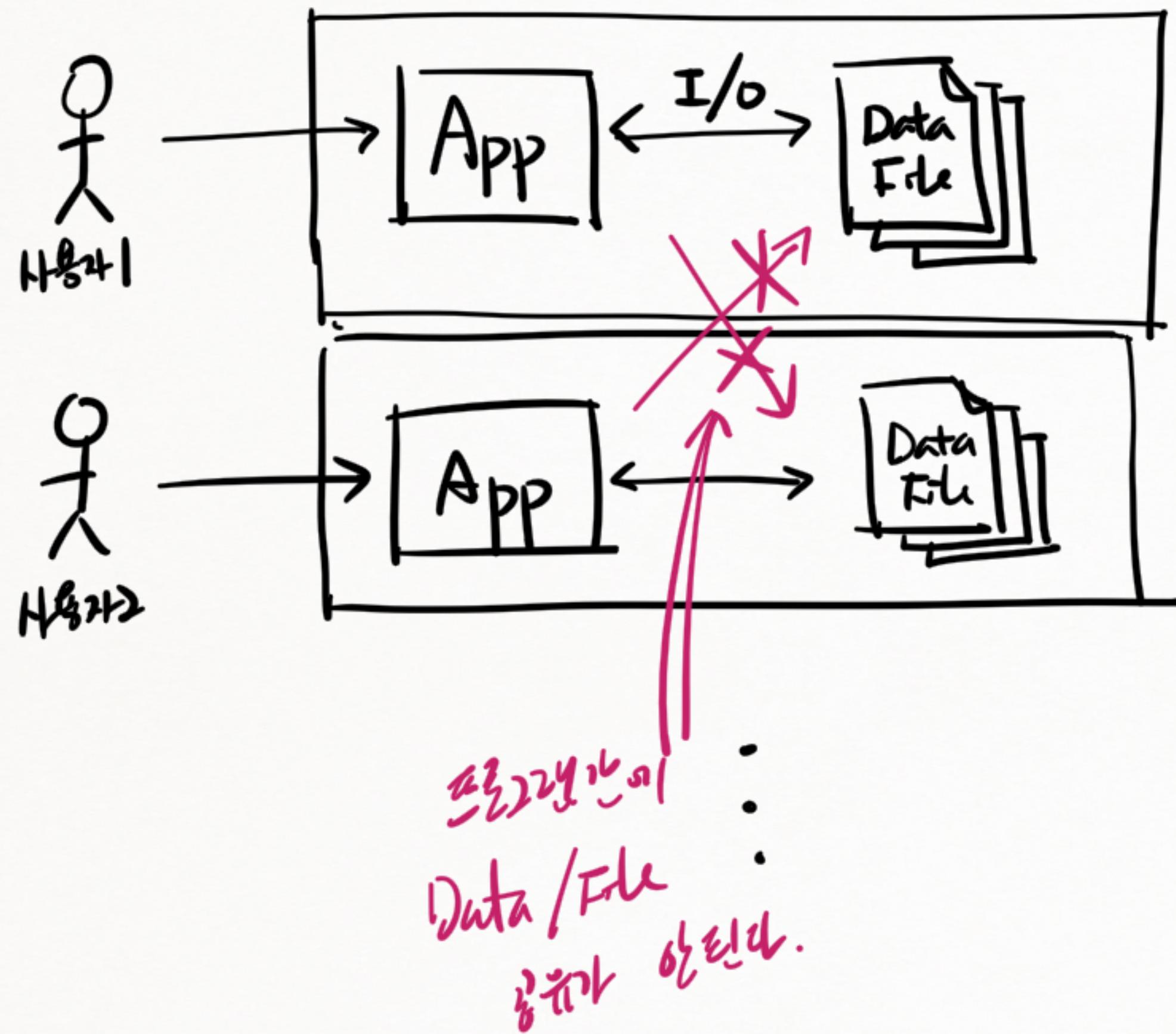


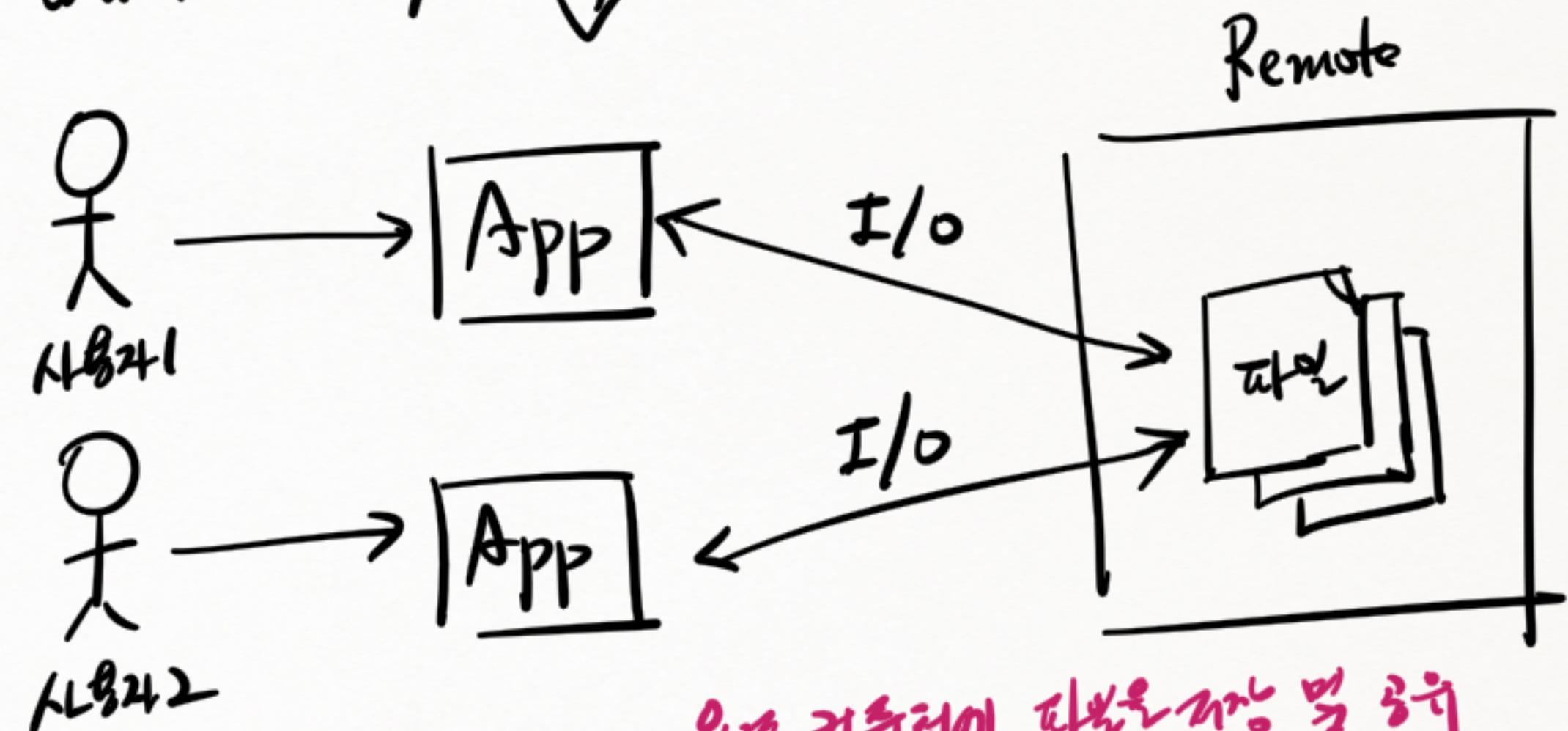
37. 바이오킹을 통한 데이터 공유

① 현황



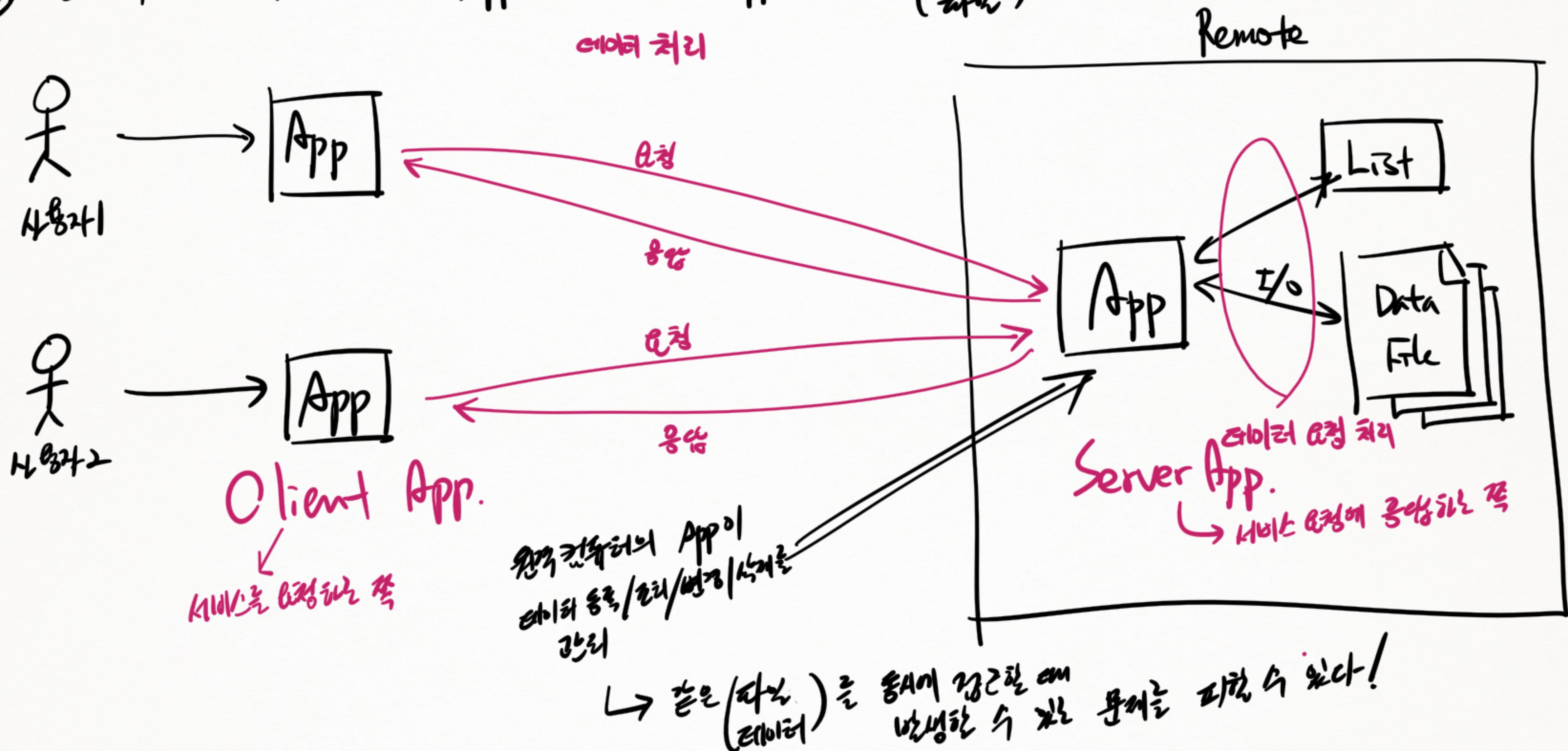
App을 실행하는 컴퓨터
데이터 파일이 로컬에 저장된다.
↓
App 간에 데이터를 공유할 수 없다!

② 데이터 파일은
별도의 컴퓨터에 분리/공유
가능할까?

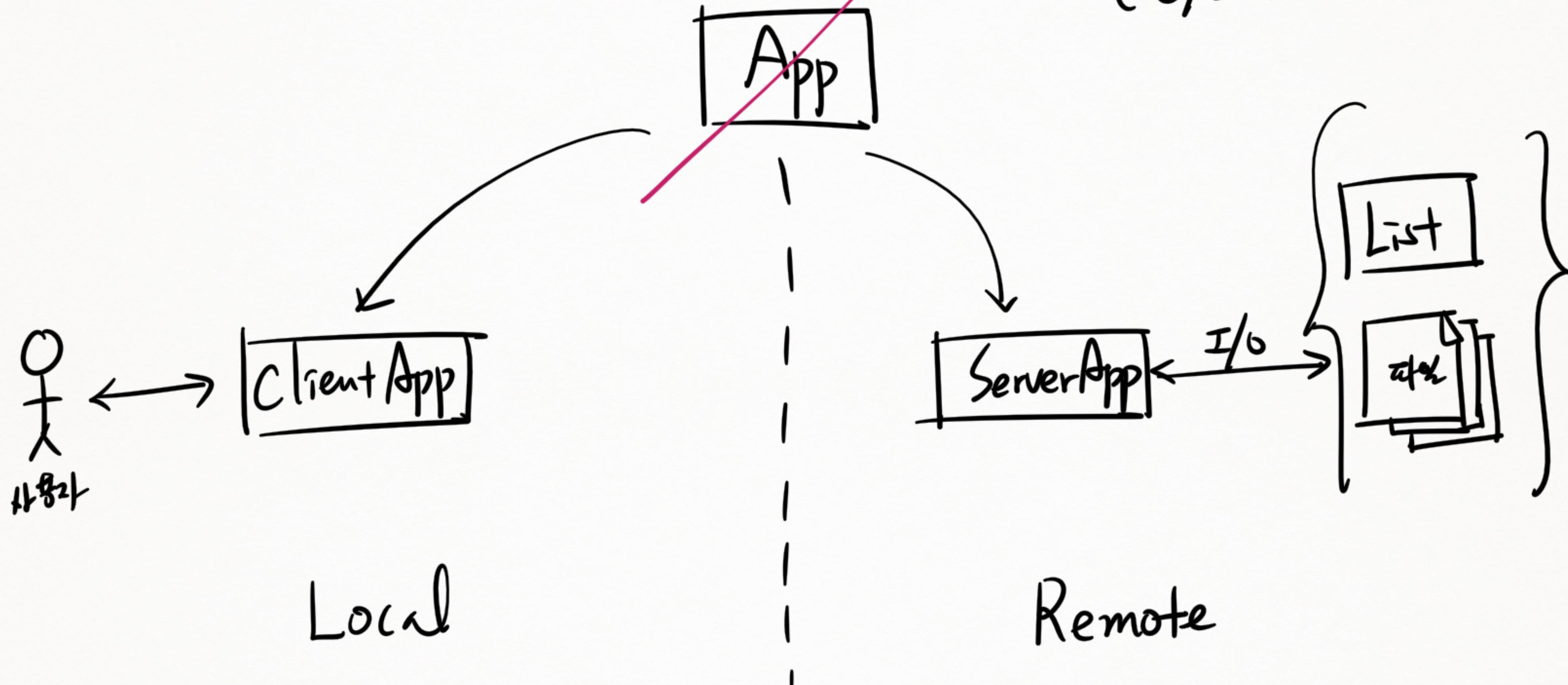


- 원격 컴퓨터에 파일을 저장 및 공유
- 동시에 여러 App이 같은 파일을 편집하다 보면
파일의 데이터가 깨질수 있다.

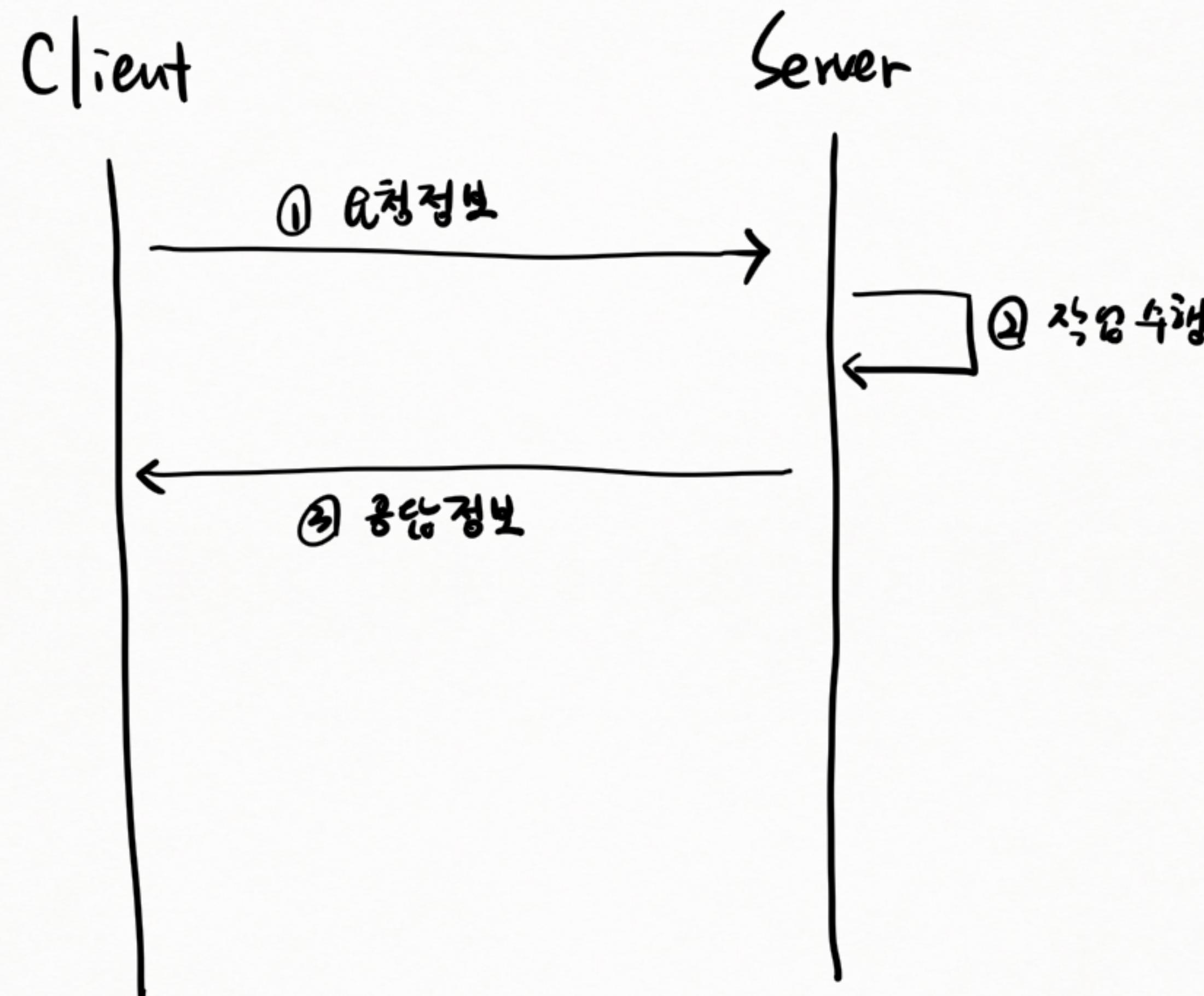
③ 데이터 관리 기능을 별도의 App. 으로 분리 - App이 직제 (데이터)
를 접근하는 것을 막는다.



