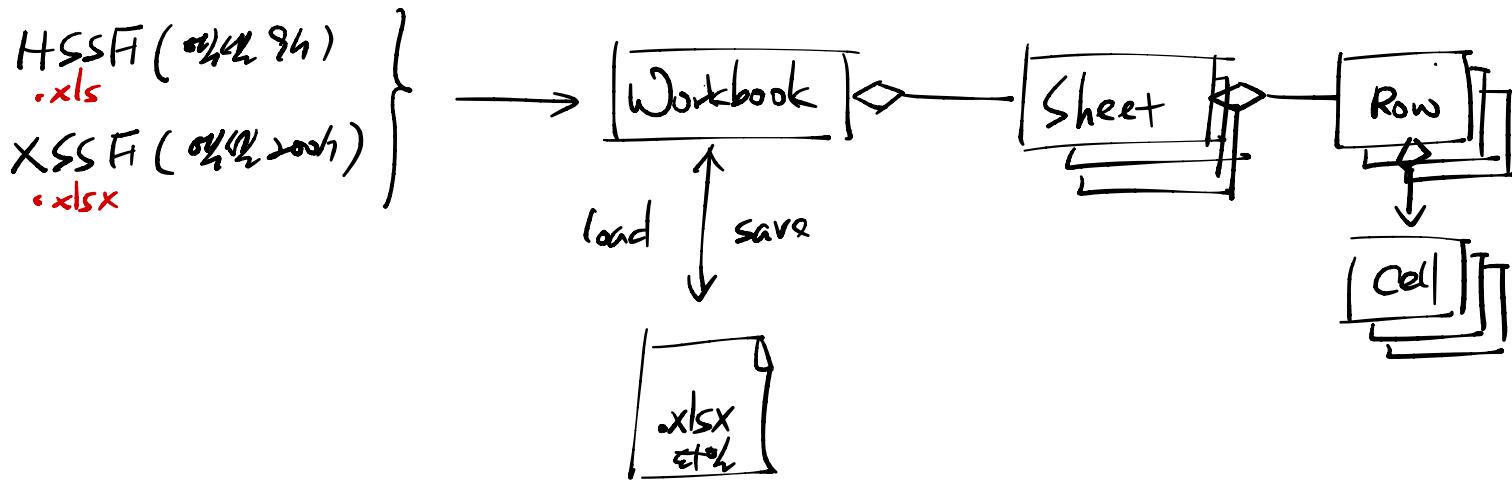


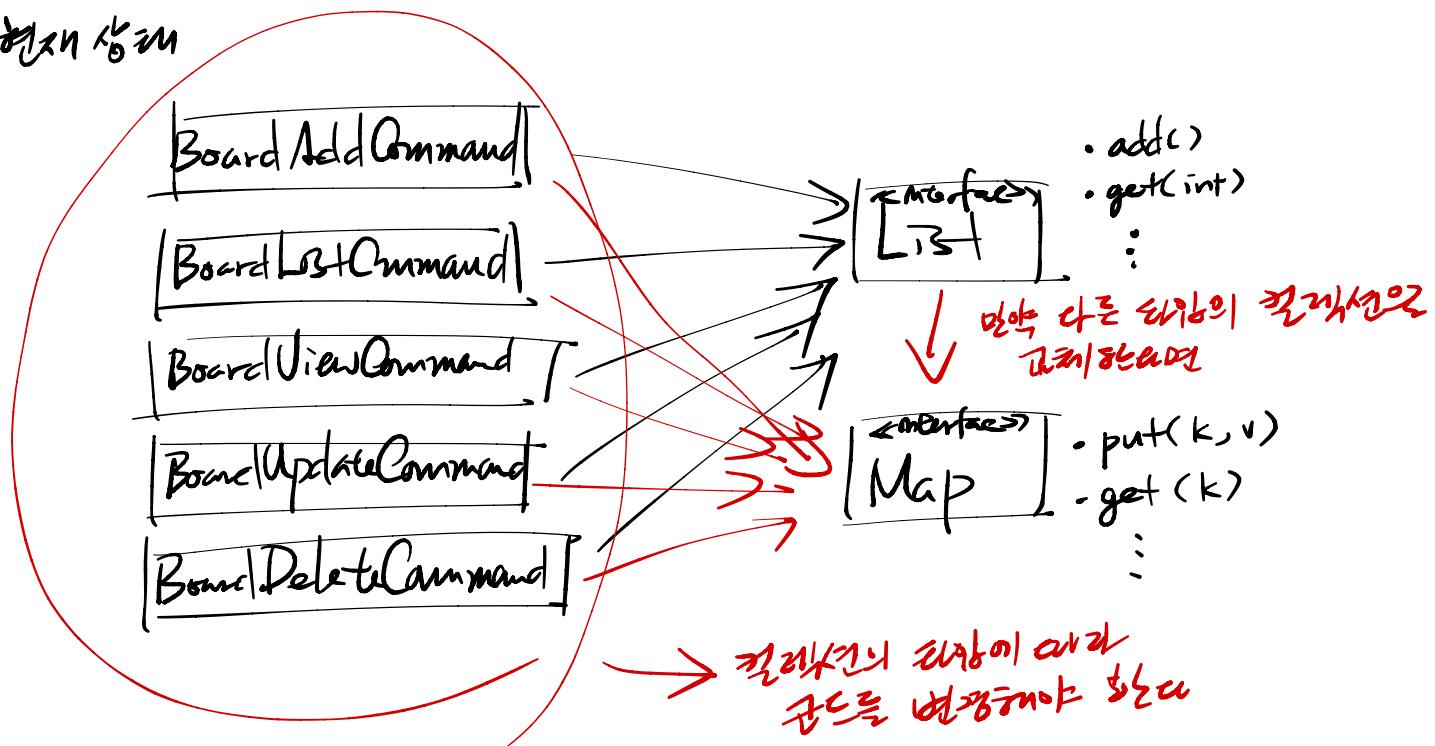
33. 콜렉터는 어떤 원칙을 따른다? : Apache POI 소개



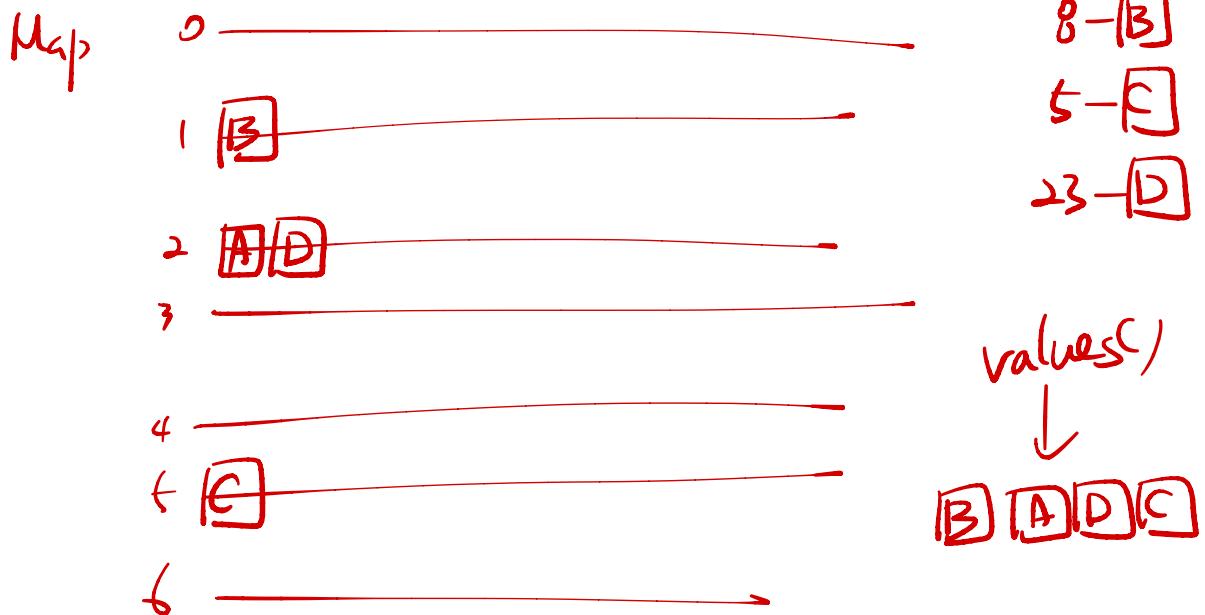
34. DAO 가 필요한 이유

↳ 데이터 베이스를 관리하기

① 테이블



* Map의 데이터를 순회할 때는 항상
꺼내서 접근해야!

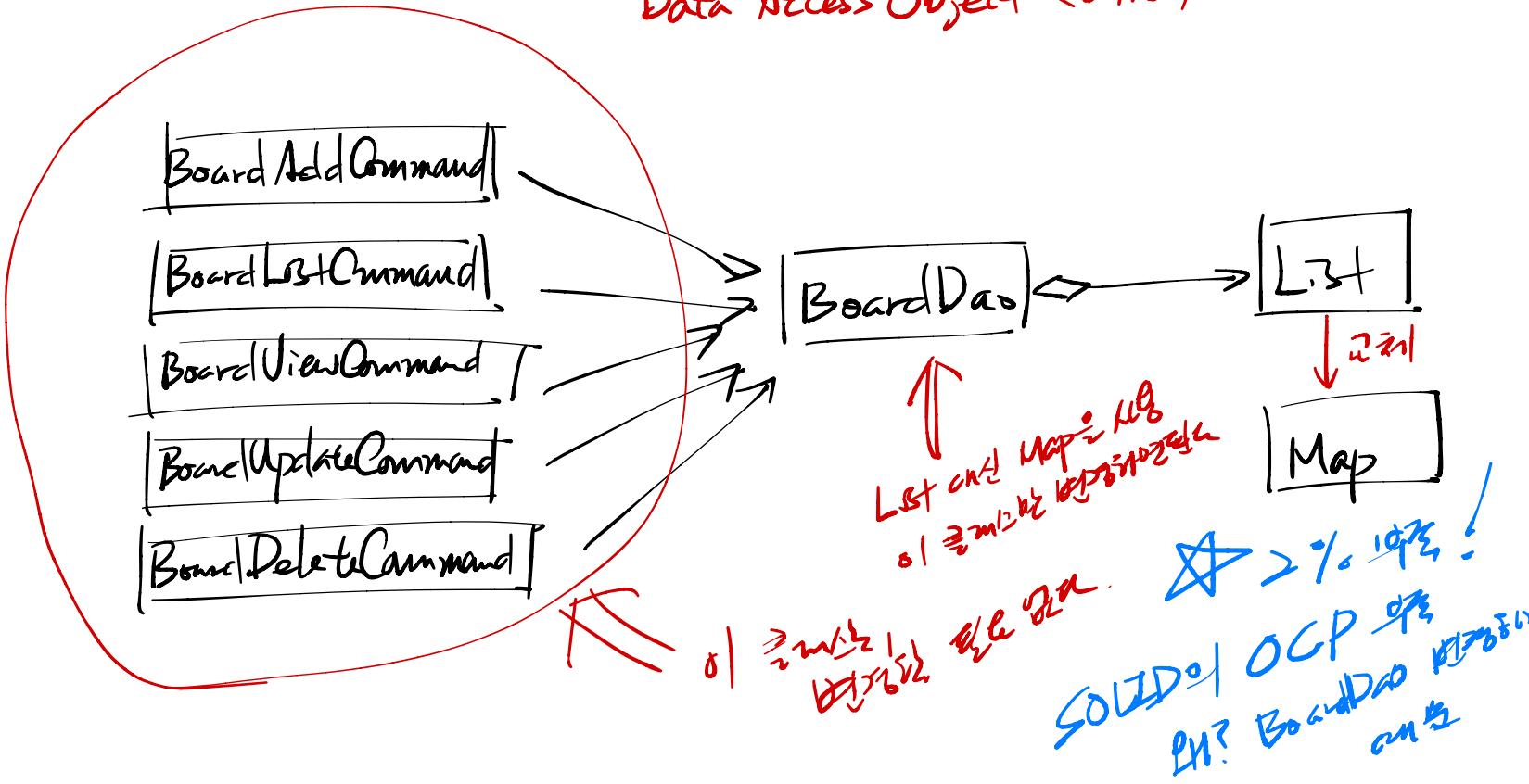


(1) 맵을 탐
key의 hash 값을 가지고
맵의 위치를 계산하는 방법이

35. 글로벌 키워드를 제거해보자

↳ 제거한 키워드는 뭘까? → 함수가 기반이다.

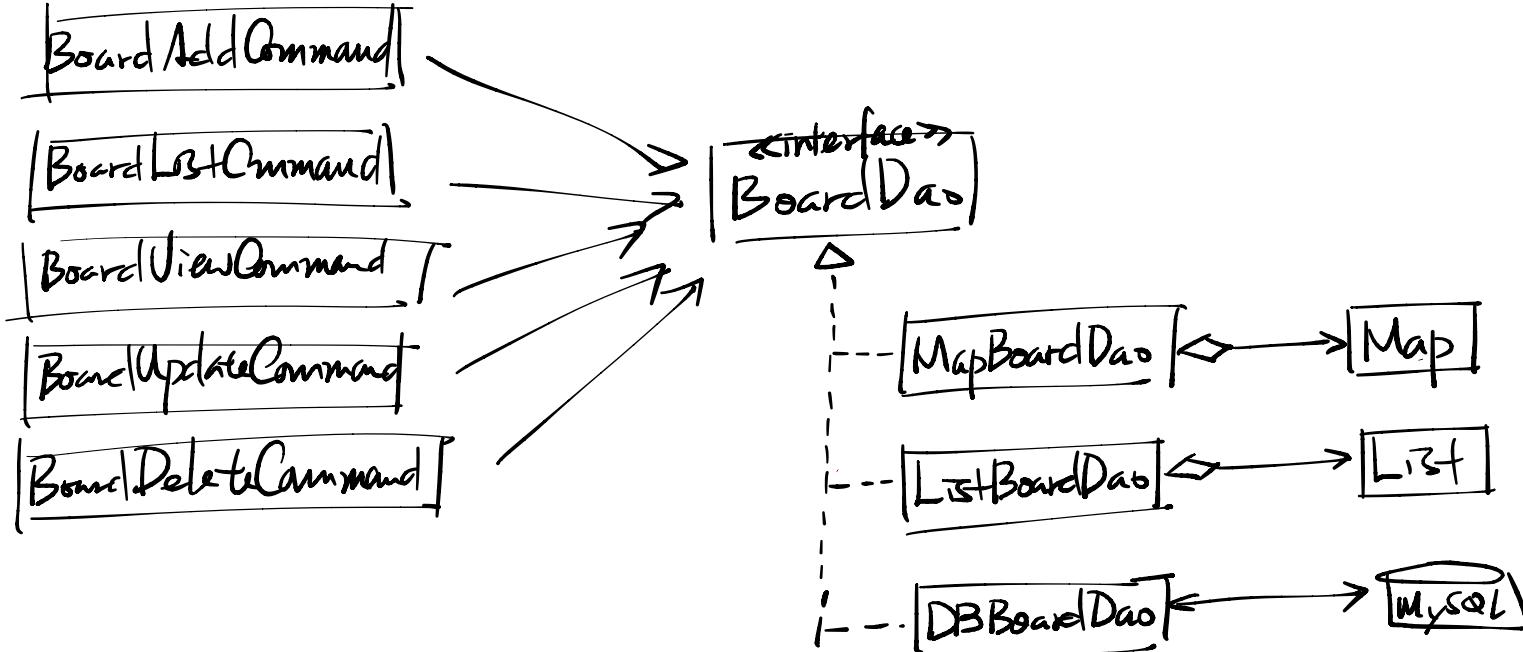
Data Access Object (DAO)



