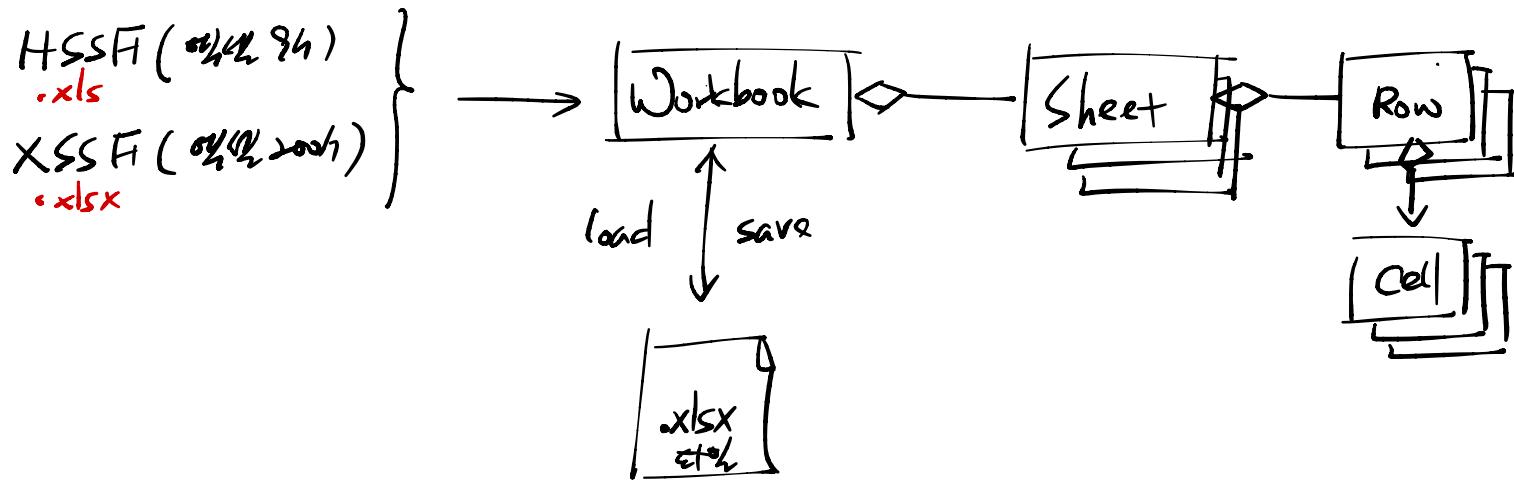


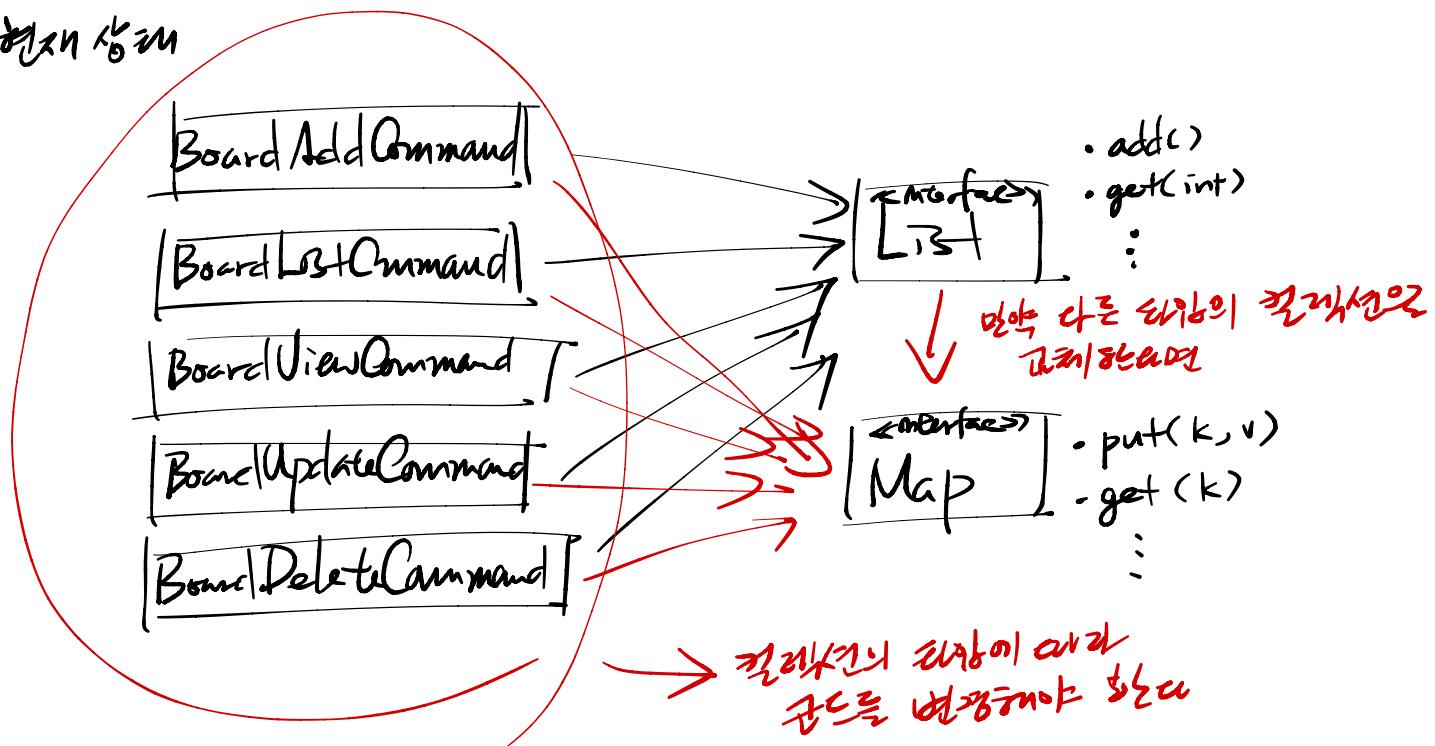
33. 콜렉터는 어떤 원칙을 따른다? : Apache POI 소개



34. DAO 가 필요한 이유

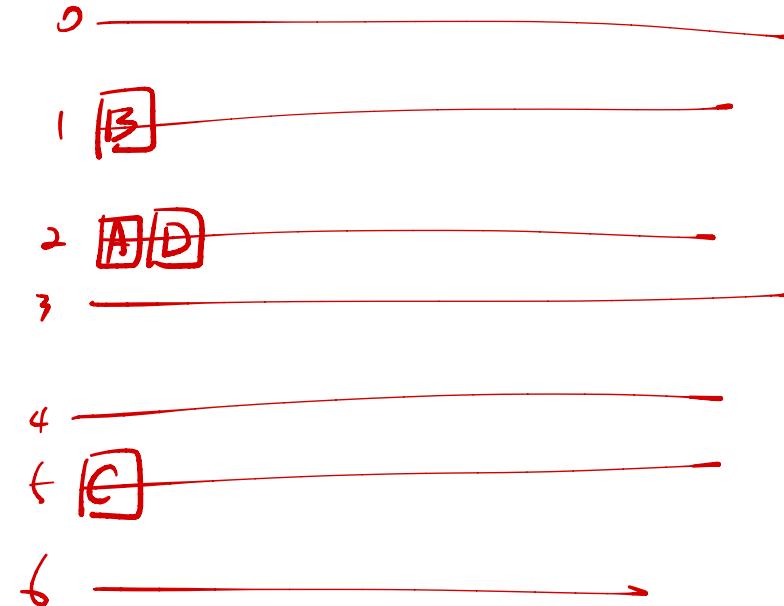
↳ 데이터 베이스를 관리하기

① 테이블



* Map의 데이터를 순회할 때는 항상
꺼내서 접근해야!

Map



② A

8-B

5-C

23-D

values()

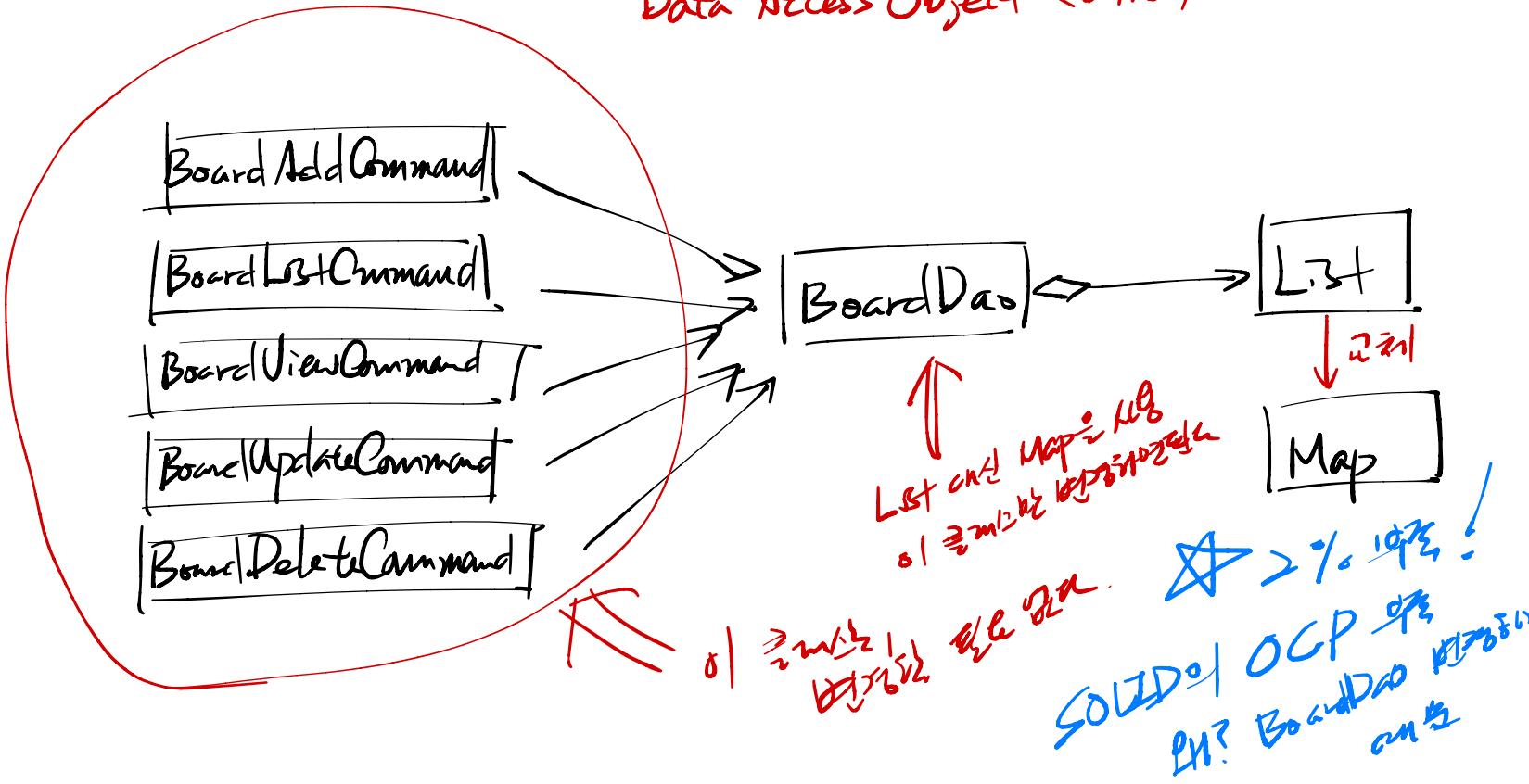
B A D C

① A B C D E
key의 hash 값을 가지고
인덱스 위치를 계산하는 방법

35. 글로벌 키워드를 제거해보자

↳ 제거한 키워드는 뭘까? → 함수가 기반이다.

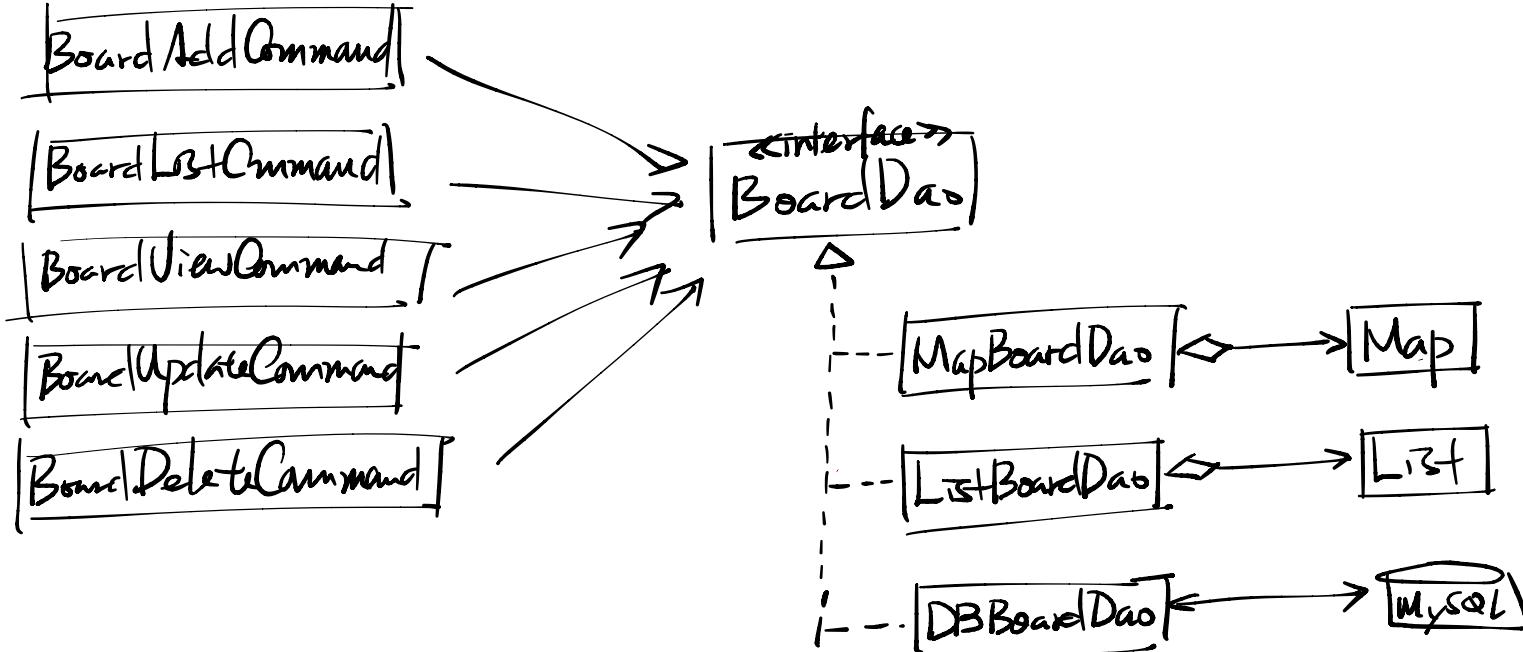
Data Access Object (DAO)



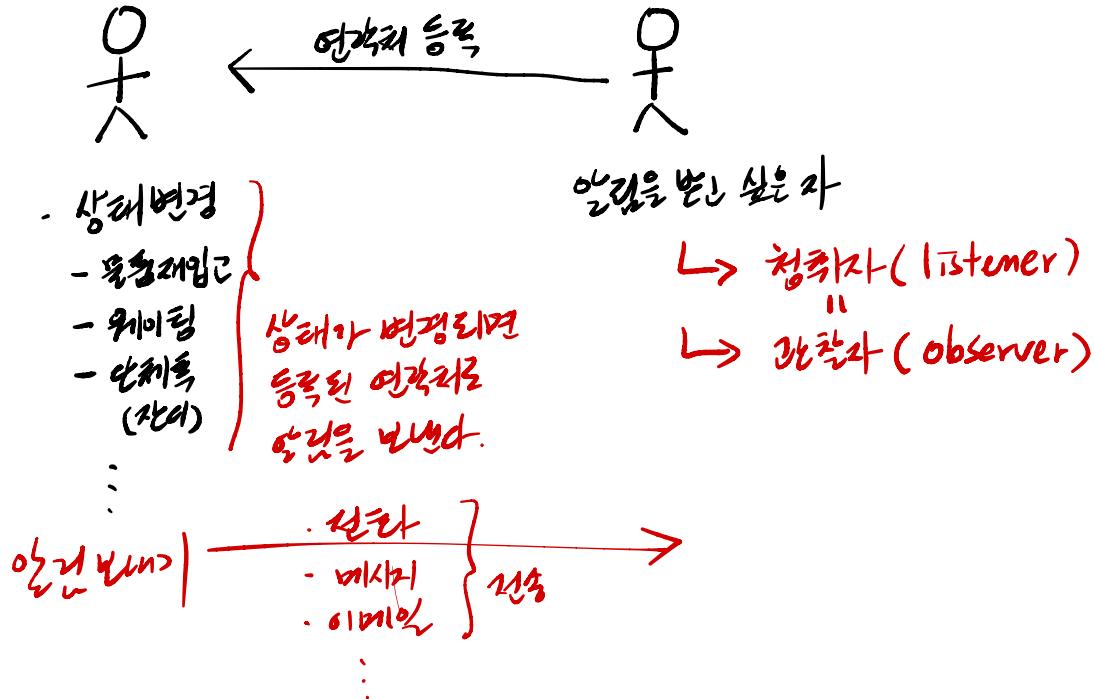
35. 글로벌 키워드는 final이면 됨

↳ 제한된 개수의 경우 → 제한된 개수의 제한된 개수.