* System Architecture : Client / Server Architecture
(C/S Architecture)



* client / server 통신 규칙(protocol)



* client / server 통신 규칙 (protocol)

① 요청 정보 (JSON 문자열)

```
{  
  "command": "레이타이머 / 명령",  
  "data": "JSON 문자열"  
}
```

↓ 예

```
{  
  "command": "board/insert",  
  "data": {"title": "...",  
           "content": "...",  
           ... :  
           }  
}
```

"레이타이머 / 명령"

기사로: board
회원: member
독서록: reading

서버의 DAO 메서드명.

JSON 문자열

command 값: 일반 문자열
data 값: JSON 문자열

- * client / server 통신 규칙 (protocol)

② 응답 정보 (JSON 문자열)

```
{  
    "status": "실행결과",  
    "result": "JSON 문자열"  
}
```

↓ 성공 예

```
{  
    "status": "success",  
    "result": "[{"no": 12, ... }]"  
}
```

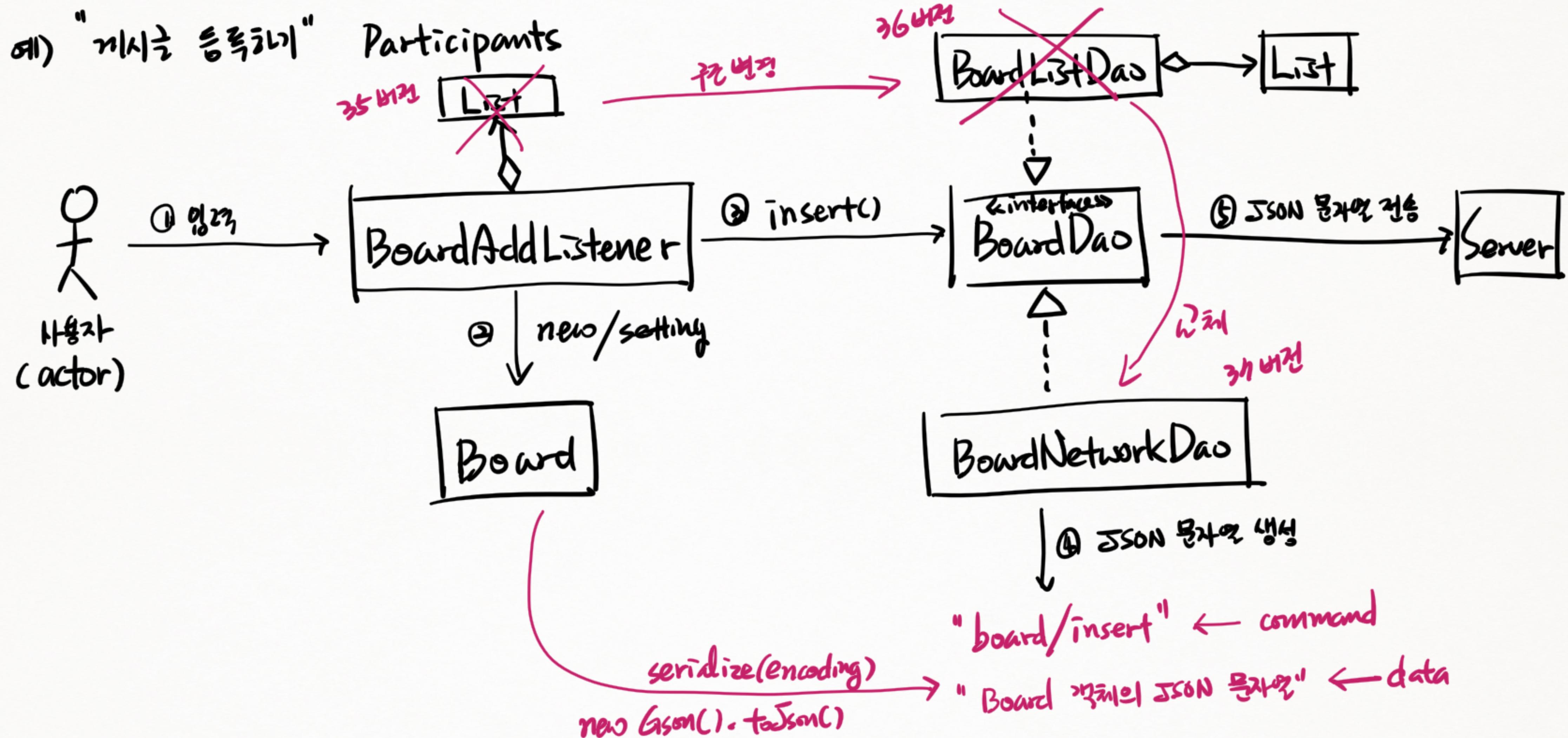
- * 실행 결과
"success": 성공
"failure": 실패

↓ 실패 예

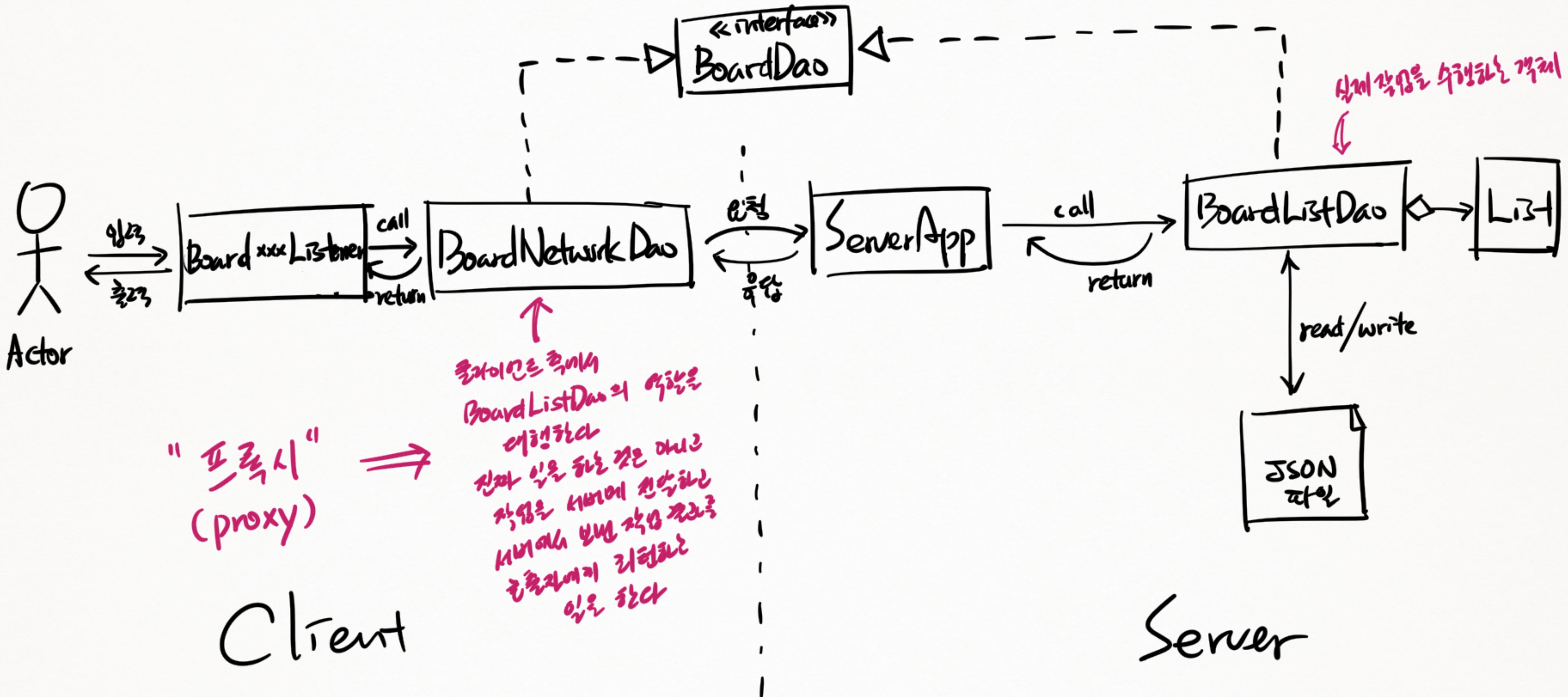
```
{  
    "status": "failure",  
    "result": "해당 번호의 데이터가 없습니다!"  
}
```

* 요청 정보 뷰티가 작업에 참여하는 객체들과 실행흐름

예) "게시글 등록하기" Participants

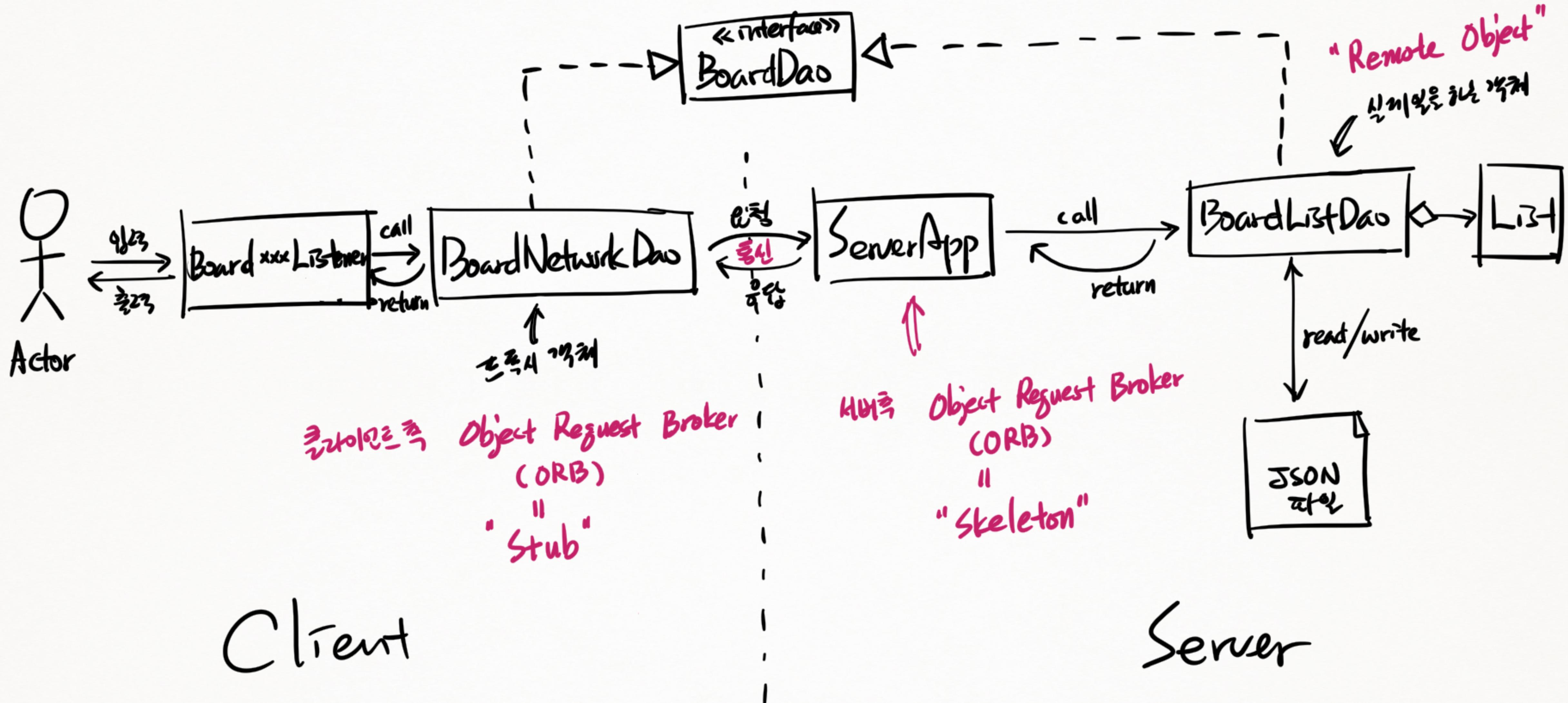


* DAO 와 Proxy 패턴 (GoF)



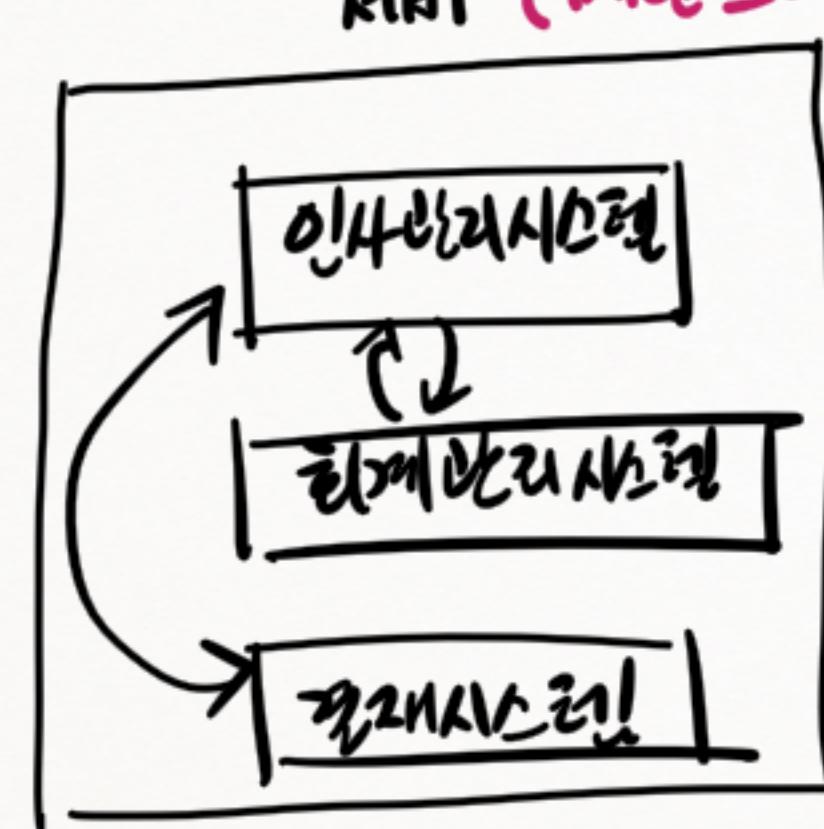
* DAO 와 Proxy 패턴 (GoF)

기억할자의 명칭



* 온라인 컴퓨팅

① 중앙 집중식 컴퓨팅



문제점

② 온라인 컴퓨팅

⇒ Rest API (RESTful)

→ 크기↓ 가격↓ 용량↓ (down sizing)

(원격스테이션) 서버

↳ Unix

↳ Solaris, HP-UX, IBM AIX, ...

