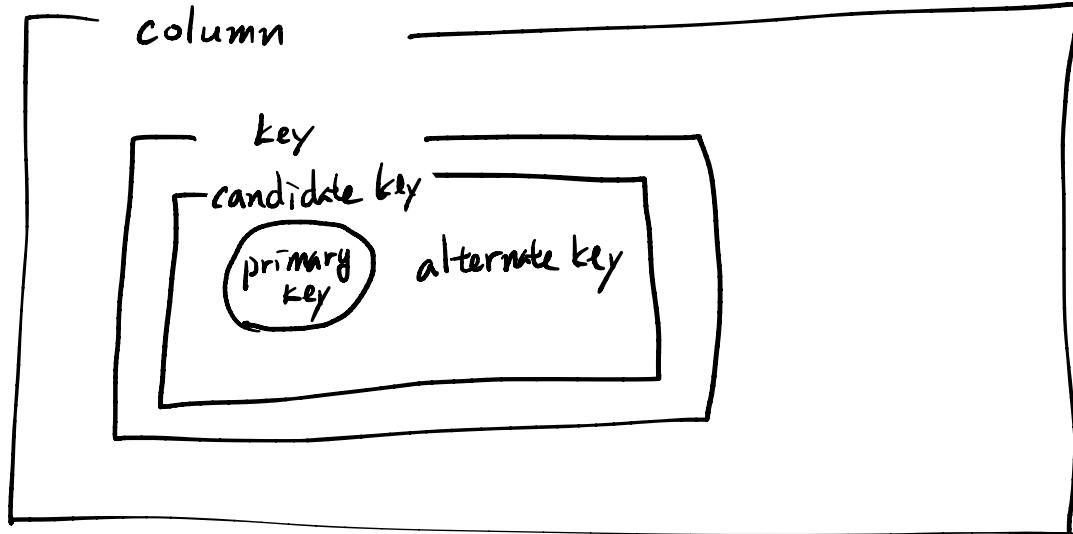
⑨ 人體, 骨盆

select user, host from mysql.user

110 . DBMS ୱେଳେ - SQL (Structured Query Language)

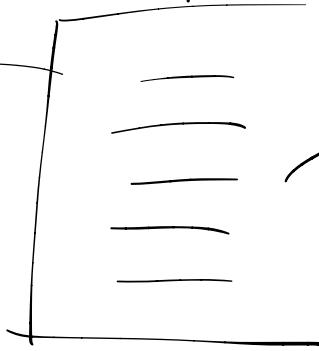
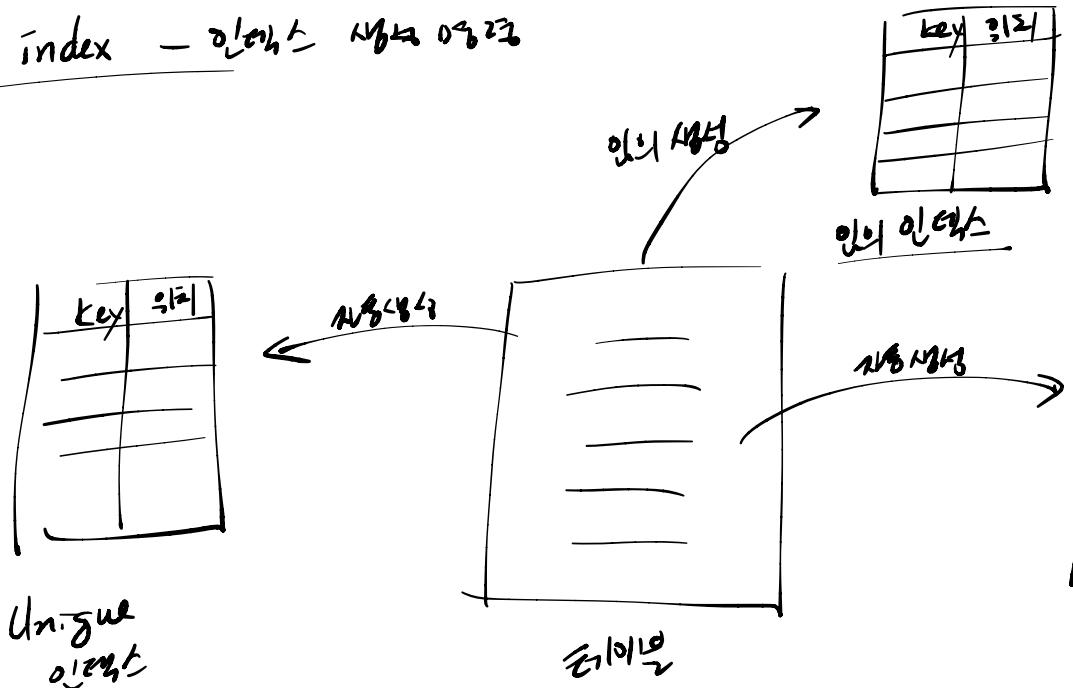


* key - 데이터를 구분할 때 사용하는 키임



(인공키)
artificial key
" "
surrogate key
(서로이기)

* index - depth $\approx 10^3$



Unique
object

현대문

~~key~~ 2121

primary key
주키

primary key
인덱스
1) 인덱스
2) 풀 텍스트, 챕터 텍스트
ElasticSearch

* projection vs selection

선택
제거
select.m

no	name	class	working	tel
			F	
			N	
			N	
			F	
			N	

"projection" - 테이블에서 원하는
필드만 선택

select no, name, class
from test1;

select no, name, class
from test1
where working = 'F';

"selection" - 테이블에서 원하는
행만 선택

(102)

* view et table V_2^{ex}

select *
from worker

view

~~view 2
M-15+ an
mru 2
mru 2
mru 2
mru 2
mru 2
mru 2~~

~~object !~~

filter
filter

select no, name, class
from test1
where working = 'Y'

select

no	name	class

no	name	class	working	fe

* char(n) vs varchar(n)

데이터베이스를 생성할 때
지정한 charset에 따라 한글자의 메모리 크기가 결정된다.

정형 → char(5) 
지정된 글자 개수가 상관없이
무조건 5글자를 차지할 메모리 크기 갖는다.

غير정형 → varchar(5) 
글자 개수 만큼
메모리 크기를 갖는다.