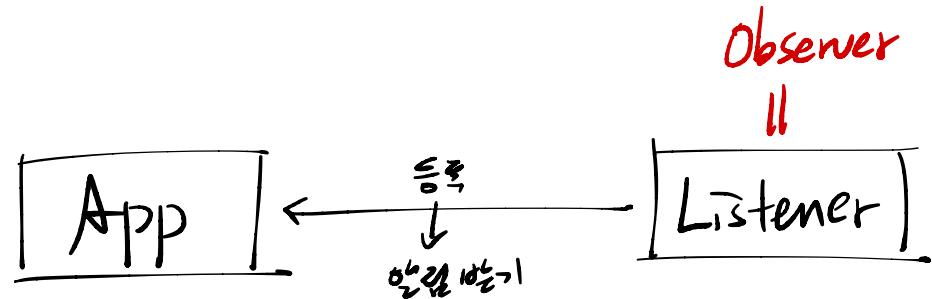
Data Access Object (DAO)



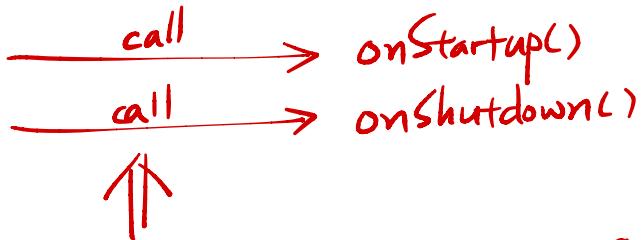
* 36. 알림 패턴 : GOF의 Observer 패턴
(설명문서)



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

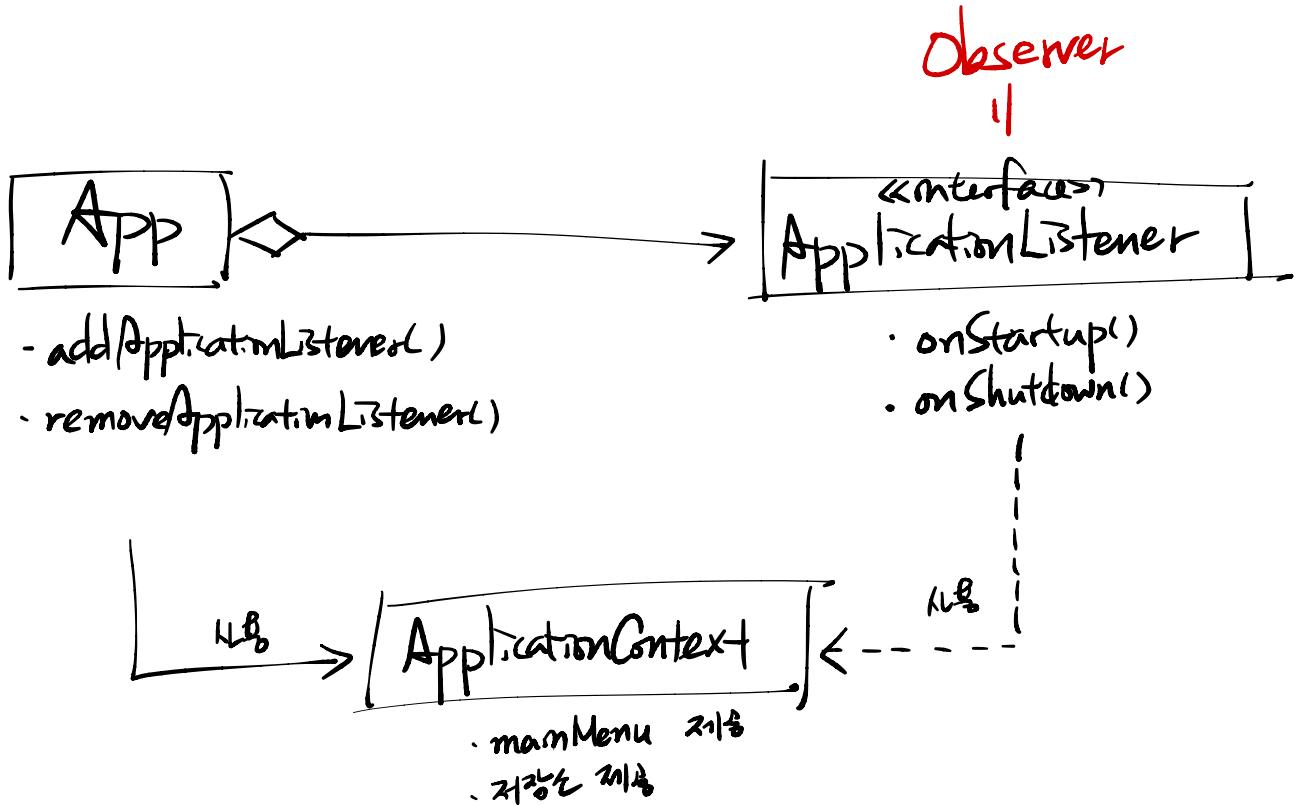


- 설정에 따라
- 사용자
- 풍경



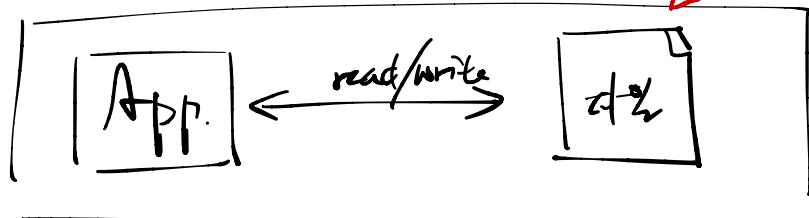
설정에 따라 설정에 따라 설정에 따라
설정에 따라 설정에 따라 설정에 따라

* GOF의 Observer 패턴 대처



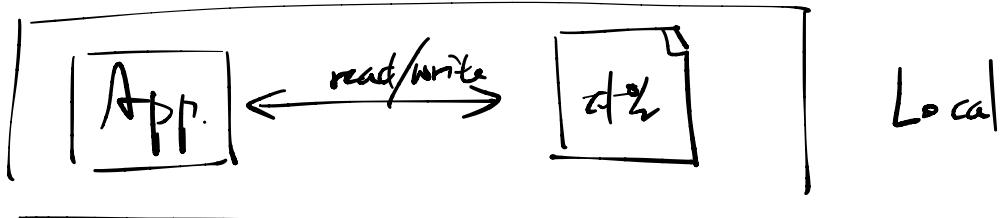
31. Application 간에 데이터 공유

현재 상태



App.의
데이터로 데이터를
Local

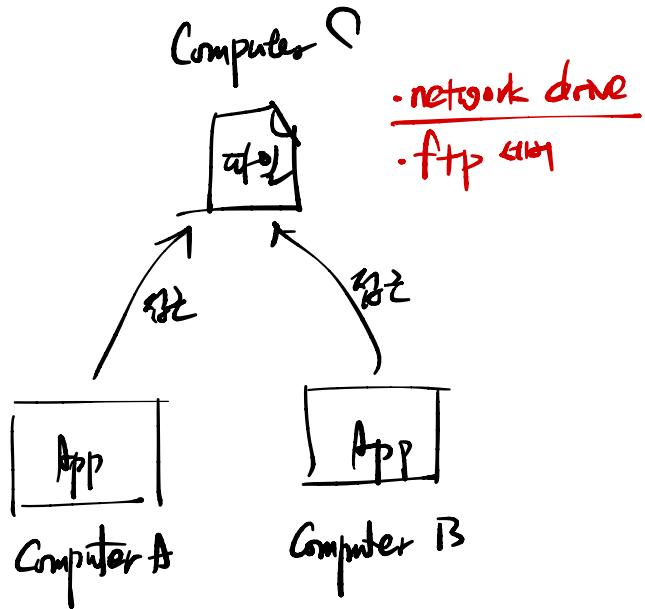
App.-간의
데이터 공유는!



•
•
•

31. Application 간에 파일 교환

① 파일로드된 파일 공유



※ 문제점

- 동시에 여러 App. 실행
문제점이 있다



파일을 네트워크에서 찾.

31. Application 간에 데이터 교환

② 멀티프로세스 애플리케이션

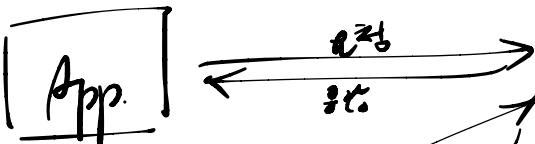
- networking prog.
- multi-tasking prog.
 - ↳ multi-threading
 - ↳ synchronous

애플리케이션
Data는 쓰고 읽을 때
Data를 주고 받음

Computer A



Computer B

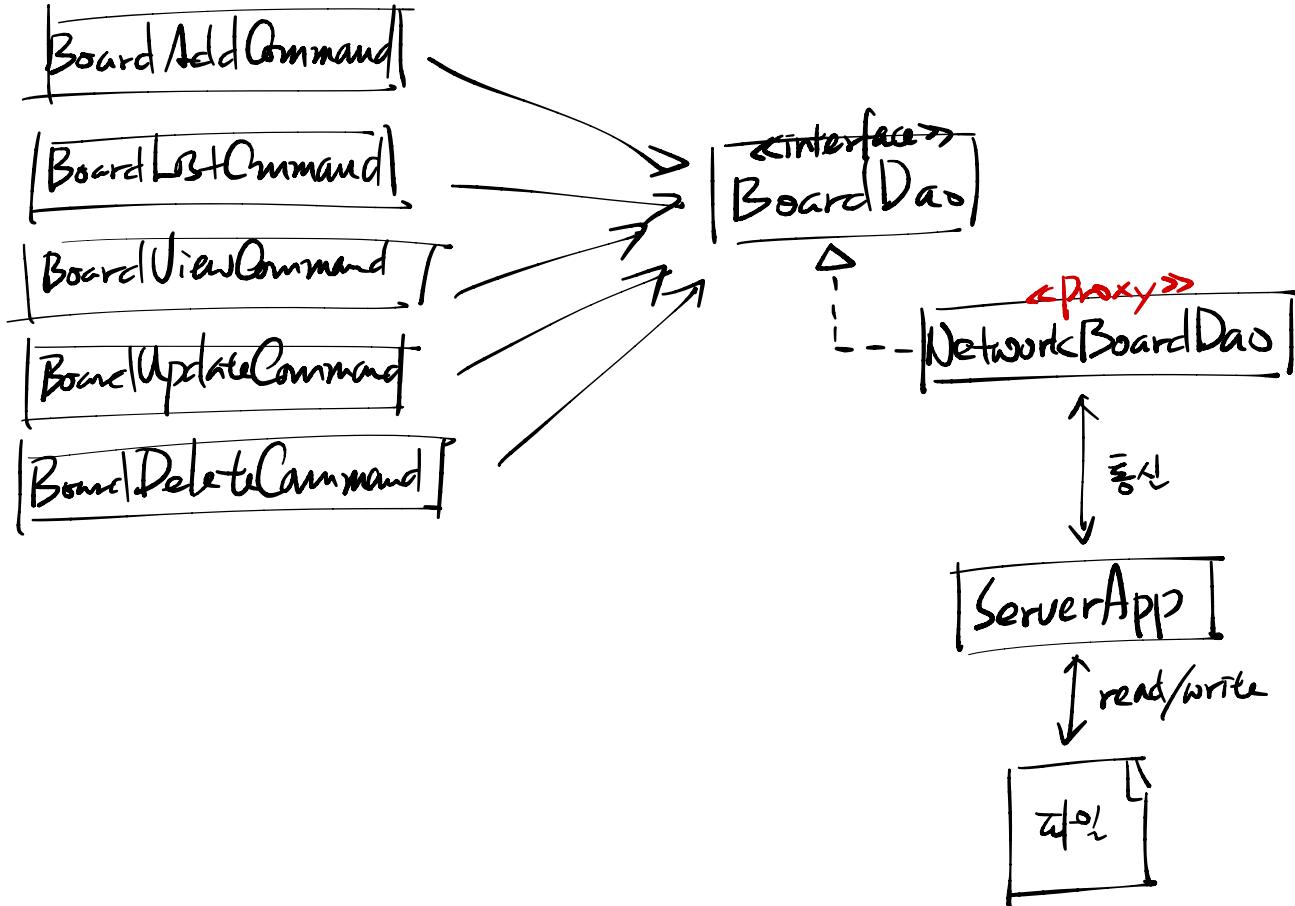


Computer C

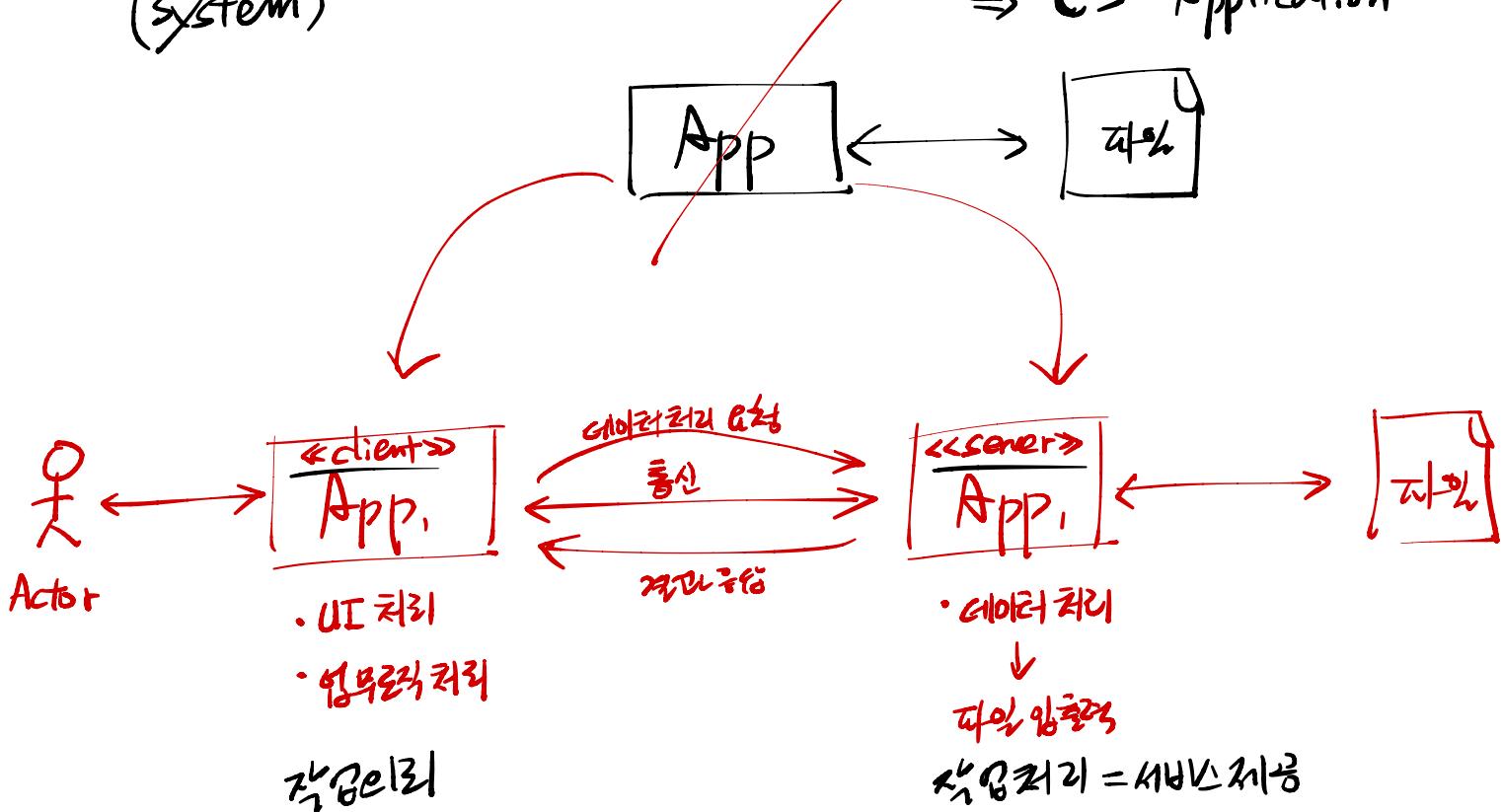


networking

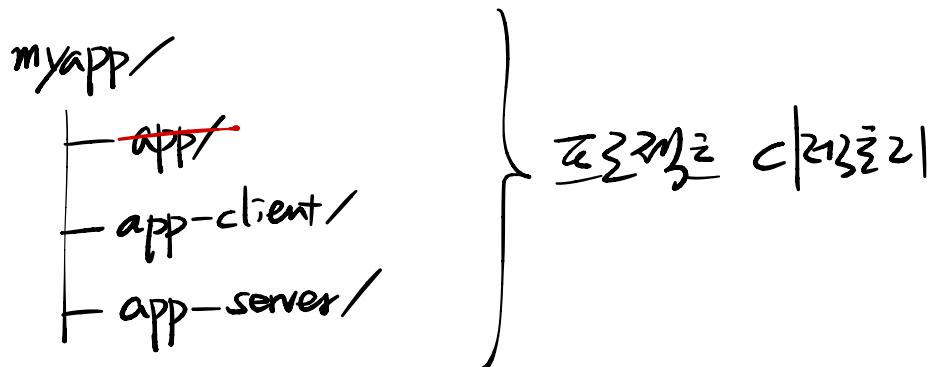
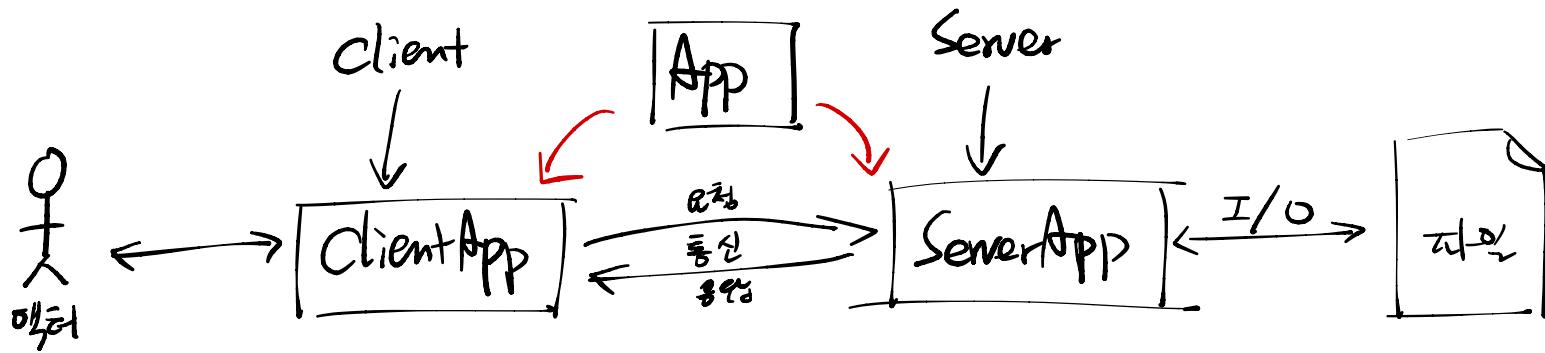
进程의 통신
进程의 통신 및 시스템 쪽
Higher level 프로세스