(원격스테이션) 터미널

↳ Unix

(원격스테이션) 터미널

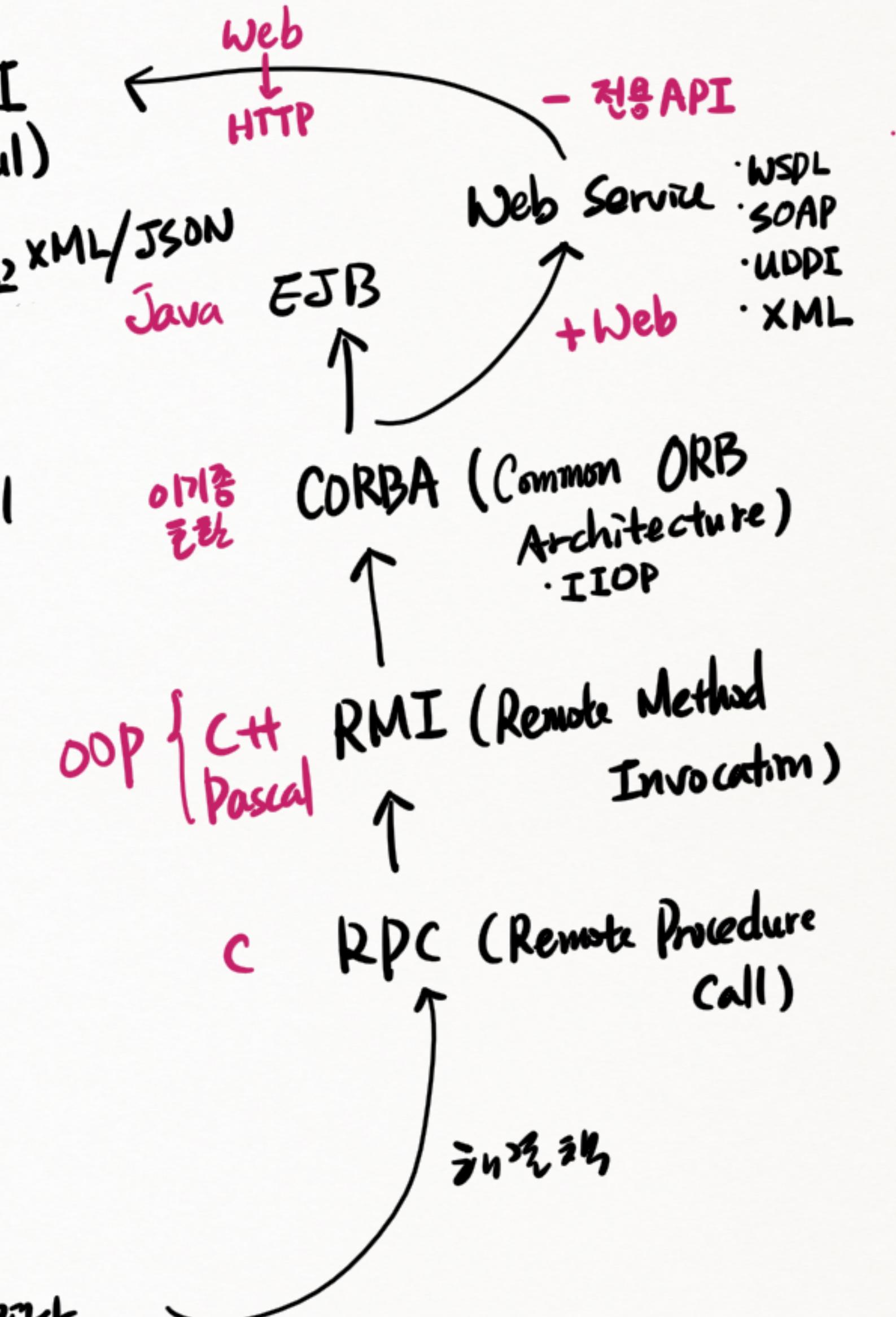
↳ Unix

인사관리 시스템

회계관리 시스템

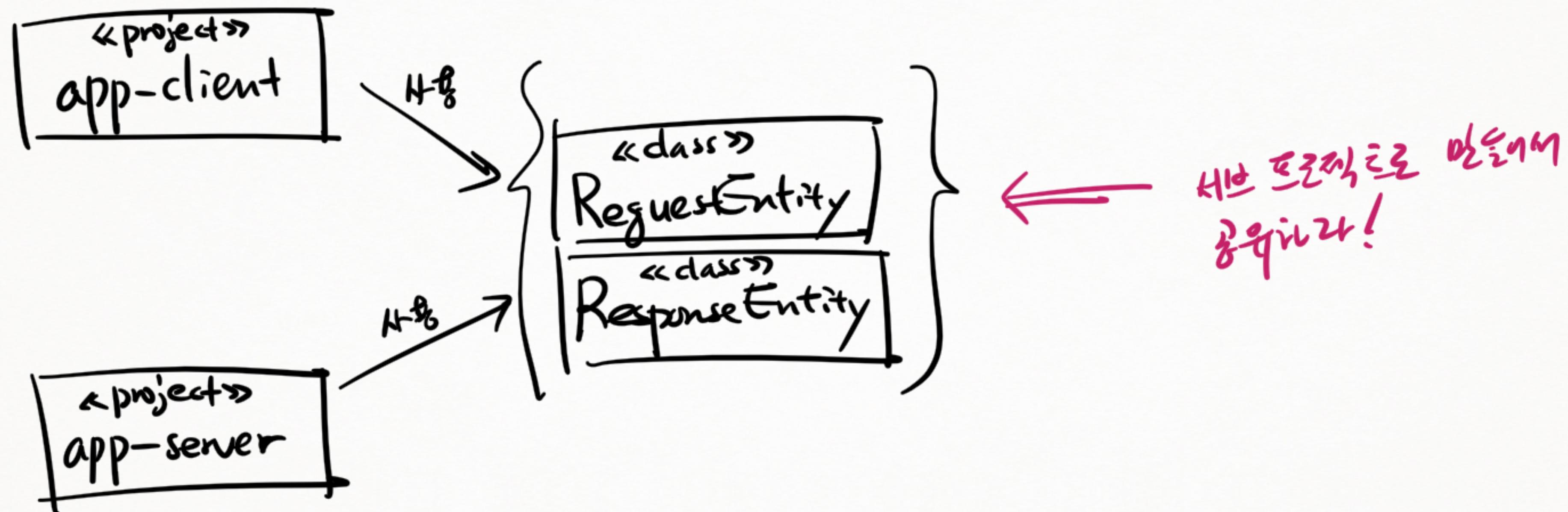
회계 시스템

call
call

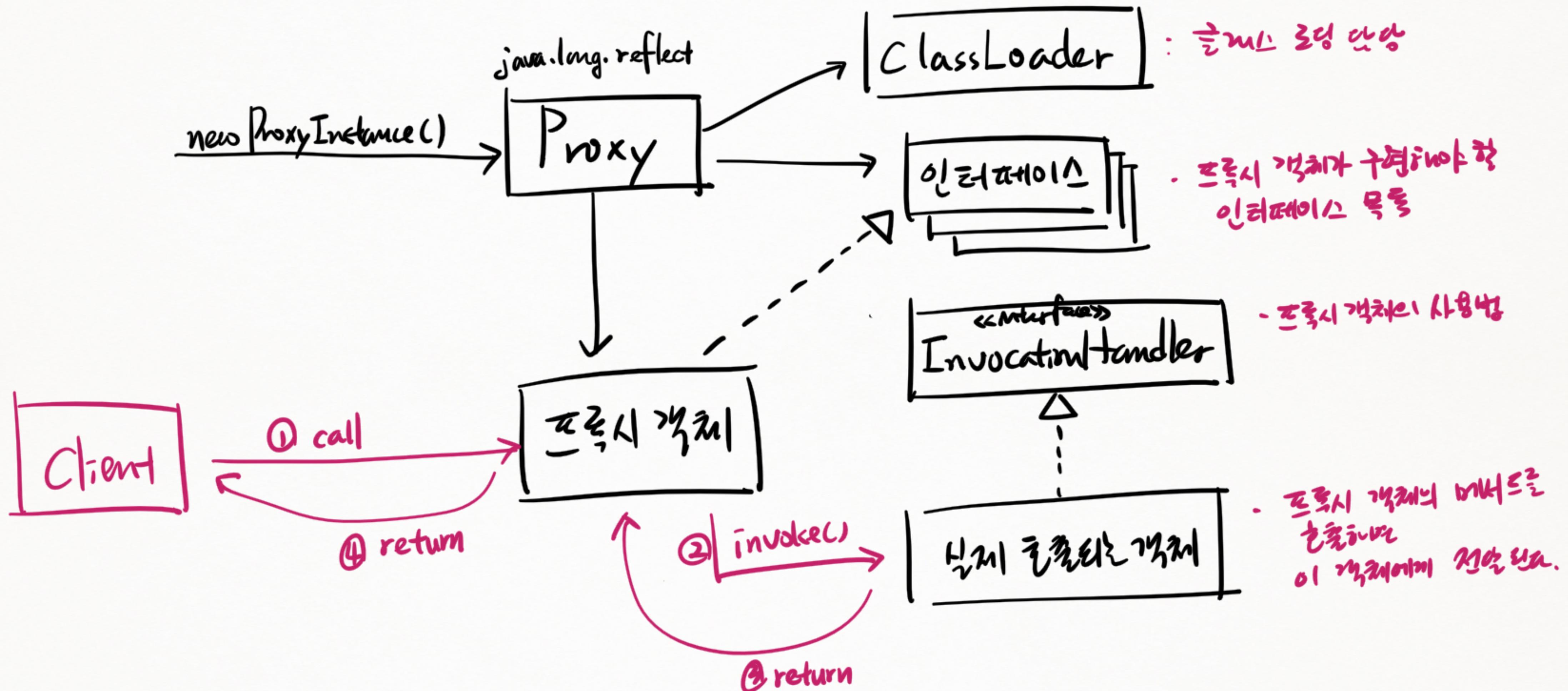


문제점

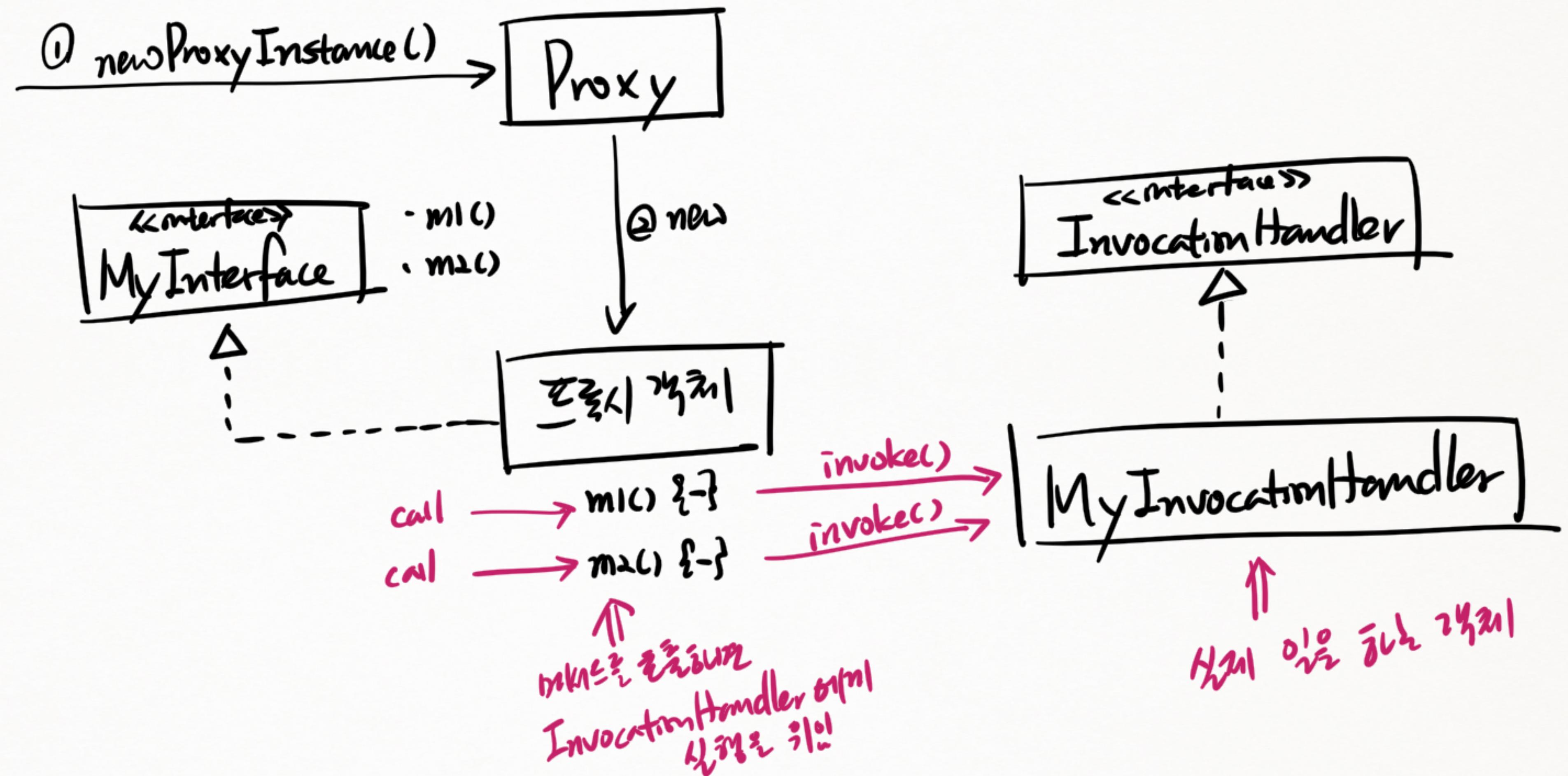
* Project 간에 헤더 공유하기



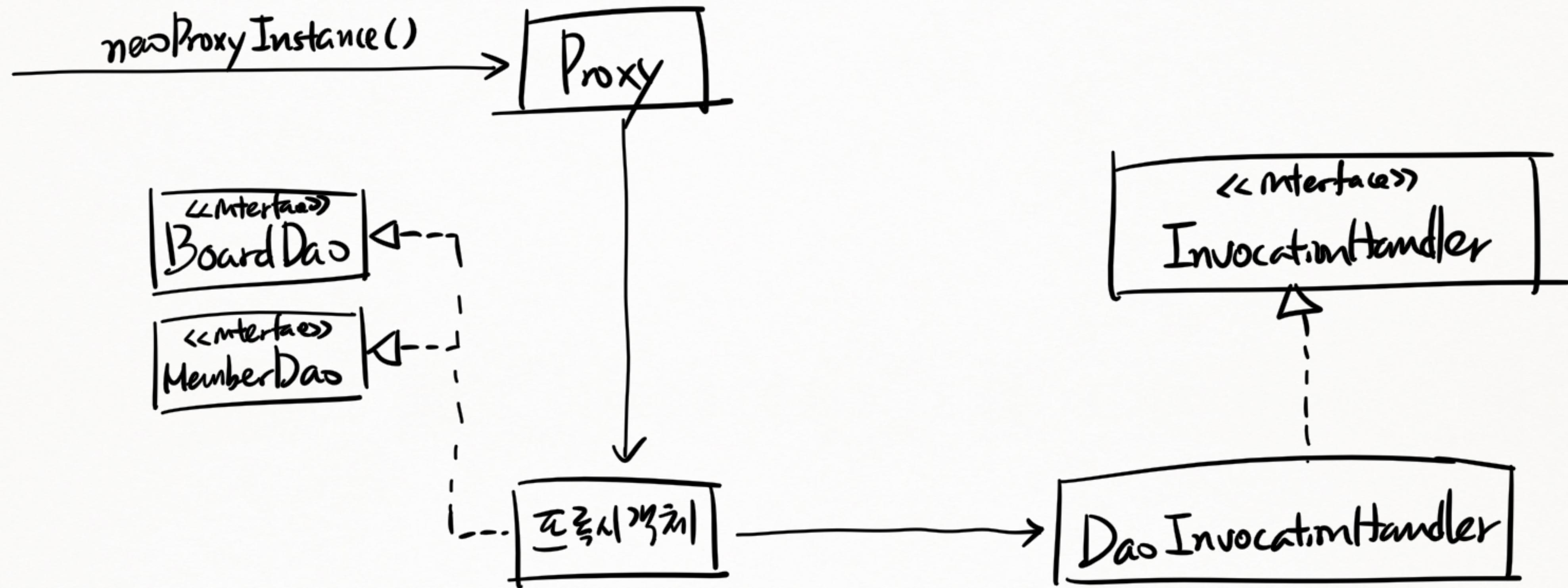
3.8. 프록시 객체 자동 생성



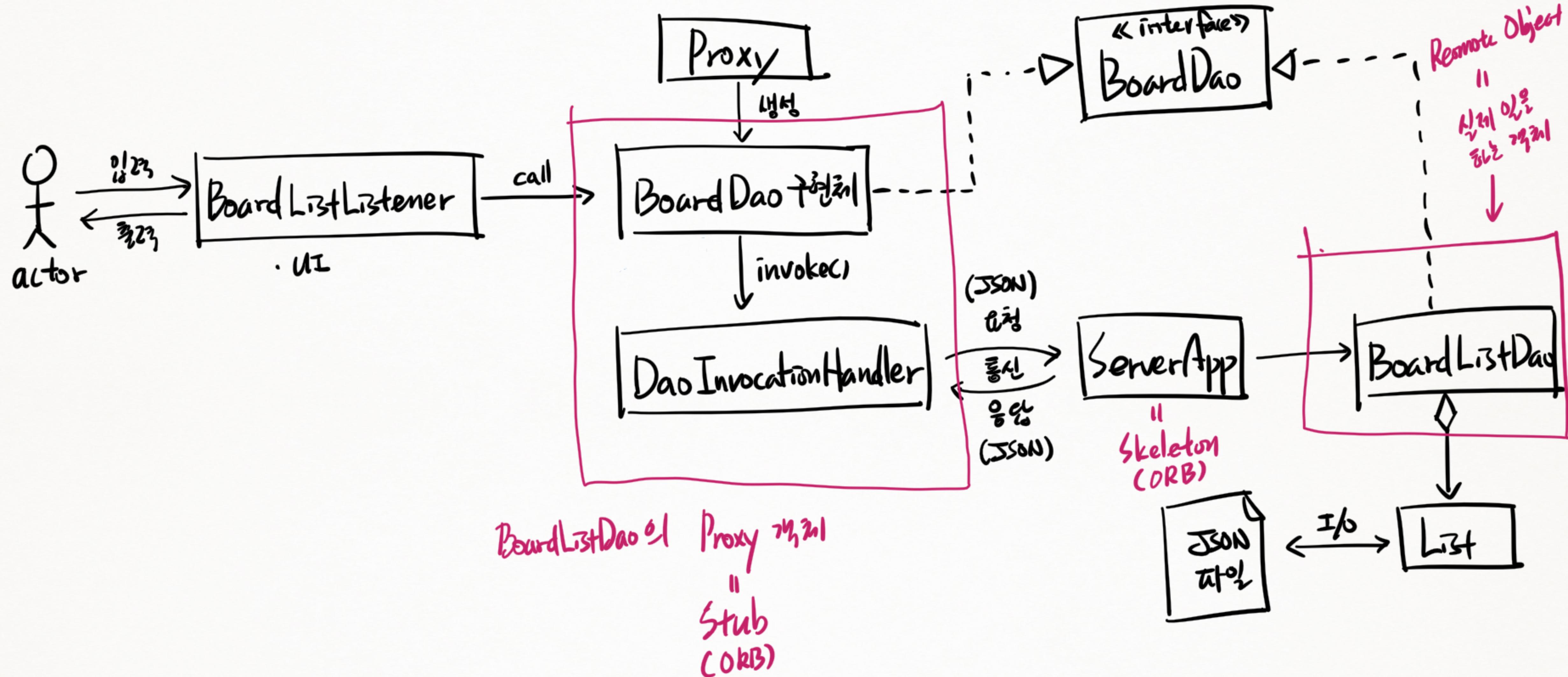
38. 프록시 객체 사용 예제 - ②



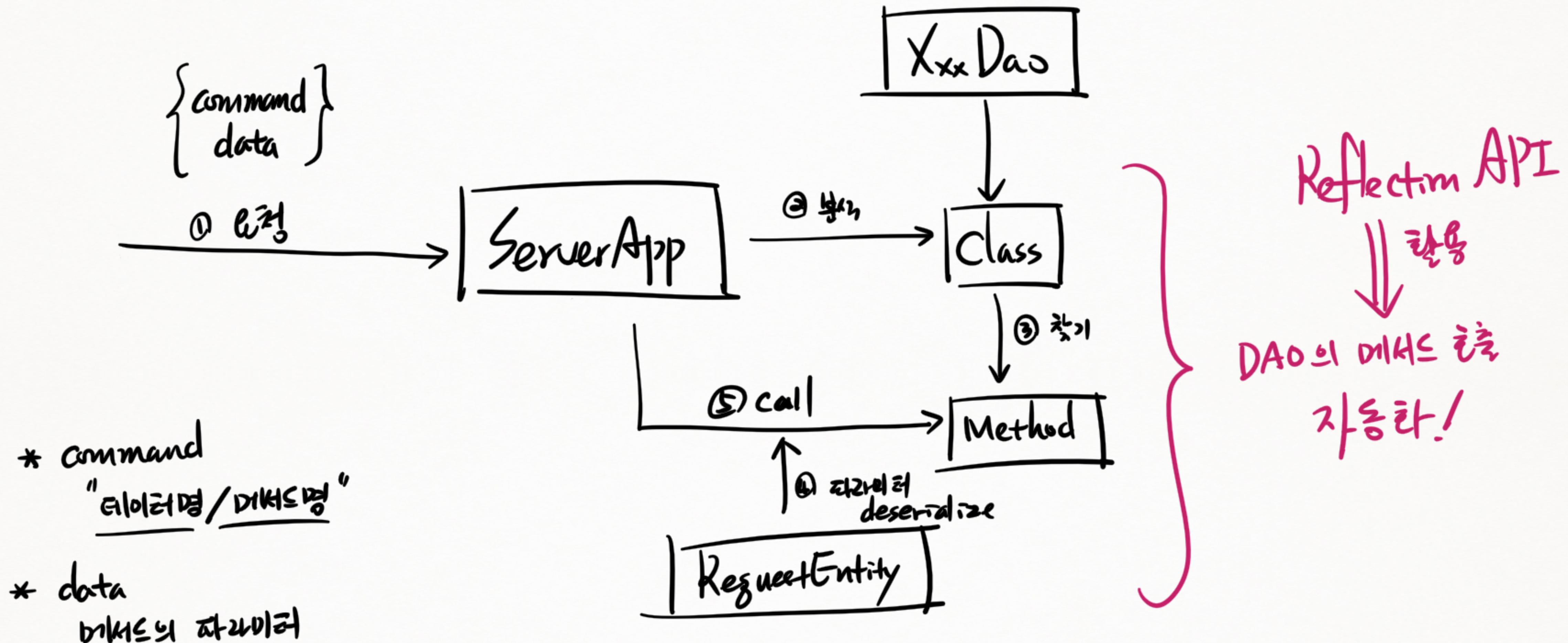
38. 프록시 패턴의 사용 예제 - Participants



38. 프록시 개념의 적용 예제 - Participants II



39. Reflection API를 사용하여 DAO 객체의 메서드 호출을 자동화하기



40. 예외 처리 예제