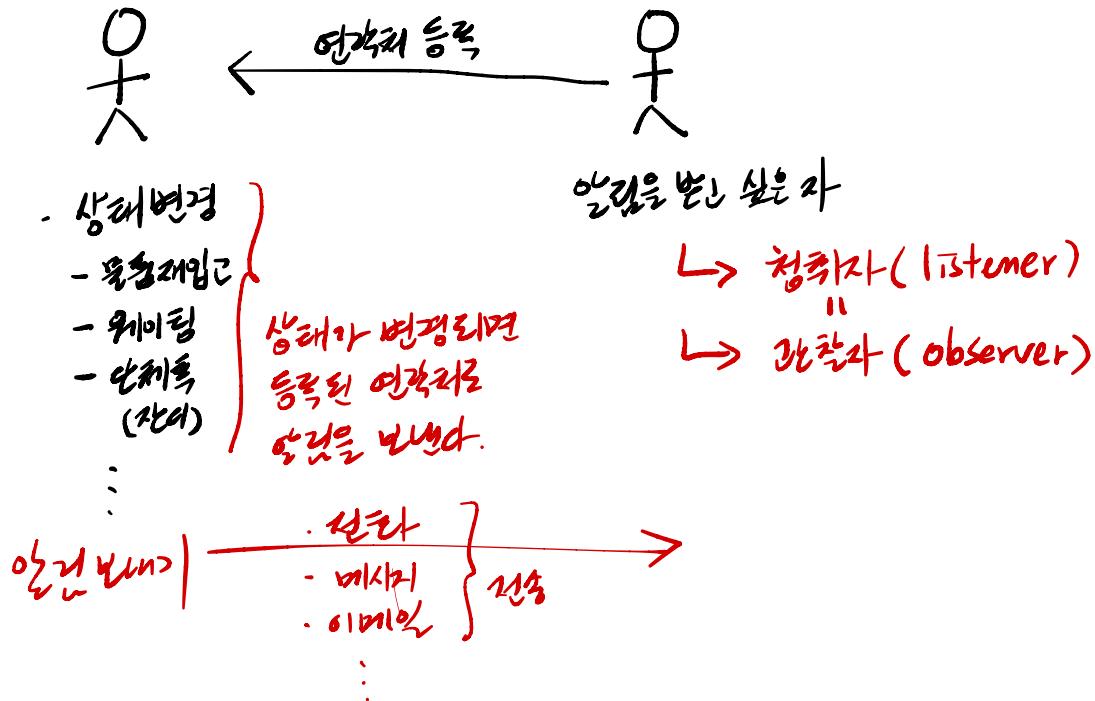
35. 글로벌 키워드는 final이면 됨

↳ 제한된 개수의 경우 → 제한된 개수의 제한된 개수.

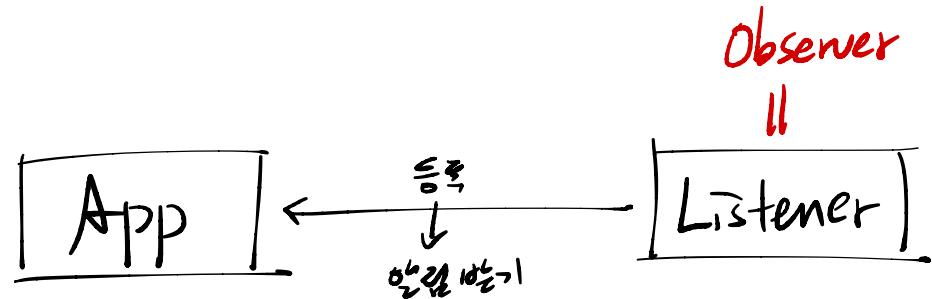
Data Access Object (DAO)



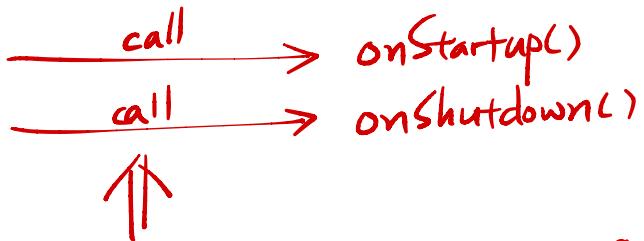
* 36. 알림 패턴 : GOF의 Observer 패턴
(설명문서)



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

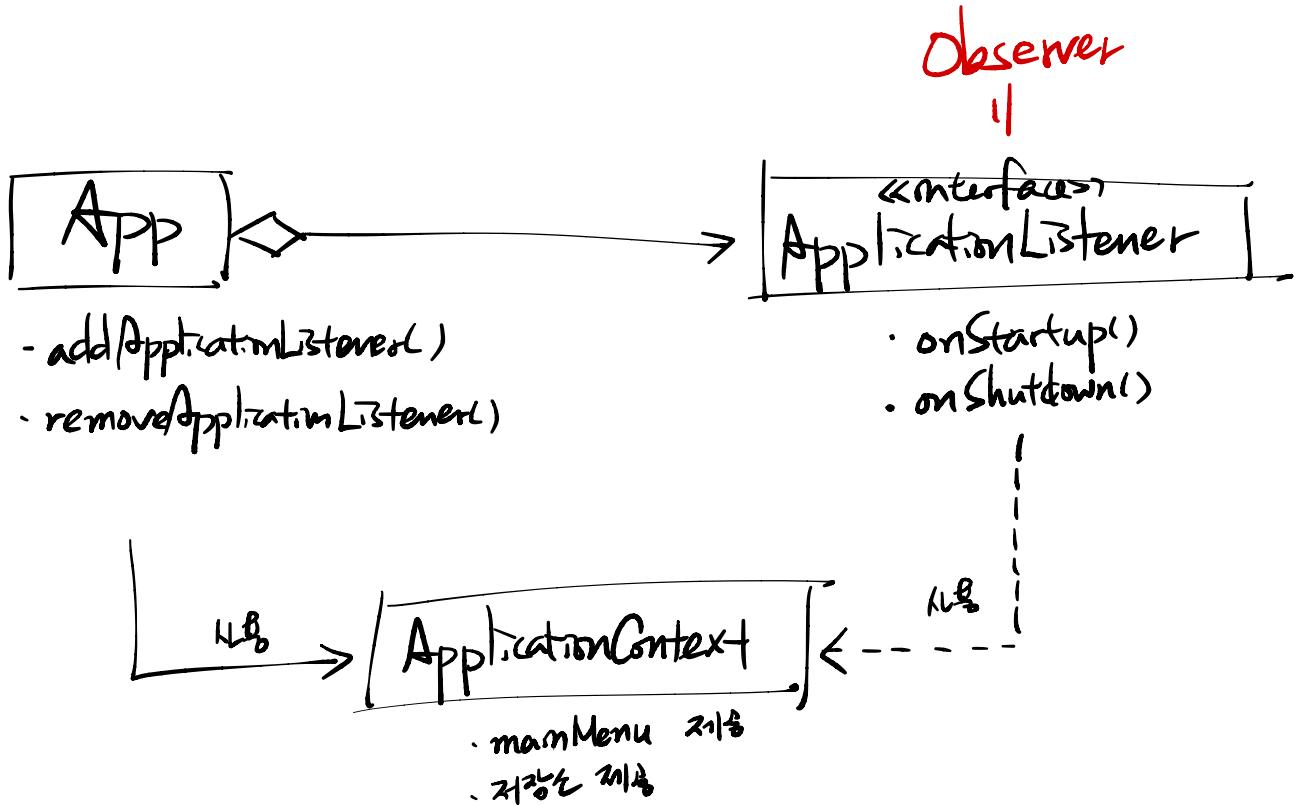


- 설정에 따라
- 사용자
- 풍경



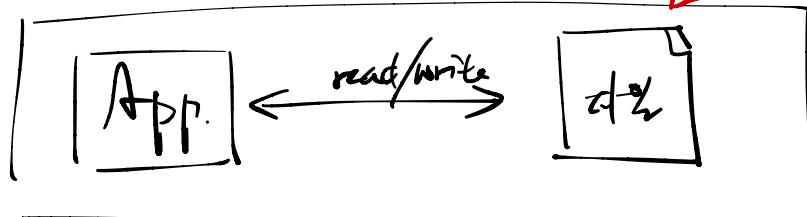
설정에 따라 설정에 따라 설정에 따라
설정에 따라 설정에 따라 설정에 따라

* GOF의 Observer 패턴 대처



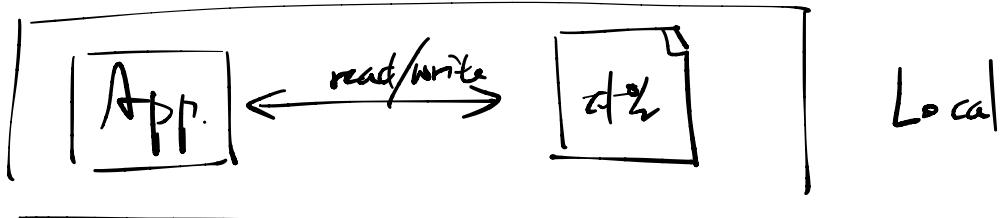
31. Application 간에 데이터 공유

현재 상태



→ App.의
데이터로 데이터를
Local

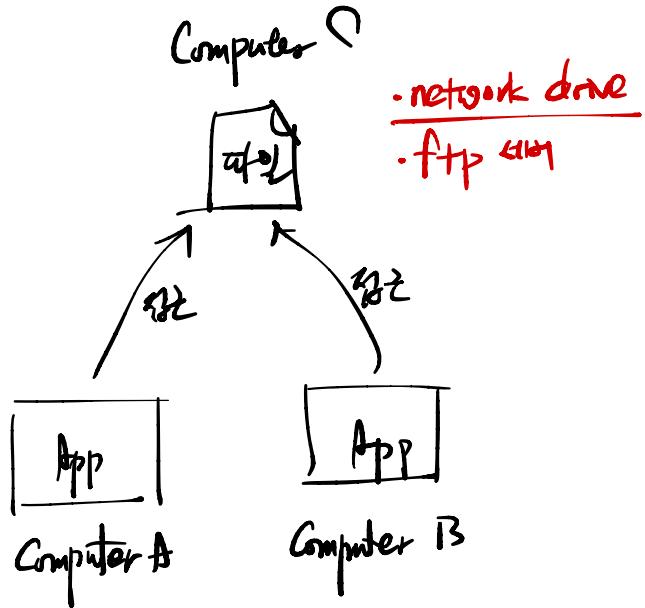
App.-간의
데이터 공유는!



•
•
•

31. Application 간에 파일 교환

① 파일로드된 파일 공유



파일 공유점

- 동시에 여러 App. 파일을
교환할 수 있다



파일을 쉽게 찾을 수 있다.

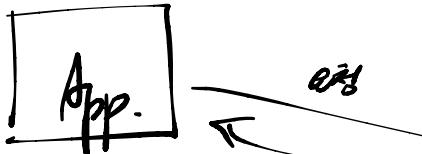
31. Application 간에 데이터 교환

② 멀티프로세스 애플리케이션

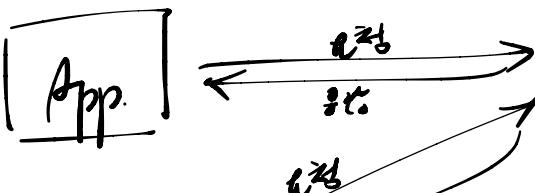
- networking prog.
- multi-tasking prog.
 - ↳ multi-threading
 - ↳ synchronous