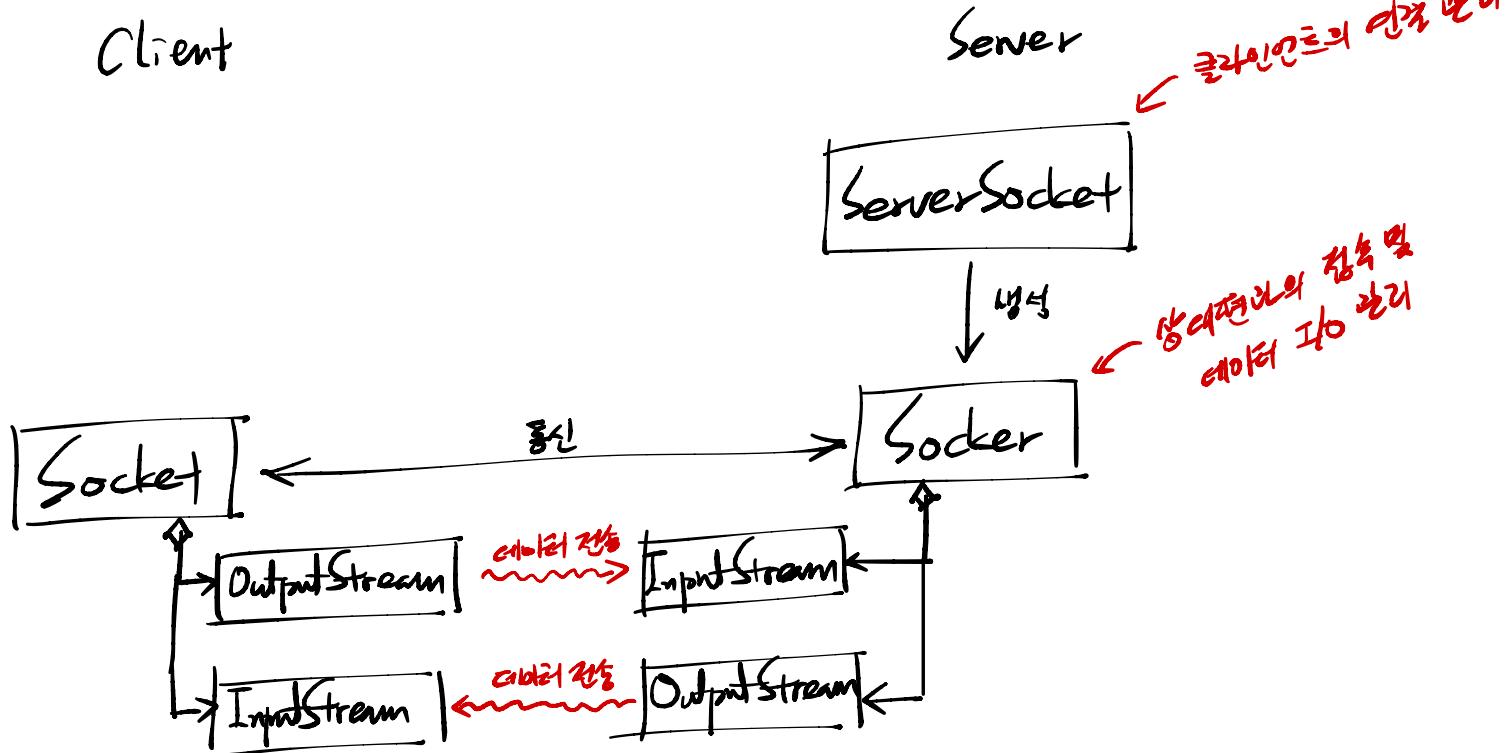
* Software Architecture : Client / Server Architecture
(System) ⇒ CS Application



* CS Architecture 구조

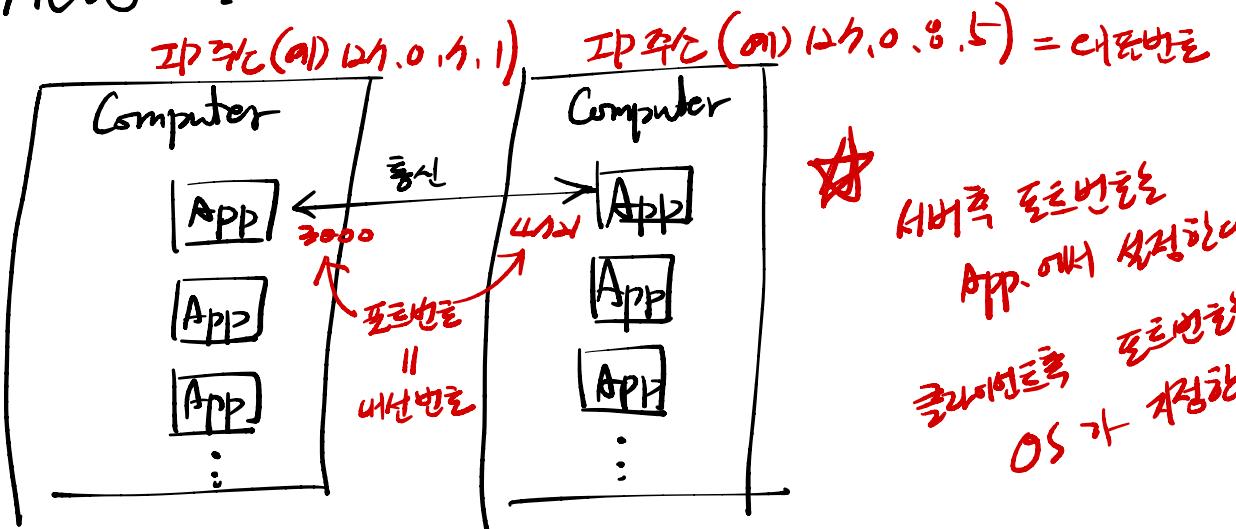


* Networking API API



* 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?
(로컬 IP)

OS가 제공합니다.

* 특별한 IP 주소

127.0.0.1

Local
주소
||
loopback 주소

||
"localhost"

* Protocol : 데이터 송수신 규칙

client

컬렉션 - "users"

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

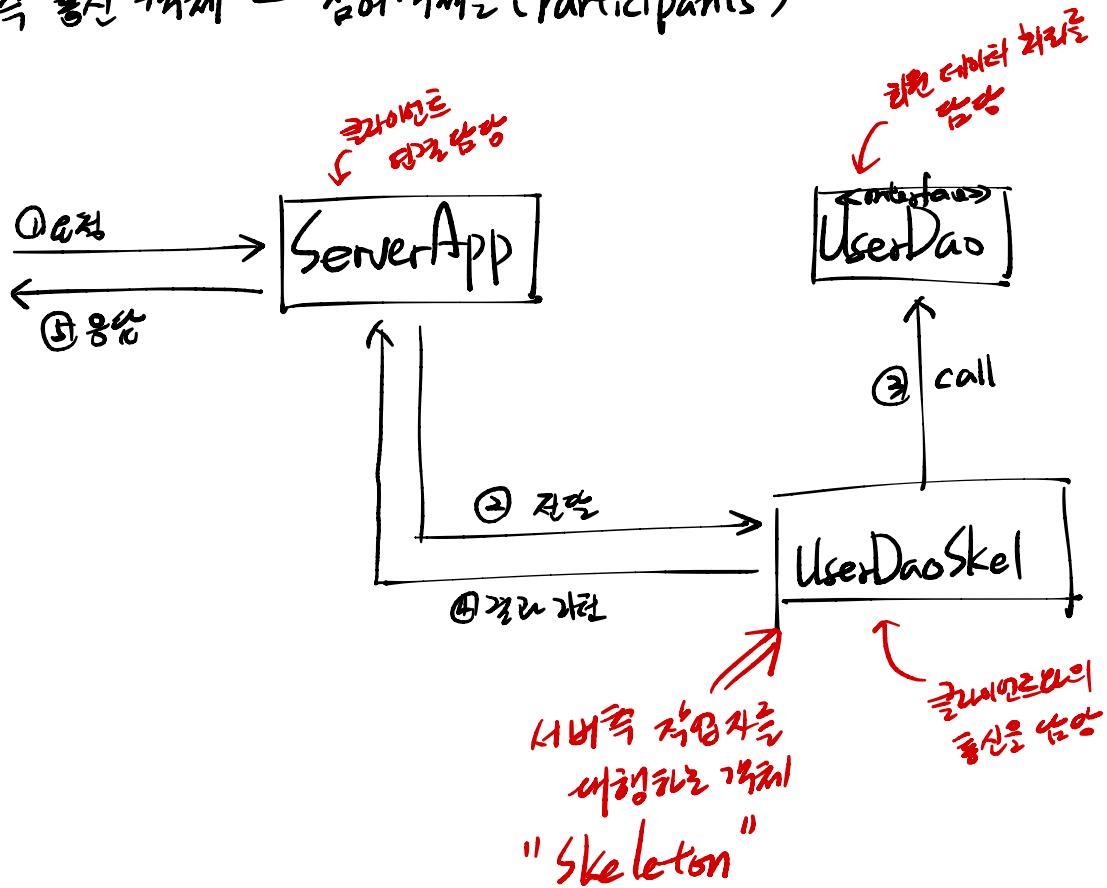
전송 데이터 - *

Server

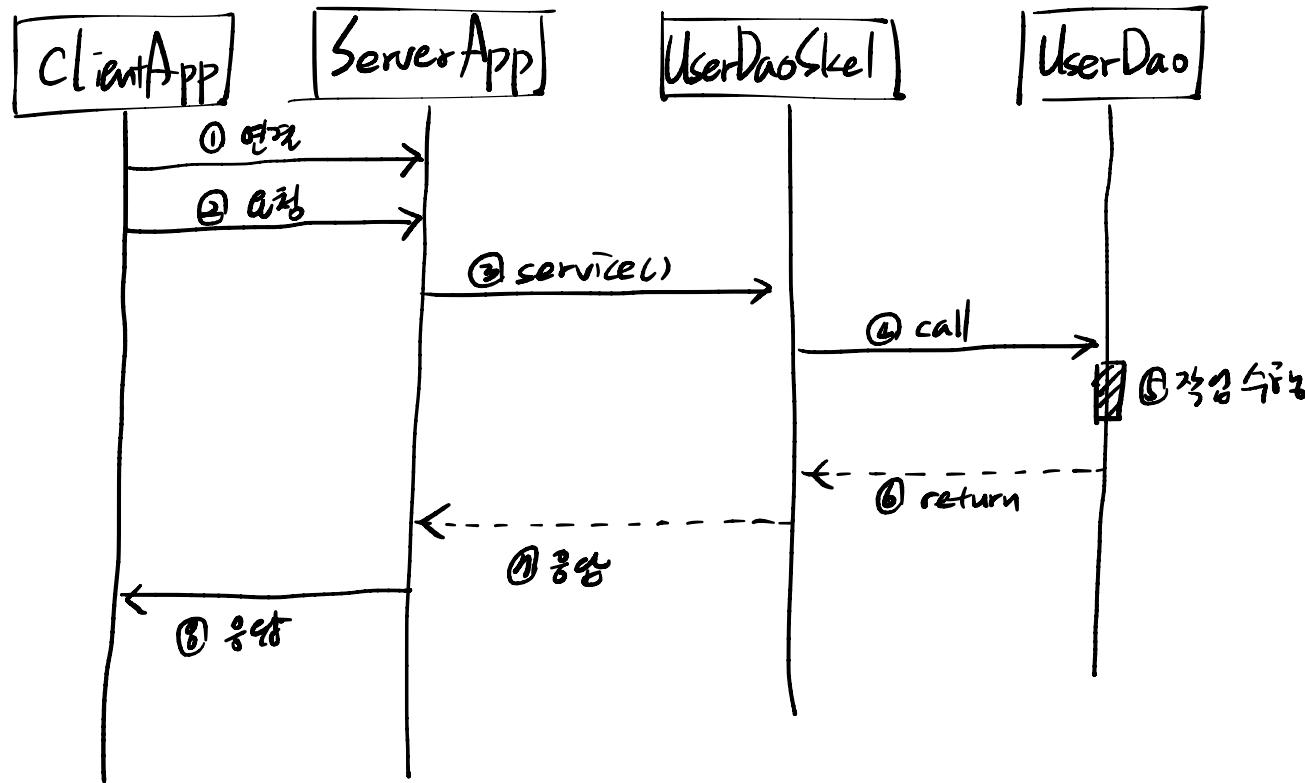
응답 상태 - "success" | "failure"

응답 데이터 - *

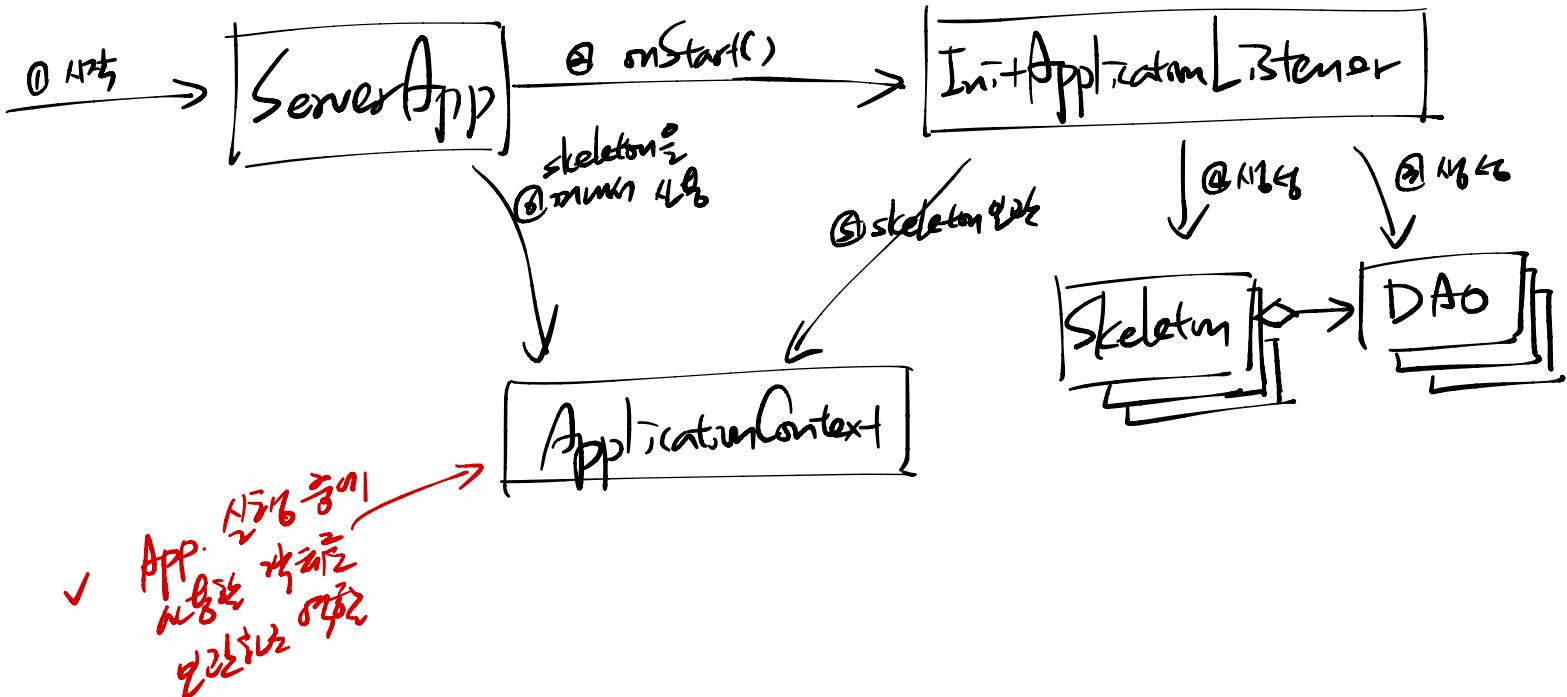
* 서버측 통신 구조 - 참여자들 (Participants)



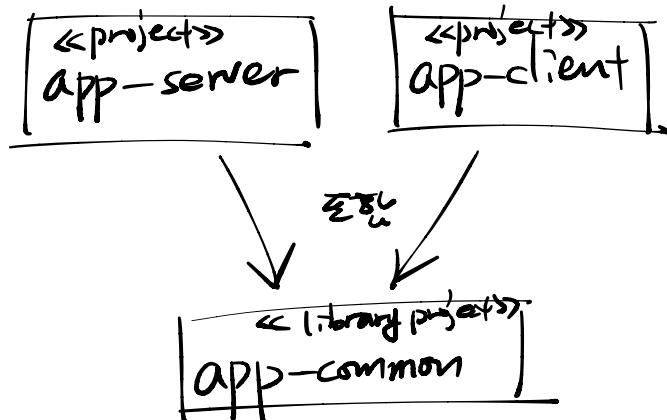
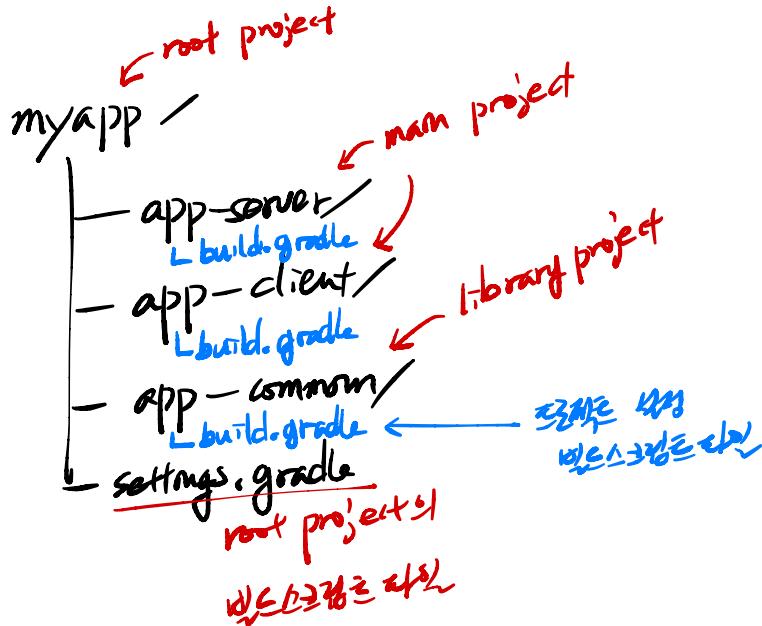
* 서버측 통신 구조 - Sequence Diagram (클라이언트가 서버의 서비스를 호출하는 과정)