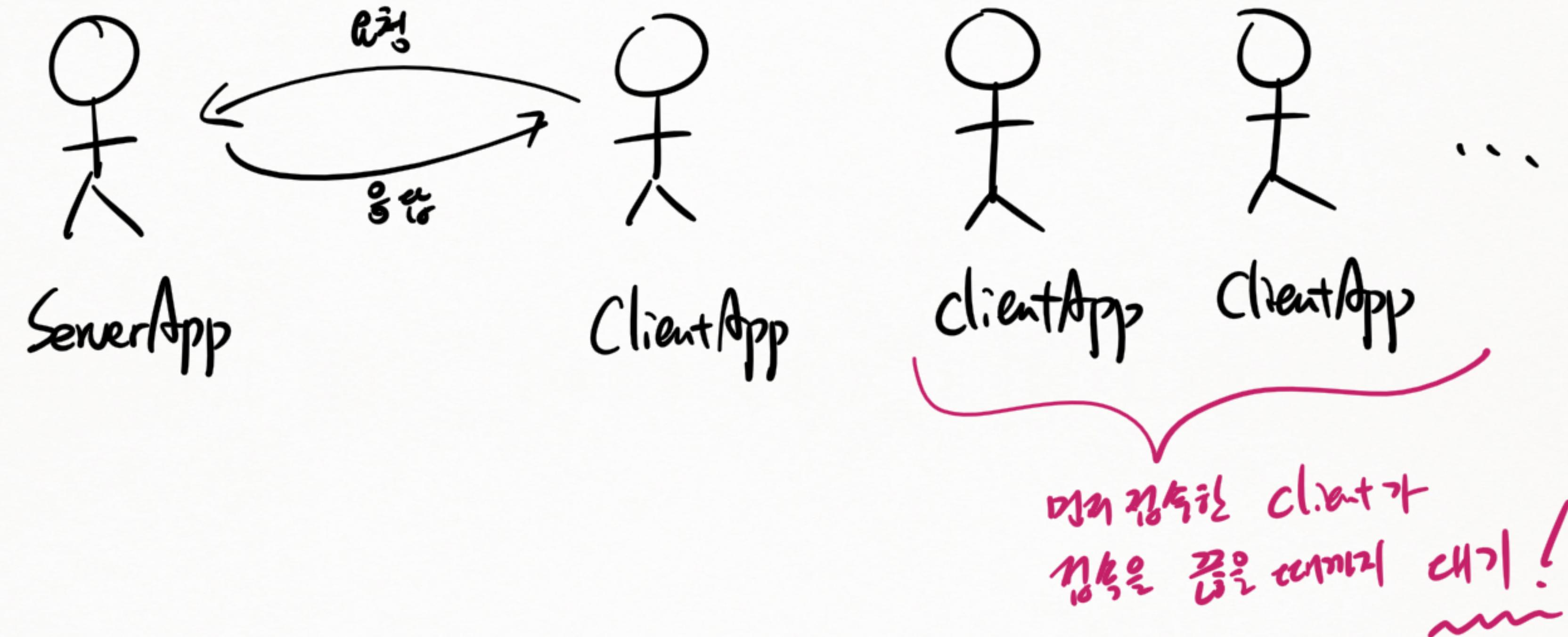
① client 편

```
try {  
    ==  
    menu.execute();  
    ==  
} catch (Exception e) {  
    ==  
}
```

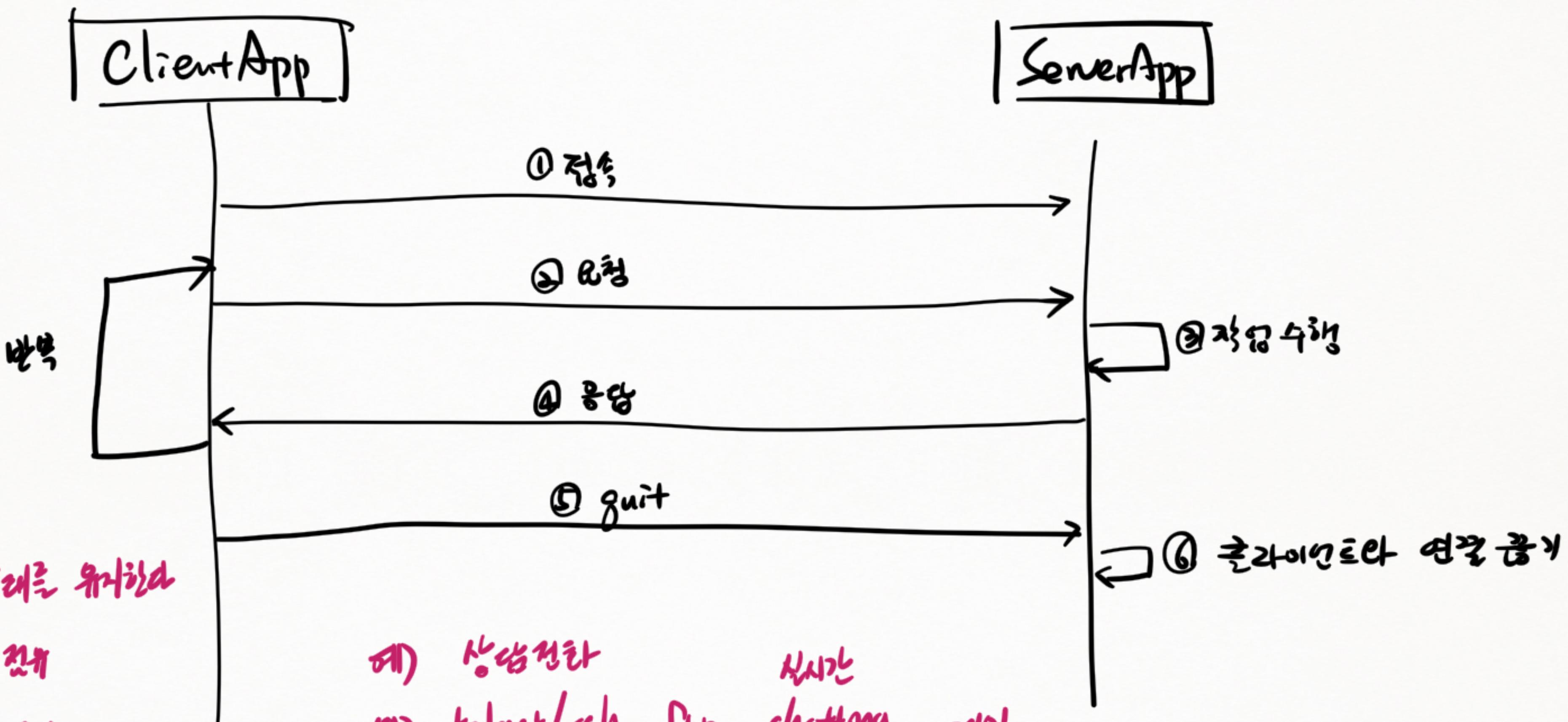
② server 편

```
try {  
    ==  
    dao.invoke(...);  
    ==  
} catch (Exception e) {  
    ==  
}
```

41. 예전 카카오로 페치를 스터디북으로 만들기 - Stateful 방식



41. 여러 주체들로 퍼진을 관리하고 처리하기 - Stateful 방식

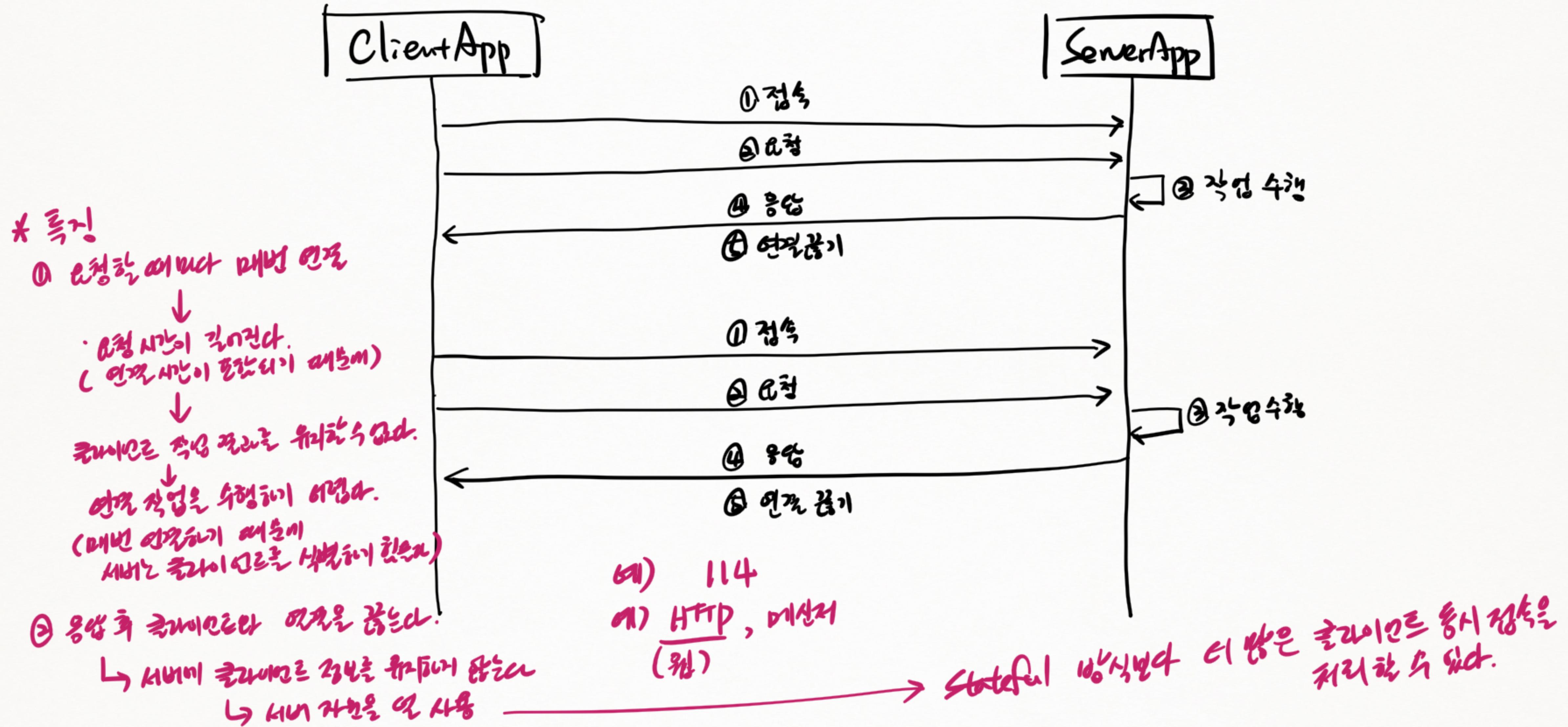


* 특징
① 접속과 연결된 상태를 유지한다

- 새끼 자리를 찾기
 - 동시에 많은 클리어언트 접속을 유지할 수 있다

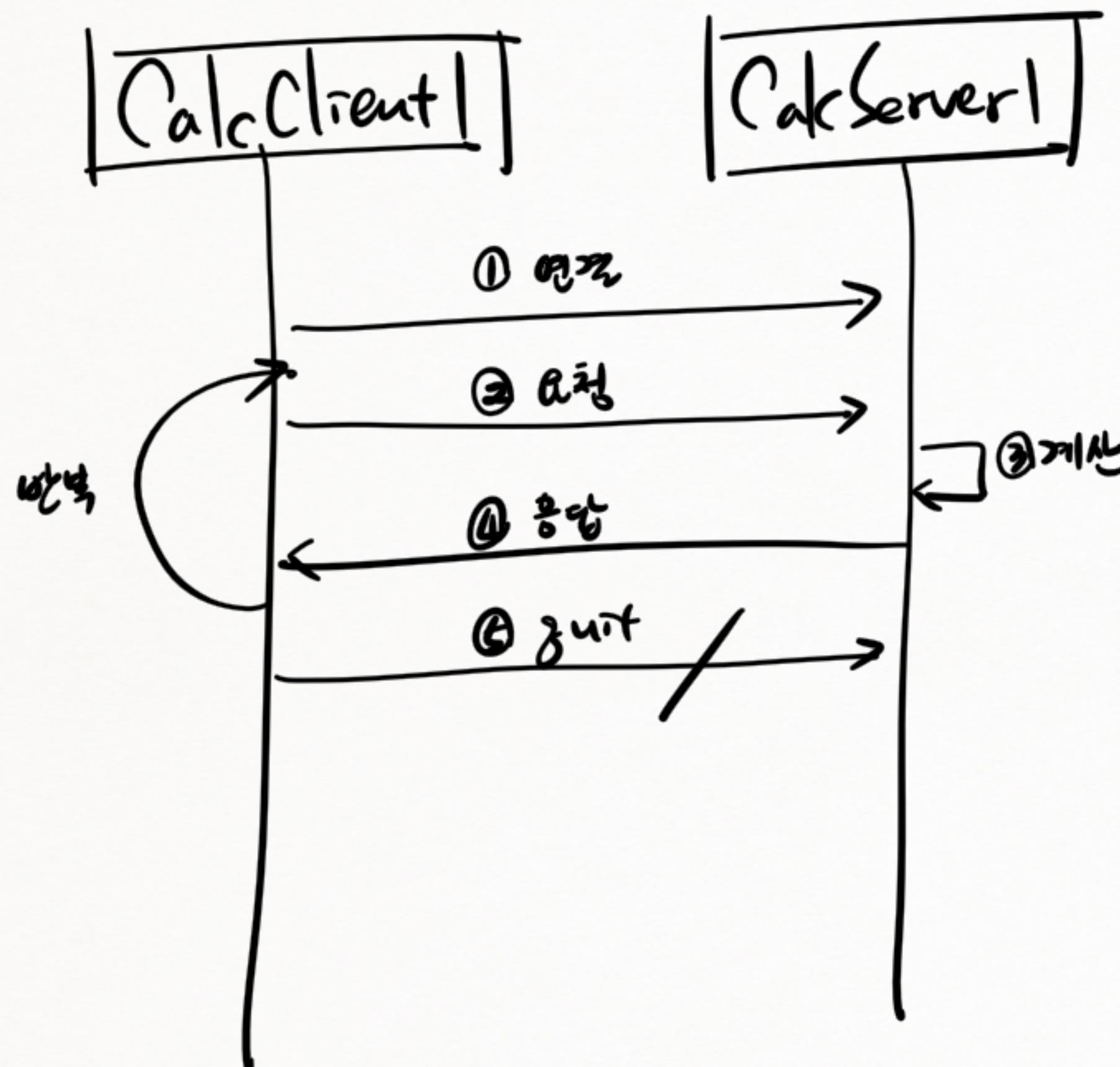
② 콜라이언트 정보 유지 → 작업 결과 유지 → 연간 작업을 처리하기 수월.