애플리케이션
Data는 쓰고 읽을 때
Data를 주고 받음

Computer A



Computer B

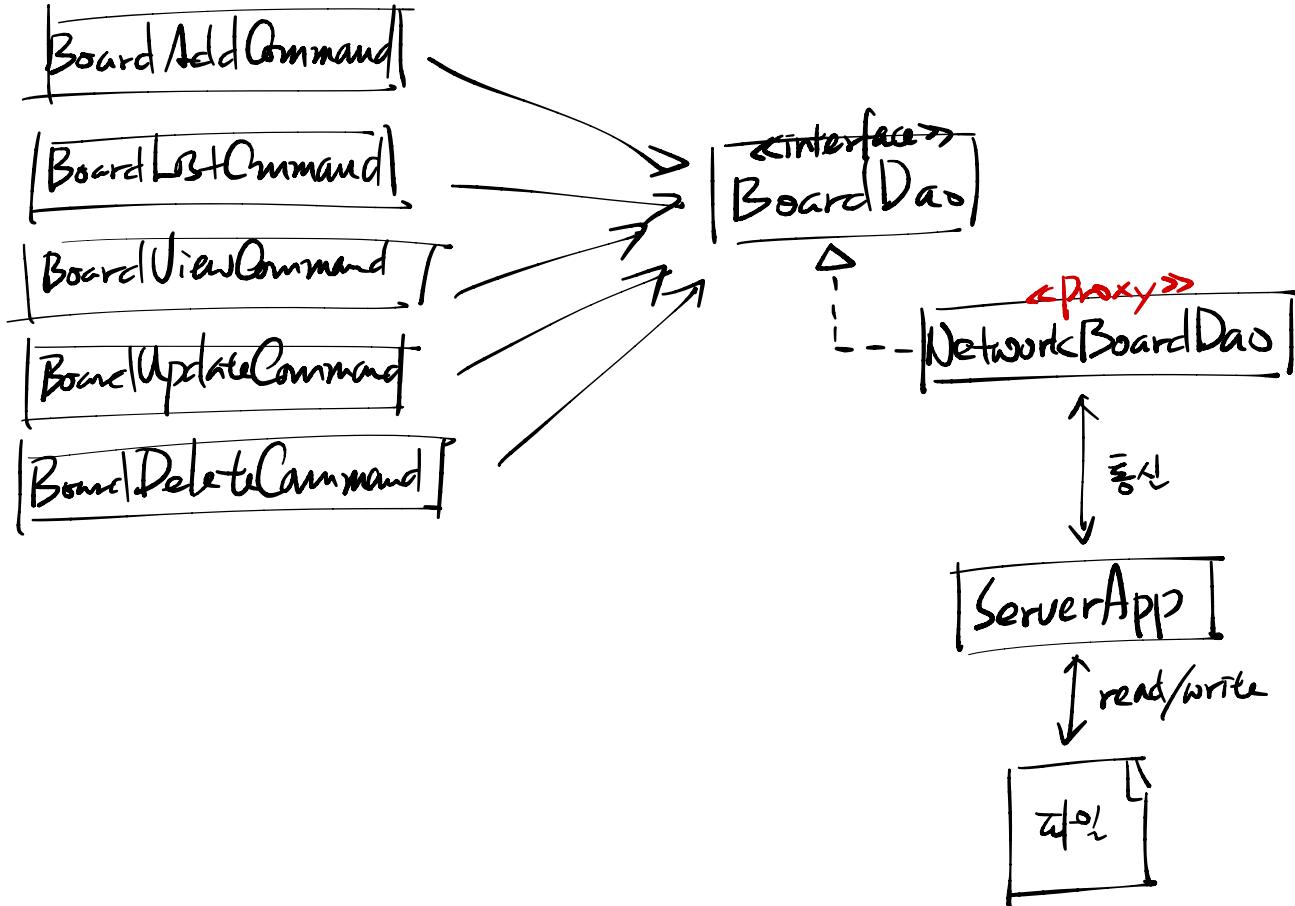


Computer C

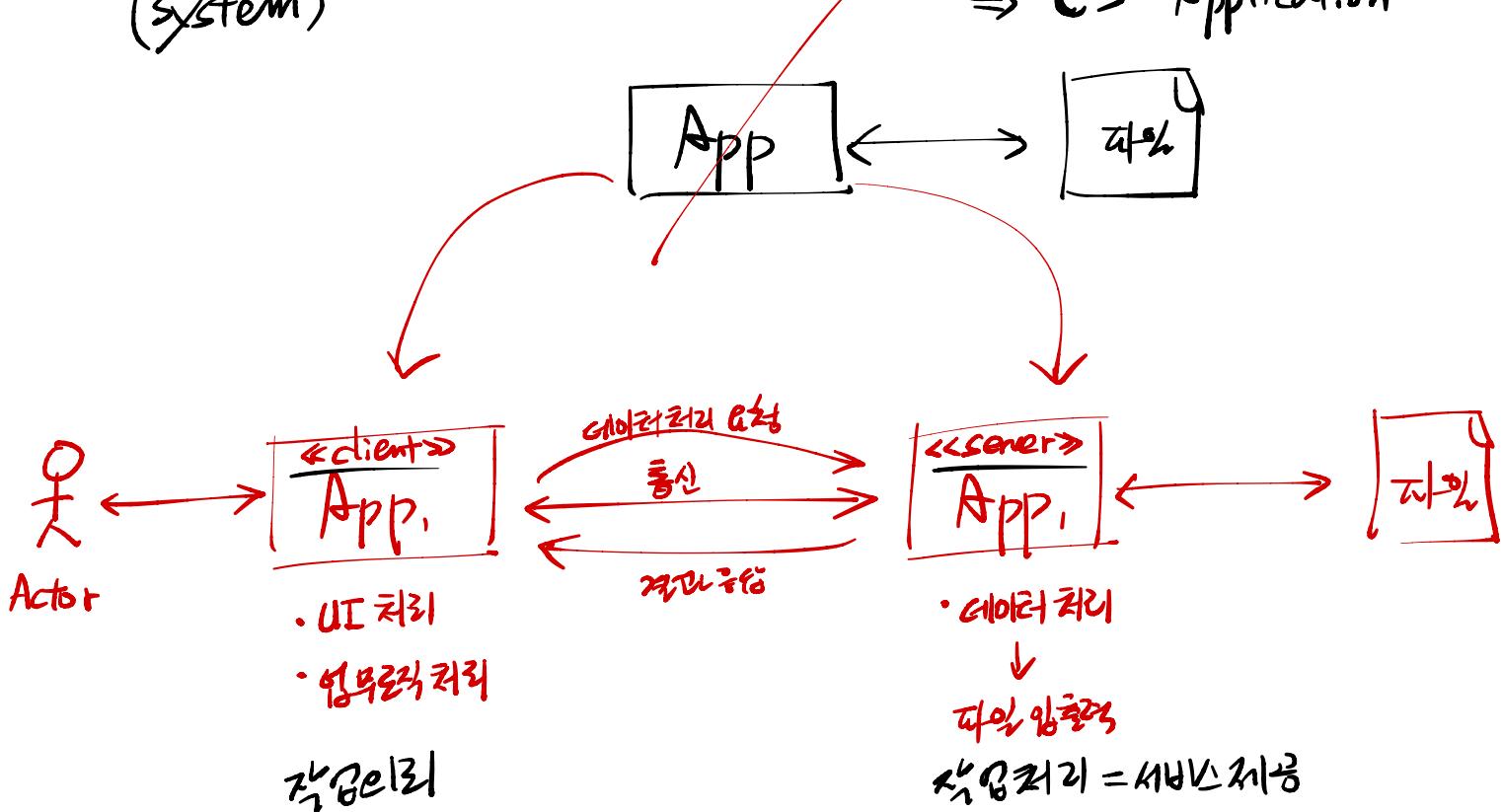


networking

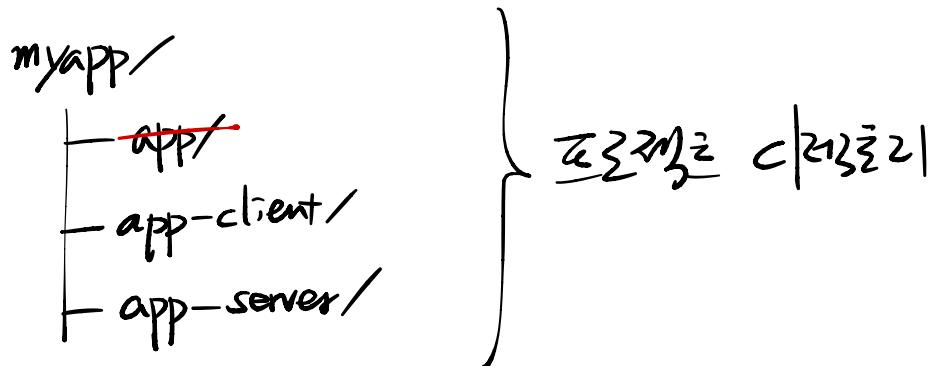
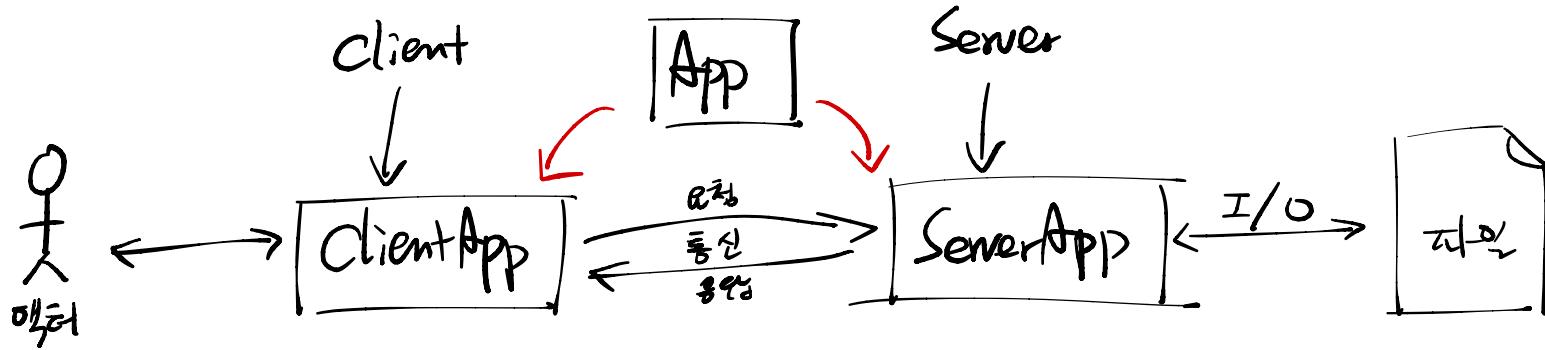
단계 32번의 단계 32번 일정 시스템
(process) (process) 단계 32번의 일정 시스템
단계 32번의 일정 시스템



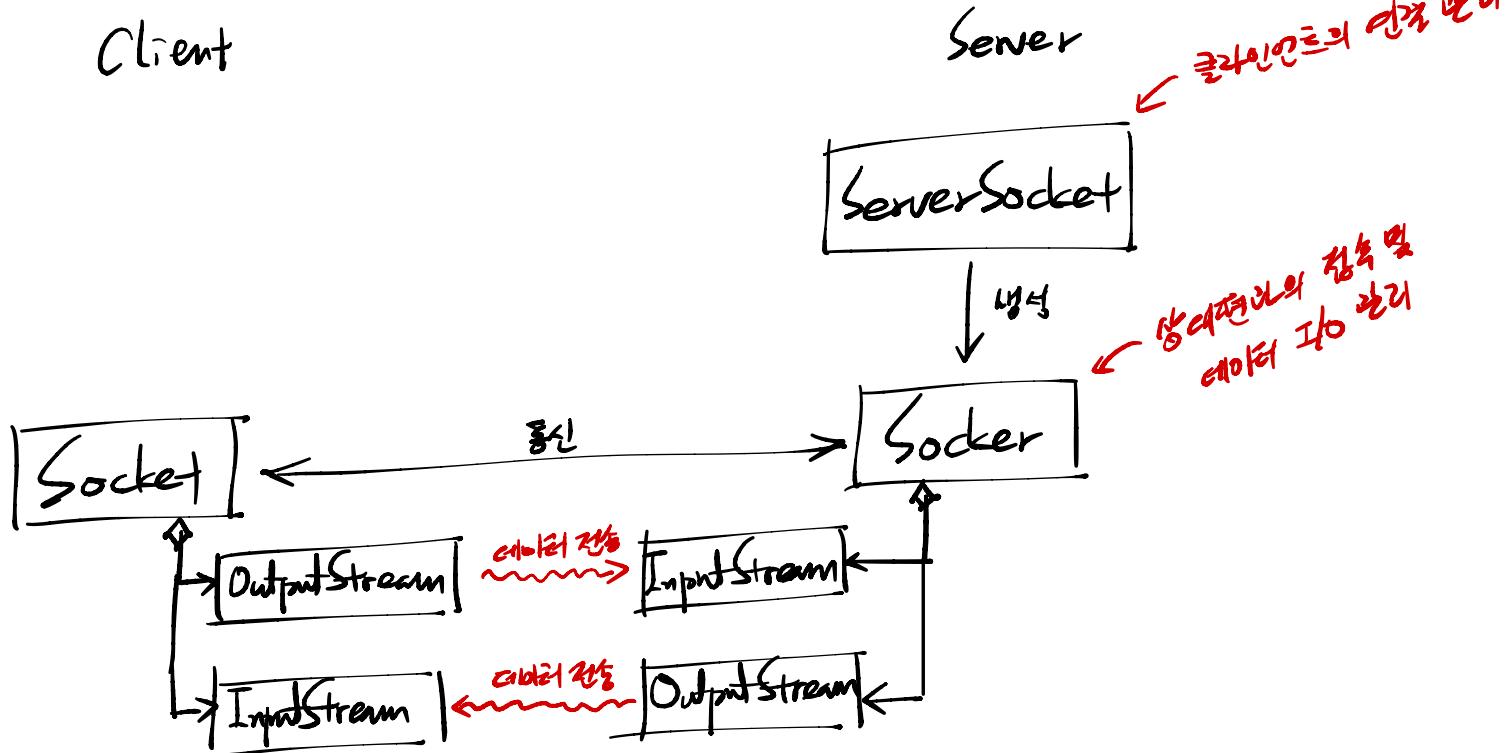
* Software Architecture : Client / Server Architecture
(System) ⇒ CS Application



* CS Architecture 구조

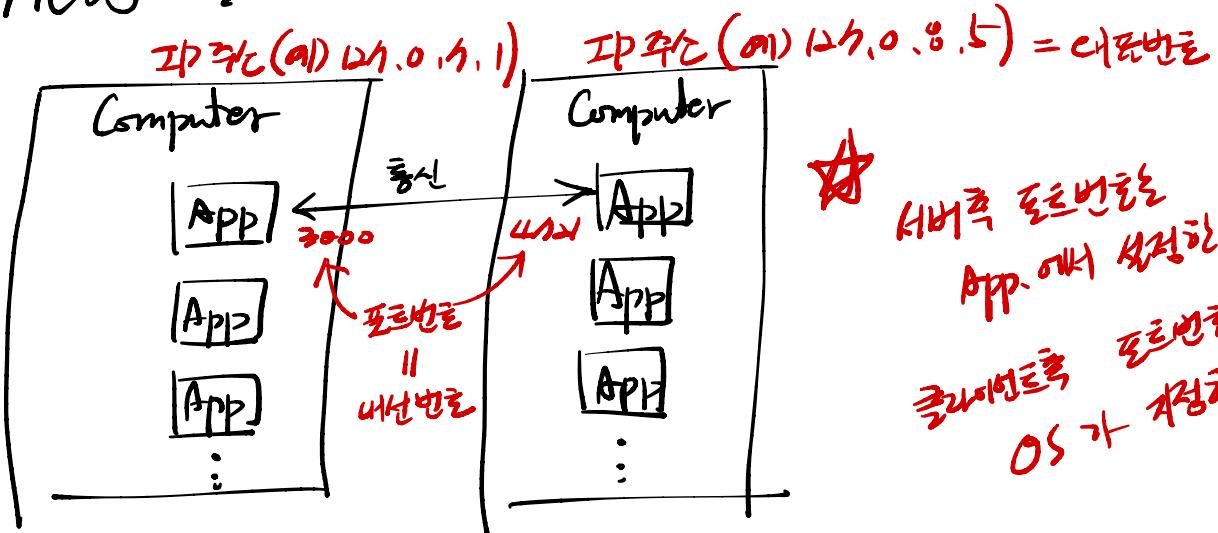


* Networking API API



* ServerSocket

new ServerSocket(포트번호, 대기열크기)



☆
내부적 포트번호는
App.에서 설정된다
클라이언트 접속 번호
OS가 지정한다.