* ServerApp in Skeleton

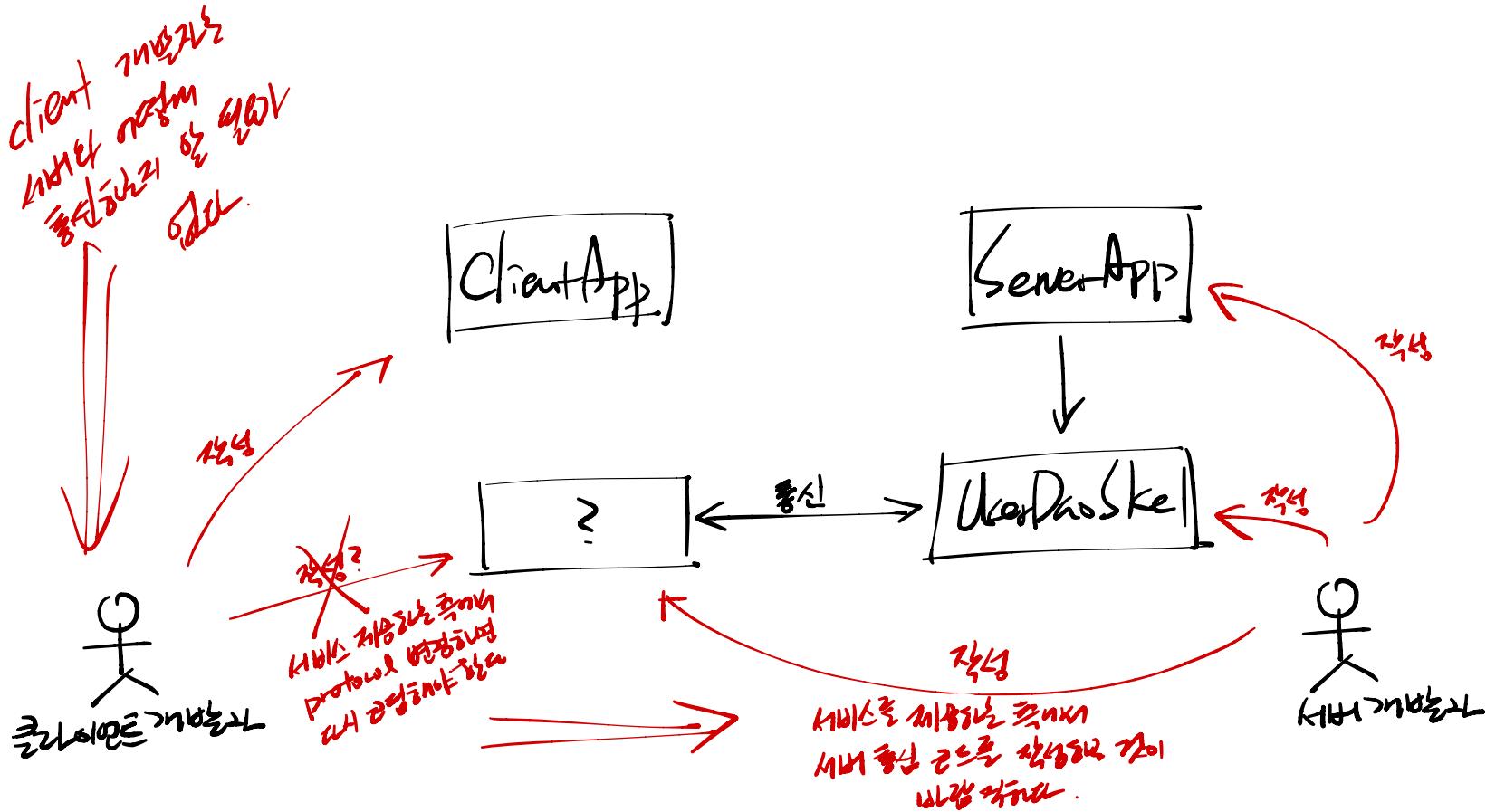


* 2019-2021 အချက် - အကျဉ်းချုပ်၏ ရေးဆွဲမှုပါမ်း

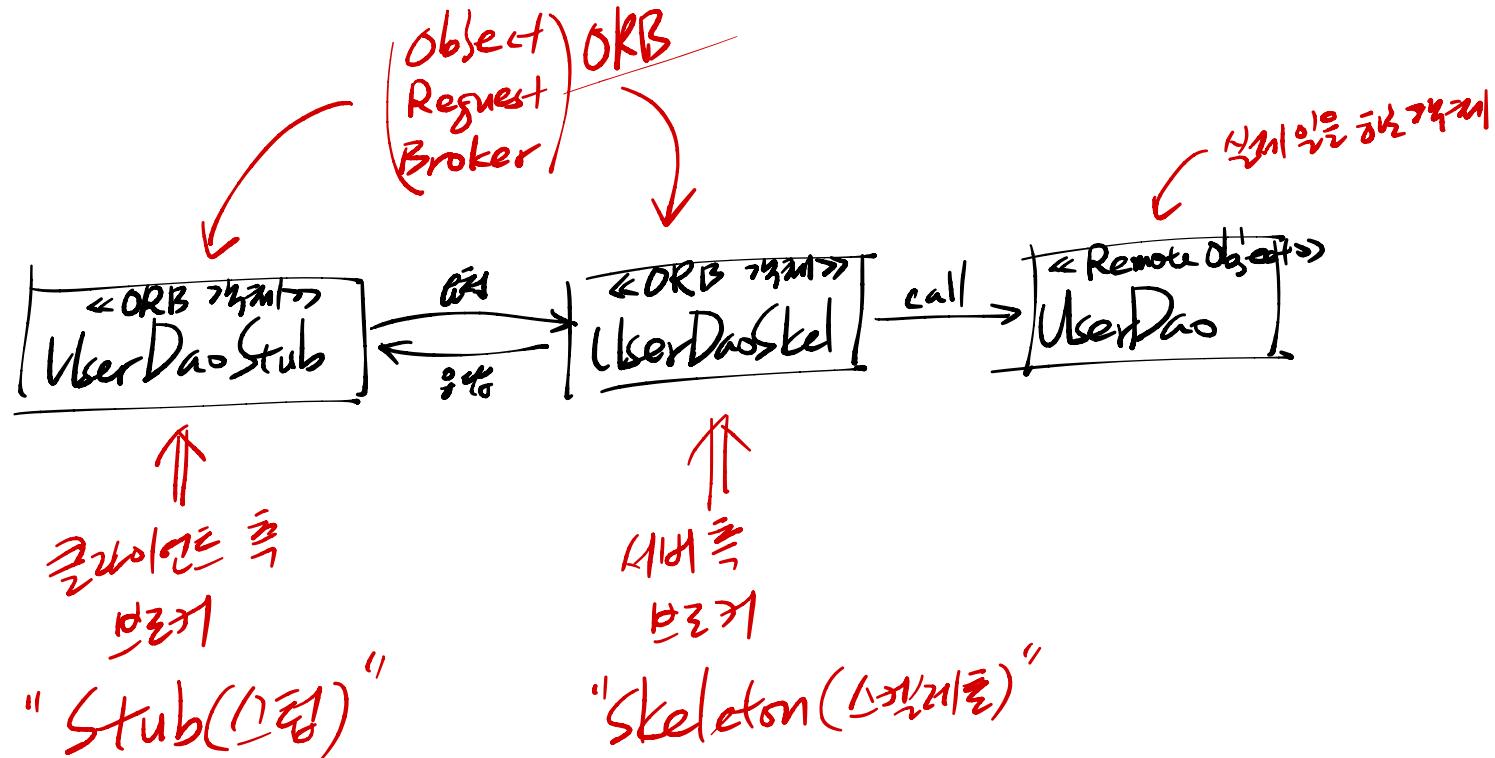


- VO ၁၃၈
- Net ၁၃၈

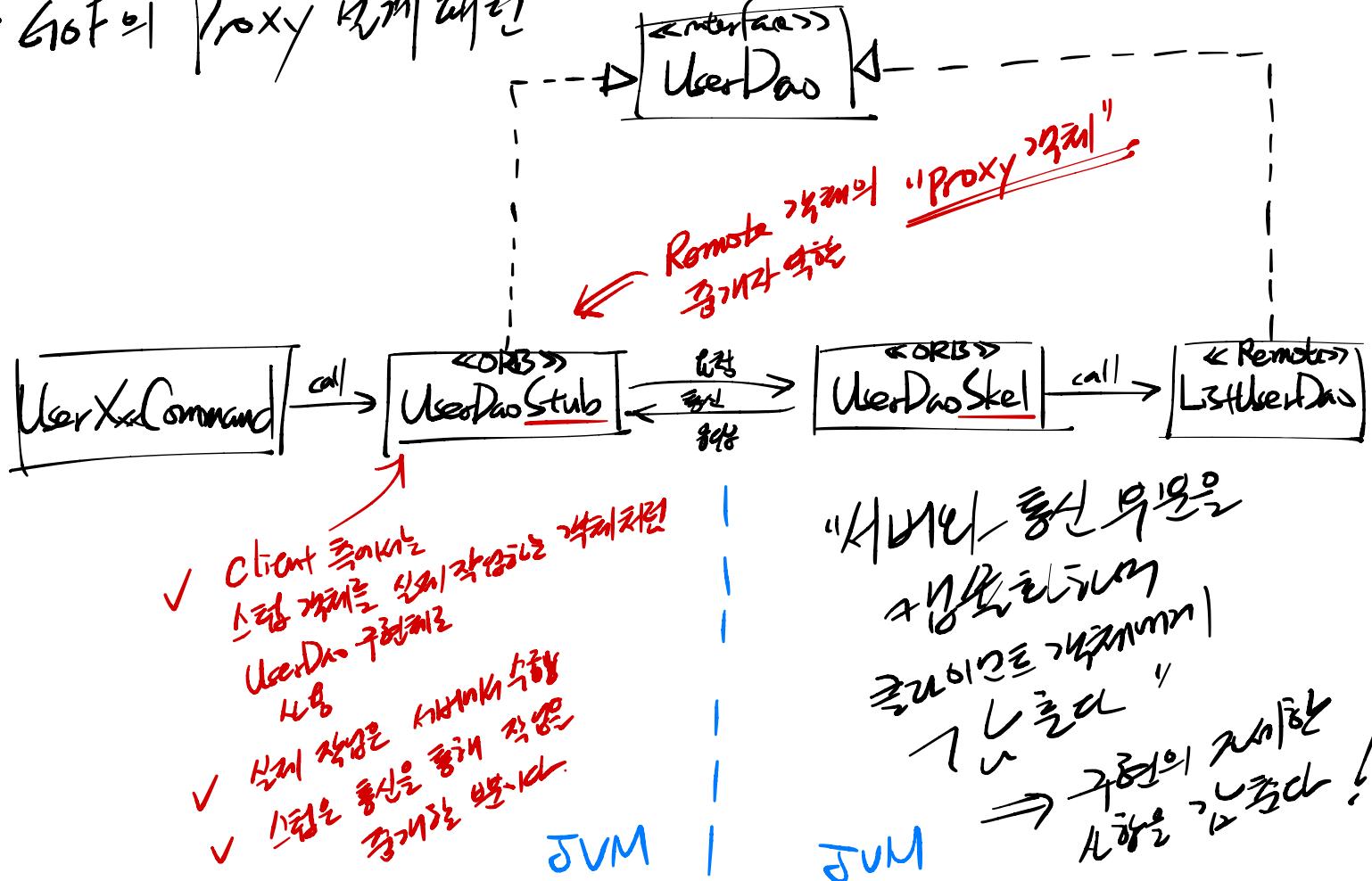
* Server 및 Client



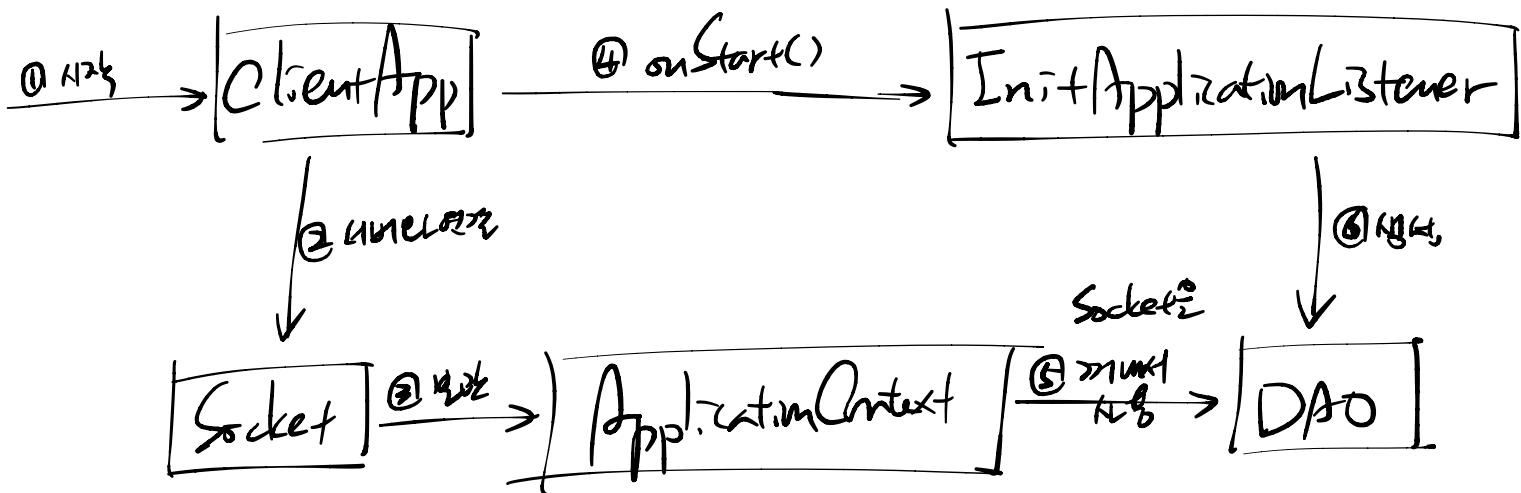
* Skeleton in Stub



* GOF의 Proxy 패턴 구현

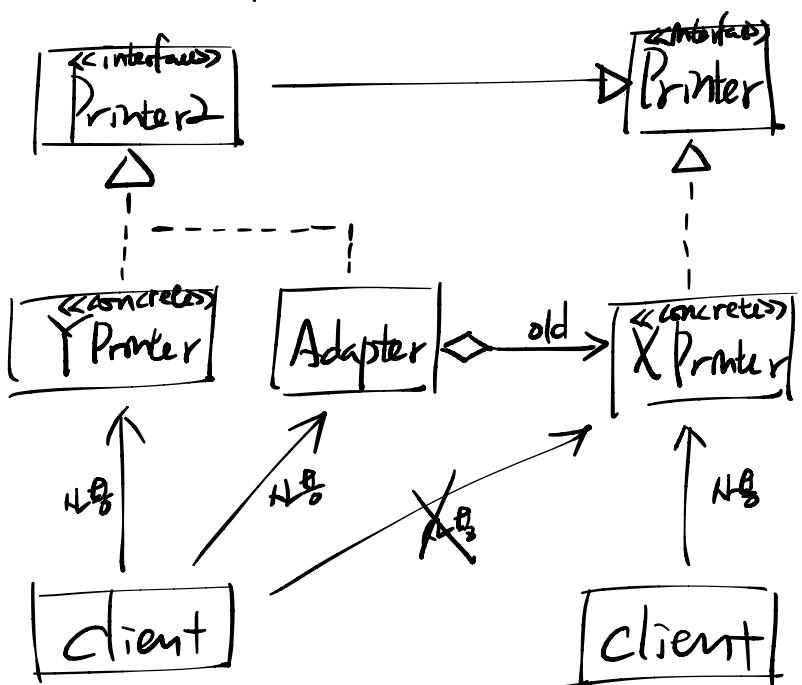


* Client Application & Stub

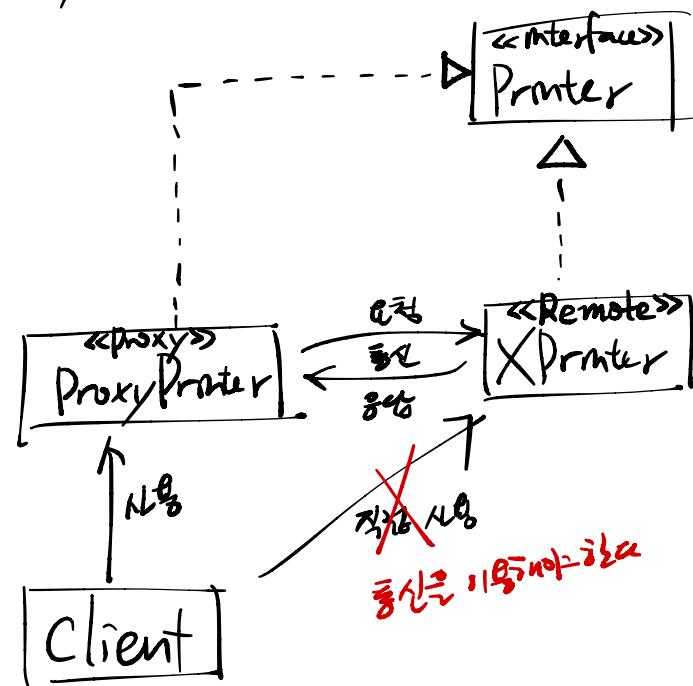


* Adapter 퀵질 vs Proxy 퀵질

① Adapter

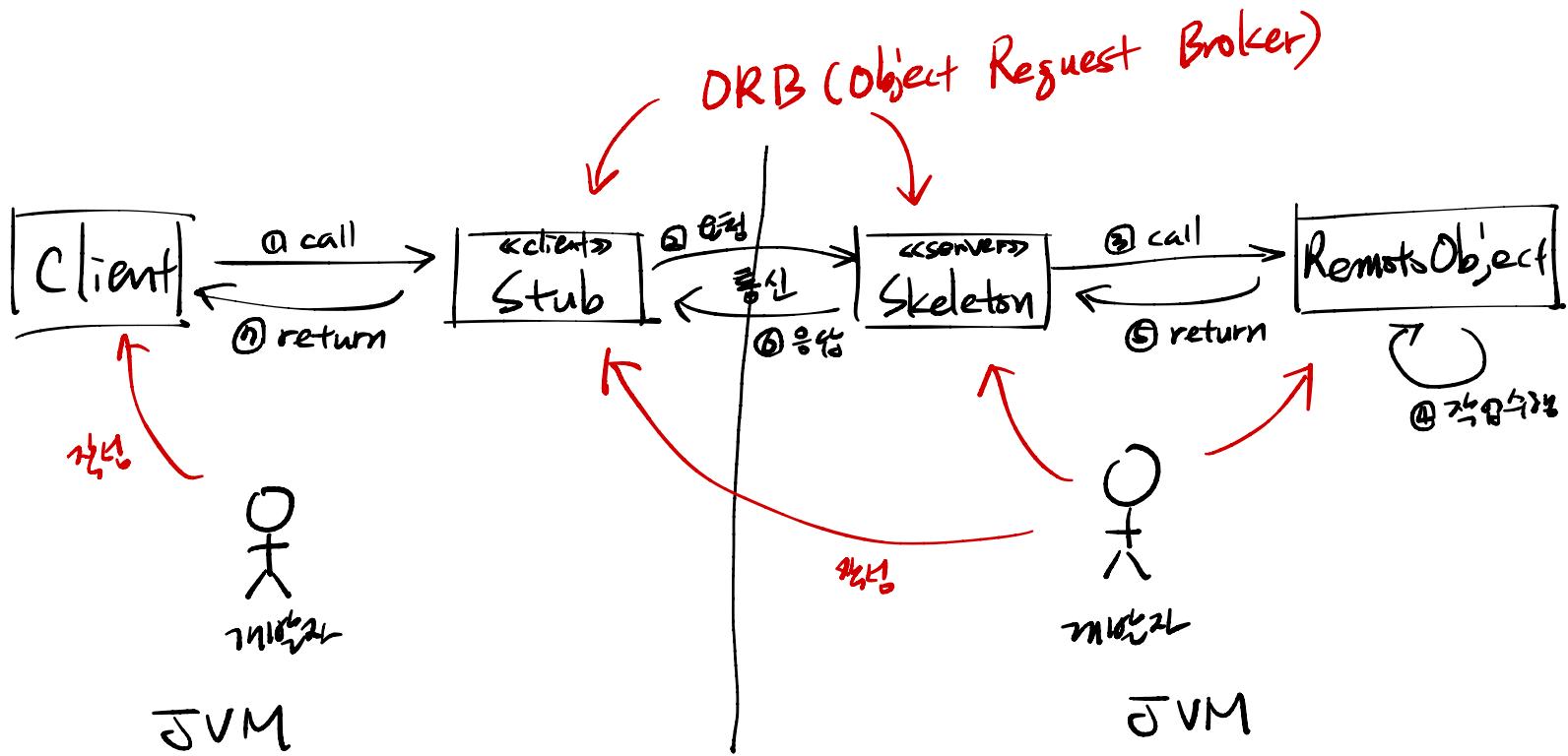


② Proxy



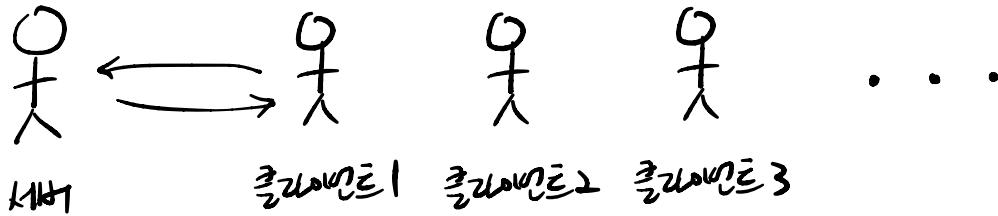
* Remote Object 개념

↳ 다른 JVM에서 만든 객체
(프록시)

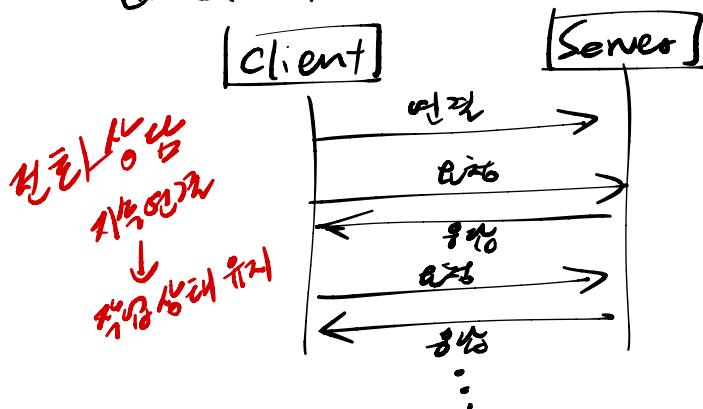


38. Stateful vs Stateless

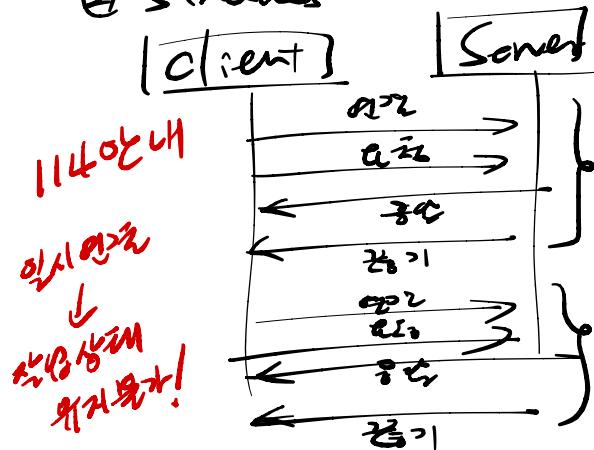
- * Stateful 웹서버 - 한번 연결한 후 여러번 요청/응답 처리