* row et column

번호	이름	이메일	주민번호	성별
1	홍길동	hong@	1111	010-
2	김민정	leem@	2222	010-
3	유비누	yoo@	3333	010-
4	이정자	ahn@	4444	010-

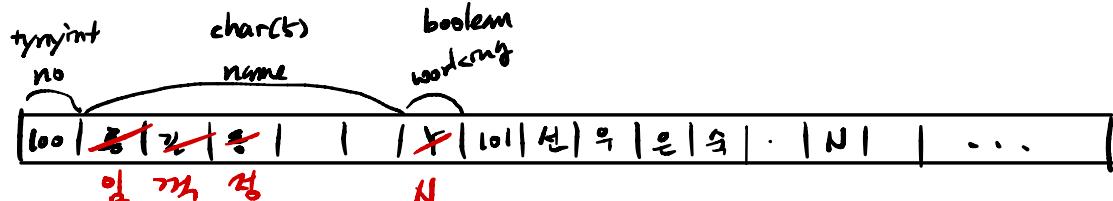
row = record = tuple

column = field = attribute

* 파일과 연결/삭제

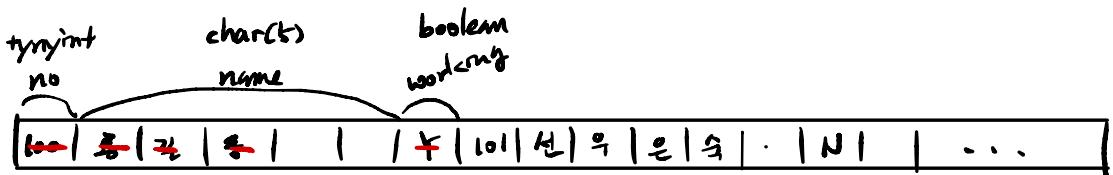
자연 이야기 1번정

↳ 기존 학습을 재학습
부분은 예상



자율 놀이터 설치

↳ 그간 갚을 사랑한
의원을 찾기로



- ↑
○ 으로 쓰여야 시기다

▶ 아이디를 제거하려면
위의 아이디를 앞으로 쓰여야 한다.

↓ ↳ 뒤에 있어야 한다

○ 다른 글자나 시그니처가 있다면
그 글자 그대로 아이디와 짜증인 부호를
○ 으로 뒤에 쓰는 것이다.

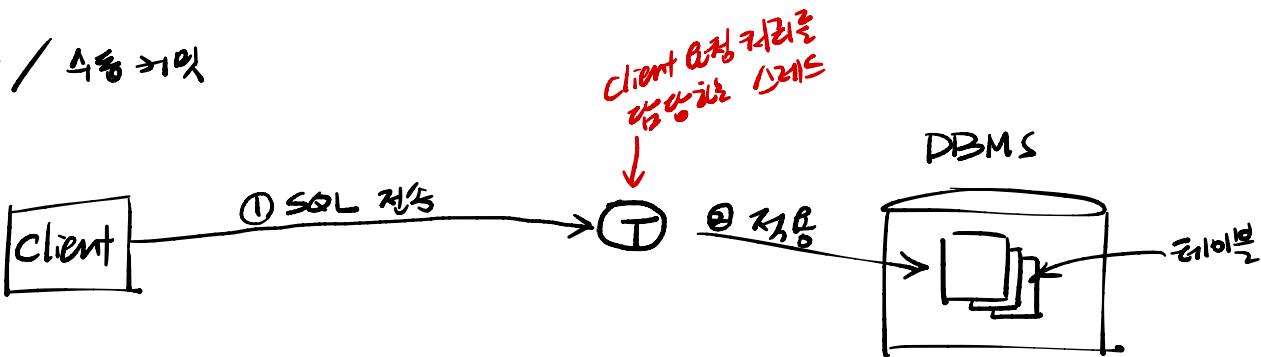
자본의 특성 부분을
살펴볼 수 있다

- ① 설교내용 부제를 제1주제 대이어로
 나타내거나,
② 제2주제 위의 대이어로
 부제를 부제으로 이어가야 한다.

* autocommit / 속도↑이익

① autocommit

(insert
update
delete)

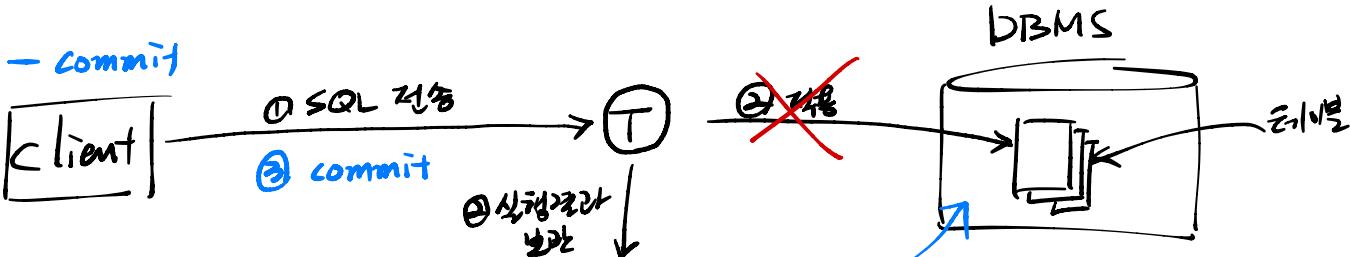


데이터를
동시에
변경할 때
자동으로
커밋
된다.

⇒ 어떤 상태로
되돌릴 수 있다.