42. 예제 클라이언트 페처를 Stateless 방식

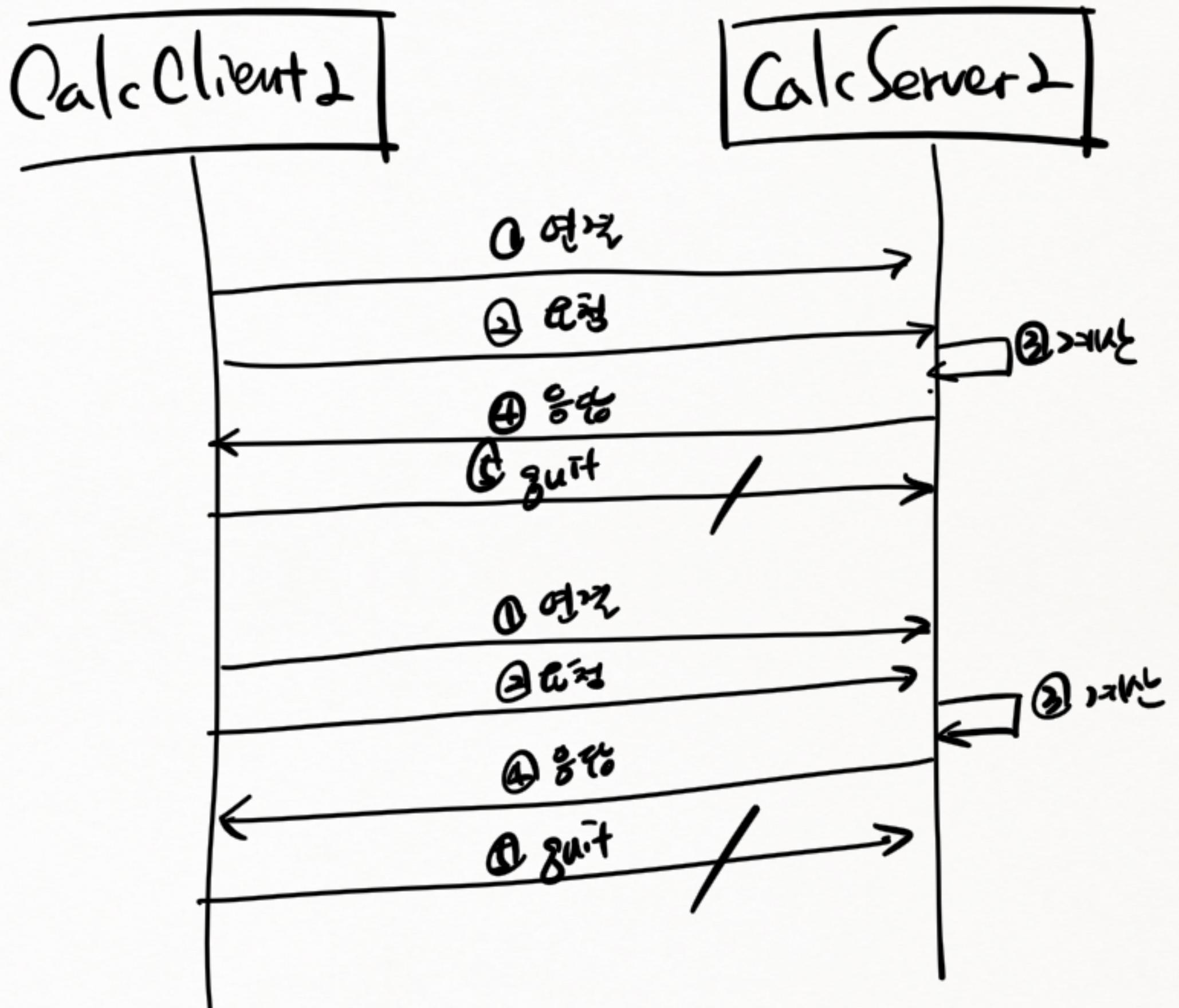


* Stateful vs Stateless

① Stateful

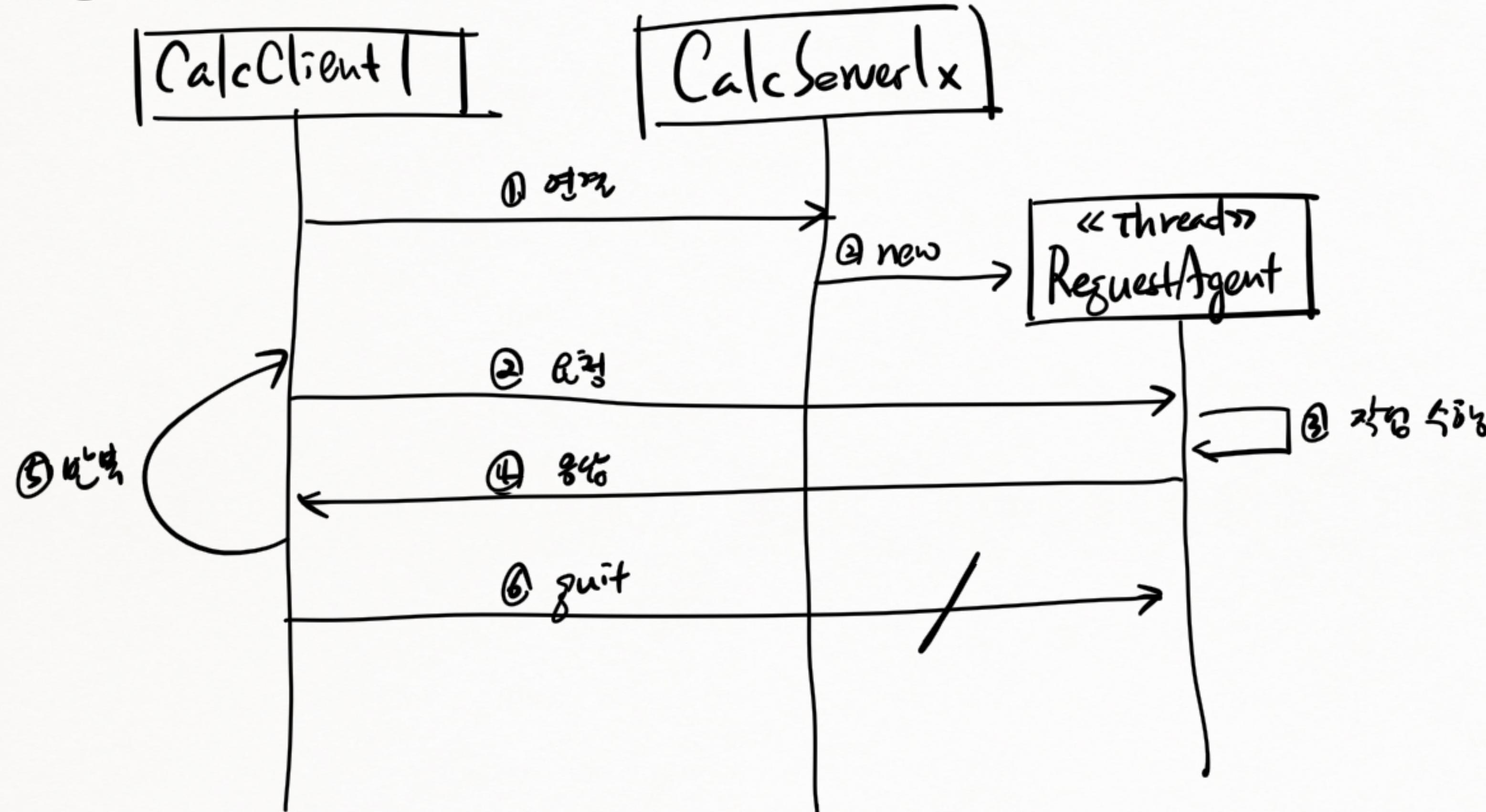


② Stateless



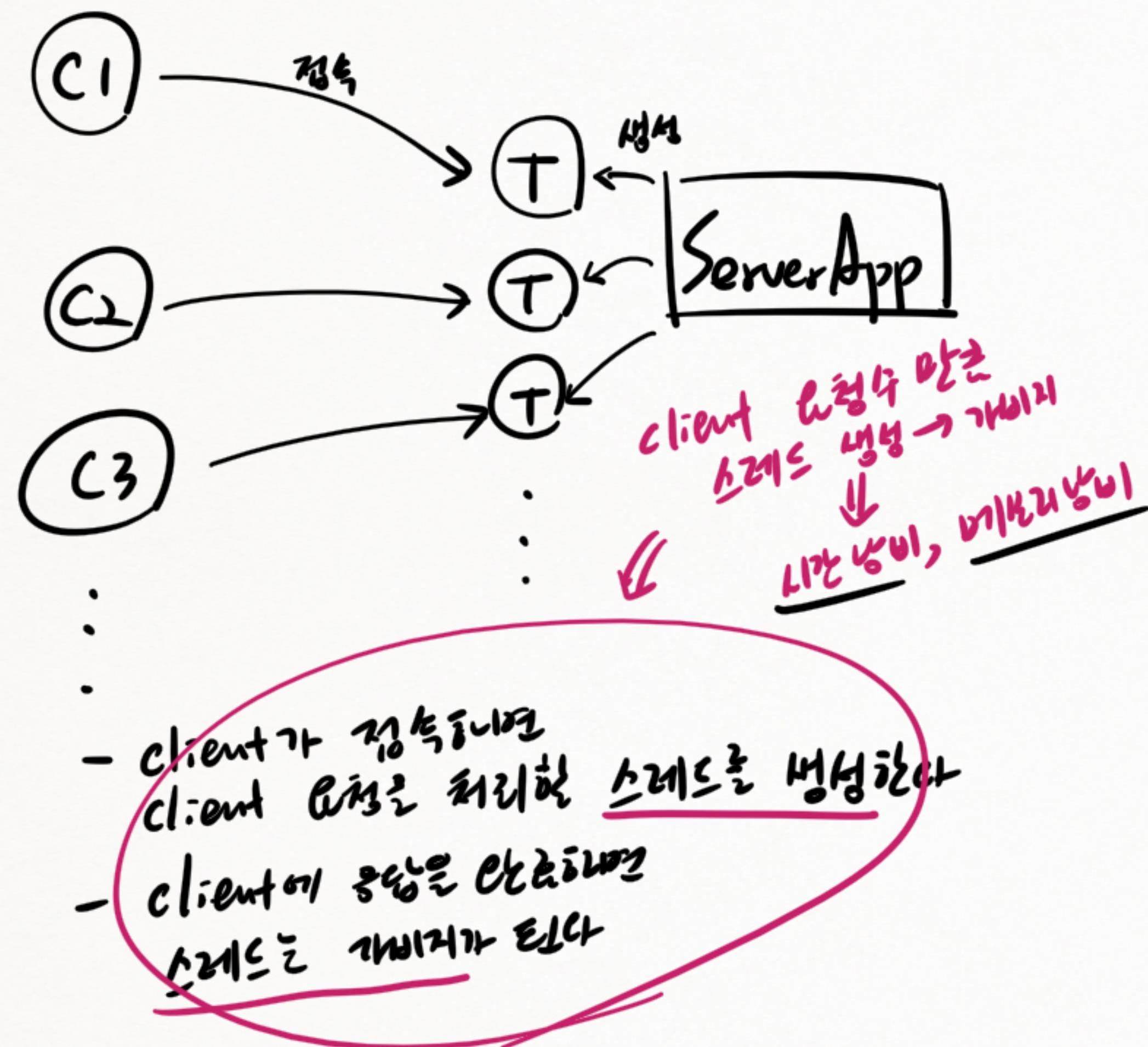
* Stateful vs Stateless

③ Stateful + Thread

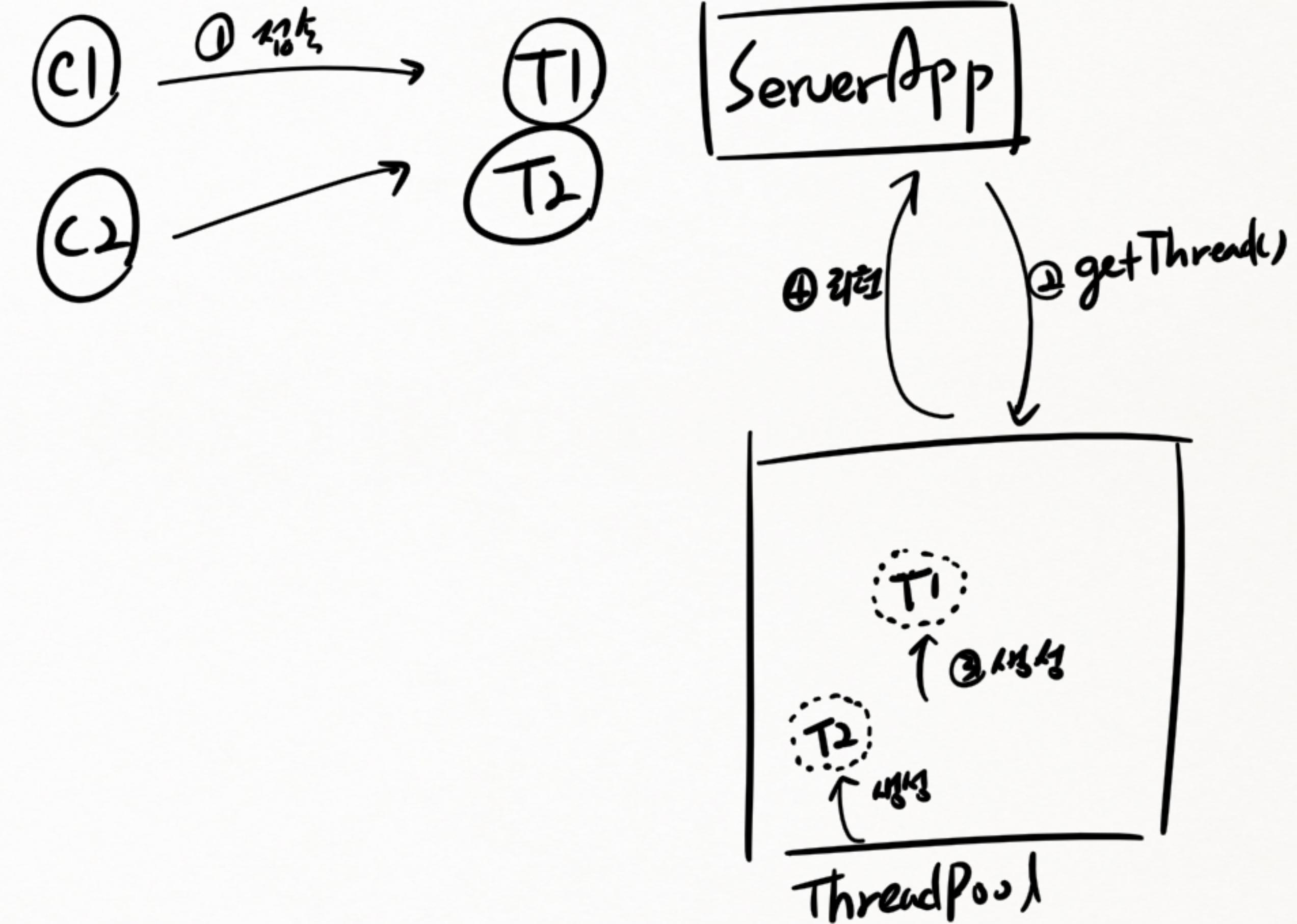


44. 스레드 재사용하기 : GOF의 Flyweight 패턴

① 스레드풀 적용 전

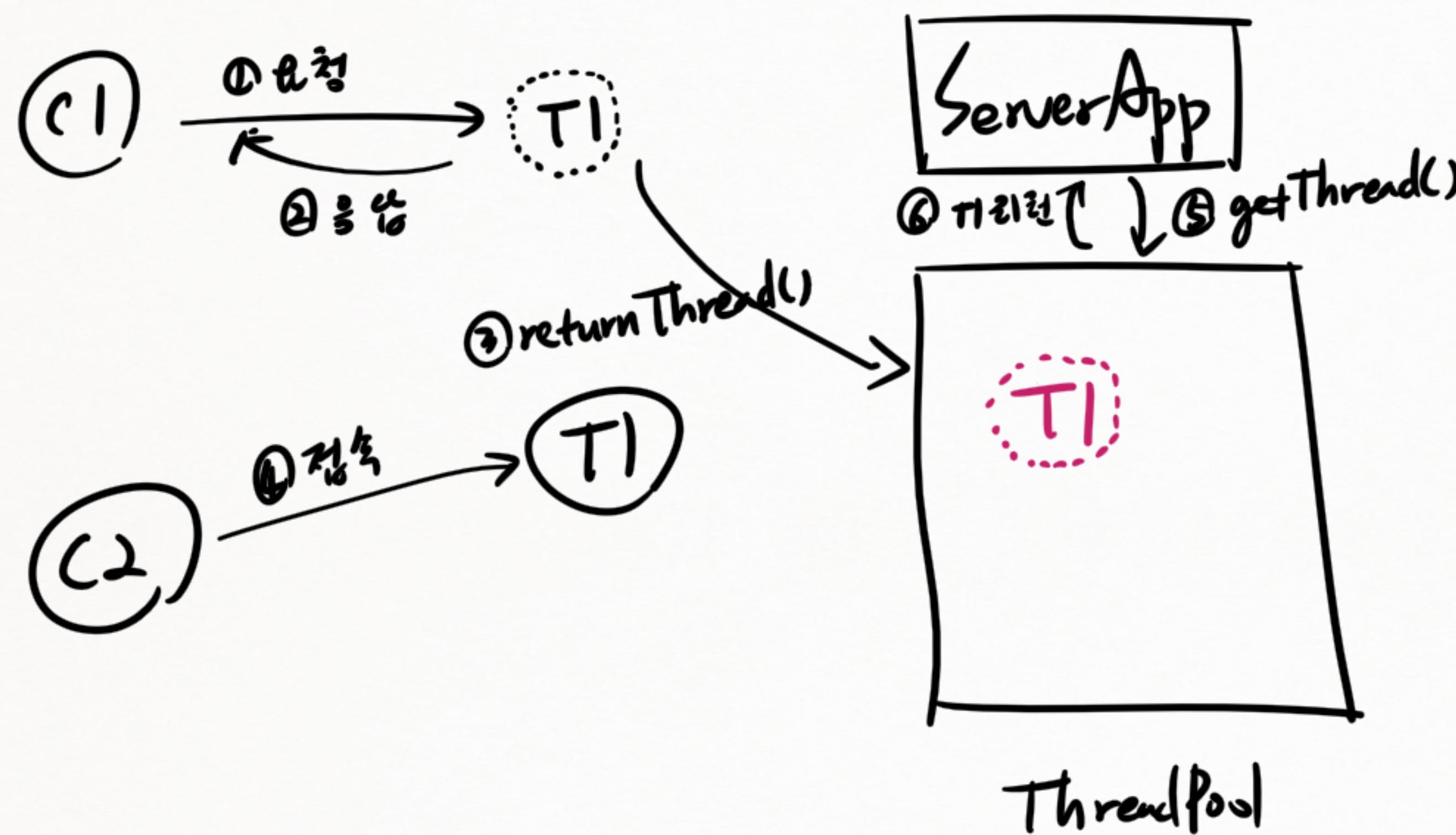


② 스레드풀 적용 후



44. 스레드 재사용하기 : GOF의 Flyweight 패턴

③ 스레드 재사용



{
· 개체 별 생성 시간이 오래 걸리는 경우
· 동일한 개체를 반복적으로 사용하는 경우}



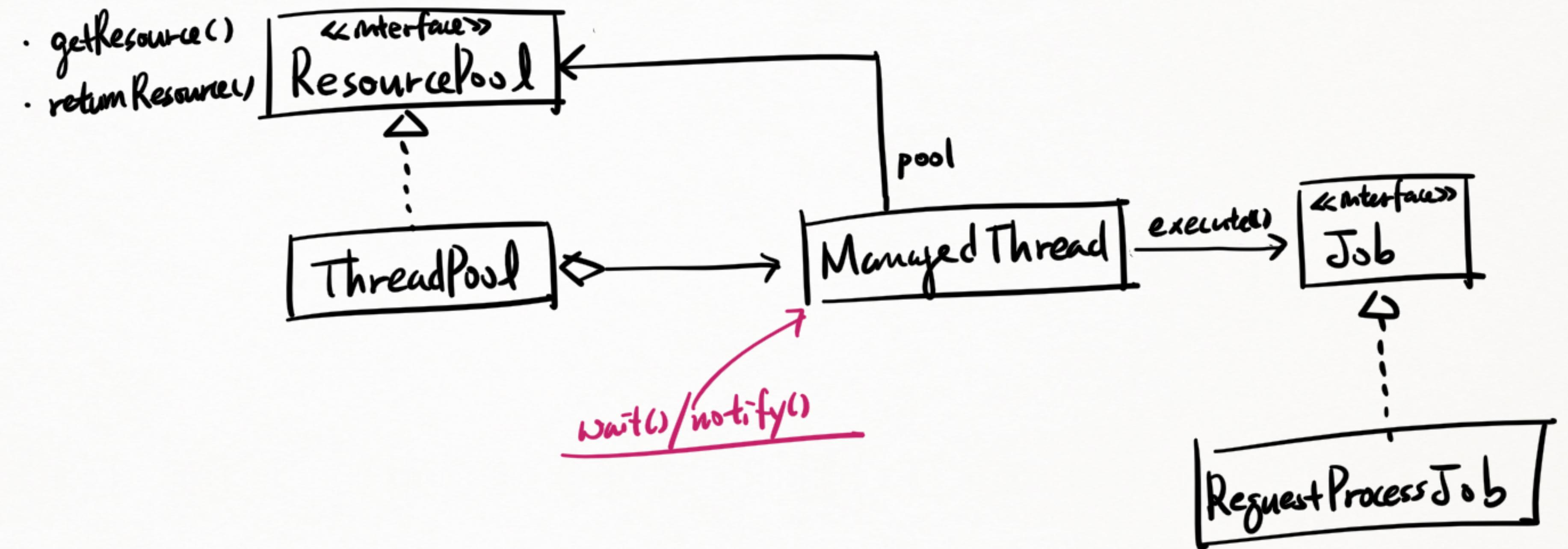
- 개체 사용후 가비지로 버리지 않고
 재활용성이 보완.
- 개체가 필요할 때,