- * Stateless 웹서버 - 한번 연결한 후 한 번 요청/응답 처리



① Statefull

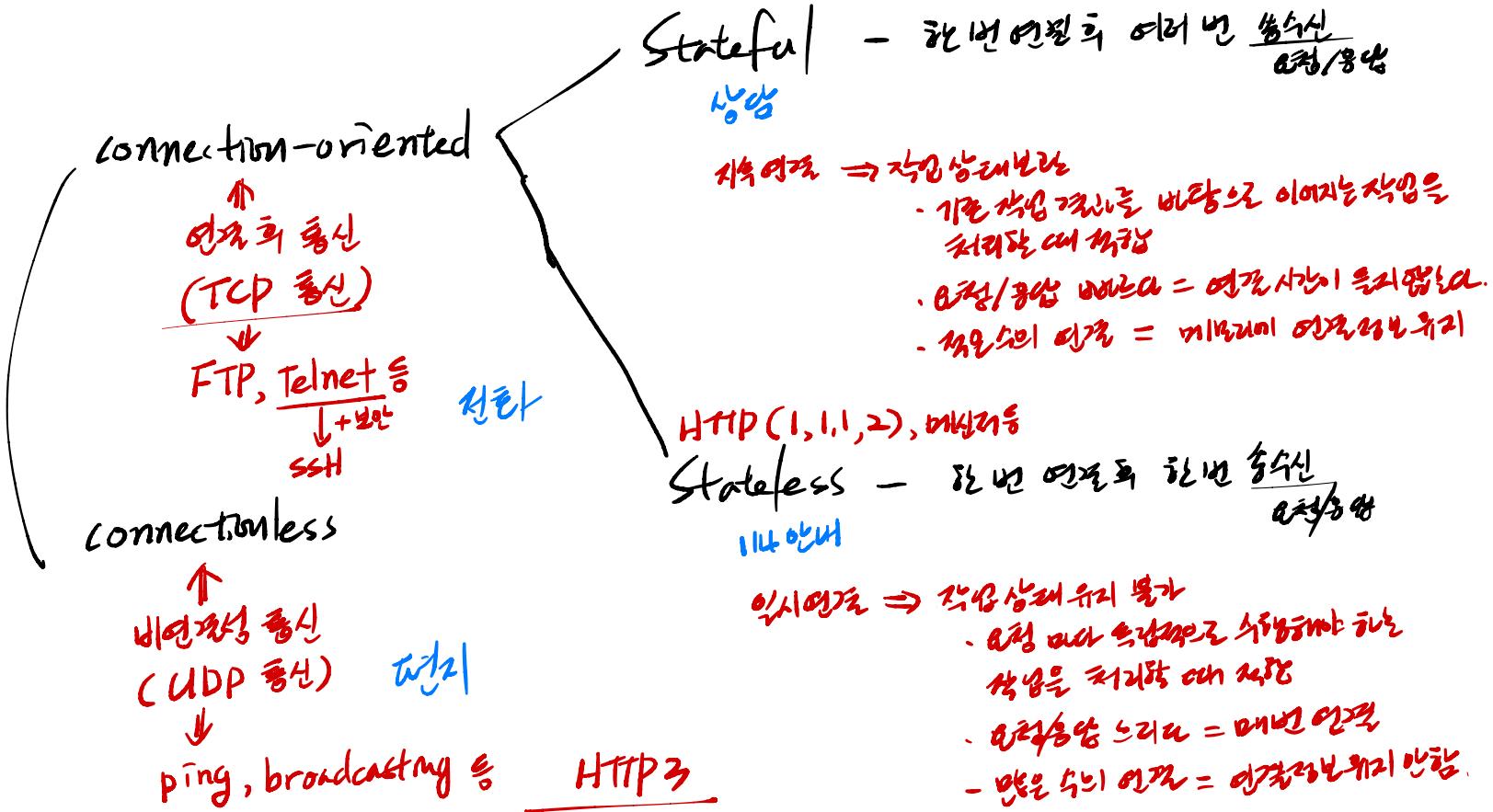


② Stateless

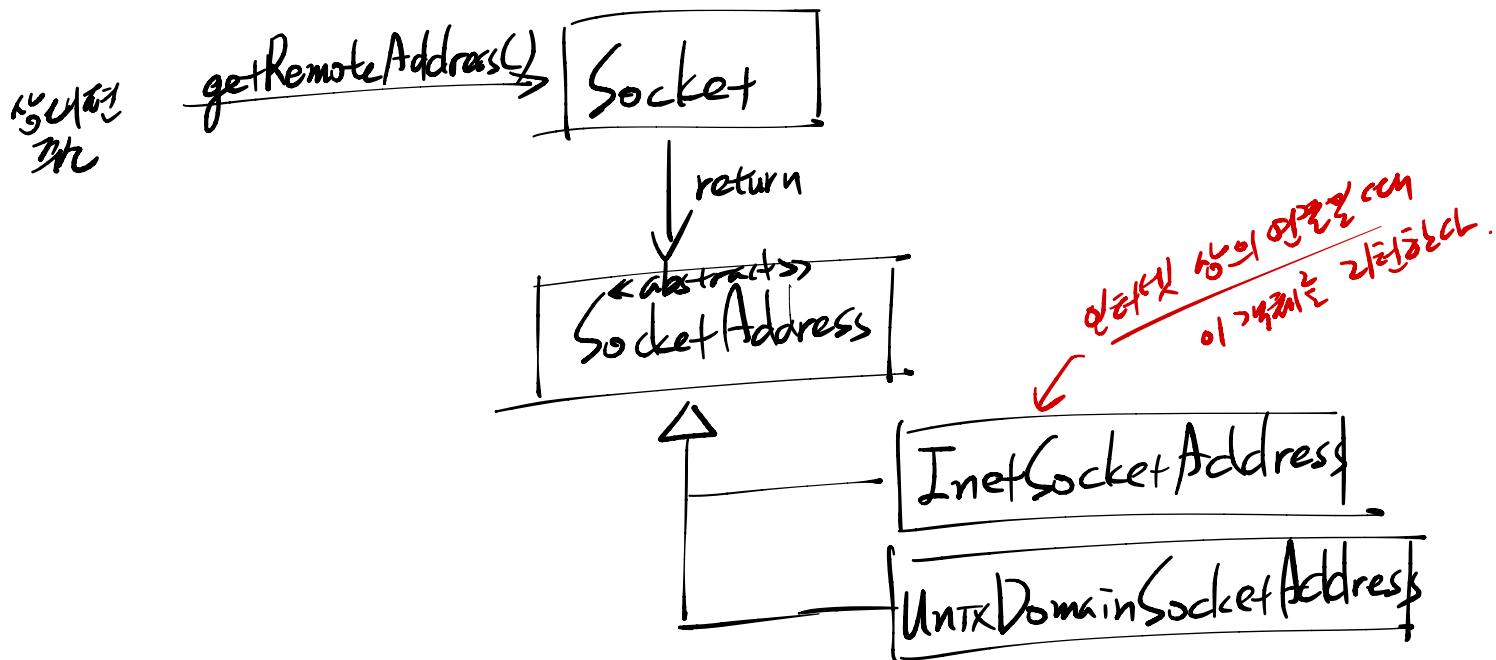


* 통신 18'시'

FTP, Telnet, SSH, 스트리밍, 채팅 등

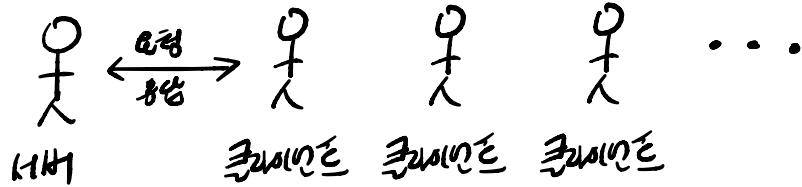


* Socket API



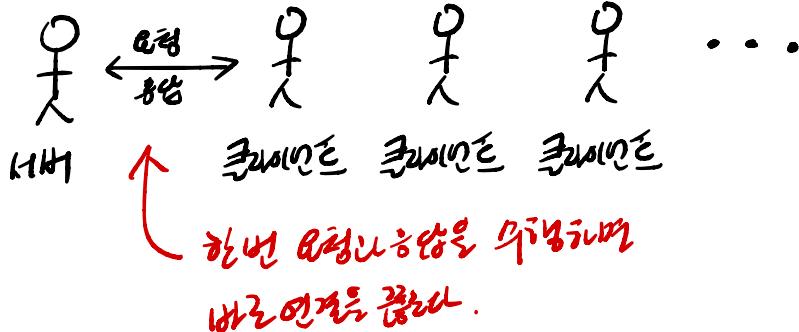
39. 미리그레드 층

① 아종 클러스터드 요청 처리 - Stateful



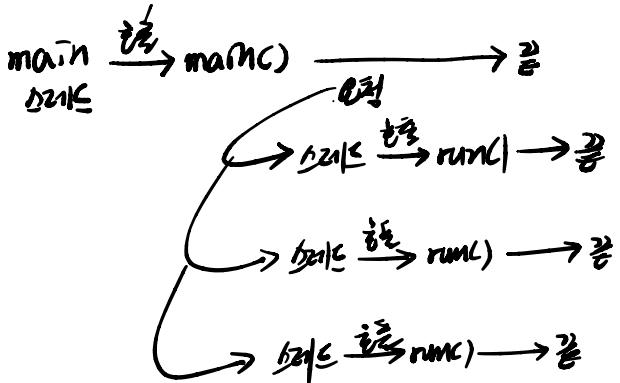
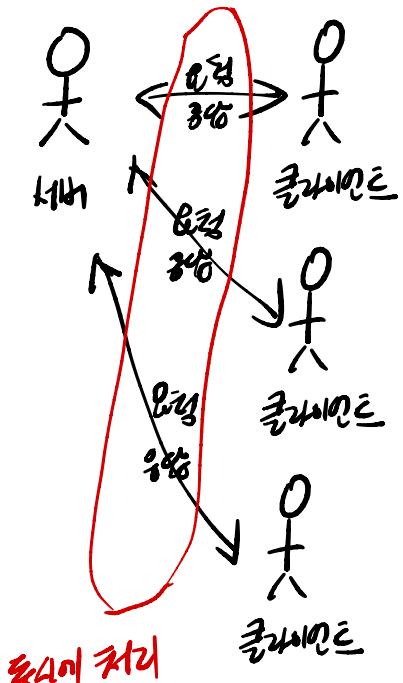
- ✓ 서비스와 연결된 클러스터드가
여기의 위치를 알기 때문에
다른 클러스터드는 가리키지 않아.
- ✓ 헤더를 통해 서로 처리된다.

② 아종 클러스터드 요청 처리 - Stateless



- ✓ 한 번 인증 → 한 번 헤더/값
 \downarrow
Stateful 헤더는 다른 서버로 전달되지 않아.
- ✓ 헤더 속성 대로 처리

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



① 쓰레드마다 세 가지 상태가 있다
그 시각화를 어떻게 해야 할까?

클라이언트 측정을 고정적으로 처리!

제거?

내장으로 처리!

클라이언트 측정 처리

main의 실행흐름
원리단계 이후 부정!