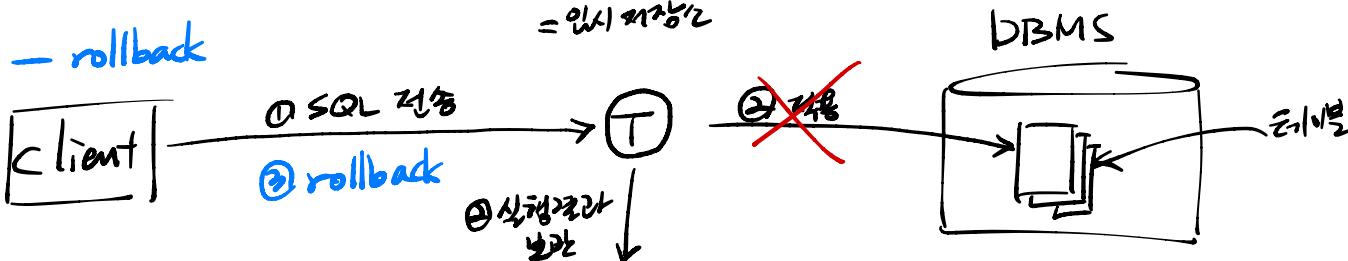
② 푸시 commit - commit

- insert
- update
- delete

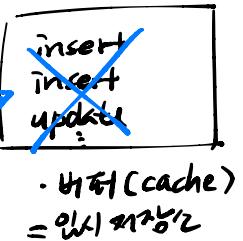


② 푸시 commit - rollback

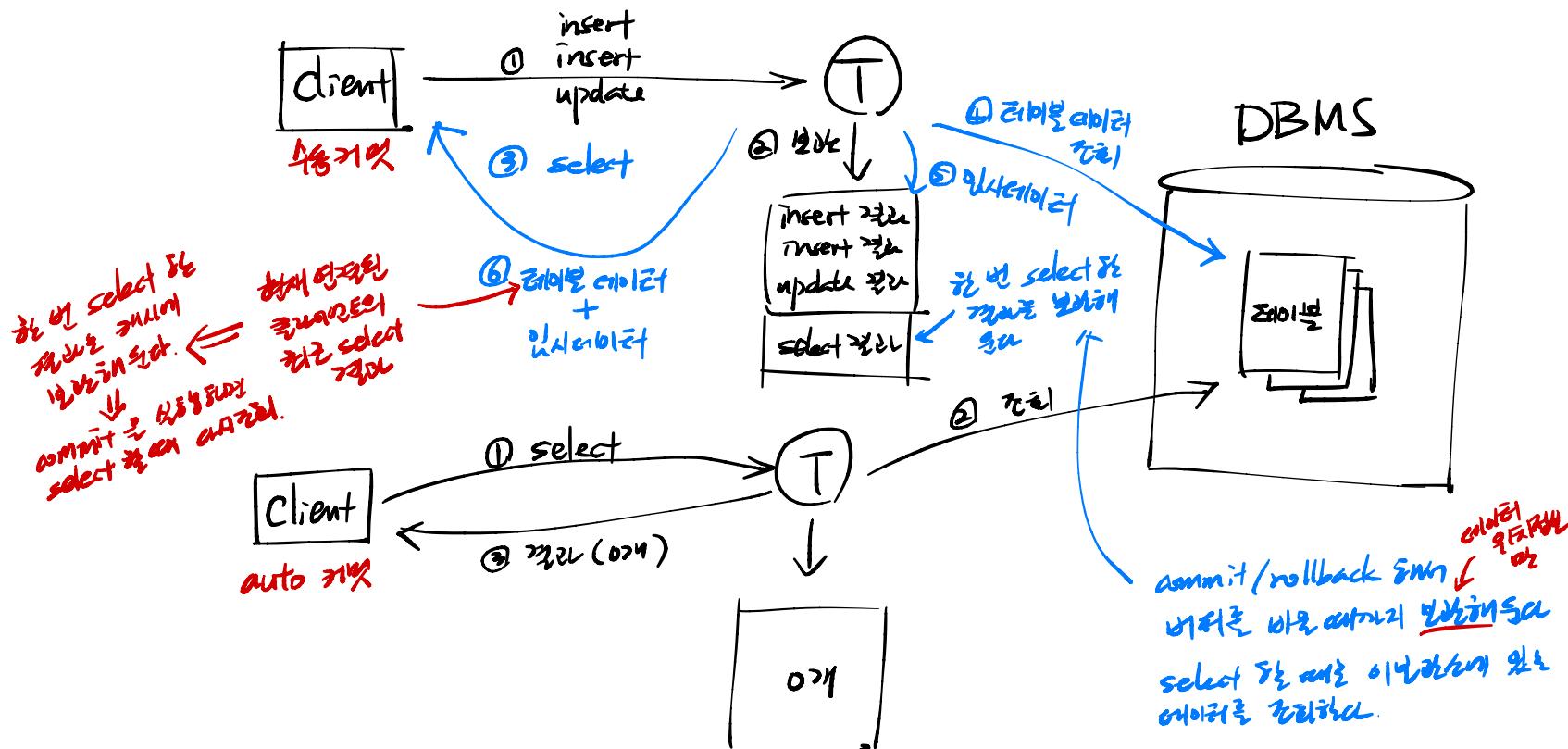
- insert
- update
- delete



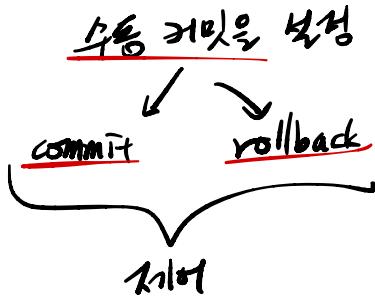
④ DB에서 일시 저장되었던
SQL 명령어를 실제 데이터베이스에 적용!
그러면 데이터를 바꾼다.