 재활용이 필요한 개체를 미리 만들어
 "개체 재사용!"



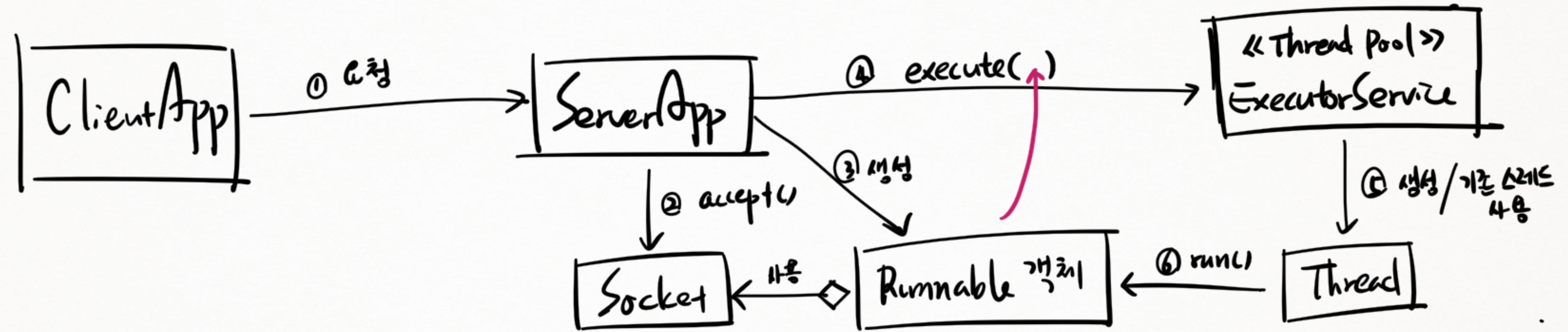
"Flyweight 패턴."

"Pooling 기법"

* Thread Pool Class Diagram

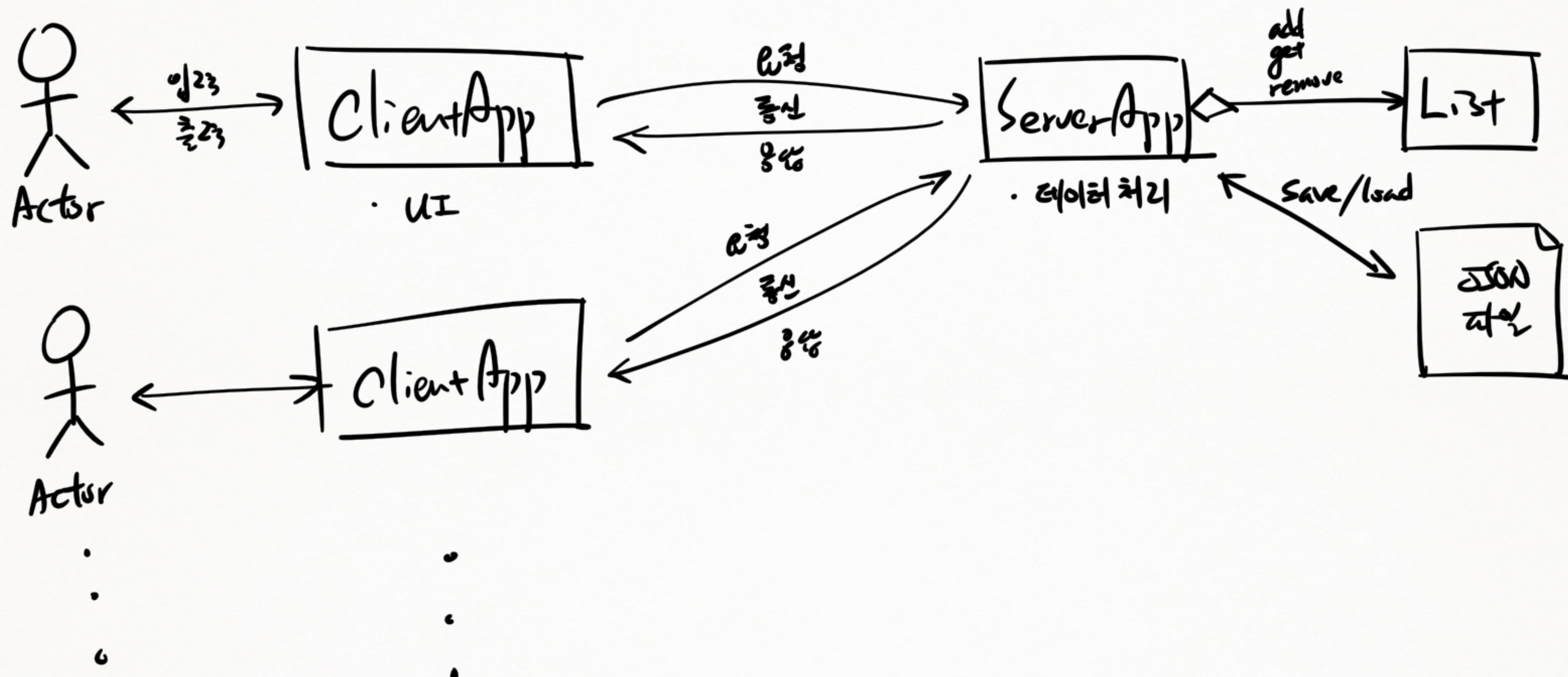


45.励志 스레드풀 구현하기 - Executors/ExecutorService 활용



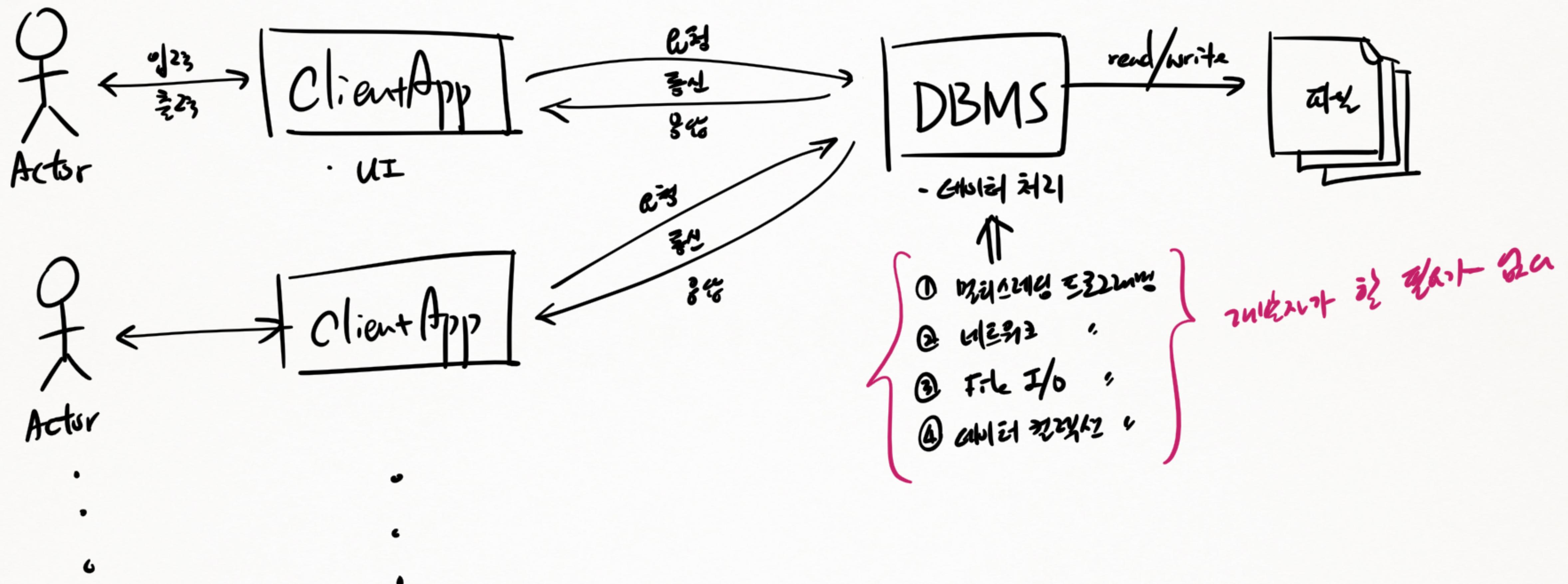
46. DBMS 도입하기

① 기본 구조

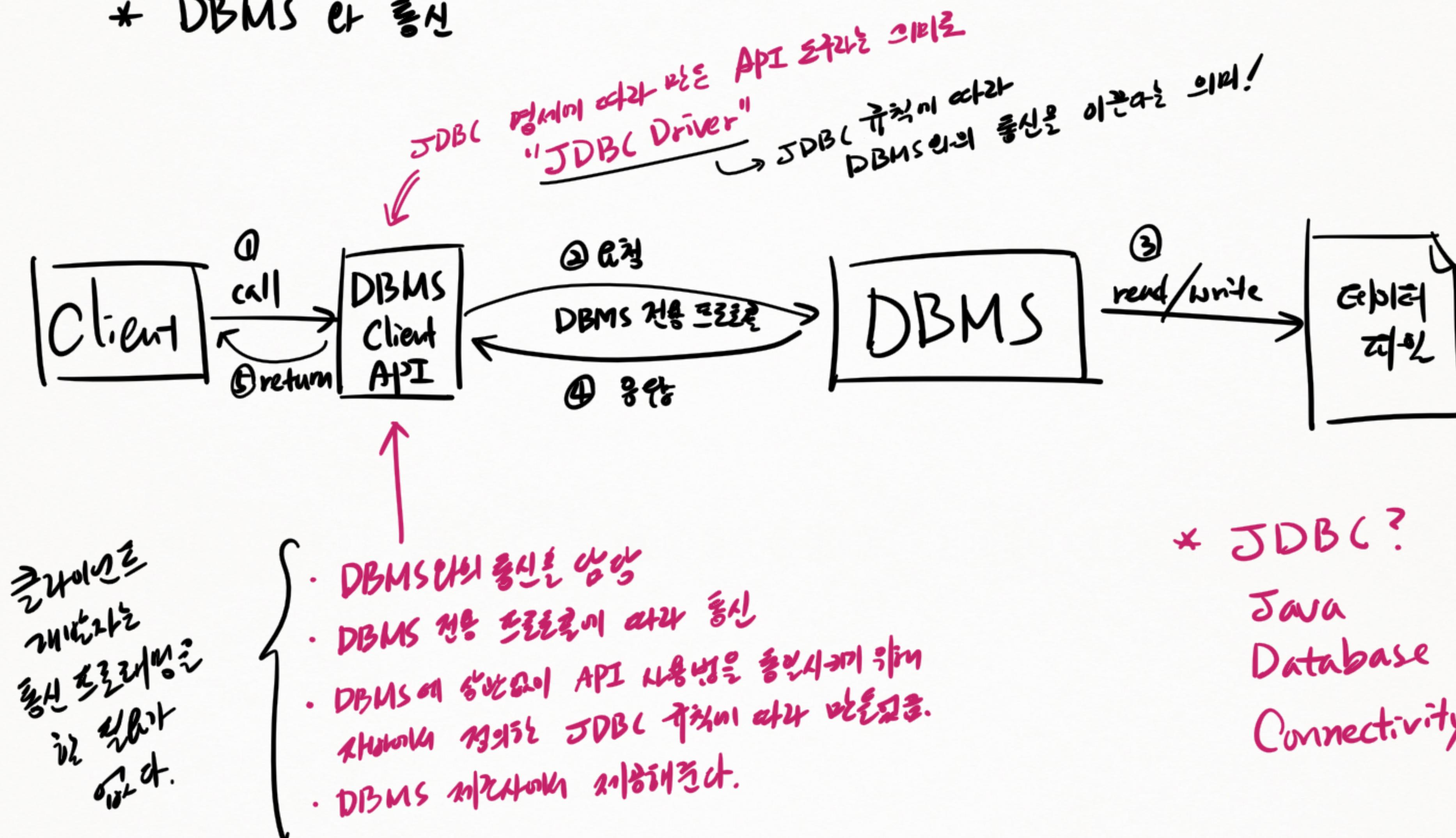


46. DBMS 도입하기

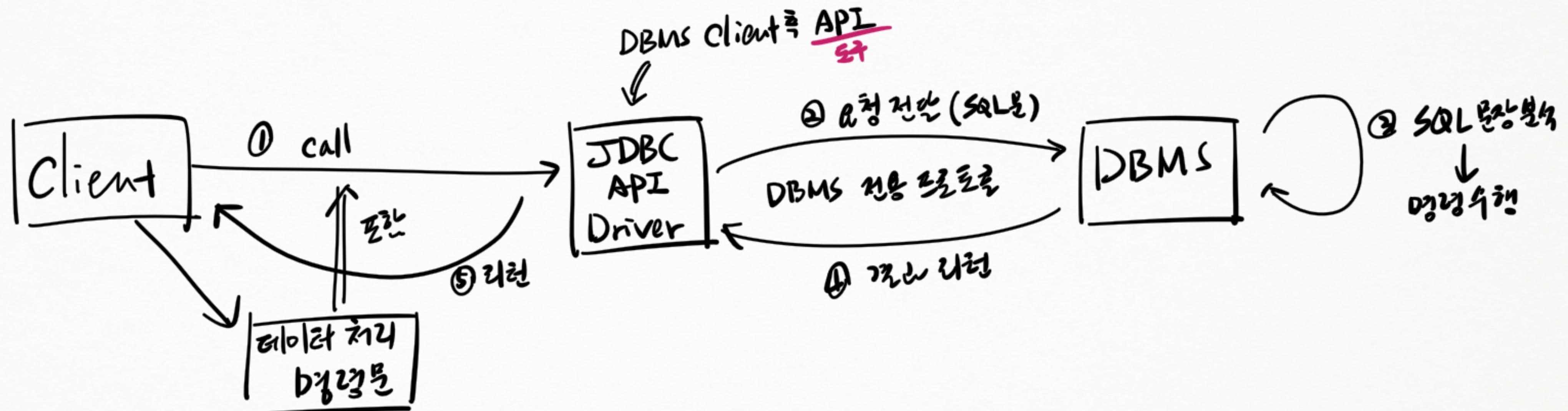
② 개발 단계



* DBMS 와 통신



* DBMS 에 데이터 처리를 요청하기



* SQL
Structured
Query
Language

구조적인
문법을 갖춘
데이터 처리
명령언어

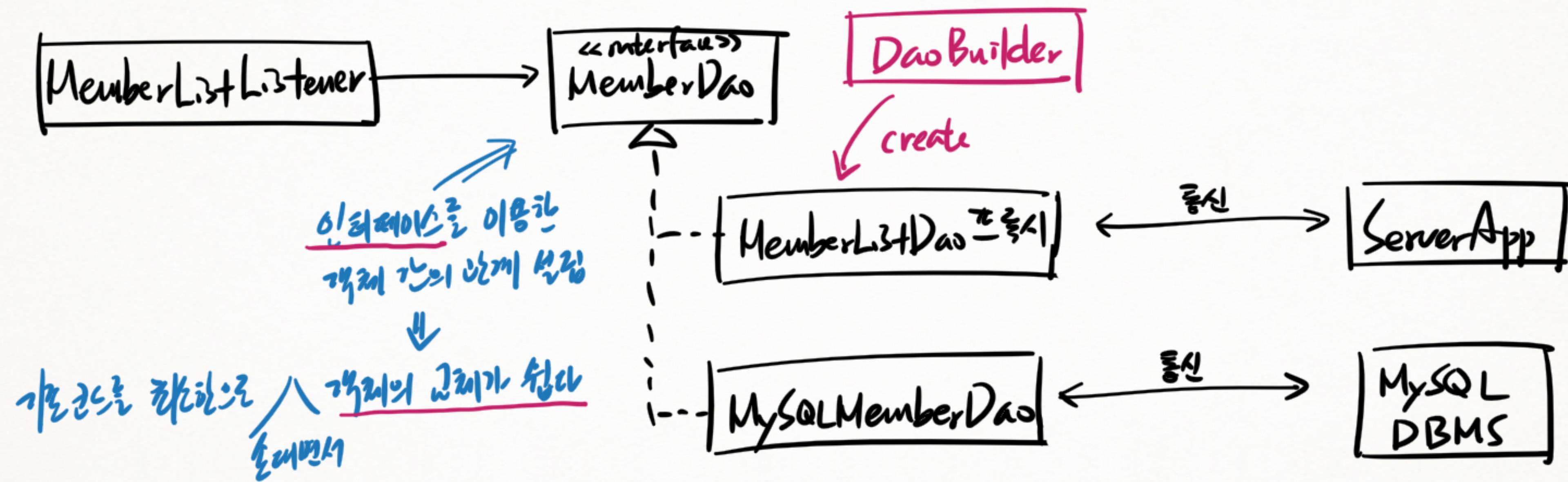
- 표준언어
 - ↳ DBMS 응용제작.
(DBMS와 상관없이
사용할 수 있다.)