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

class 씨닝 extends Thread {

void run() {

여기 내용

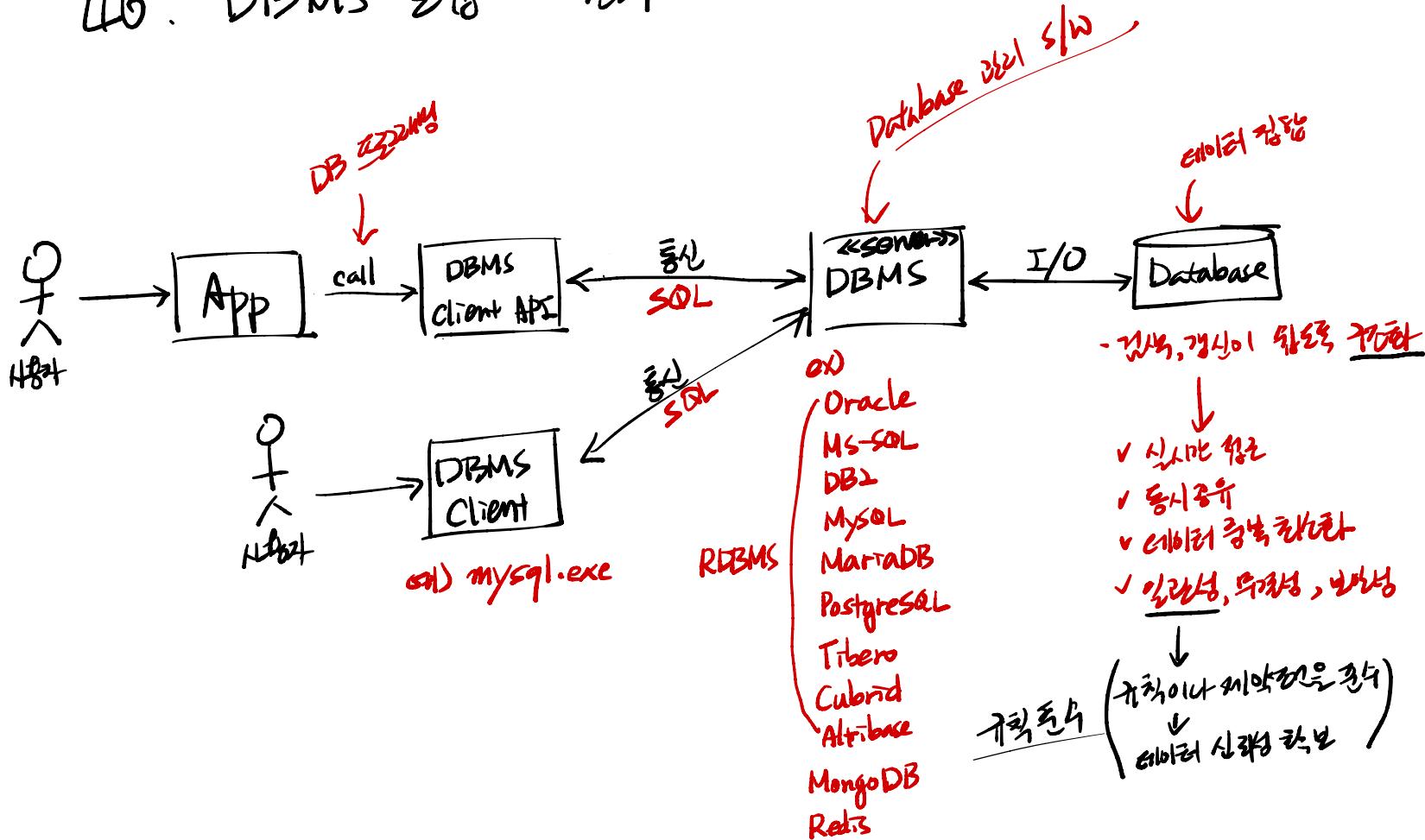
}

}

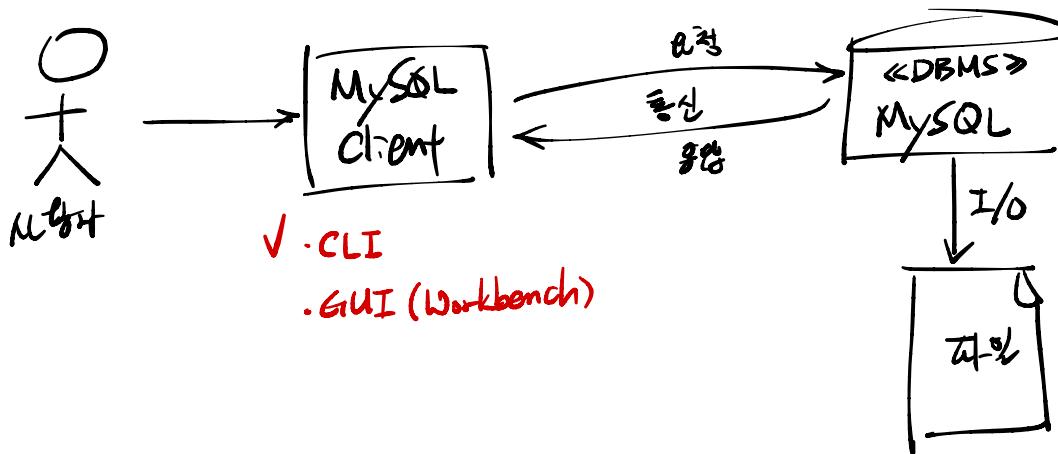
쓰레드.start();

여기서 쓰레드가 시작된다.

40. DBMS 솔루션 - 토론



40. DBMS 속성 - MySQL 설치



4. DBMS 은호 - DBMS 사용

① 은호

```
$ mysql -u root -p
```

DBMS 은호이므로
↑ user id
↑ 암호입력

```
$ mysql -h 은호 -u oracle -p
```

← 다른 컴퓨터

② DBMS 사용 예제

```
create user 'oracle'@'은호' identified by '은호'
```

↑ 은호
↑ ID
↑ 패스워드
↑ ID 품질
↑ %. (다른 모든 품질)

은호 품질

③ 데이터베이스 만들기

```
create database eschooldb default character set utf8 default collate utf8_general_ci
```

utf8은 2Byte로
인코딩 하는
"문자집합"

↑
정确한 비교 및 정의

110. DBMS 예제 - DBMS 사용

④ 관리자에게 접근 권한 부여

grant all on studydb.* to 'study'@'localhost';
DBMS

접근권한 DBMS DB명, 테이블명 user host

⑤ 데이터베이스 목록 조회

show databases

⑥ 관리자에게 접근 권한 부여

use 관리자에게 권한 부여

⑦ 테이블 목록 조회

show tables

⑧ 테이블 구조 조회

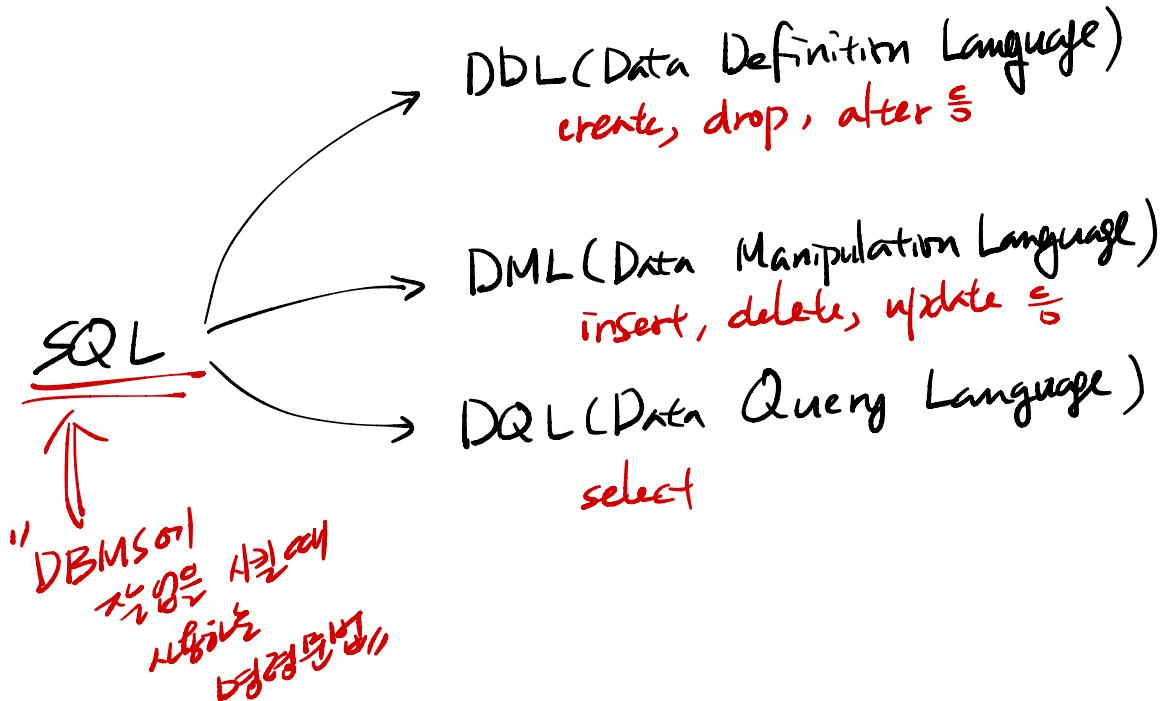
describe 테이블 이름

desc 테이블 이름

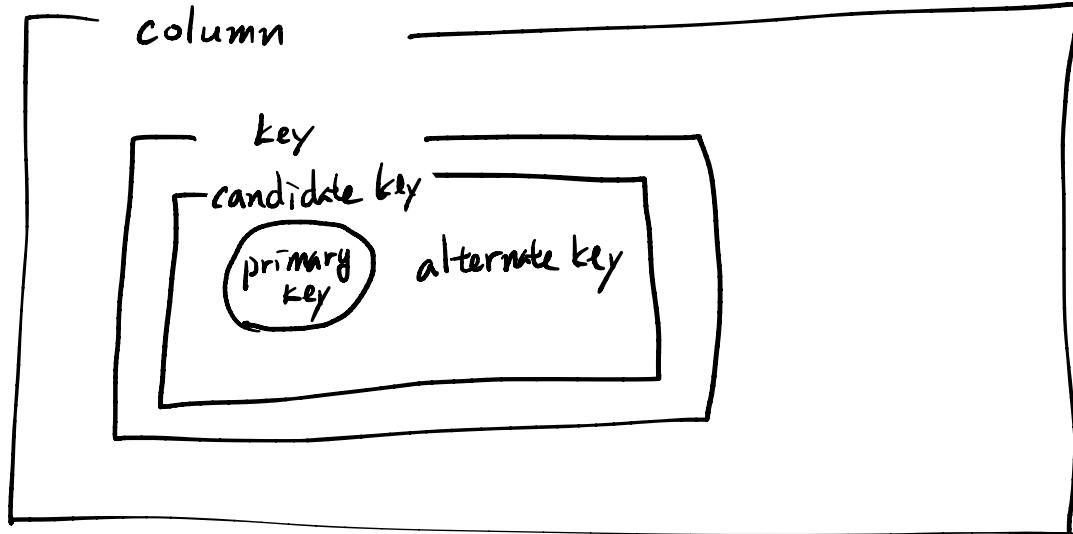
⑨ 접근자 목록 조회

select user, host from mysql.user

110 . DBMS ୱେଳେ - SQL (Structured Query Language)

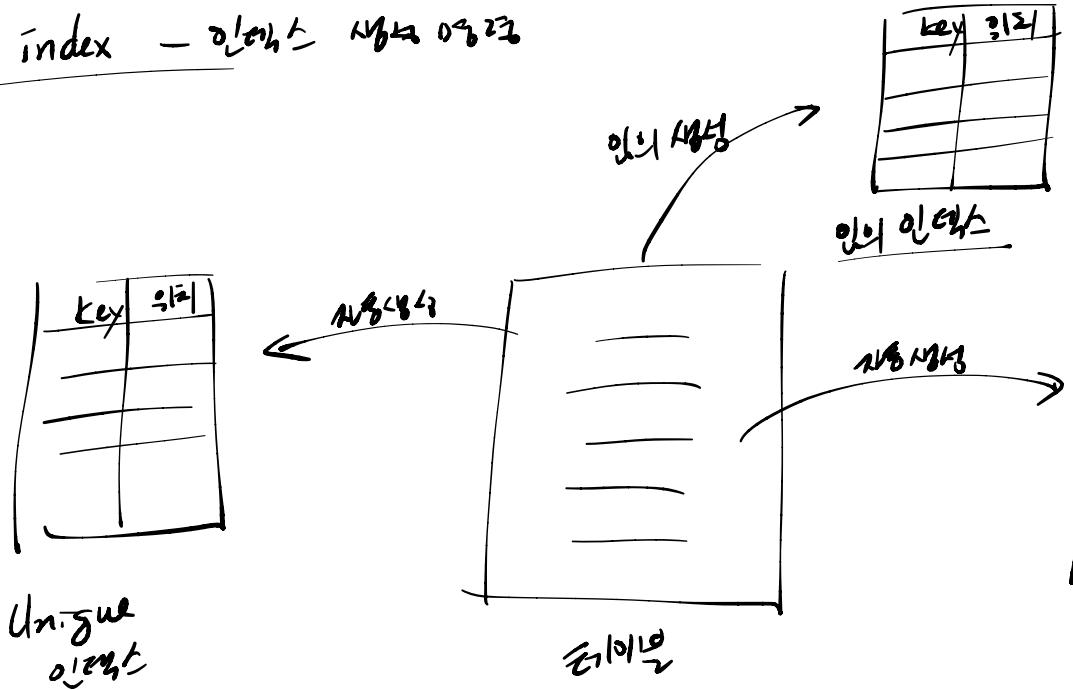


* key - 데이터를 구분할 때 사용하는 키임



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

* index - 인덱스 구조화된



✓ 고유한 고유한 원소들은 인덱스 (이후에 고유한)

✓ 고유한 고유한 원소들이 있다.

같은 원소는 아예 존재하지 않음.

primary key
인덱스

"인덱스인가"로 처리됨 not idr.
a) 풀드, 캐시
ElasticSearch

* projection vs selection

선택
제거
select.m

no	name	class	working	tel
			F	
			N	
			N	
			F	
			N	

"projection" - 테이블에서 원하는
필드만 선택

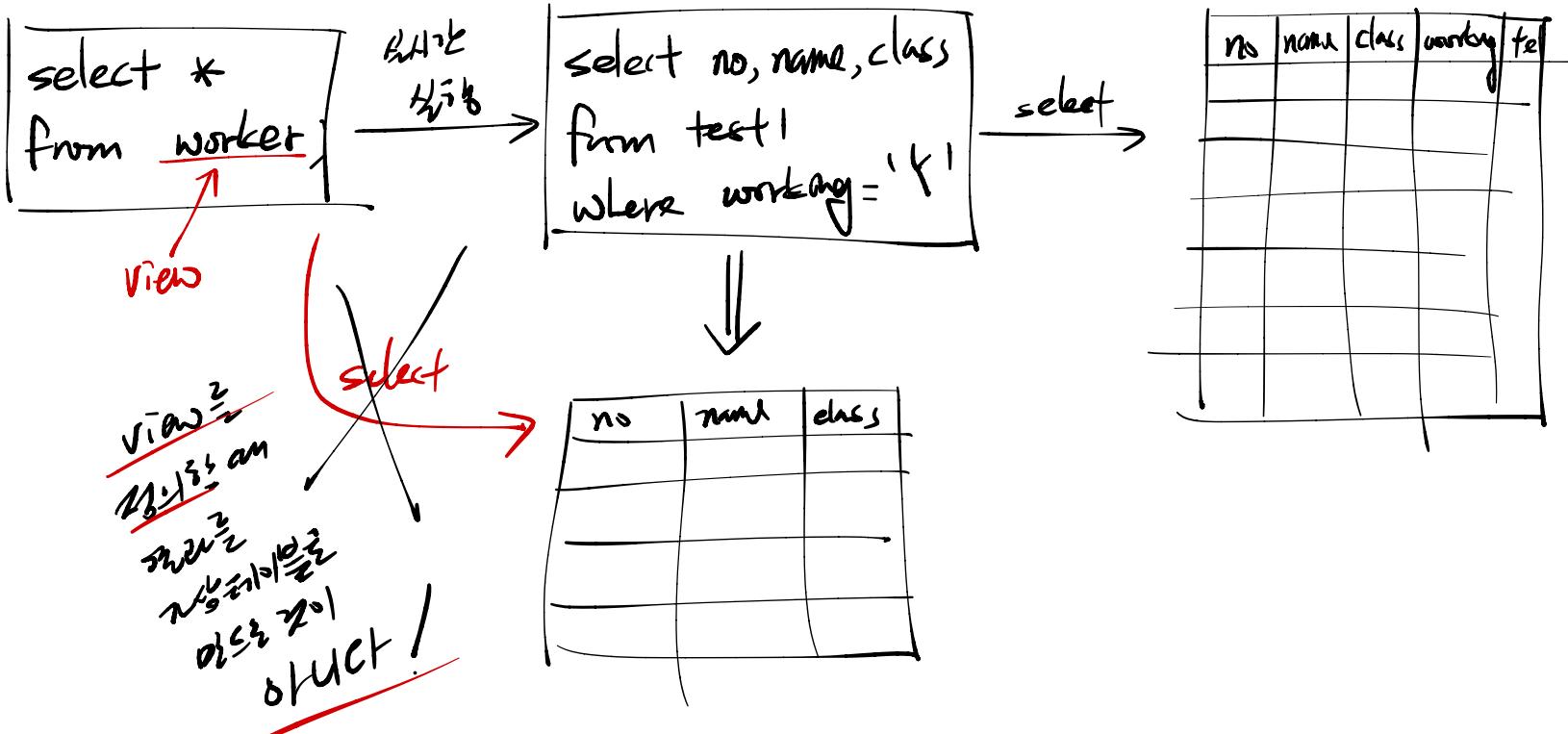
select no, name, class
from test1;

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

"selection" - 테이블에서 원하는
행만 선택

(102)

* view et table V_2^{ex}



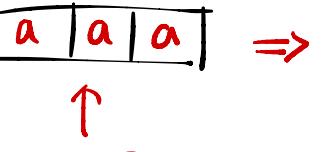
* char(n) vs varchar(n)

데이터베이스를 생성할 때
지정한 charset에 따라 한글자의 메모리 크기가 결정된다.

정형 → char(5)  $\Rightarrow 'aaa'$

지정된 글자 개수가 상관없이
무조건 5글자를 차지할 메모리 크기 갖는다.