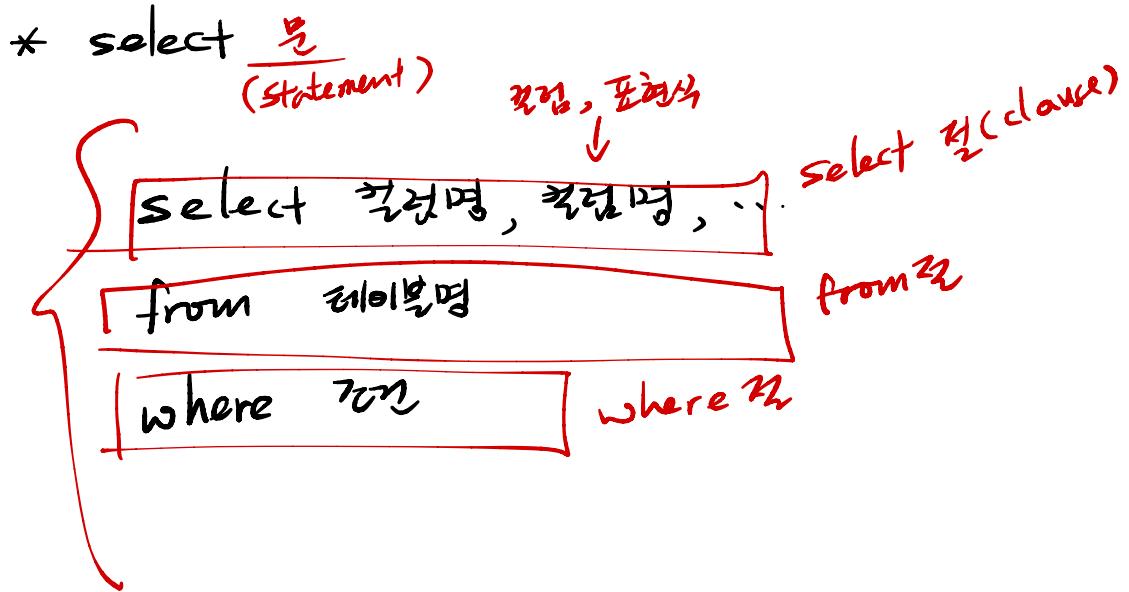


* client et commit



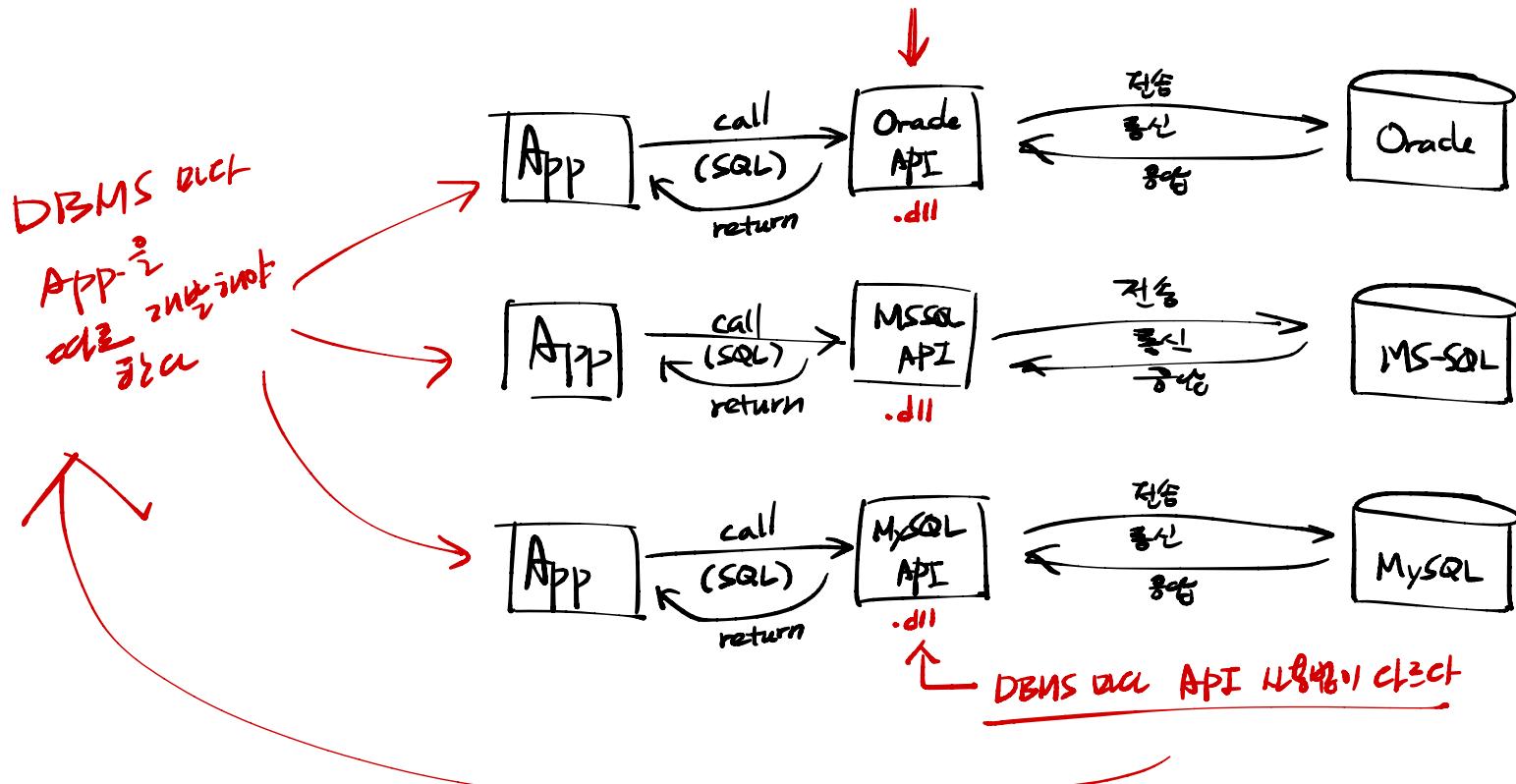
- * 트랜잭션 (transaction)
 - ↳ 여러 개의 작업을 한 단계로 묶은 거.
(insert/update/delete)





* DB 접근방법 - Vendor API = Native API

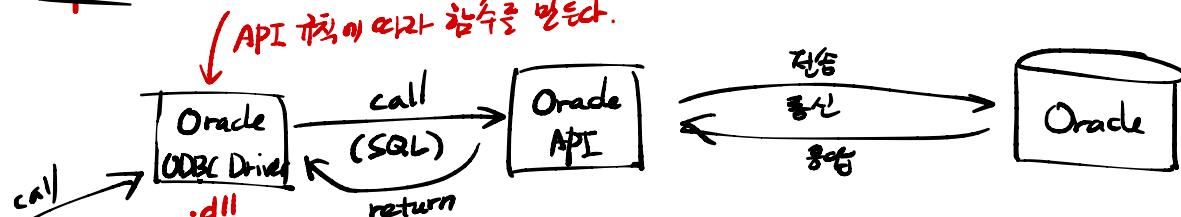
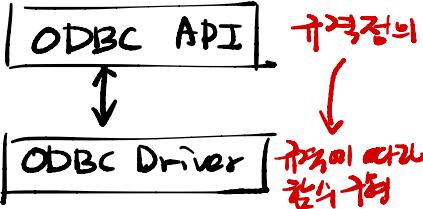
Vendor API = Native API (C/C++)



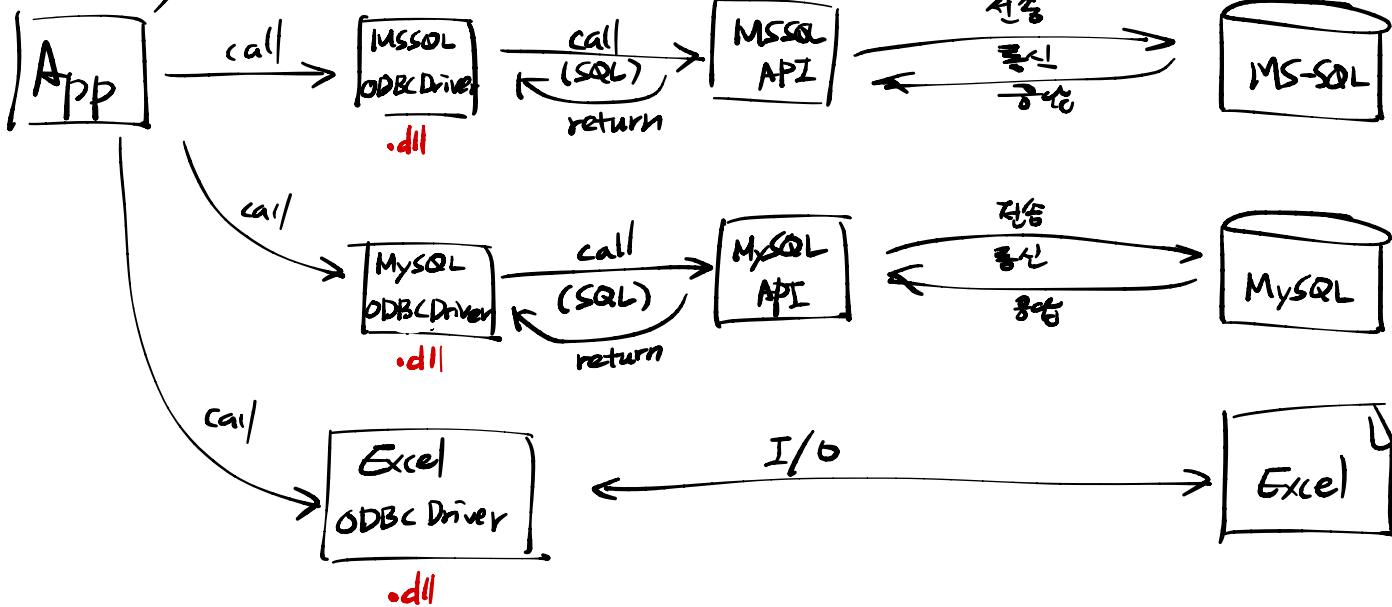
* DB 접근 방법 - ODBC API ← API 를 통한 (C/C++)