* 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"

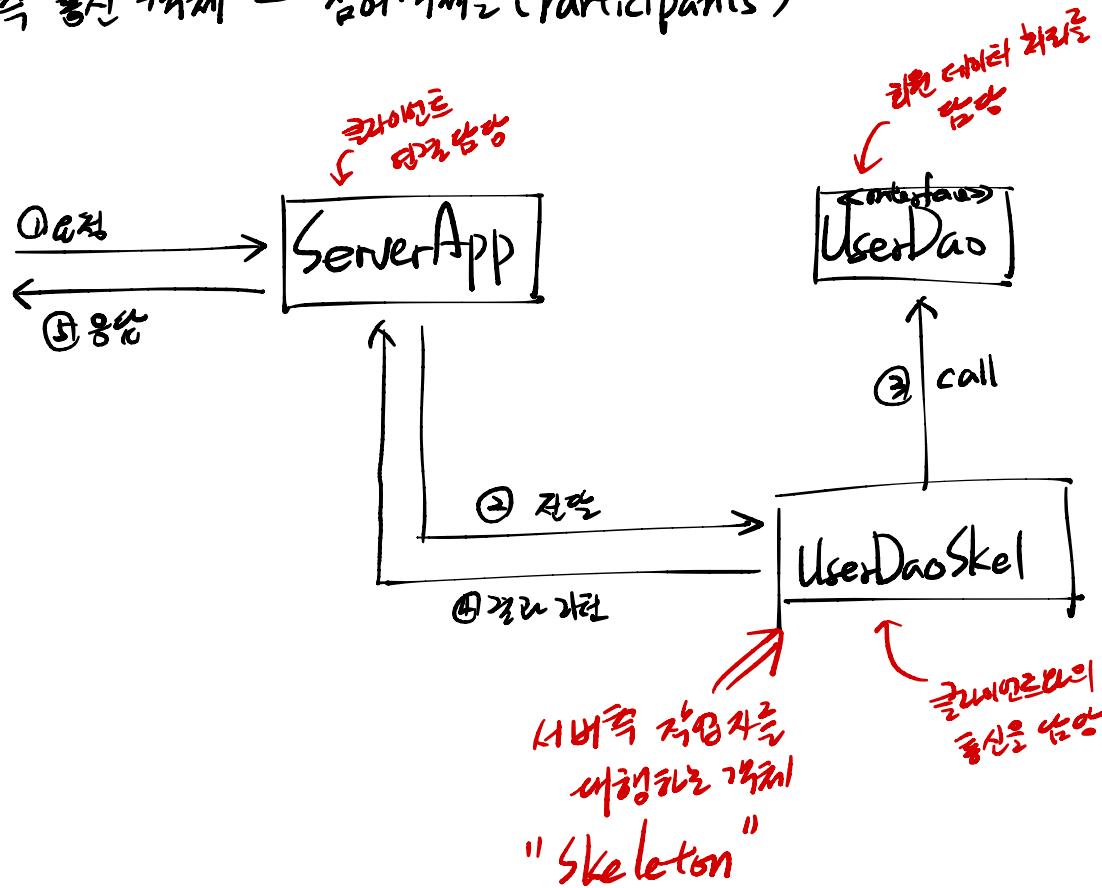
전송 데이터 - *

Server

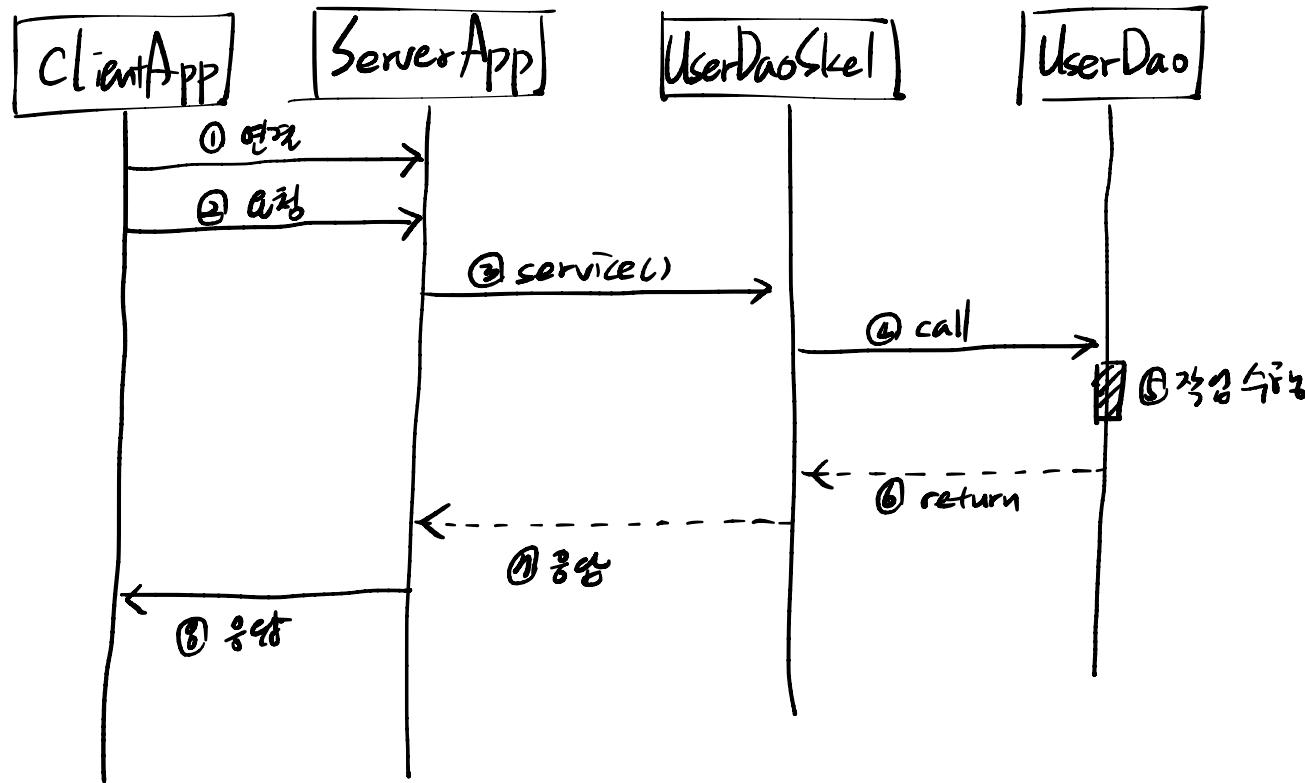
응답 상태 - "success" | "failure"

응답 데이터 - *

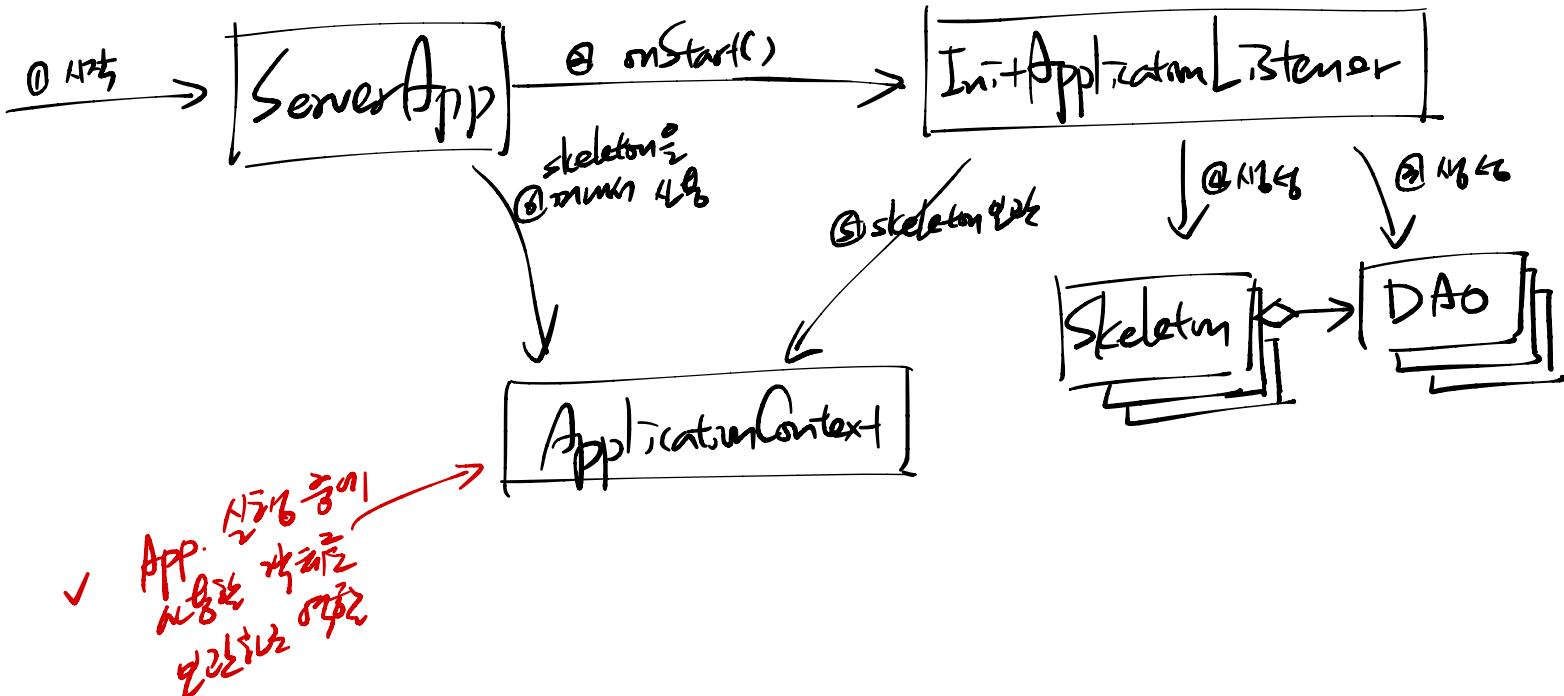
* 서버측 통신 구조 - 참여자들 (Participants)



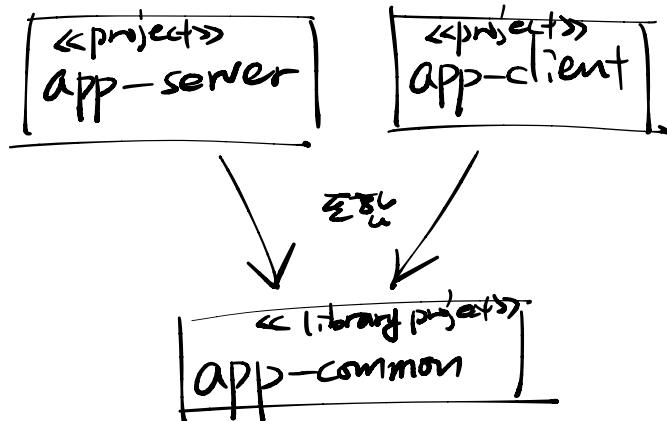
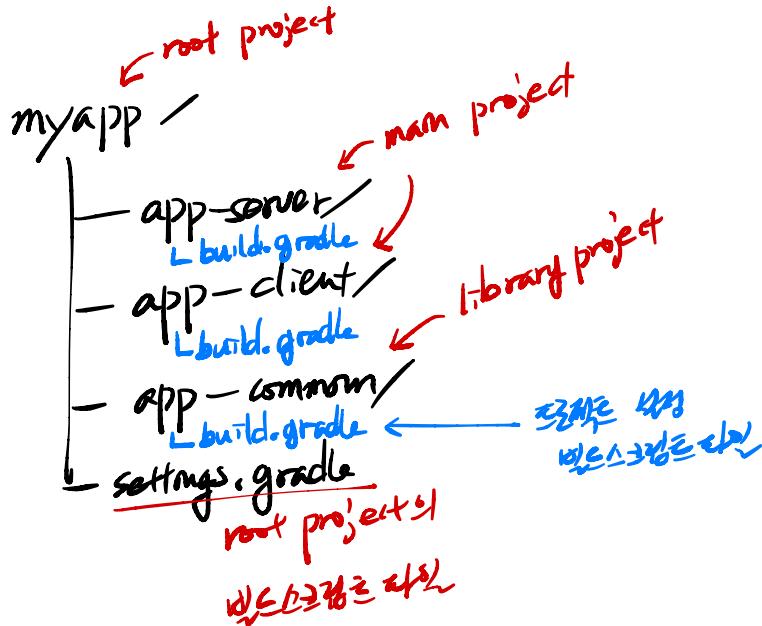
* 서버측 통신 구조 - Sequence Diagram (클라이언트가 서버의 서비스를 호출하는 과정)