SQL 문법에 맞춰
작성한다.

→ 실수 :

준비SQL + DBMS 사용 쓰임

* DAO 와 DBMS



47. SQL 쓰기법 종류

① 일상적인 SQL 쿼리를 예제로

update myapp-board set

```
title = '%s'
```

content = '%.s'

where category = %d

and board_no = %d

and password = '%c\\$'

1111' or '1'=1'

"SQL 퀼 쿼리" " хр再也 퀼 쿼리 "

100m² 100kg/m² 100kg/m² 100kg/m²

SQL 語句

1920

$\Rightarrow \underline{\text{and password}='1111' \text{ or } '1'='1'}$

47. SQL 쓰는법 - PreparedStatement!

② SQL문을 SQL비 풀어쓰니

update myapp-board set

title = ? , ?로 대체한다.

content = ?

where category = ?

and board_no = ?

and password = ?

* in-parameter 란 뭐?

PreparedStatement pstmt = con.prepareStatement(SQL);

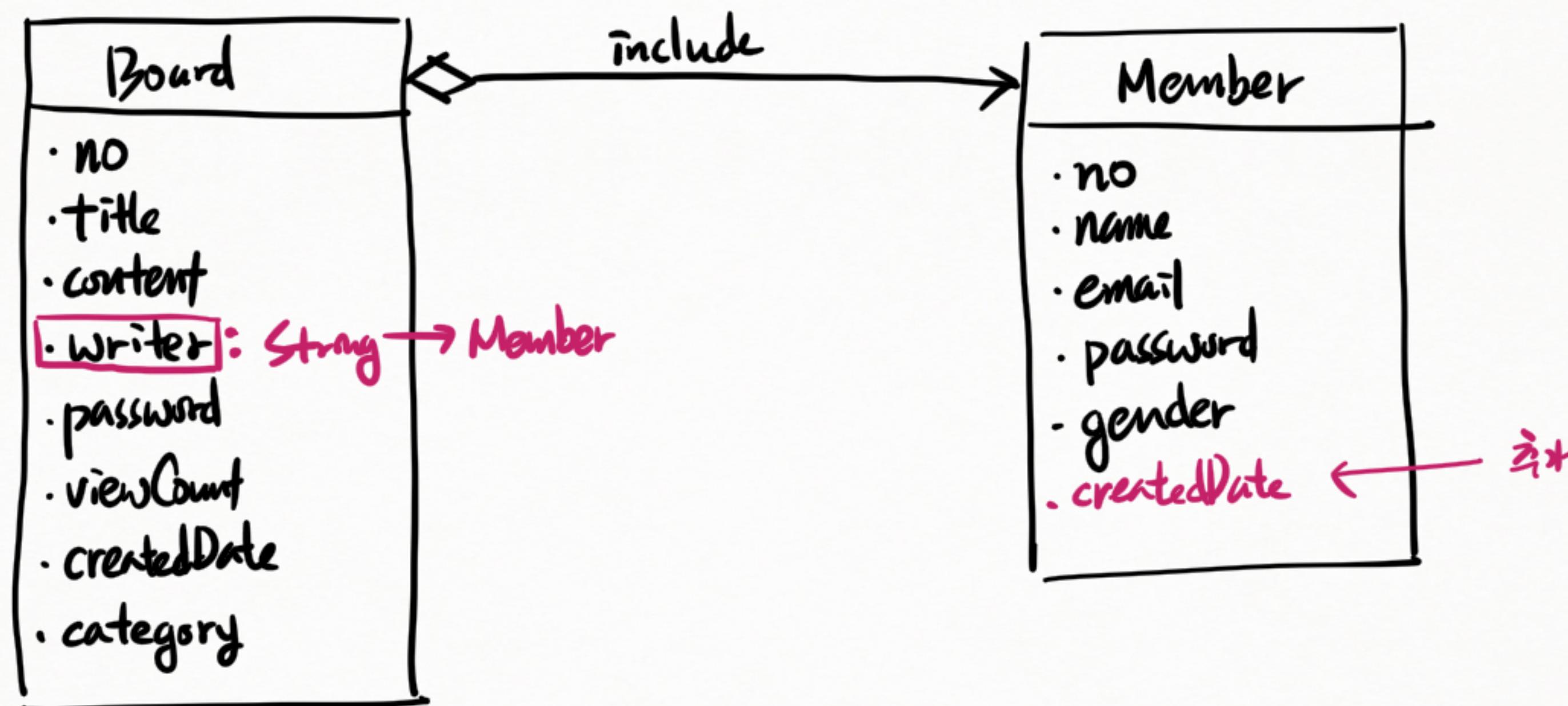
pstmt.set~~xxx~~(1, ?);

~~in type~~

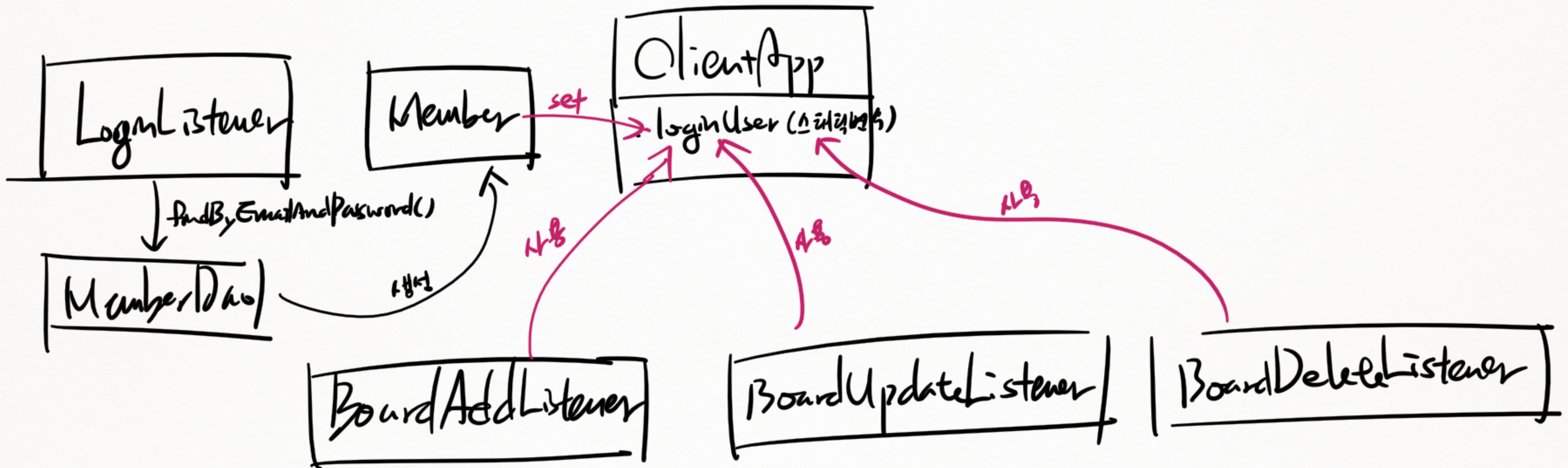
in-parameter인 줄 알았던

in-parameter인 줄 알았던

48. Entity Relationship

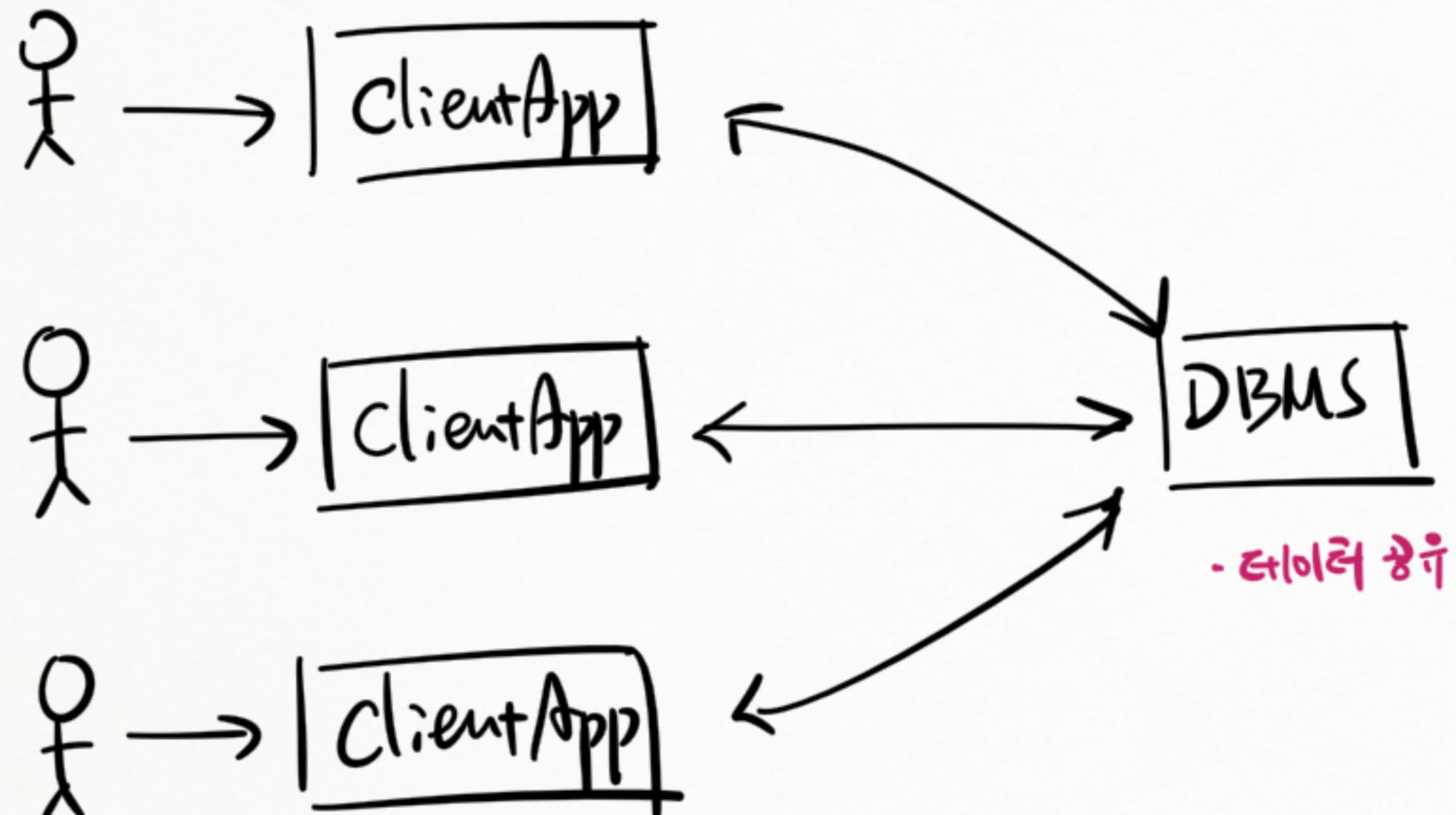


49. 로그인 흐름도



50. Application Server Architecture

① C/S Architecture



Client

- local에 설치
 - ✓ 실행할 때 local 자료 사용
 - ✓ 기능 변경 → 재설치 필요
 - ✓ C/C++, VB, PowerBuilder, Delphi