Open Database Connectivity

(API 틀에 따라 활용할 수 있는 드라이버)

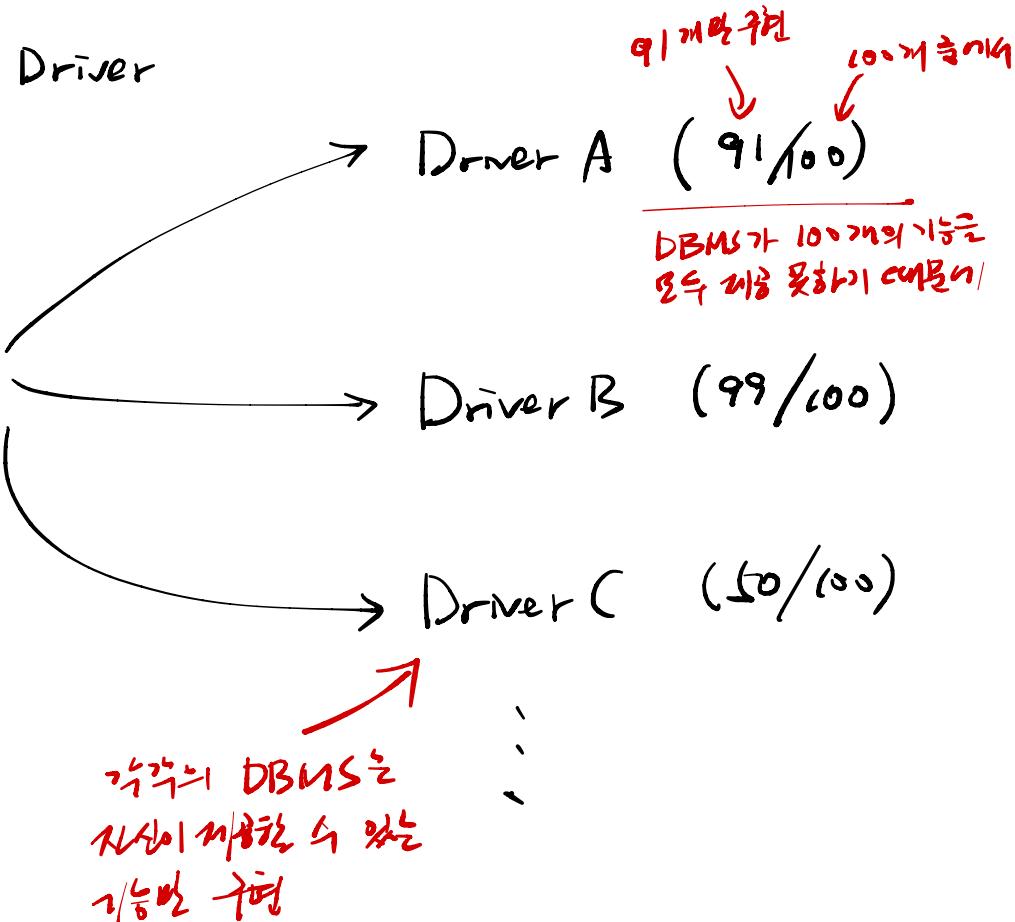


API 사용법
기본적인 접근
DBMS 드라이버
직접 API를
접근하는 것
아니면



* ODBC API 와 ODBC Driver

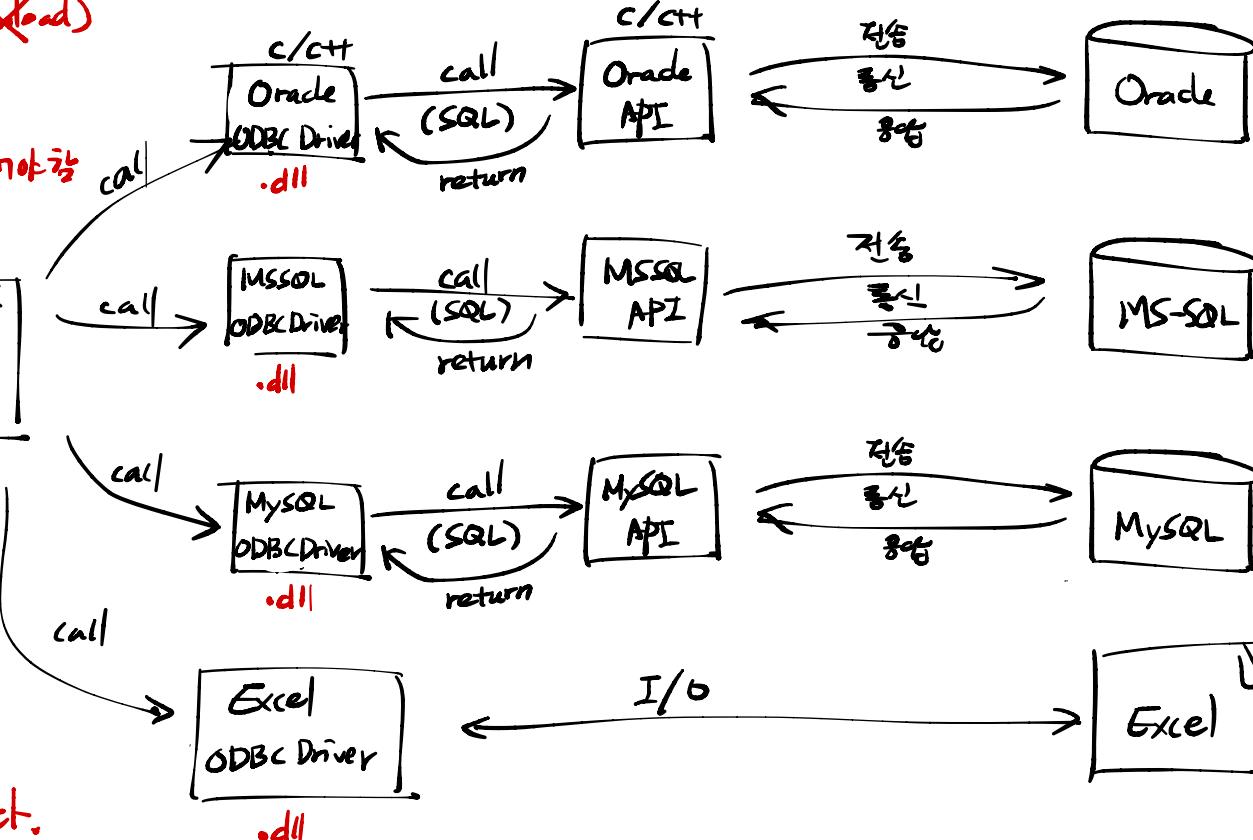
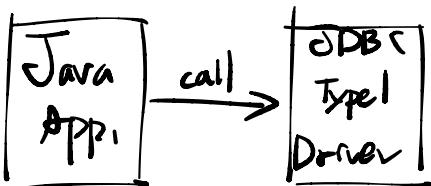
ODBC API
① 100개 품목의
기능 중의
기능을 정의하는
DBMS가 지원하는 기능
기능을 모두 지원하는
기능은 거의
기능은 거의