* ServerApp in Skeleton

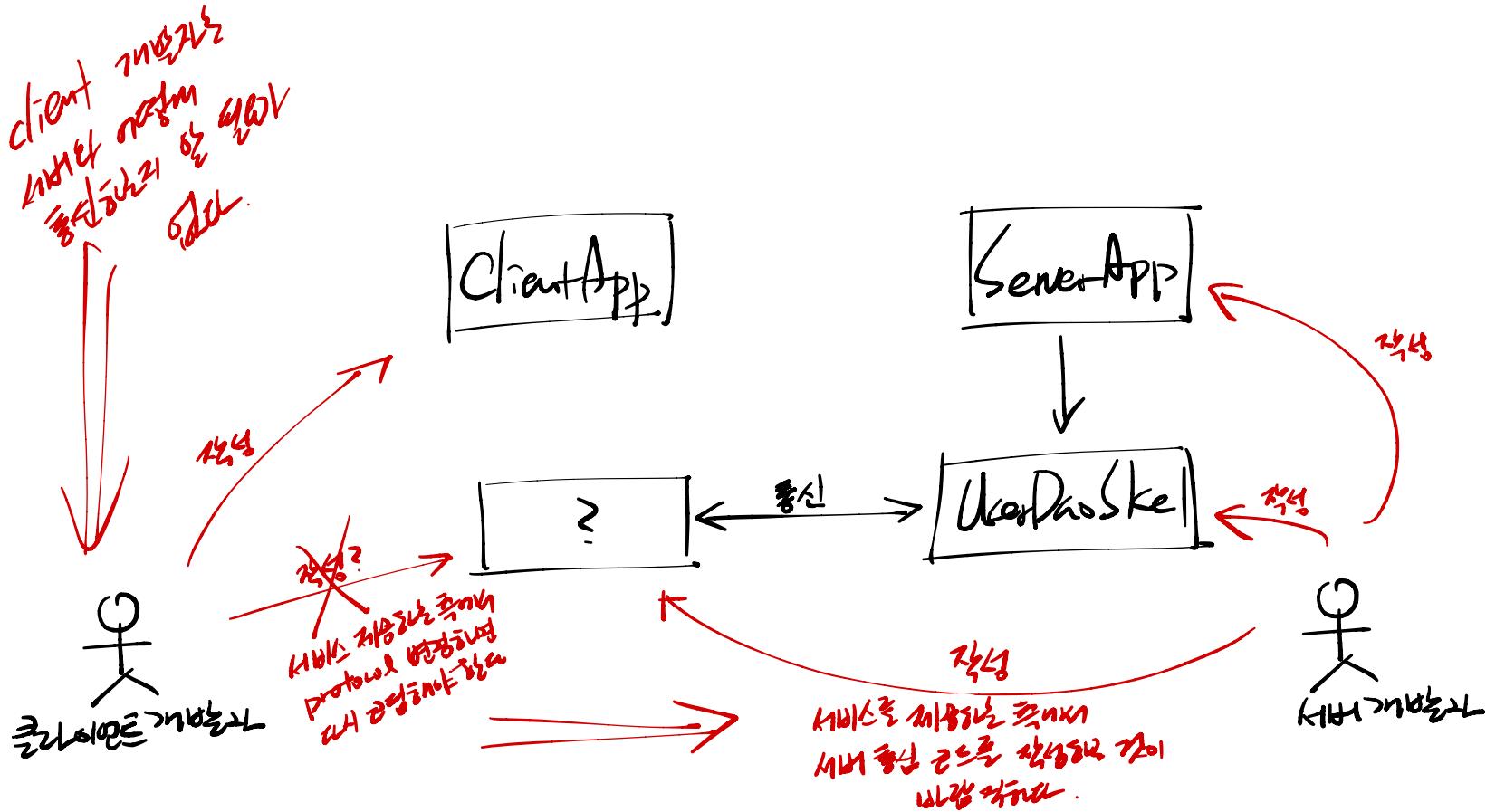


* 2019-2021 အချက် - အကျဉ်းချုပ်၏ ရေးဆွဲမှုပါမ်း

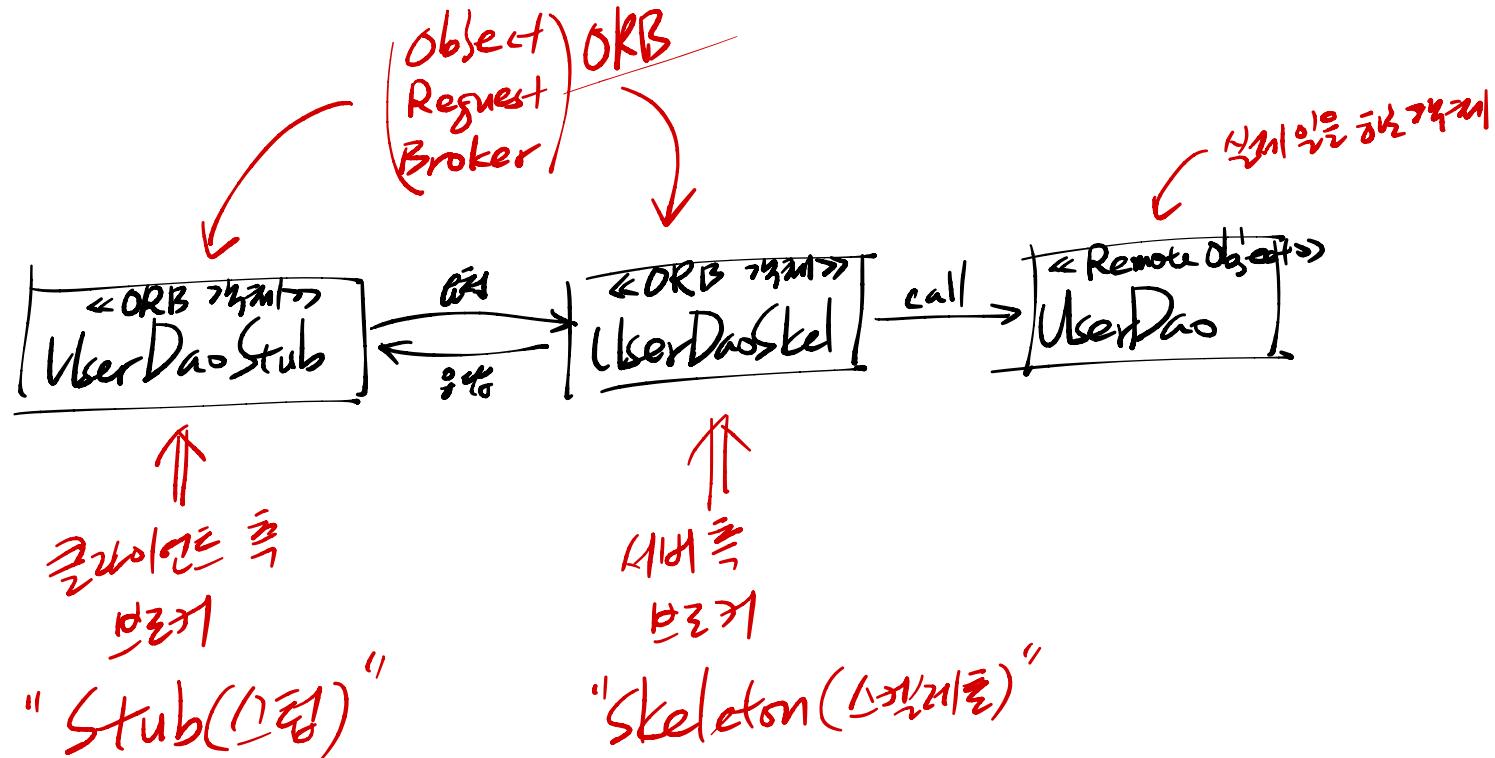


- VO ၁၃၈
- Net ၁၃၈

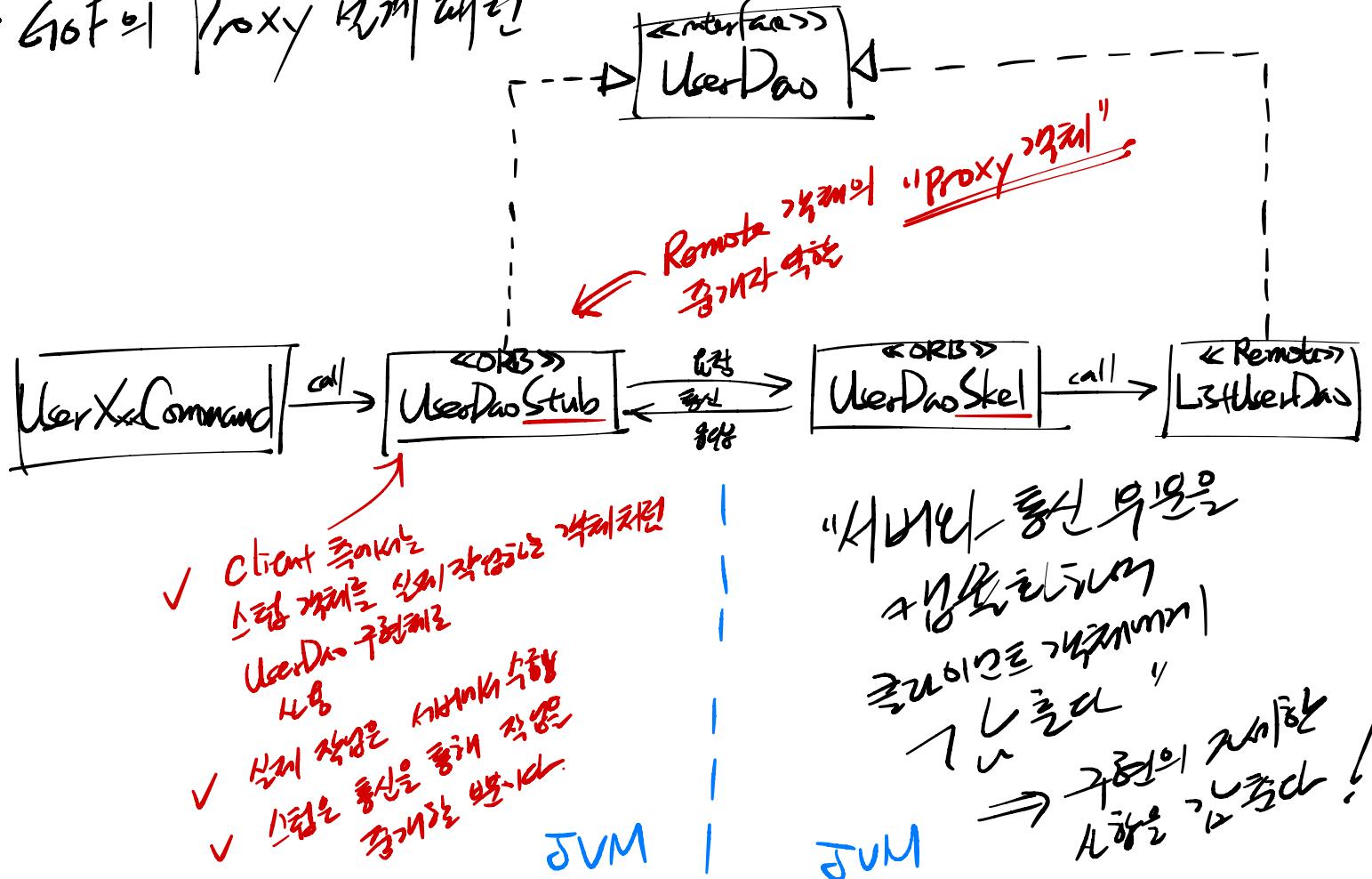
* Server 및 Client



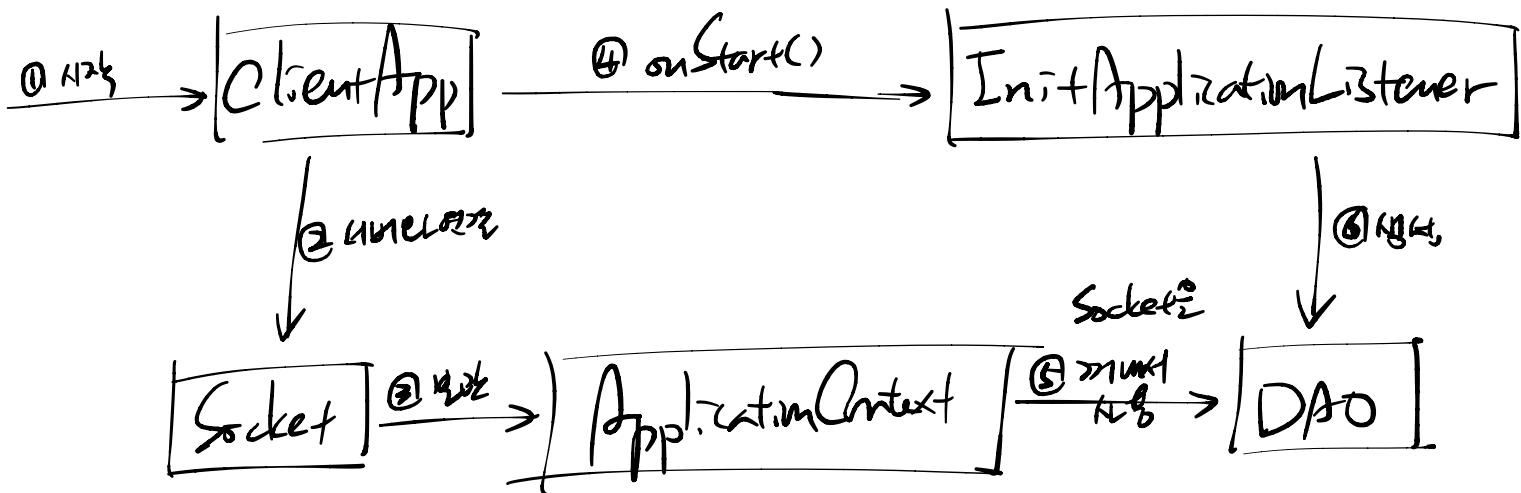
* Skeleton in Stub



* GOF의 Proxy 패턴 구현

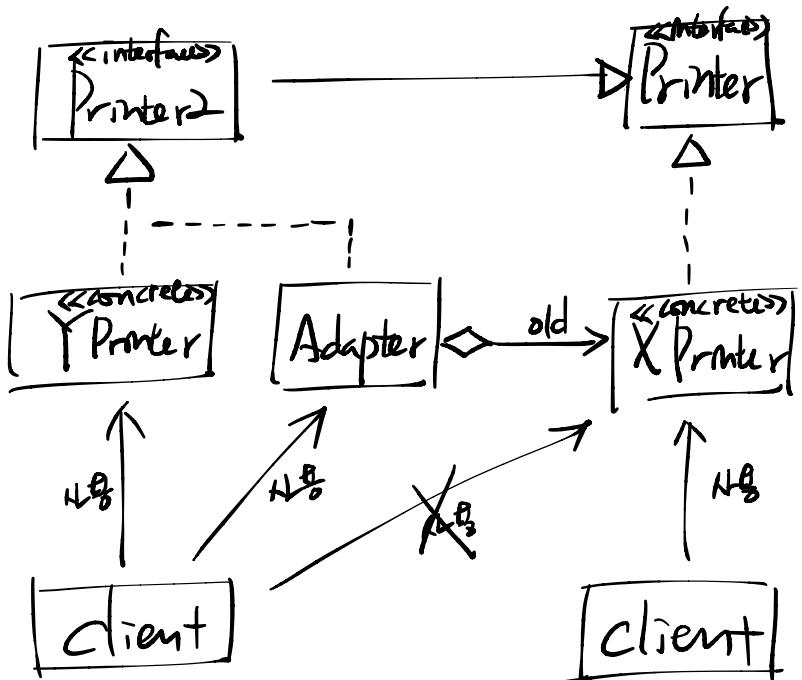


* Client Application & Stub

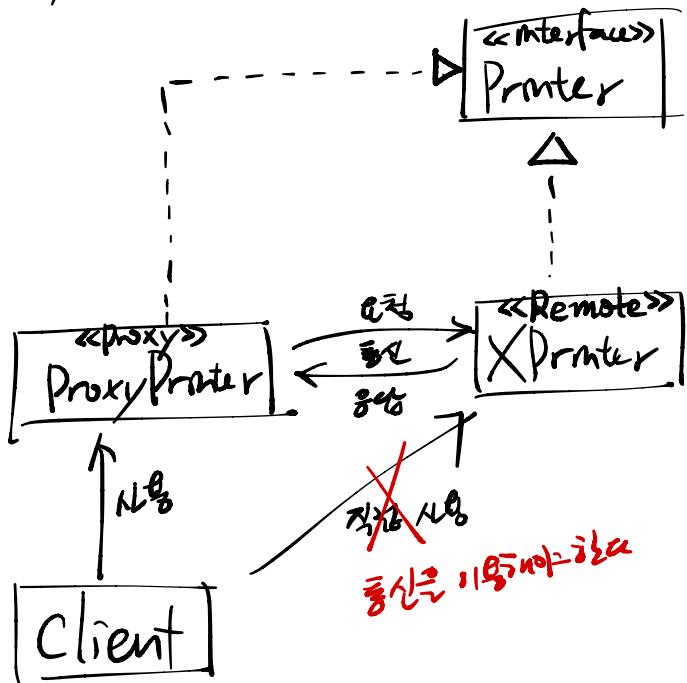


* Adapter 퀵 vs Proxy 퀵

① Adapter

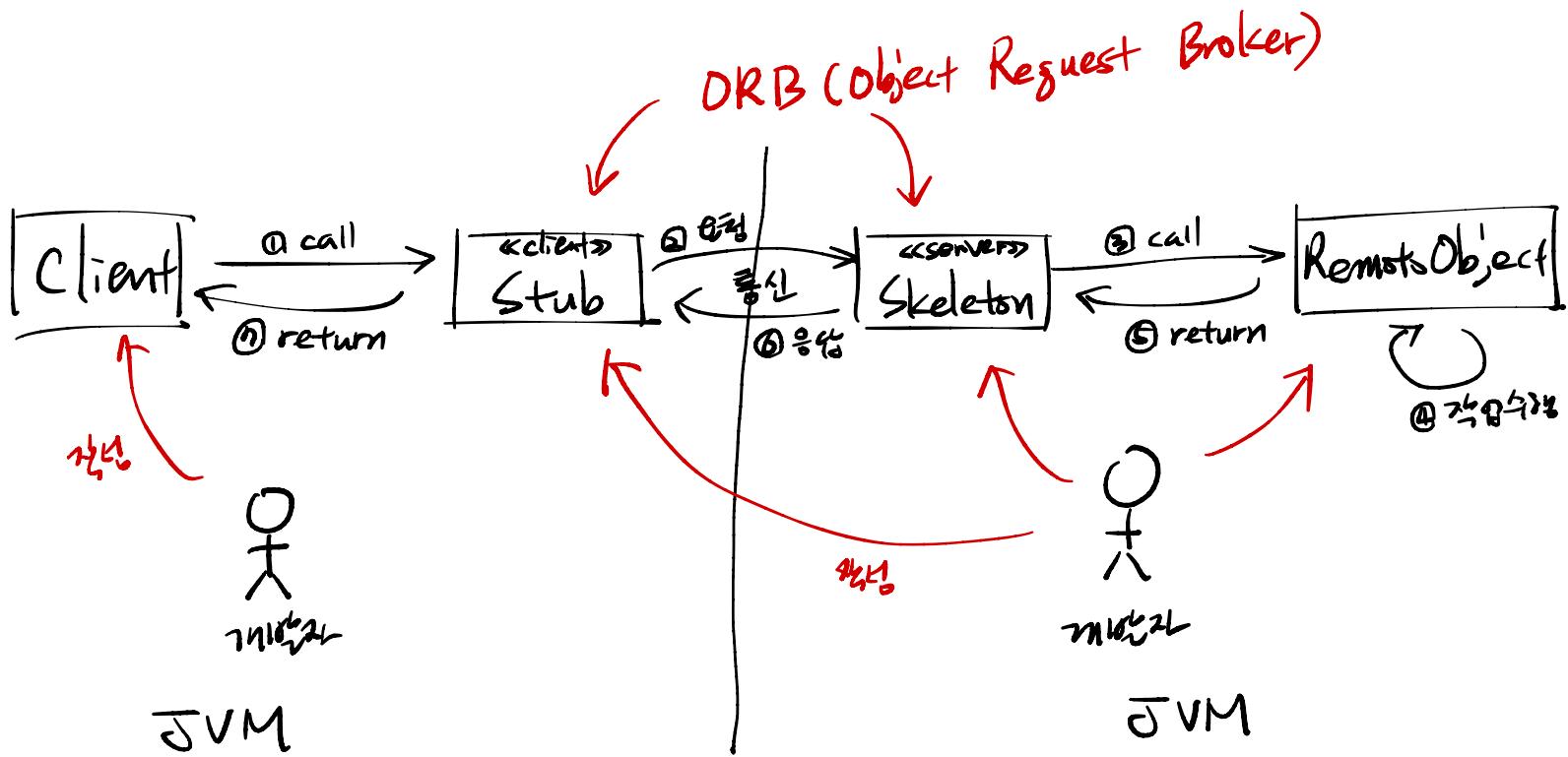


② Proxy



* Remote Object 개념

↳ 다른 JVM에서 만든 객체
(프록시)

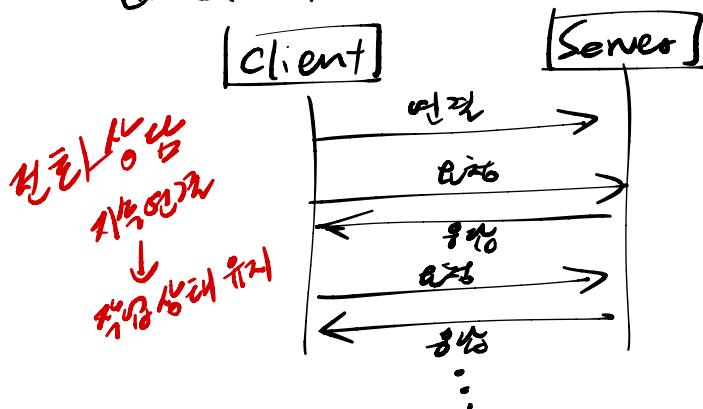