문자 2칸

غير정형 → varchar(5)  $\Rightarrow 'aaa'$

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-

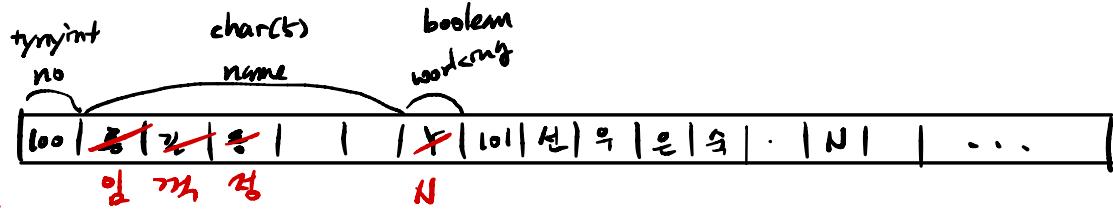
row = record = tuple

column = field = attribute

* 파일과 연결 / 출판

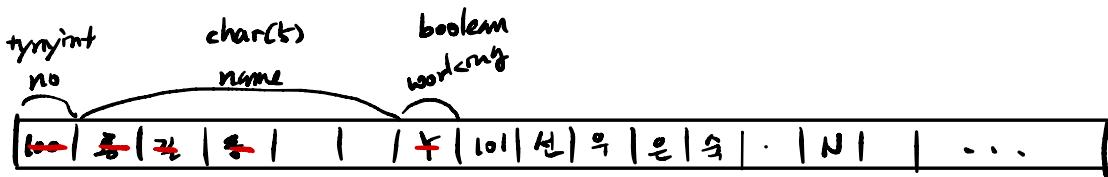
파일 데이터 1번정

↳ 기호값을 저장할
문자는 1정



파일 데이터 삭제

↳ 기호값을 저장한
위치를 초기화



↑
0 으로 초기화 시킨다

☞ 데이터를 재가공하면
위의 데이터를 앞으로 옮겨야 한다
↓
↳ 후시작하는 것이다

• 다음으로나 시거나 했거나 한 것이다

↓
그러나 그다음 데이터는 저장된 부분을
0 으로 덮어쓰는 것이다.

파일의 특정 부분을
저장할 수 있다

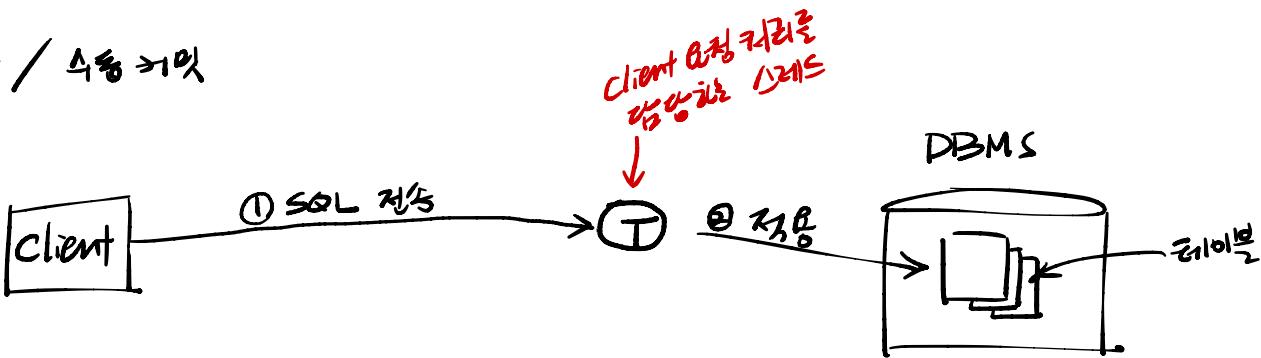
↓

- ① 저장할 위치를 어떤 데이터를
내 파일에 쓰거나,
- ② 또는 위의 데이터를
삭제한 부분으로 덮어쓰는 것이다.

* autocommit / 속도↑이익

① autocommit

(insert
update
delete)

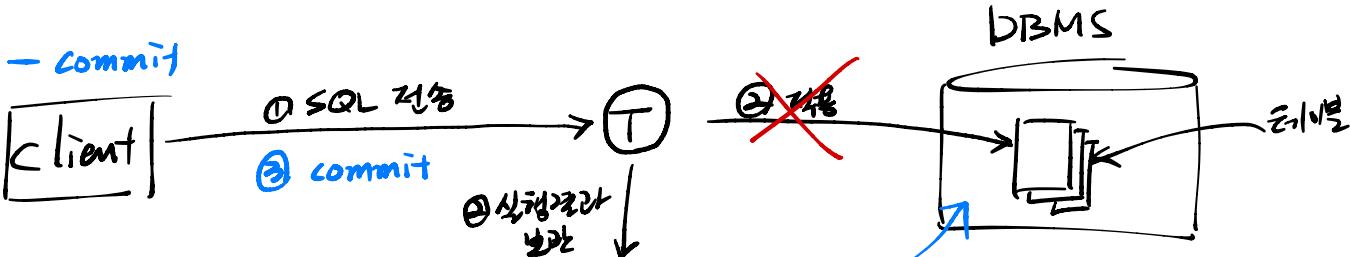


트랜잭션 처리에 번거러운 과정은
방지됨, 예전
제작 방식이다.

⇒ 초기 상태로 되돌릴 수 있다.

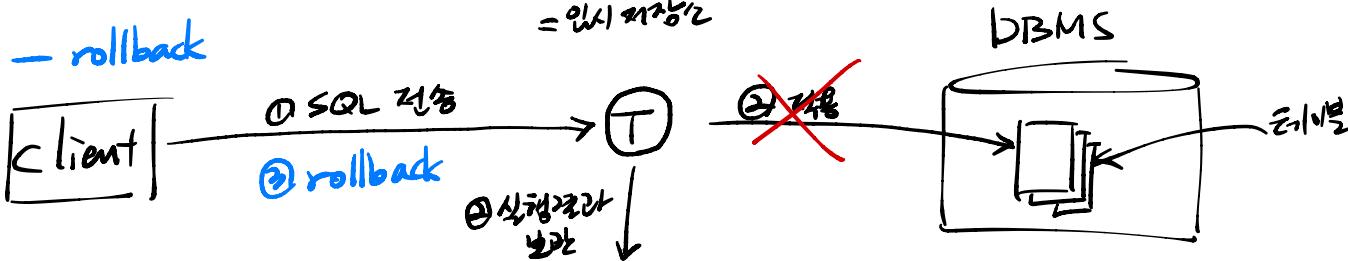
② 푸시 commit - commit

- insert
- update
- delete



② 푸시 commit - rollback

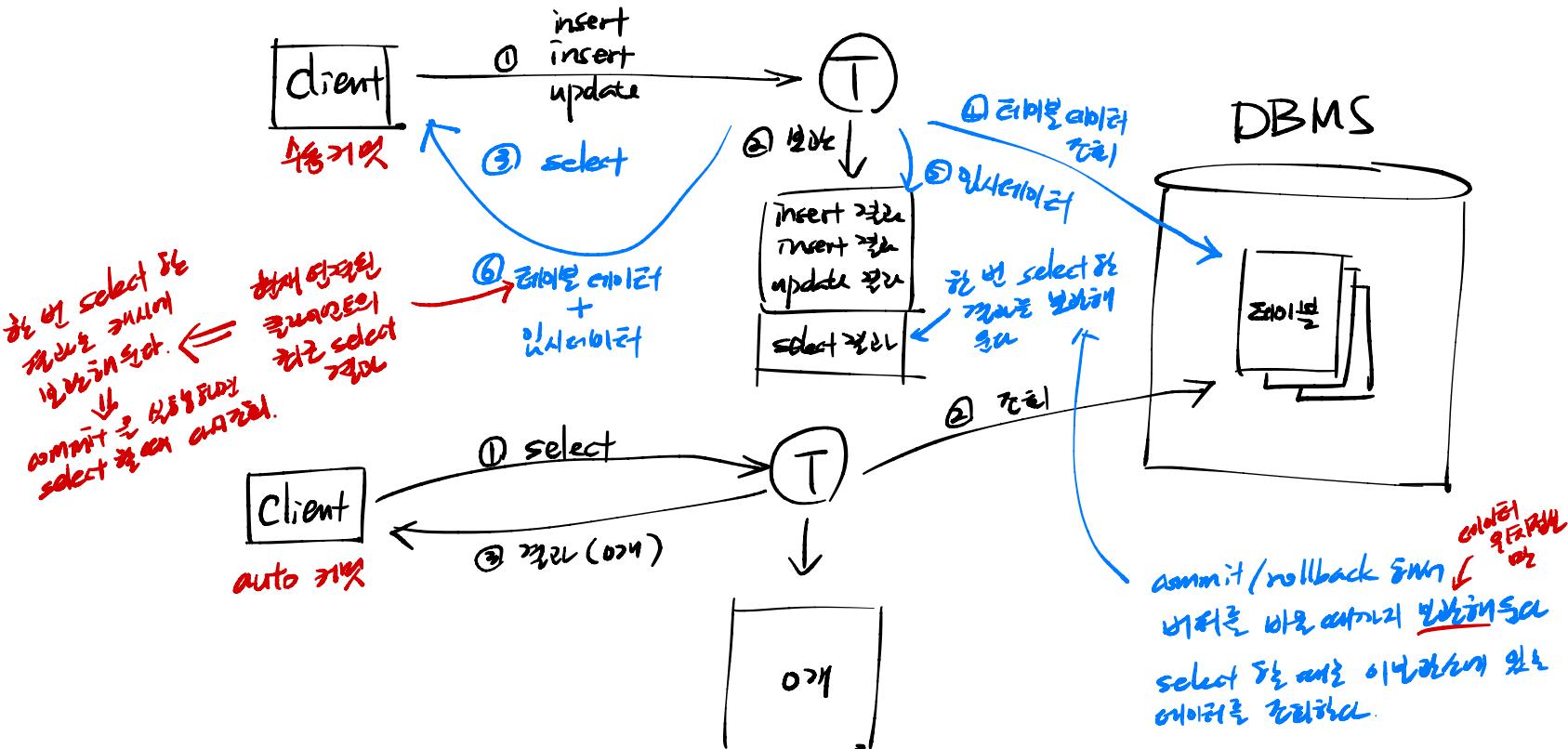
- insert
- update
- delete



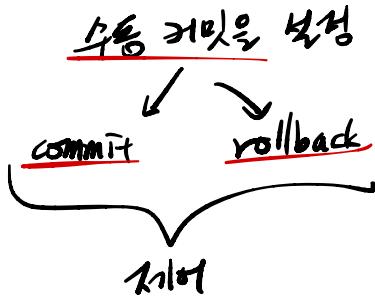
④ 푸시에 일시 멈춰버린 SQL문장을 실제 데이터베이스에 적용!

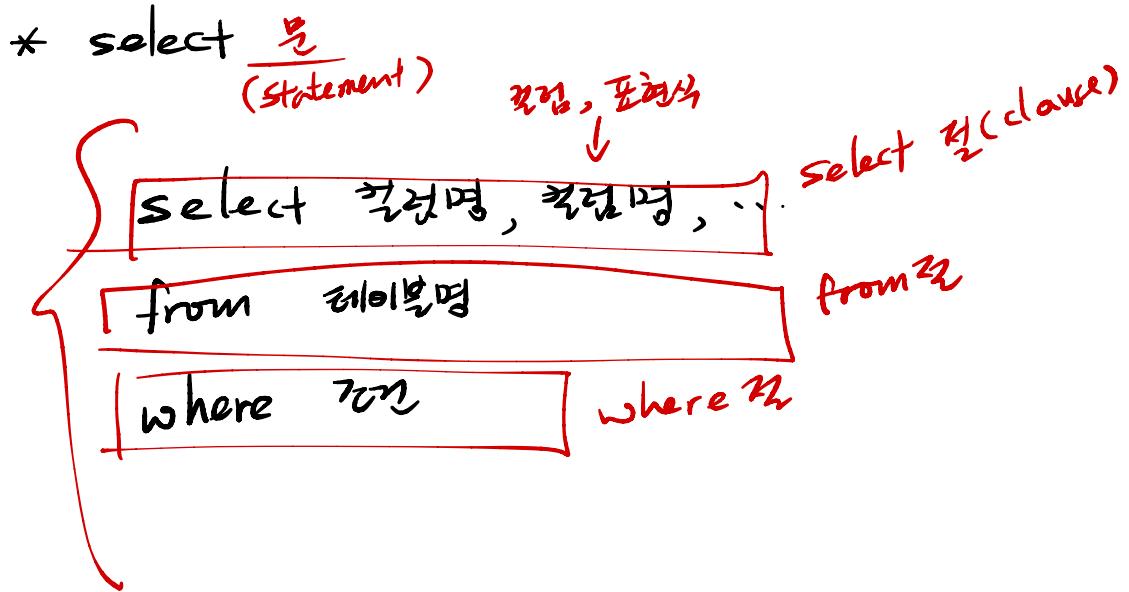


* Client et commit



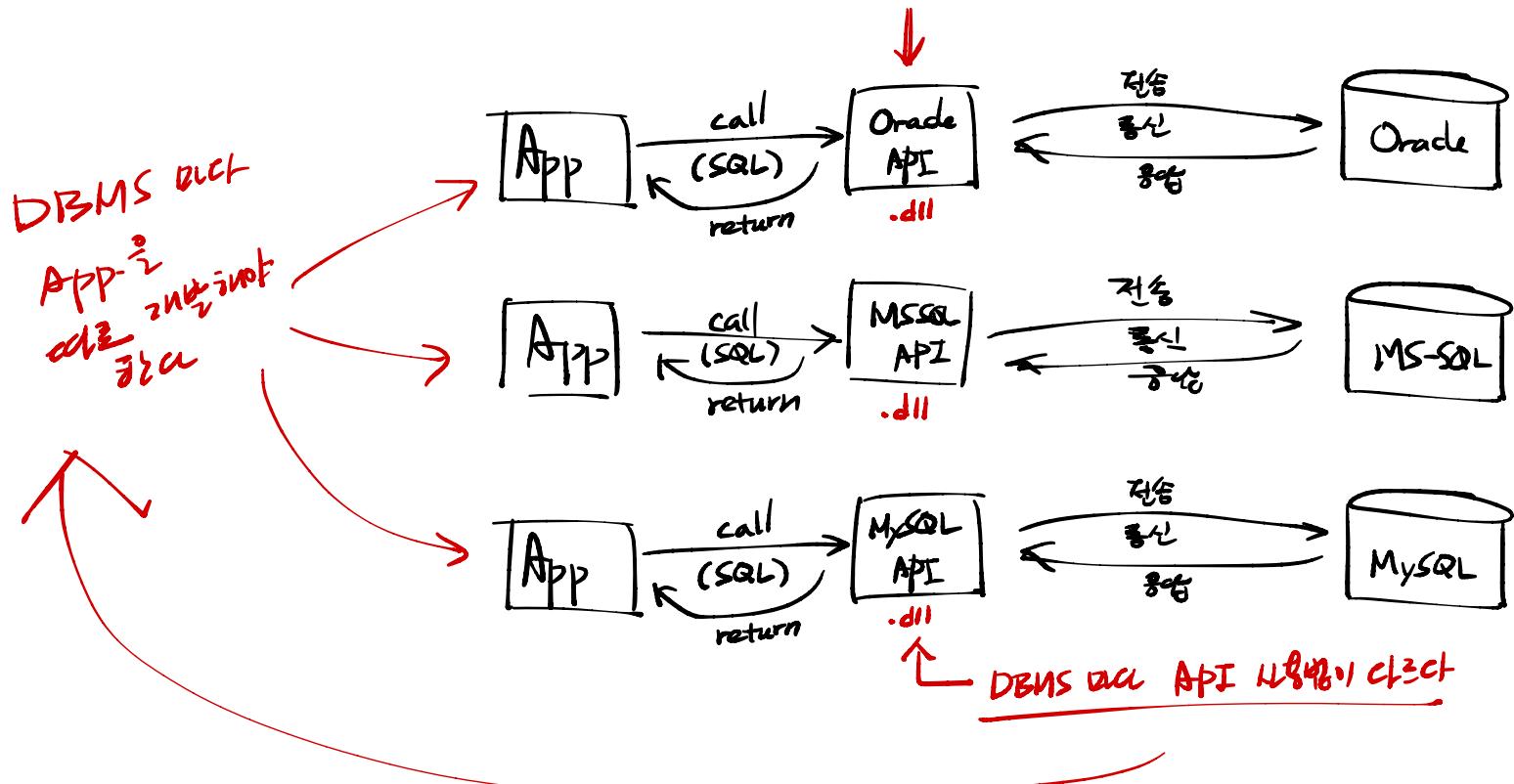
- * 트랜잭션 (transaction)
 - ↳ 여러 개의 작업을 한 단계로 묶은 거.
(insert/update/delete)





* DB 접근방법 - Vendor API = Native API

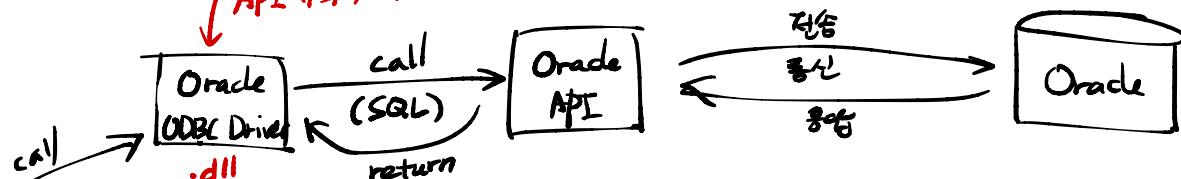
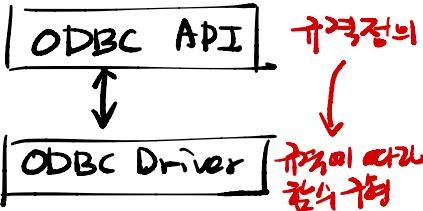
Vendor API = Native API (C/C++)



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

Open Database Connectivity

API 틀에 따라 활용할 수 있는 방식.