* DB 접근방법 - JDBC API "Type I" \Rightarrow ODBC-JDBC Bridge Driver
 Java Database Connectivity (Java API. 여러 DBMS에 대한 DB 접근 API)

Type I (1단계)

- JRE에 기본포함 (down load)
- ODBC API 툴킷 (C 함수)
 - |- OS에 종속적,
 - |- ODBC Driver 설치되어야 함
- 성능 좋지 않다.



* JDBC API

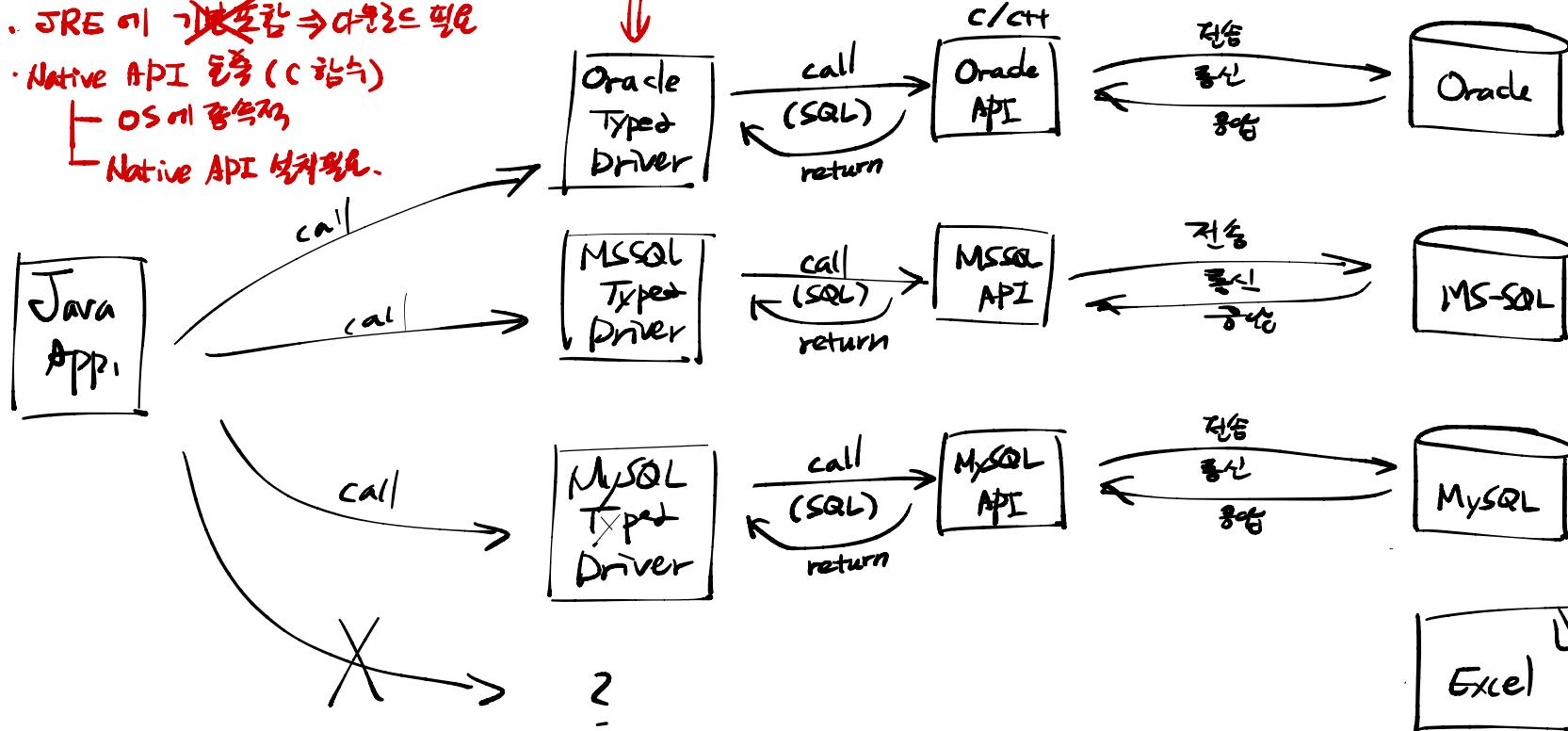
- DBMS에 자체화된 드라이버가
OS에 구애받지 않고
DBMS에 상관없이
일관된 방법으로
접근할 수 있게 해준다.

* DB 접근방법 - JDBC API Type 2 \Rightarrow Native API call Driver

Type 2 (2유형)