38. Stateful vs Stateless

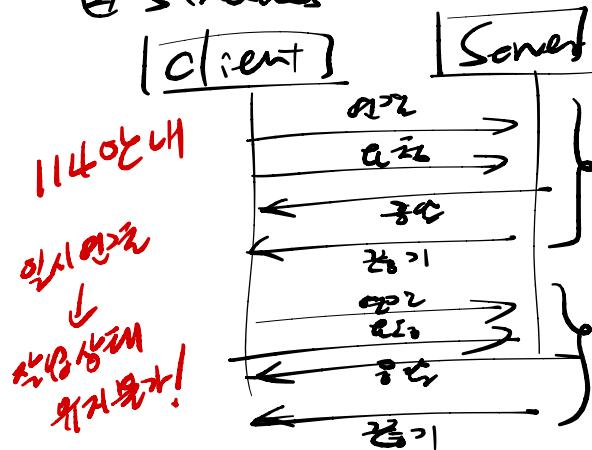
- * Stateful 웹서버 - 한번 연결한 후 여러번 요청/응답 처리
- * Stateless 웹서버 - 한번 연결한 후 한 번 요청/응답 처리



① Statefull

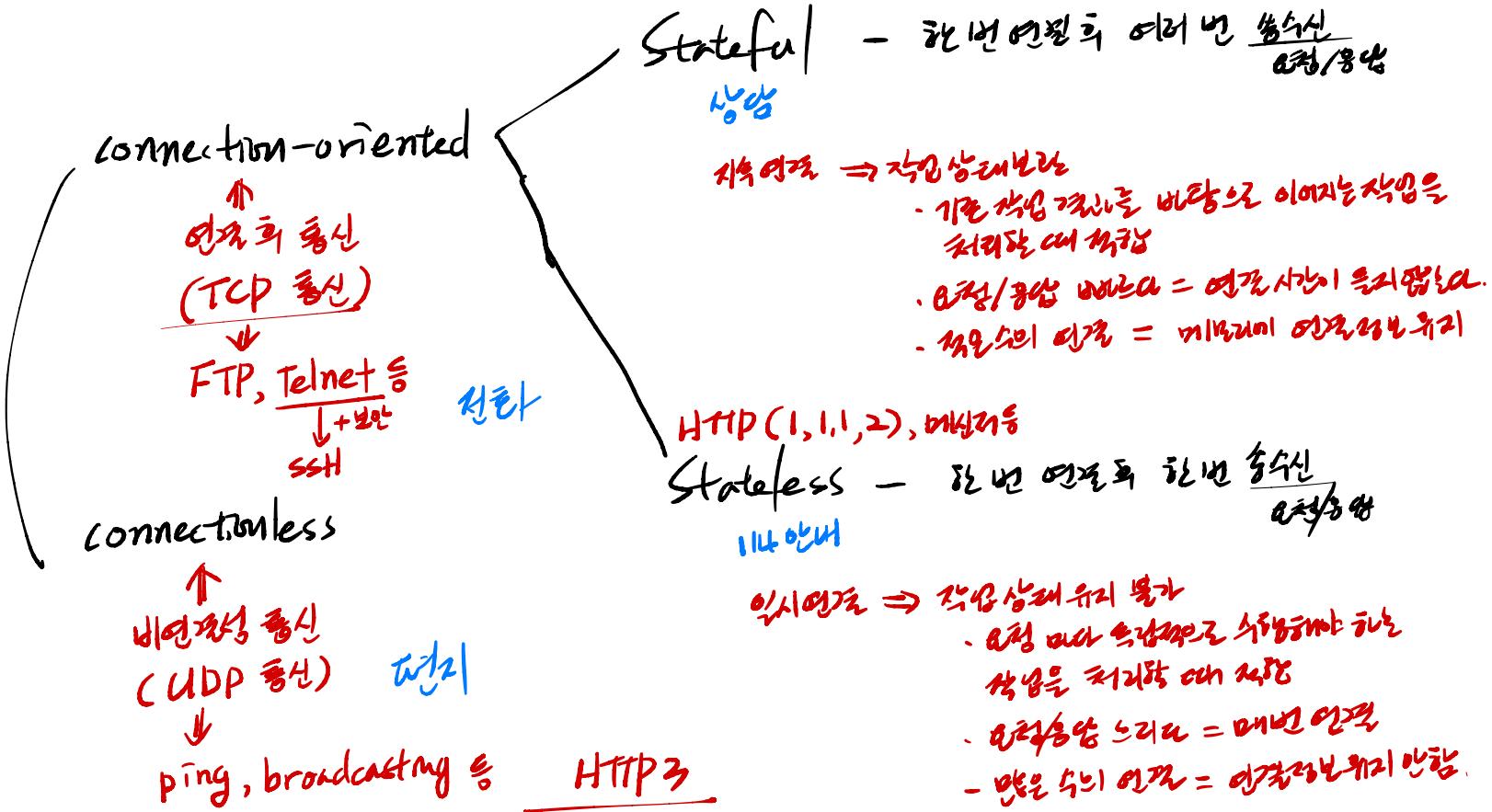


② Stateless

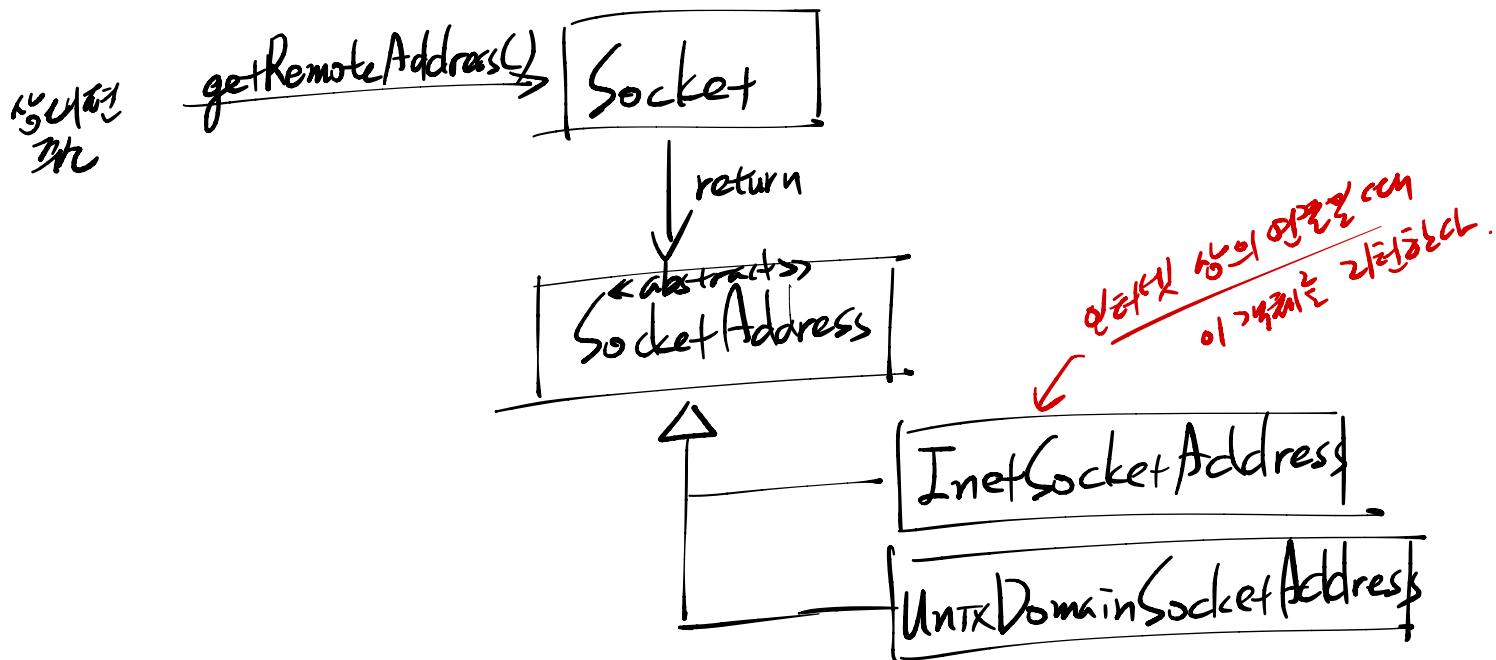


* 통신 18'시'

FTP, Telnet, SSH, 스트리밍, 채팅 등

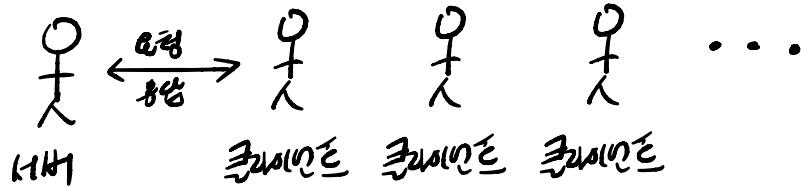


* Socket API



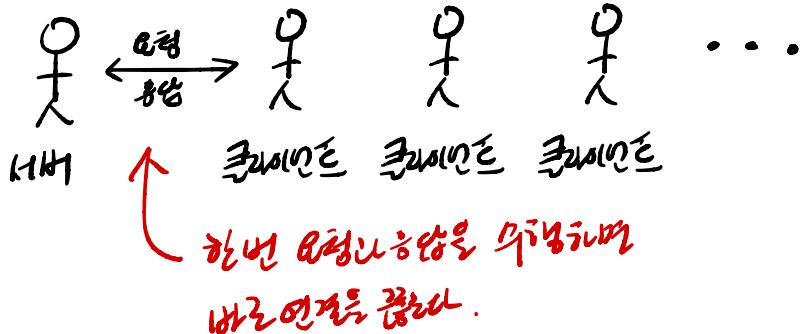
39. 미리그레드 층

① 아종 클러스터드 요청 처리 - Stateful



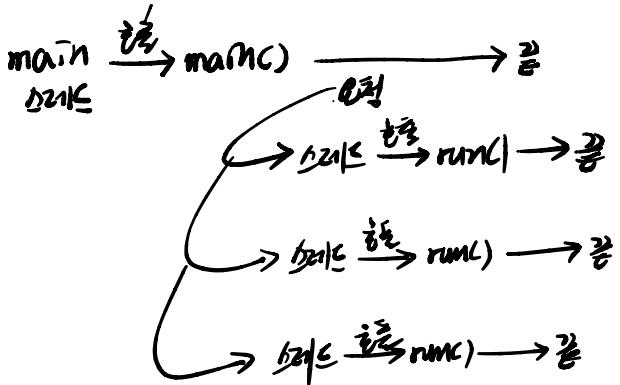
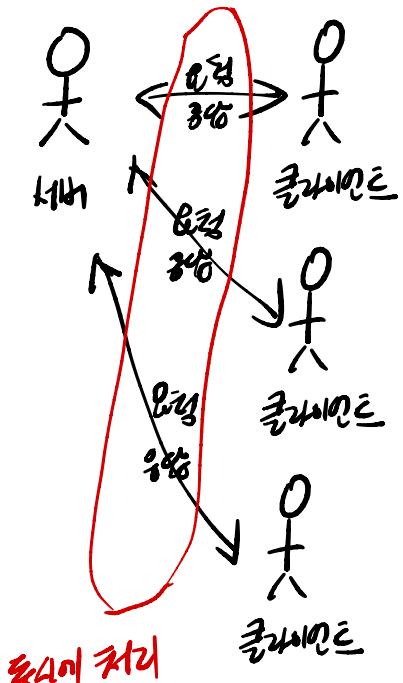
- ✓ 서비스와 연결된 클러스터드가
여기의 위치를 알기 때문에
다른 클러스터드는 가리키지 않아.
- ✓ 키값은 동일하게 처리된다.