API 사용법
기본 구조
DBMS 드라이버
직접 APP을
접근 방법은
없다

App

MSSQL ODBC Driver .dll

MSSQL API

MS-SQL

MySQL ODBC Driver .dll

MySQL API

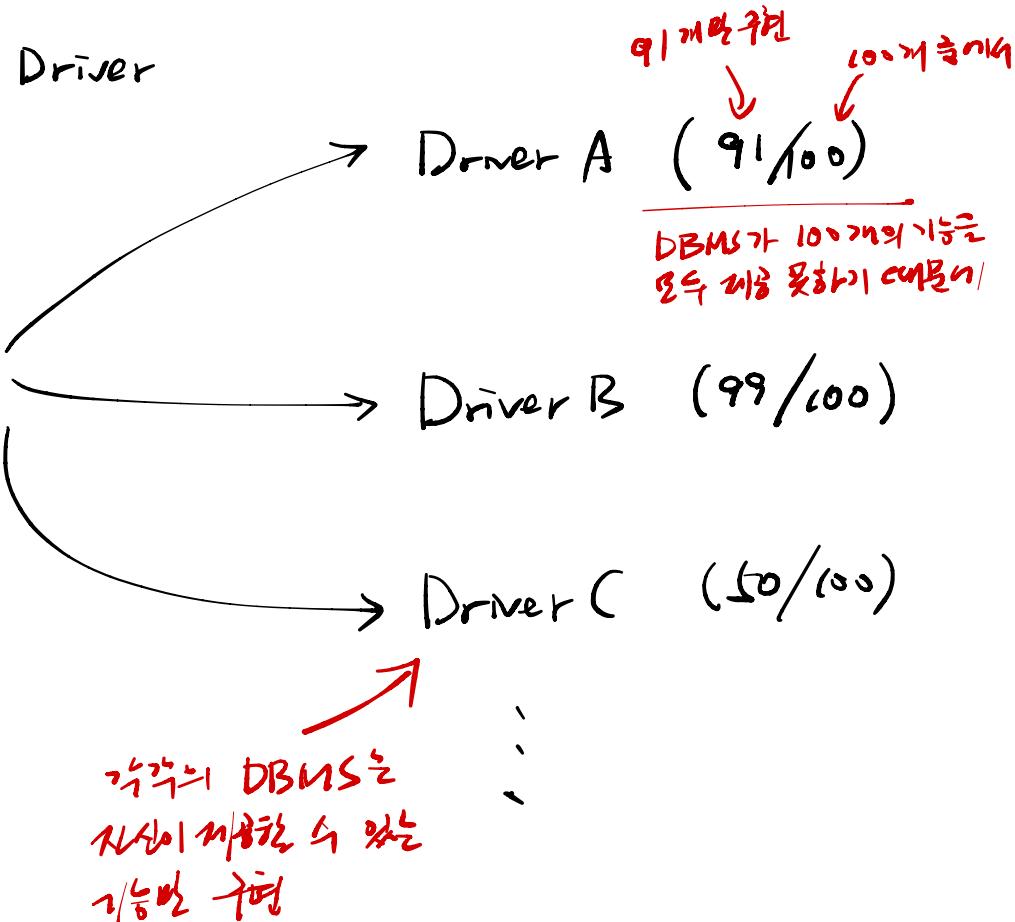
MySQL

Excel ODBC Driver .dll

Excel

* ODBC API 와 ODBC Driver

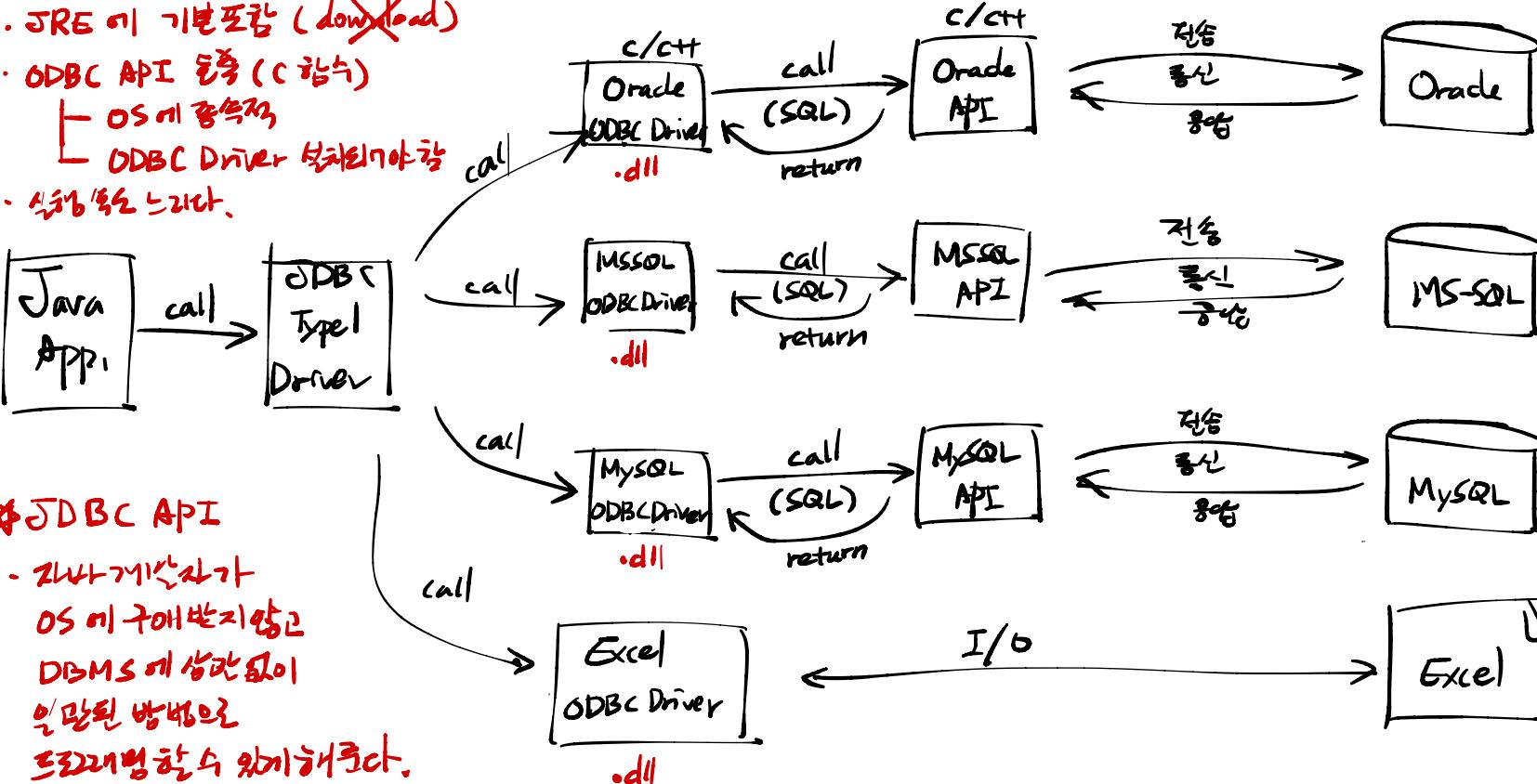
ODBC API
① 100개 품목의
기능 중의
기능을 정의하는
DBMS가 지원하는 기능
기능을 모두 지원하는
기능은 거의
기능은 거의



* 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.

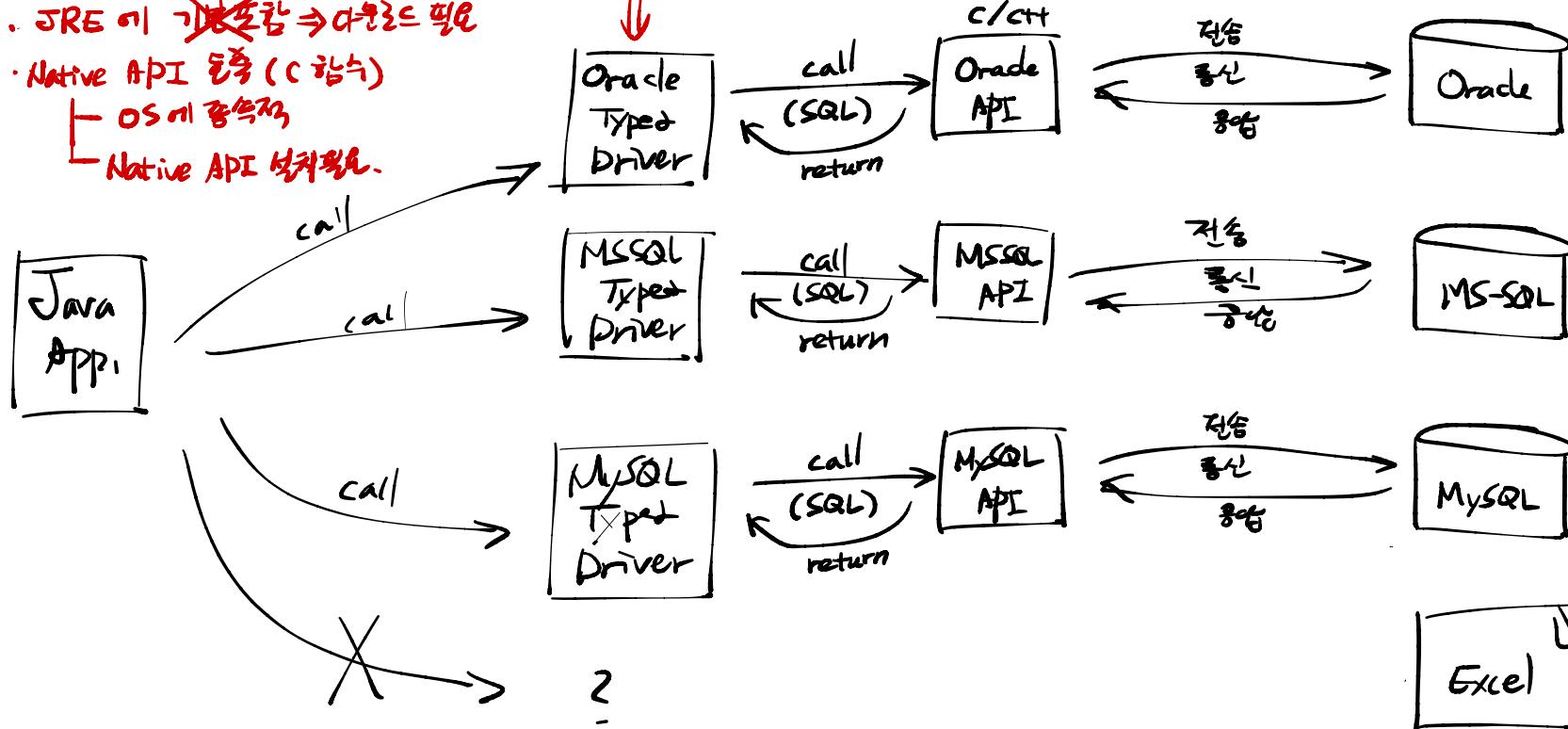


* DB 접근방법 - JDBC API Type 2 \Rightarrow Native API call Driver

Type 2 (2유형)

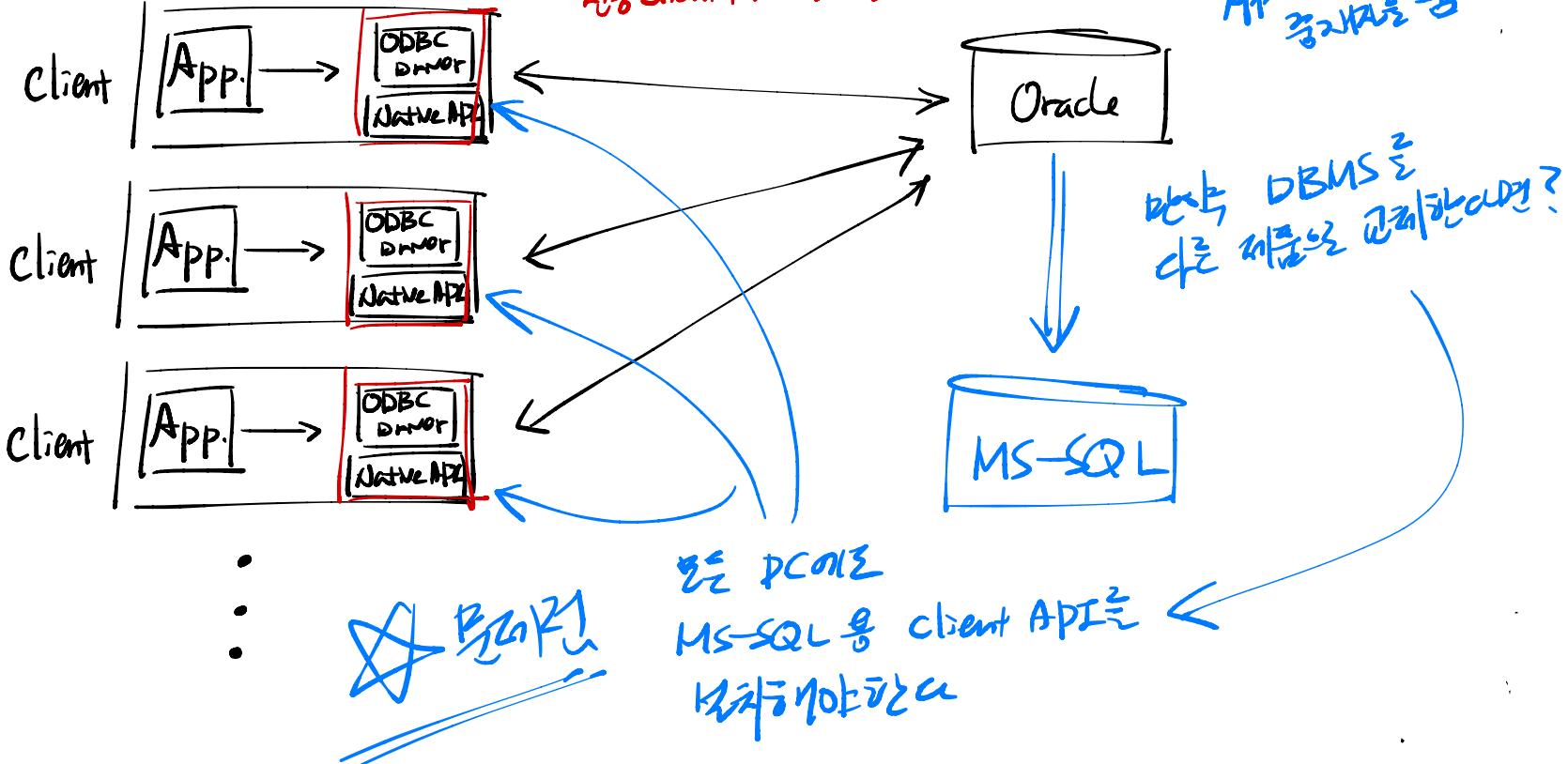
- JRE 외 ~~기존~~ \Rightarrow 다른 드라이버
- Native API 툴킷 (C 풀이)
 - OS에 종속적
 - Native API 사용방법.

JDBC API 규격이 빠트리 Vendor API 풀이



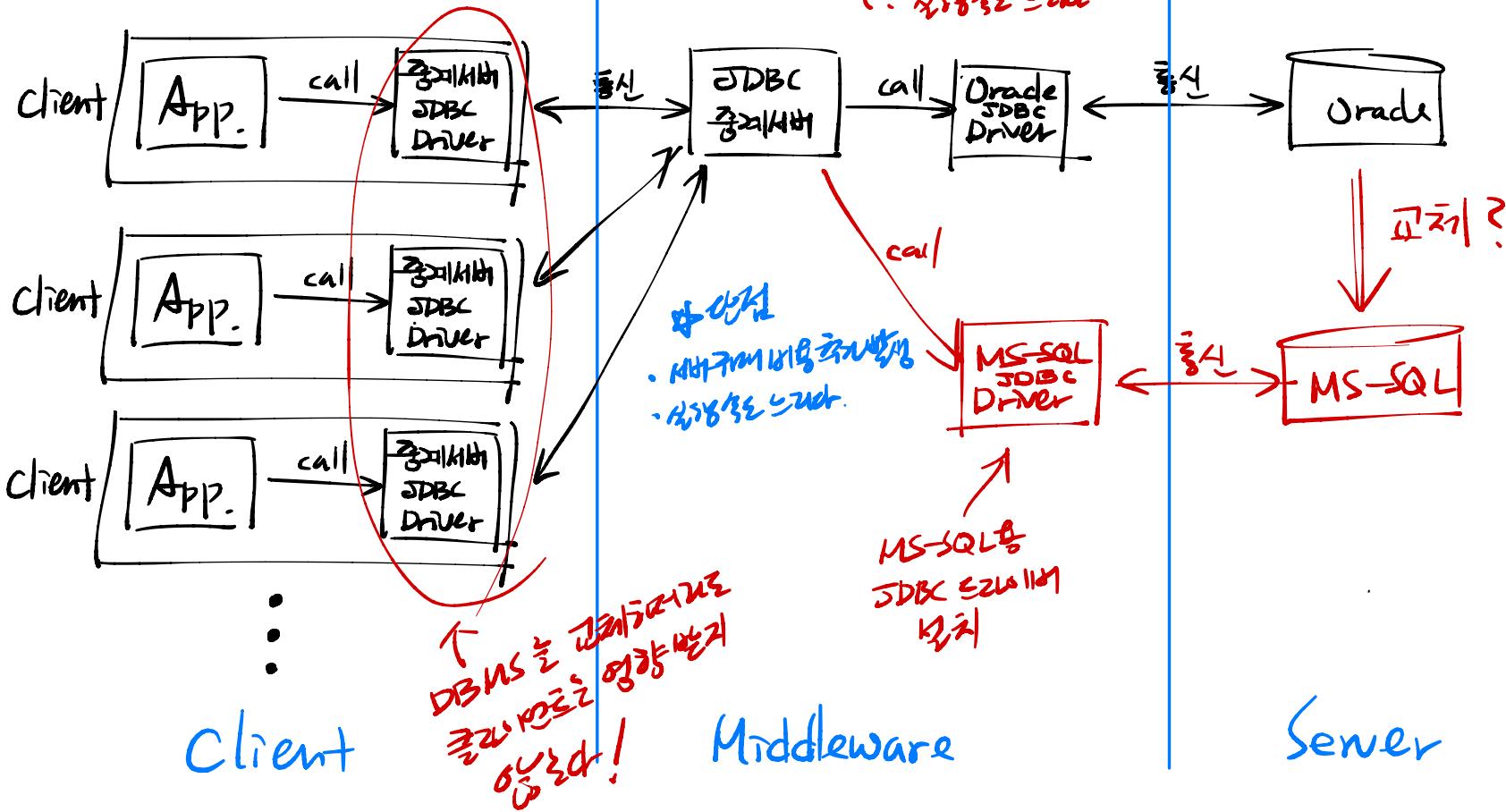
* DB 접근방법 - JDBC API Type 3 \Rightarrow Middleware Driver

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