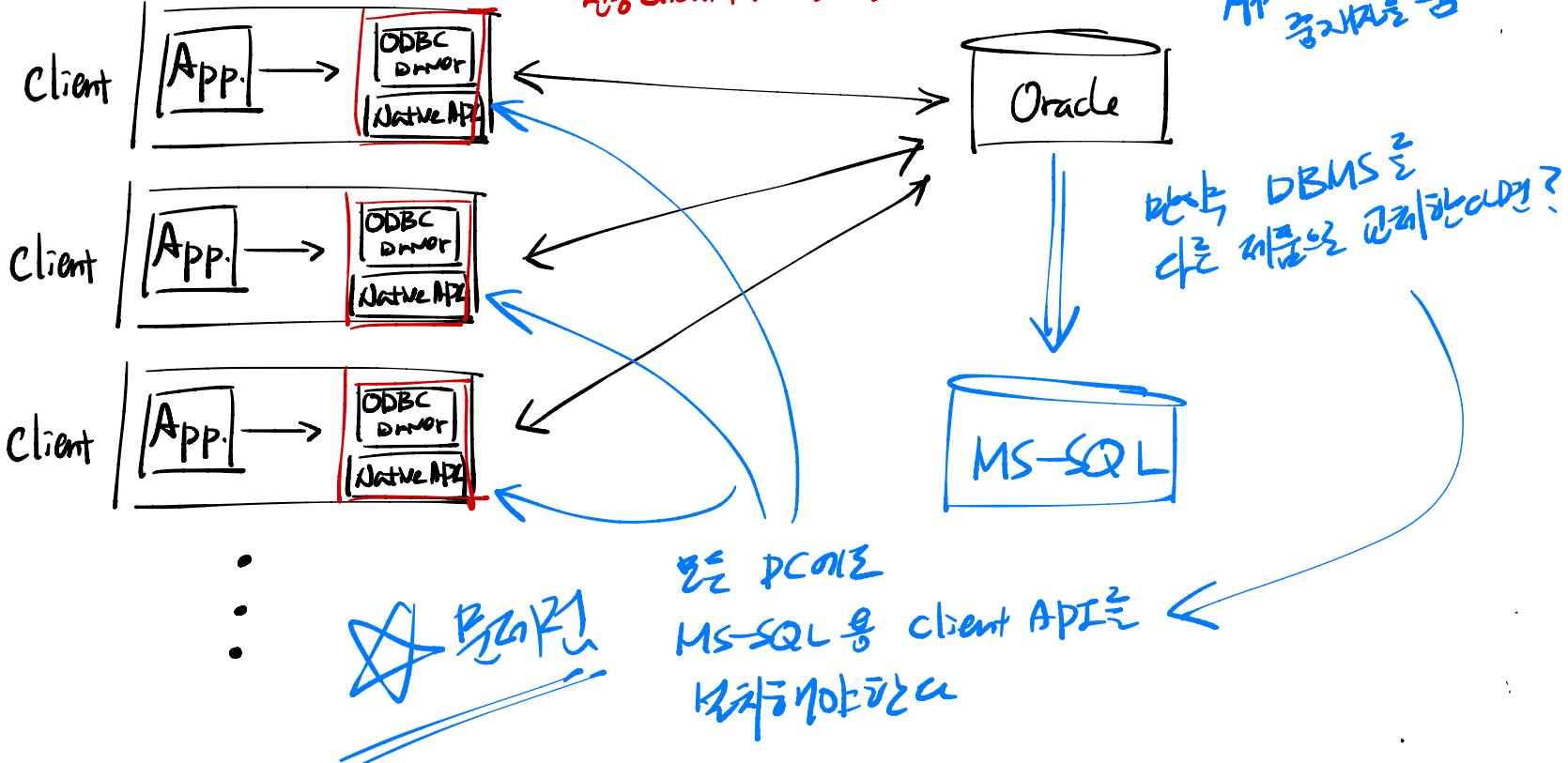
- JRE 외 ~~기본 제공~~ \Rightarrow 다른 드라이버 필요
- Native API 툴킷 (C 풀수)
 - OS에 종속적
 - Native API 사용 필요.

JDBC API 규격이 빠트리 Vendor API 풀수



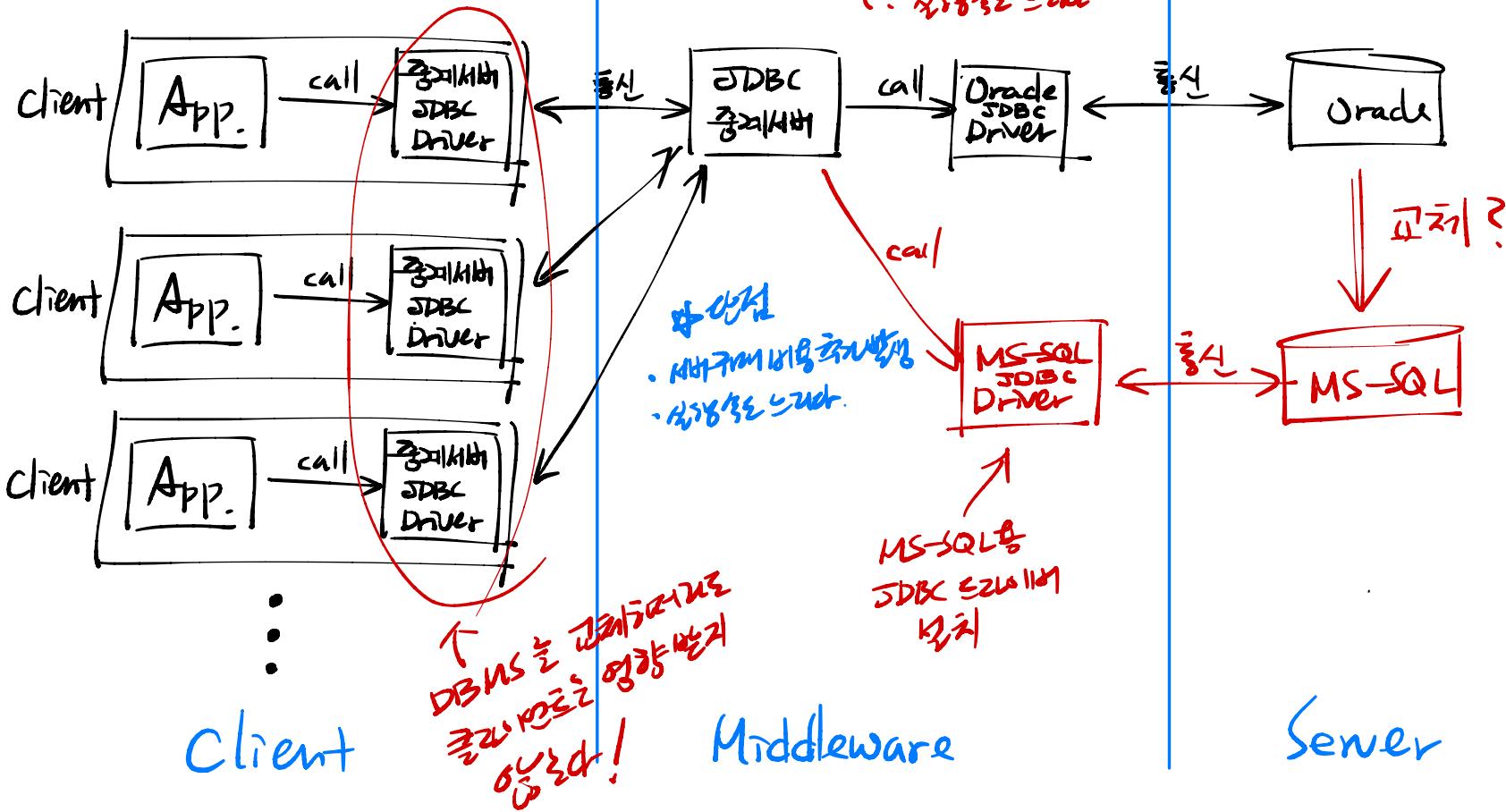
* DB 접근방법 - JDBC API Type 3 \Rightarrow Middleware Driver