② 아종 클러스터드 요청 처리 - Stateless



- ✓ 한 번 요청 → 한 번 응답/응답
- ↓
- Stateful 방식과는 대조적으로 확장성이 좋다.
- ✓ 키값은 유동적으로 처리

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



고정화되어 사용자 고려하지 않고
고정화되는 대화형은 틱

클라이언트 고정을 고려해보자!

개인?

영향으로 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

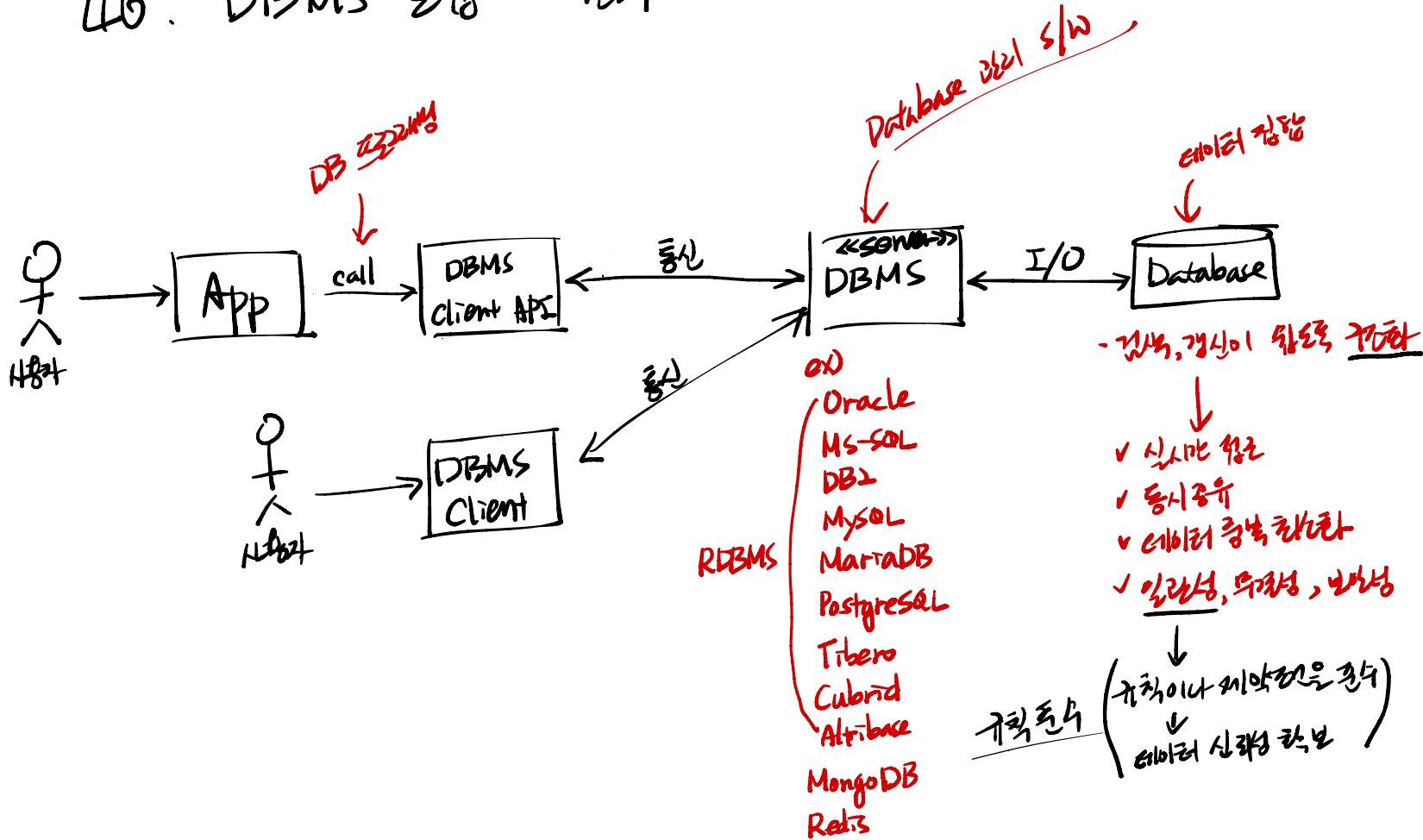
}

}

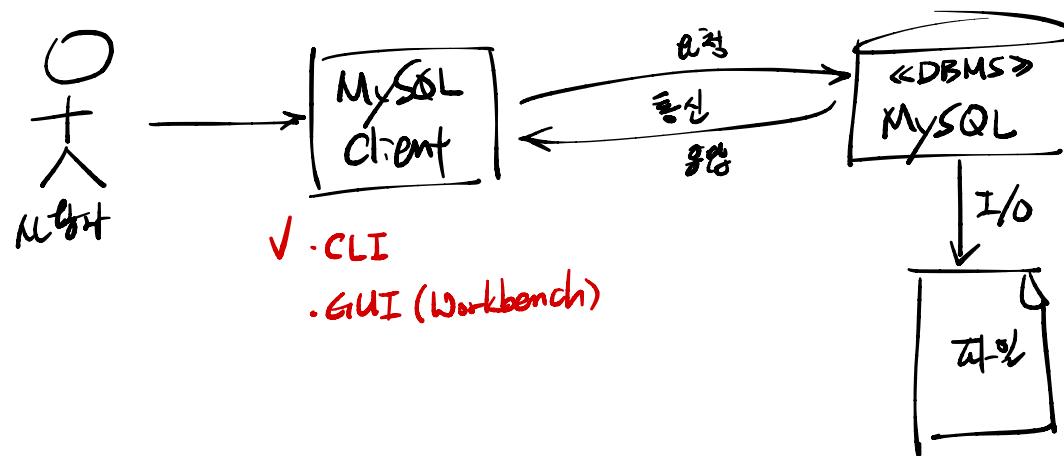
쓰레드.start();

여기서 쓰레드가 시작된다.

40. DBMS 속도 - 톤



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

② DBMS 사용자 관리

③ 01012181011 484

create database globe11010101 default character set utf8 default collate utf8_general_ci

~~도시락을 사면서~~
~~이제는 막수~~
"드디어 왔어!"

110. DBMS 예제 - DBMS 사용

④ 관리자 권한으로 MySQL 접속

grant all on studydb.* to 'study'@'localhost';
MySQL 명령
DBMS 명령
MySQL 명령
(@: MySQL 명령)
user
host

⑤ 데이터베이스 목록 보기

show databases

⑥ 데이터베이스 사용

use studydb

⑦ 테이블 목록 보기

show tables

⑧ 테이블 구조 보기

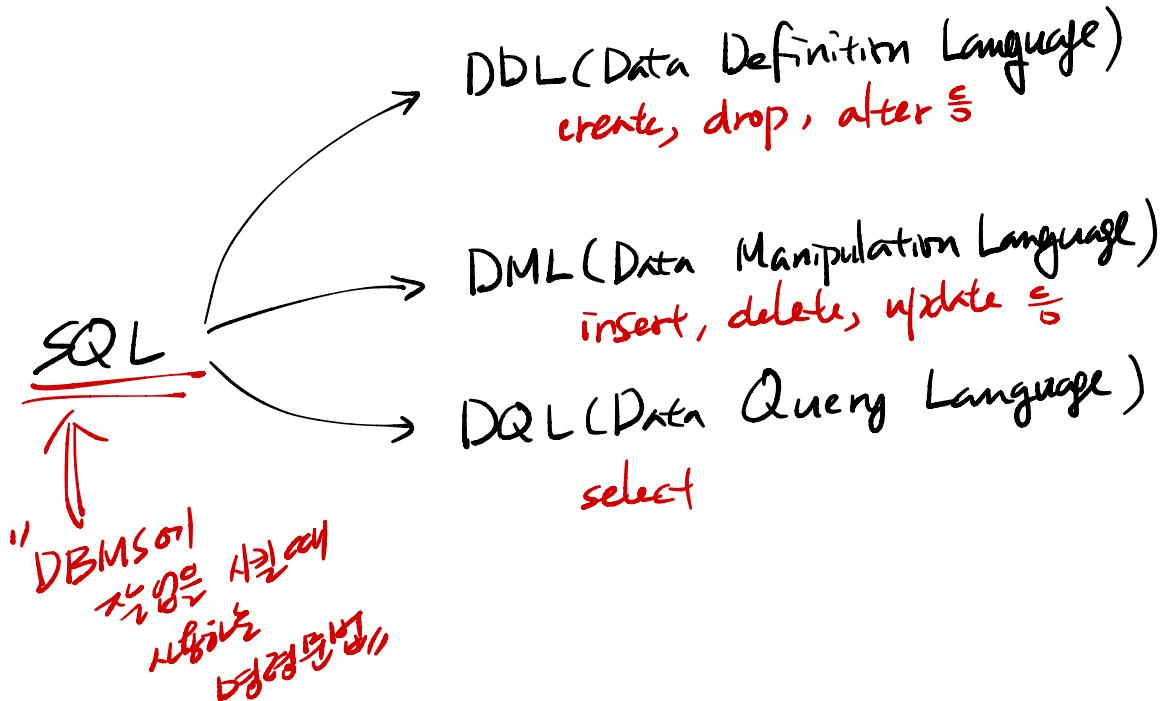
describe studydb.t1

desc studydb.t1

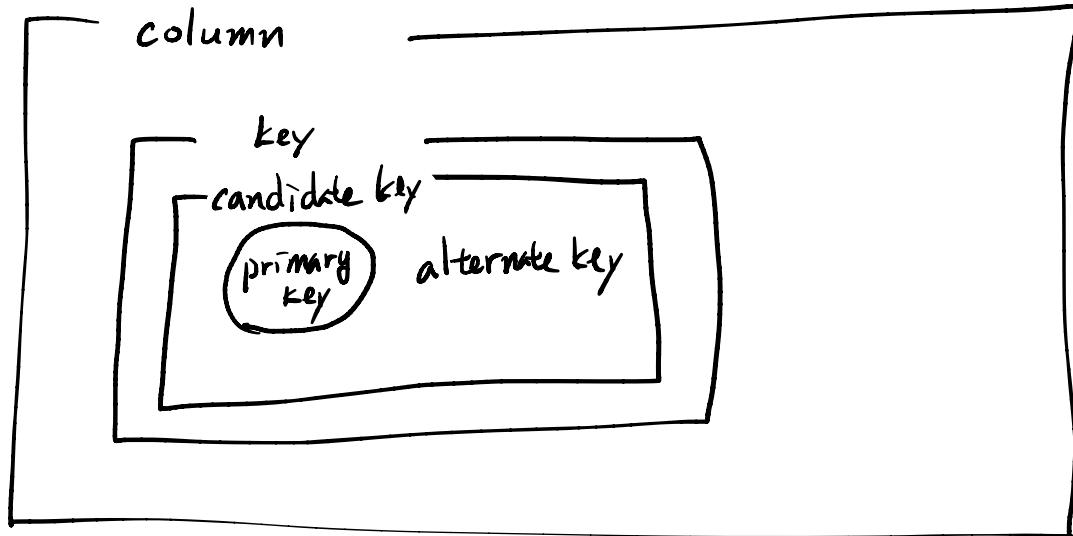
⑨ 사용자 목록 보기

select user, host from mysql.user

110 . DBMS ୱେଳେ - SQL (Structured Query Language)

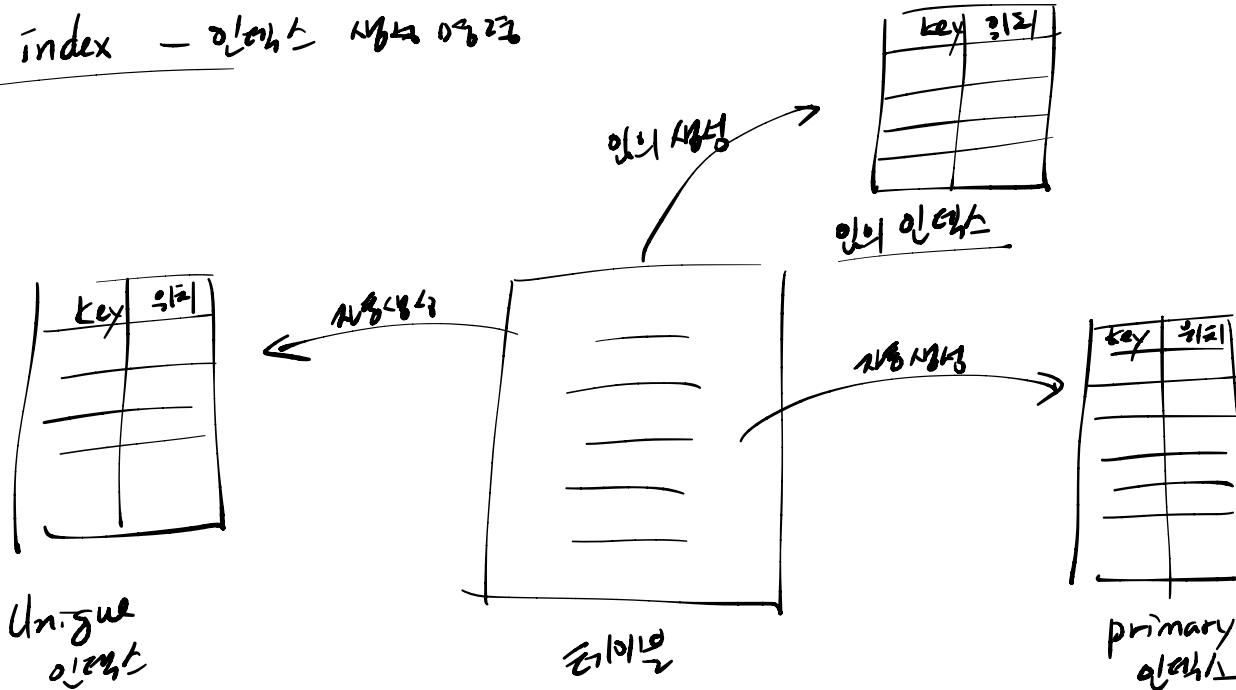


* key - 데이터를 구분할 때 사용하는 키임



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

* index - 인덱스 구조화된



✓ 고유한 고유한 원소들은 인덱스 (이후에 고유한)

✓ 고유한 고유한 원소들이 있다.

같은 원소는 아예 존재하지 않음.

고유한 원소들

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

* projection vs selection

선택
제거
select.m

no	name	class	working	tel
			F	
			N	
			N	
			F	
			N	

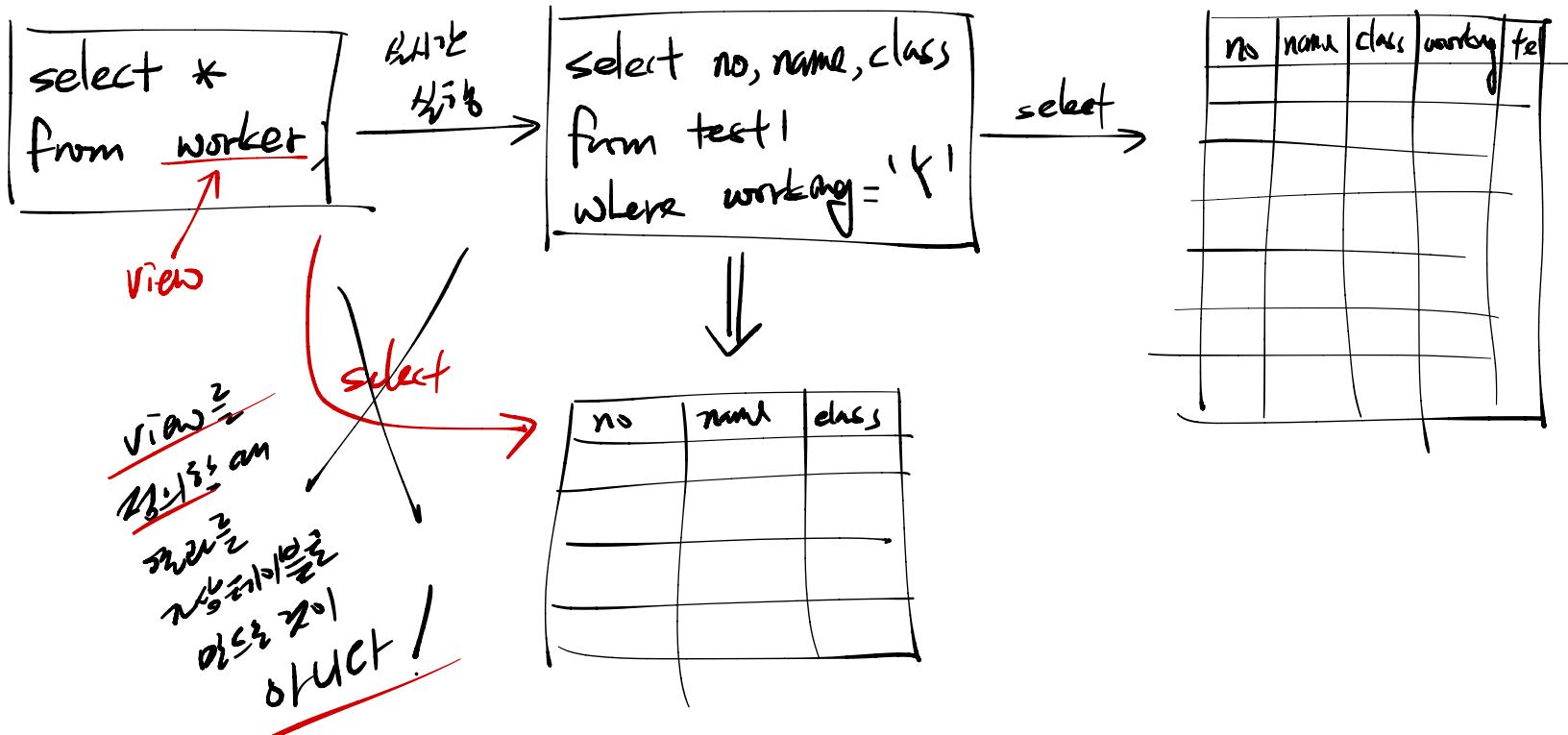
"projection" - 테이블에서 원하는
필드만 선택

select no, name, class
from test1;

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

"selection" - 테이블에서 원하는
행만 선택

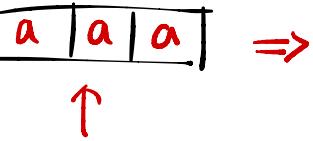
* view et table V_2^{ex}



* 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.
문제 2개

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-

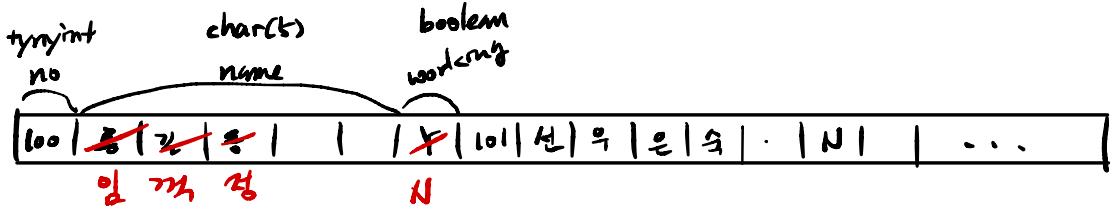
row = record = tuple

column = field = attribute

* 파일과 연결 / 출판

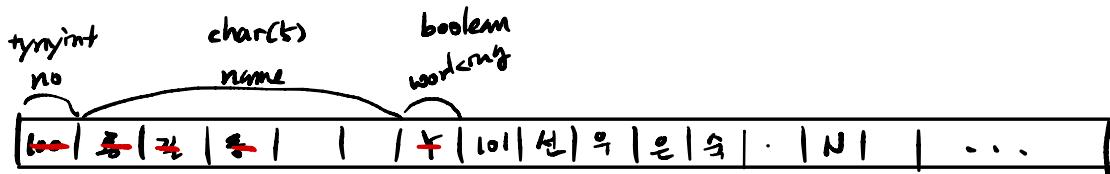
파일 데이터 1번정

↳ 기호값을 저장하는
문자열 형



파일 데이터 삭제

↳ 기호값을 저장하는
문자열을 초기화



↑
0 으로 초기화 시킨다

※ 데이터를 재가공하려면
위의 데이터를 앞으로 옮겨야 한다
↓
↳ 후시작이 가능하다

• 0으로 초기화 시킨다. 0은 이전에

그대로 그동 데이터로 저장된 부분을
0으로 덮어쓰는 것이다.

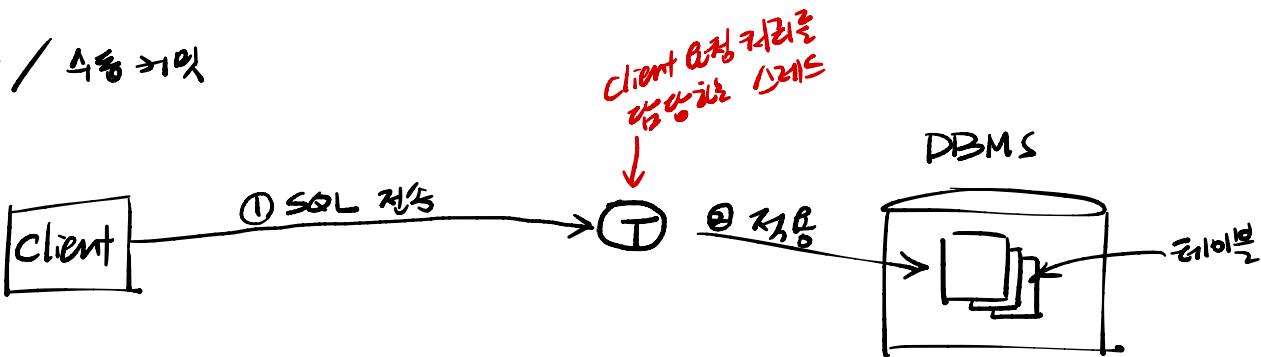
파일의 특정 부분을
저장할 수 있다

- ① 저장할 내용을 먼저 데이터를
내 파일에 쓰거나,
② 또는 위의 데이터를
삭제할 부분으로 옮겨야 한다.

* autocommit / 속도↑이익

① autocommit

(insert
update
delete)

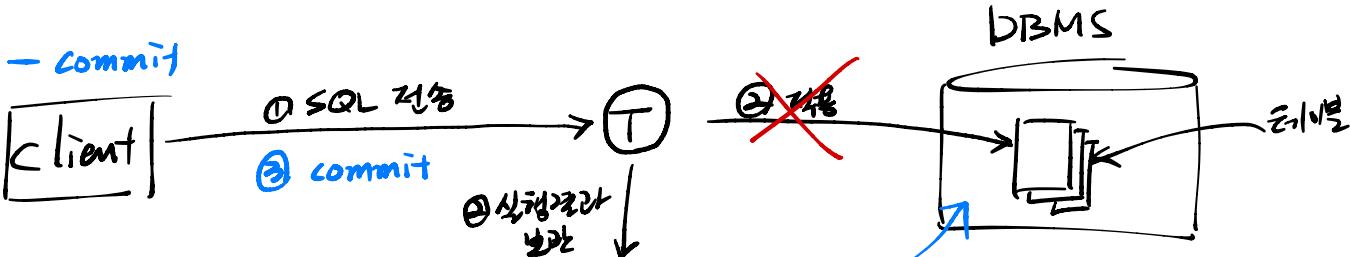


데이터를
동시에
변경할 때
자동으로
커밋
된다.

⇒ 어떤 상태로
되돌릴 수 있다.

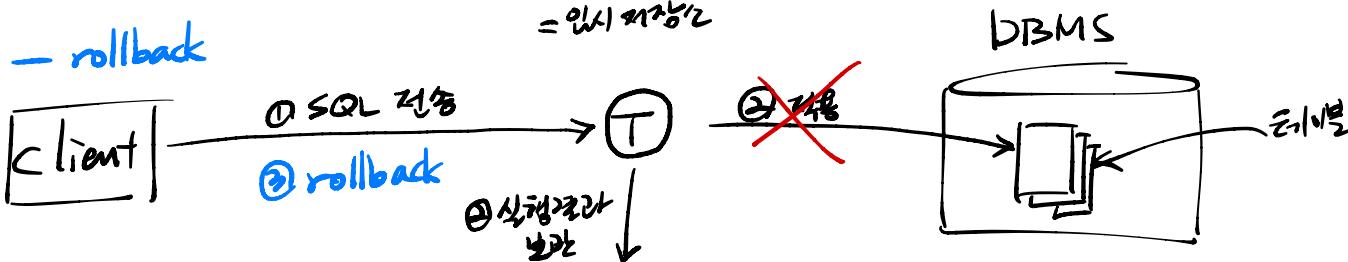
② 푸시 commit - commit

- insert
- update
- delete

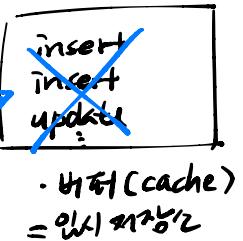


② 푸시 commit - rollback

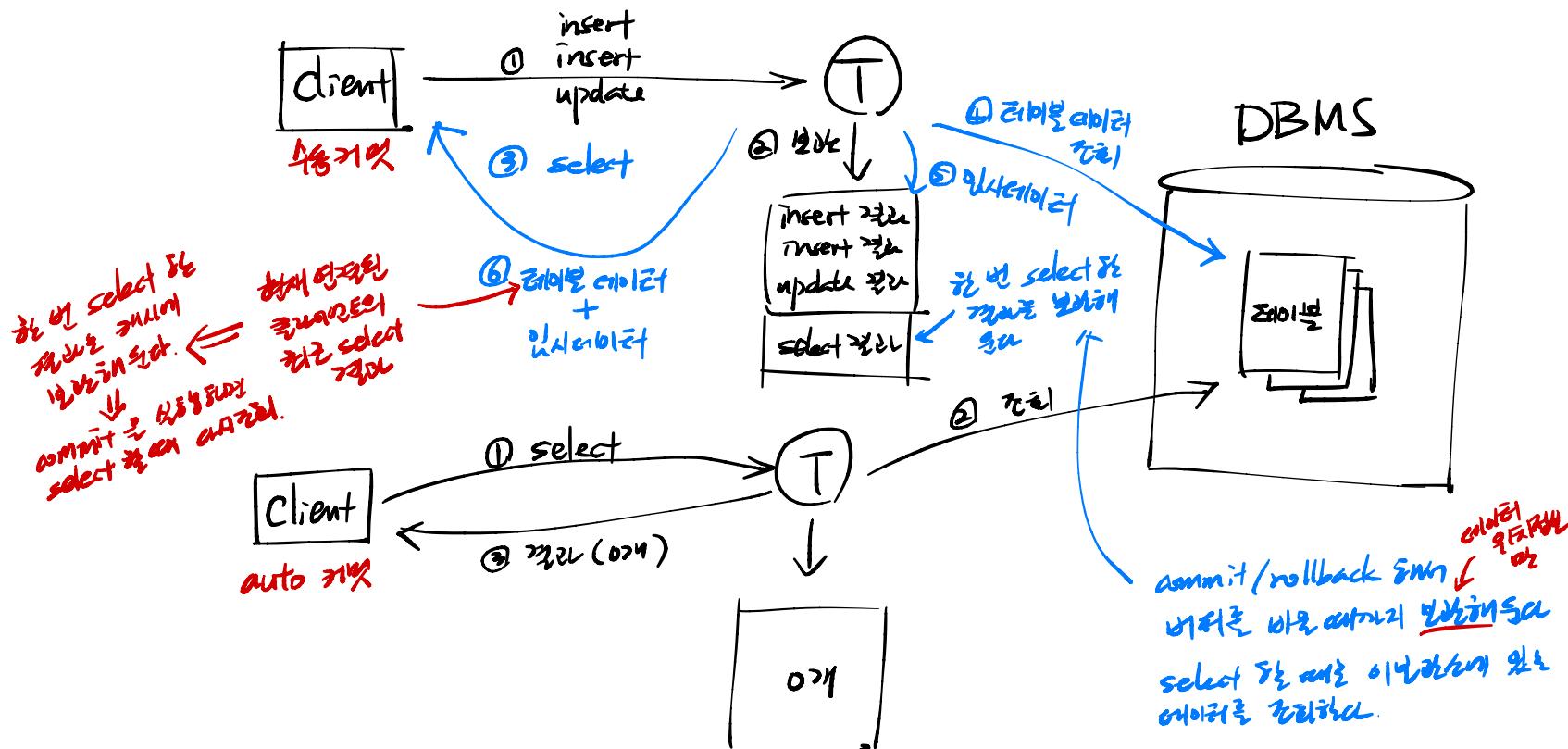
- insert
- update
- delete



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)