1) Type1, Type2 Driver의 단점 \Rightarrow DBMS에 접속하기 위해
Client API 사용 필요



* DB 프로그래밍 - JDBC API Type 3

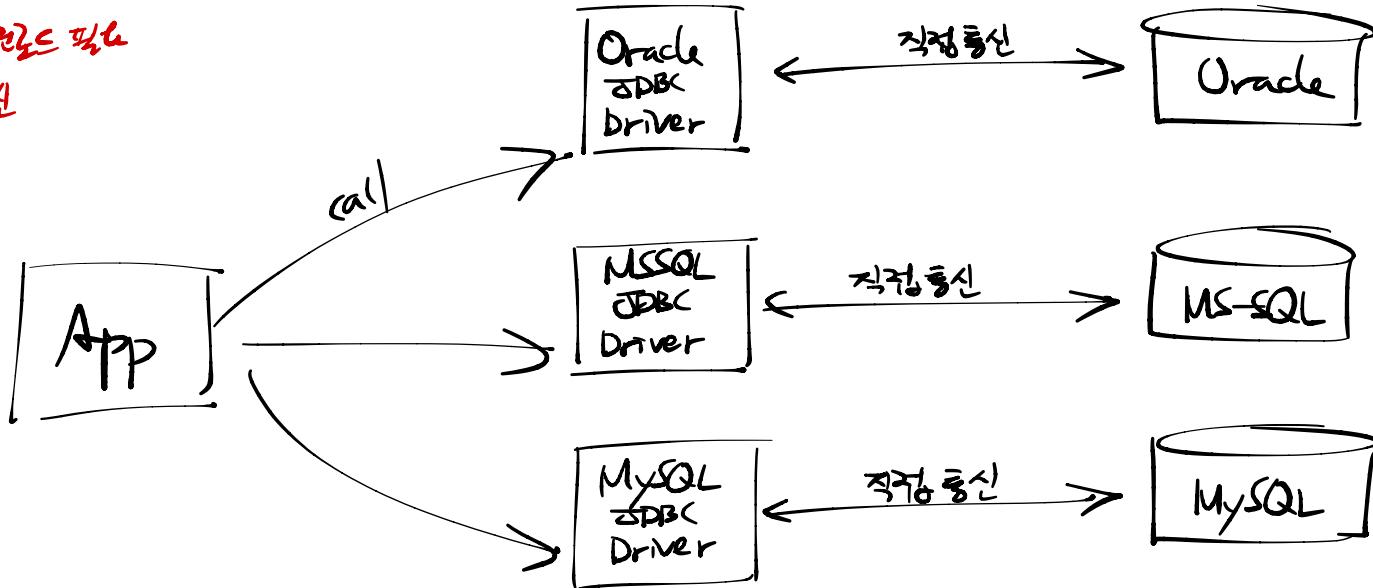
- { 기반 사용 → 구매 품목
- 중재서버 품목
- 설정값 등록



* DB 프로그래밍 - JDBC API Type 4 ⇒ Network Protocol Driver

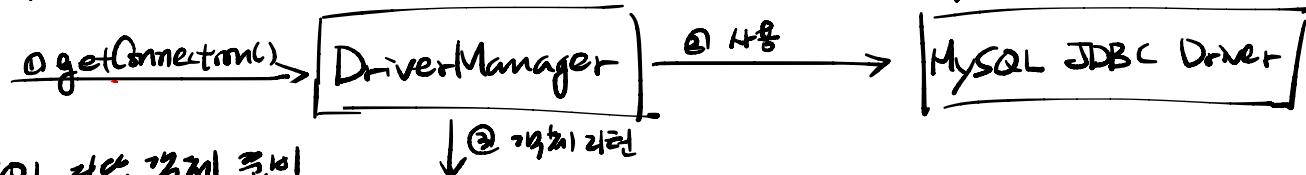
JDBC Type 4

- DBMS 용 디렉트 플러그인
- 시내부 직접통신

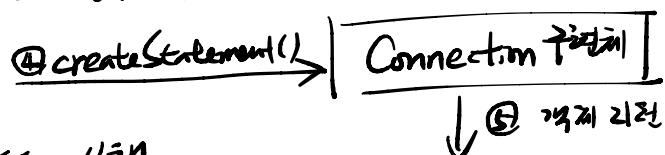


* DB 프로그래밍 - JDBC API 사용법 : insert/update/delete 처리

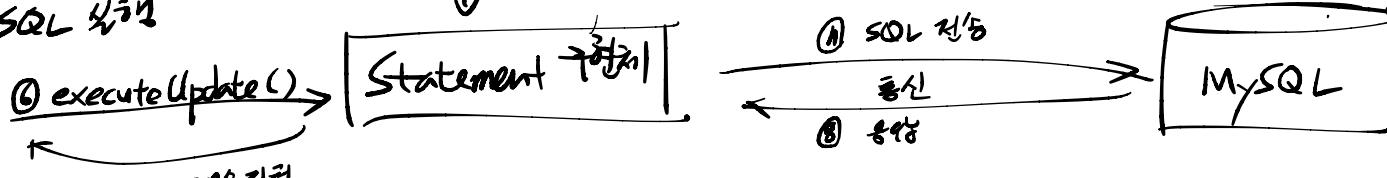
1) DBMS 연결



2) SQL 처리 객체 준비



3) SQL 처리



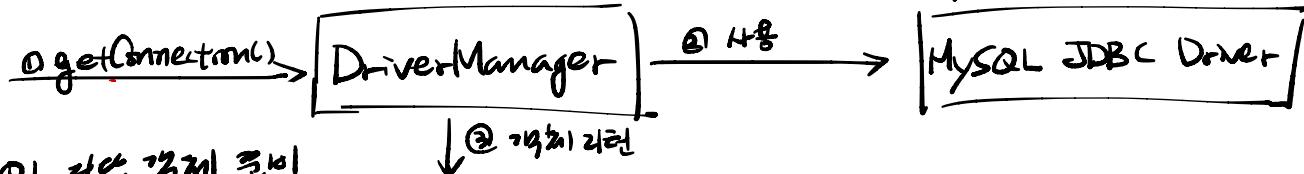
insert → 행 추가
update → 행 수정
delete → 행 삭제

① SQL 처리
② 통신
③ 응답



* DB 프로그래밍 - JDBC API 사용법 : select 처리

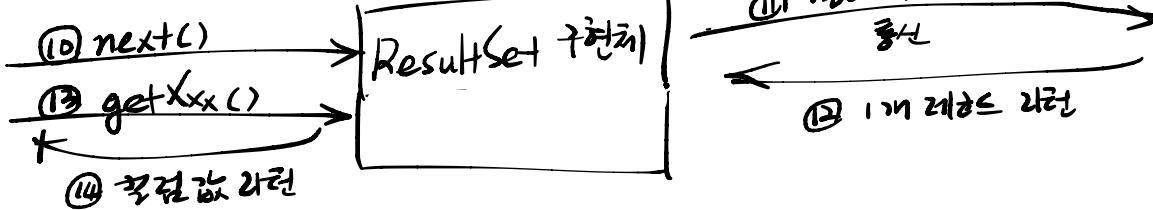
1) DBMS 연결



3) SQL 처리



4) 결과 가져오기



* DAO et JDBC Driver