Server

- client에서 직접 접근
 - client가 해킹을 당하면 서버가 위험
 - 보안에 취약

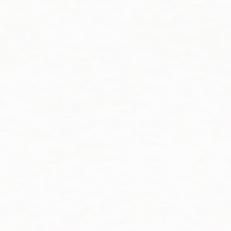
Global Business



경쟁이 심화



제조기 라이즈사이클이 끊어짐



업무 조직의 변화가 끊어짐



업무 변경이 끊어짐 → 업무 시스템 변경이 끊어짐

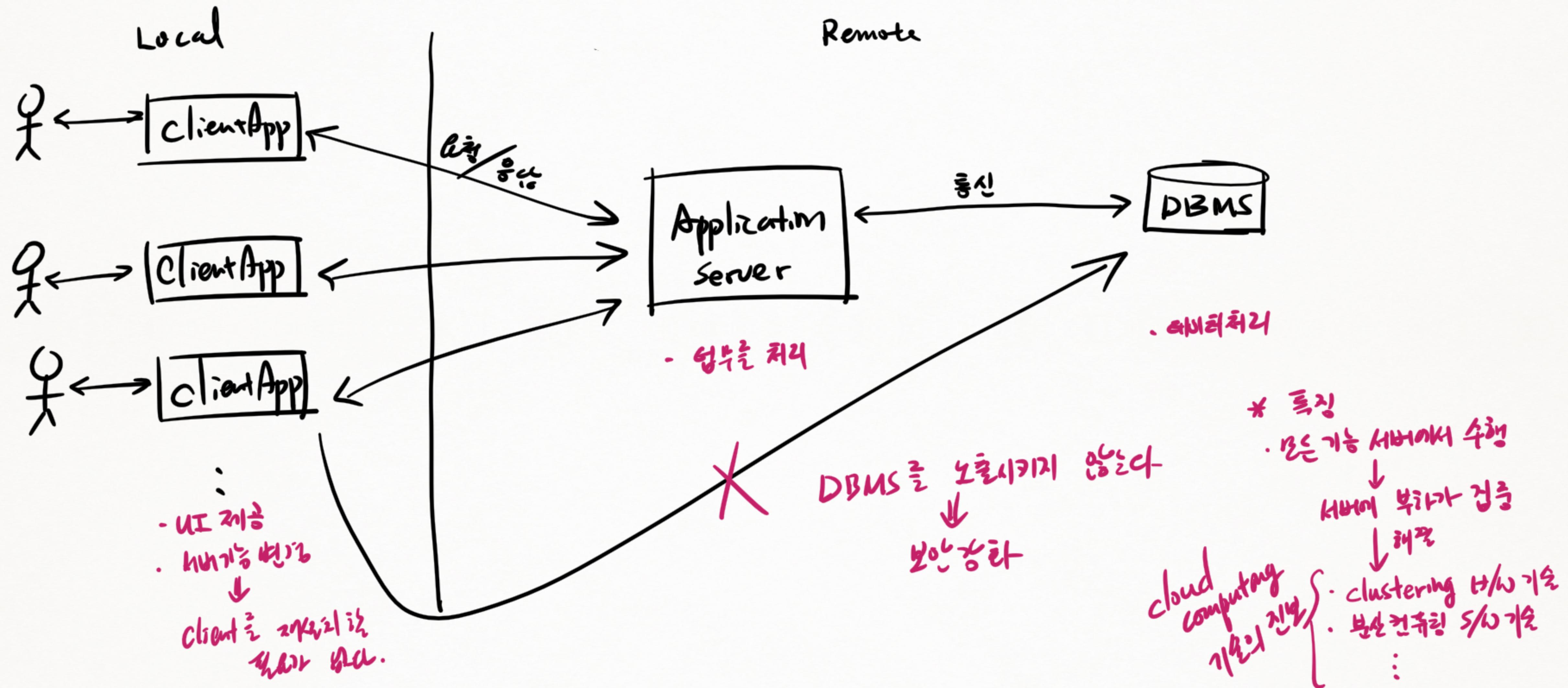
유지보수 기능 끊어짐

제작비가 끊어짐

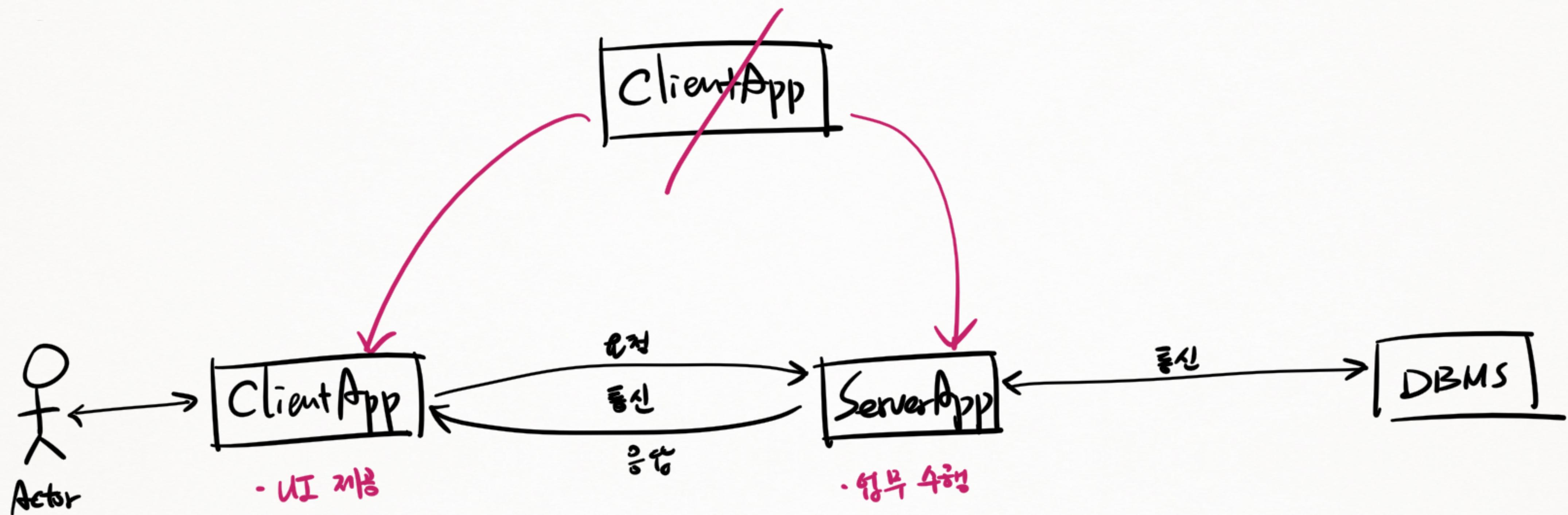


50. Application Server Architecture

② Application Server Architecture



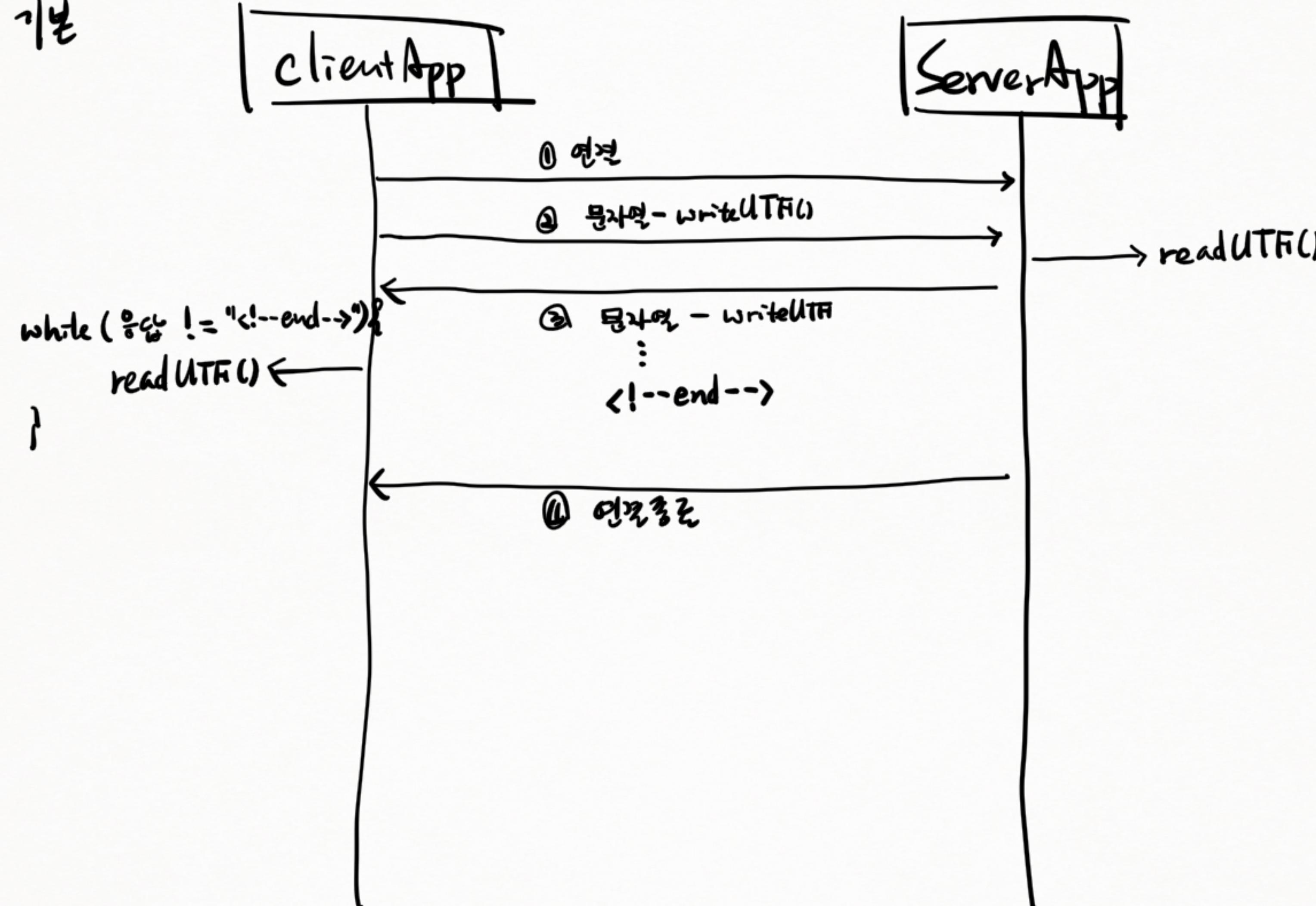
50. Application Server Architecture



50. Application Server Architecture - Protocol

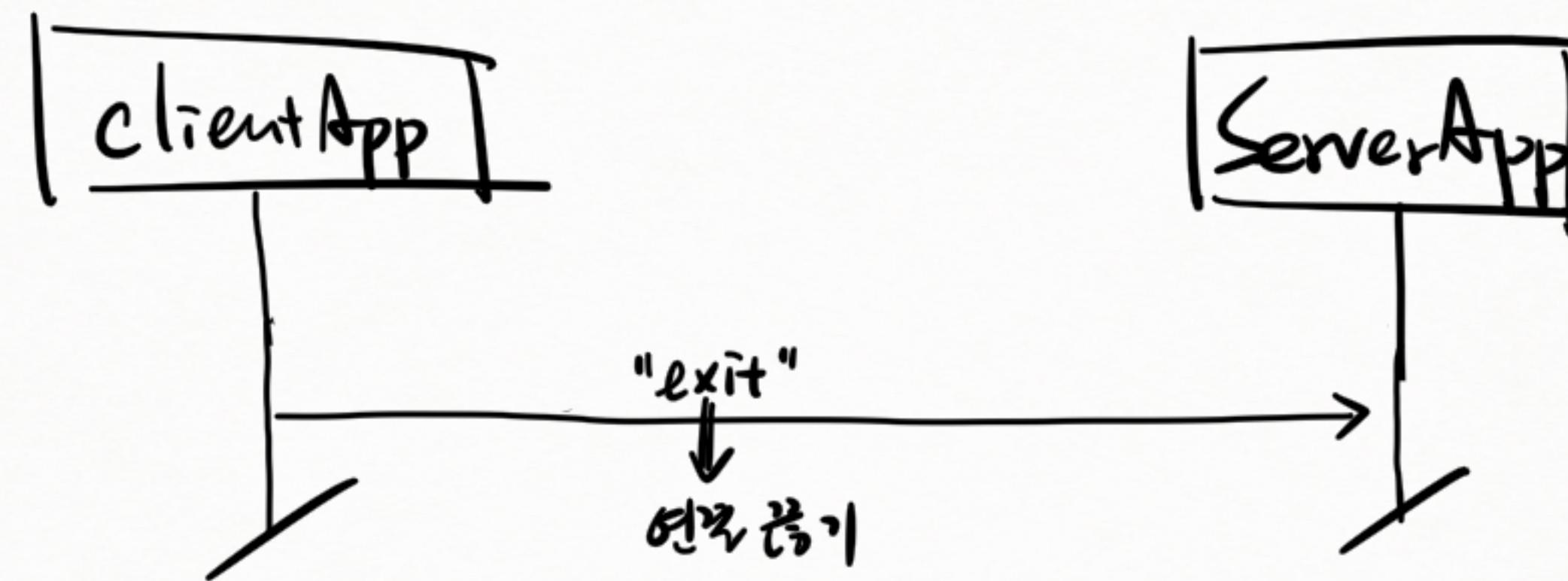
① 요청/응답 기본

Pseudo code
기호적 표현

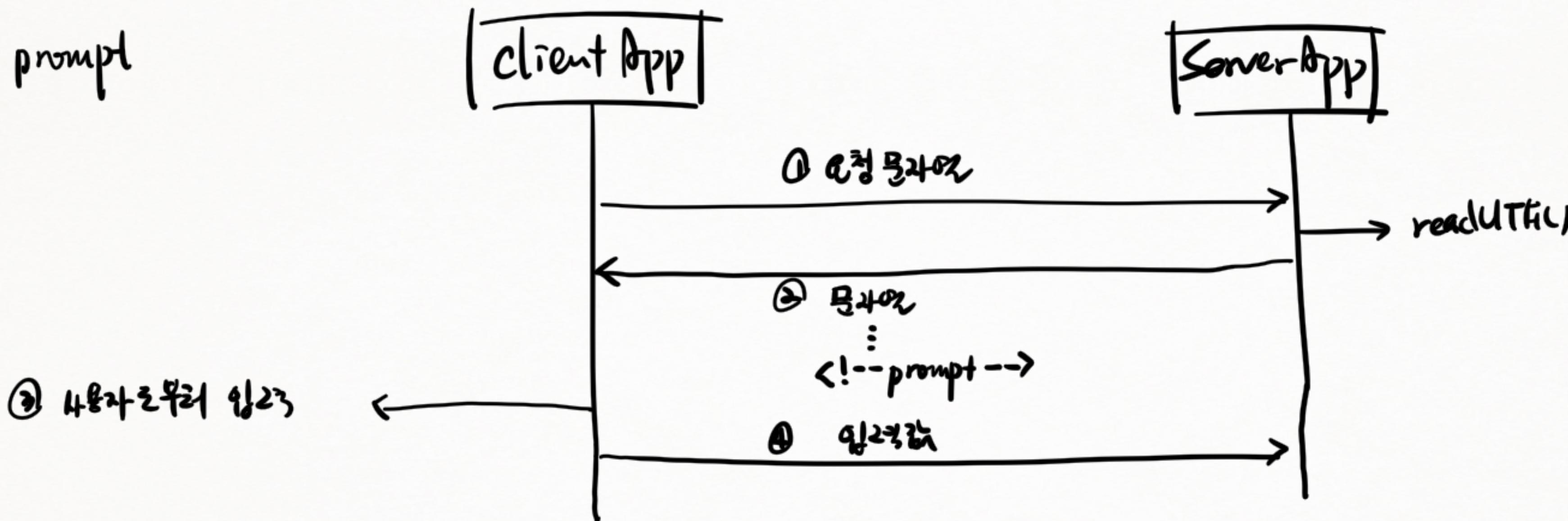


50. Application Server Architecture - Protocol

② exit

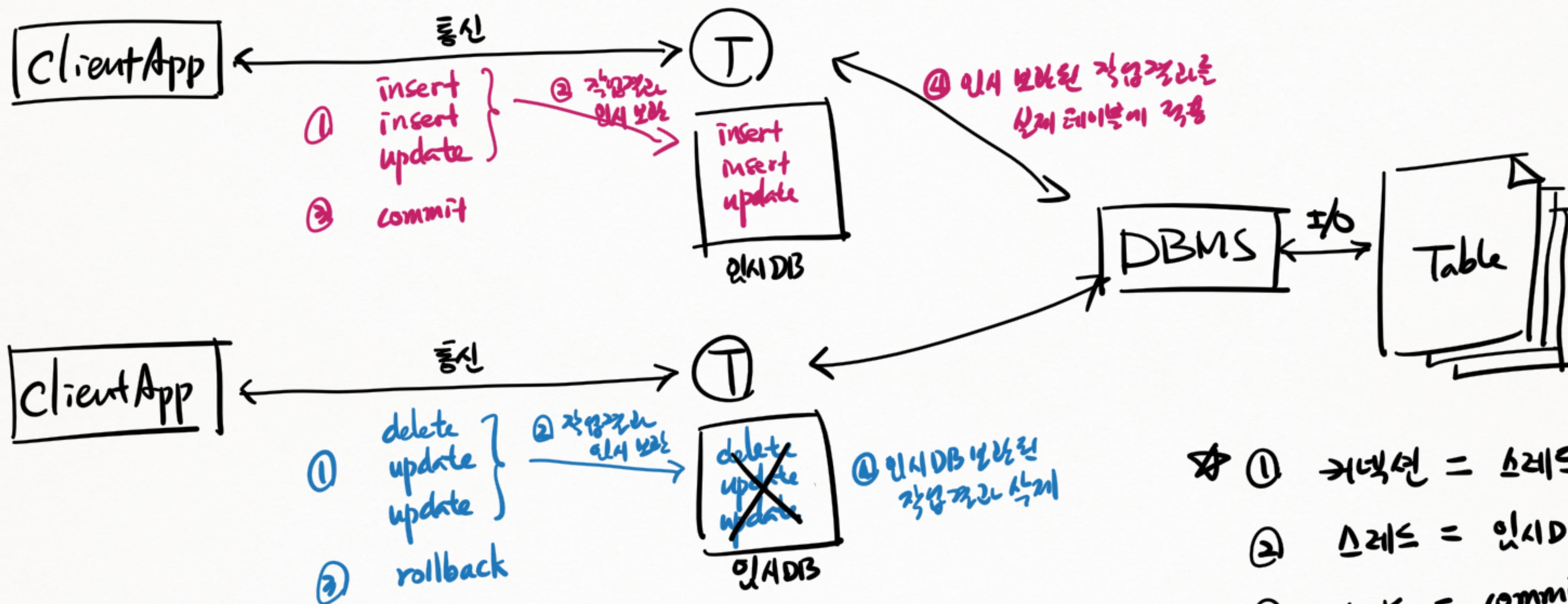


③ prompt



51. DB ConnectionPool 사용

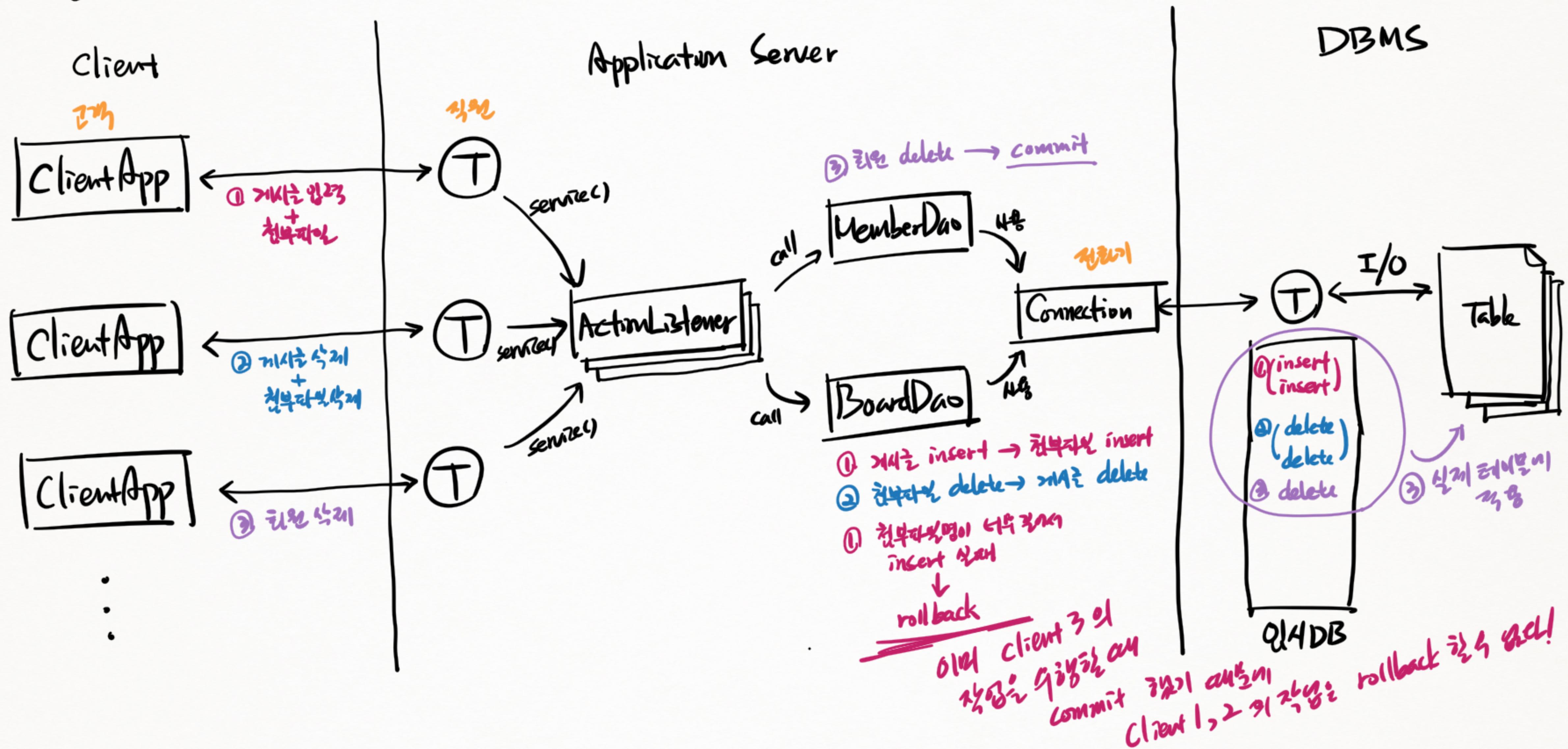
- ① DBMS 작업 사용 - autocommit = false



- * ① 커넥션 = 소레드
- ② 소레드 = 임시DB
- ③ 소레드 = commit/rollback
- ④ 클라이언트 별로 독립적인 작업을 수행
"클라이언트 간에 작업의 간섭이
발생하지 않는다."

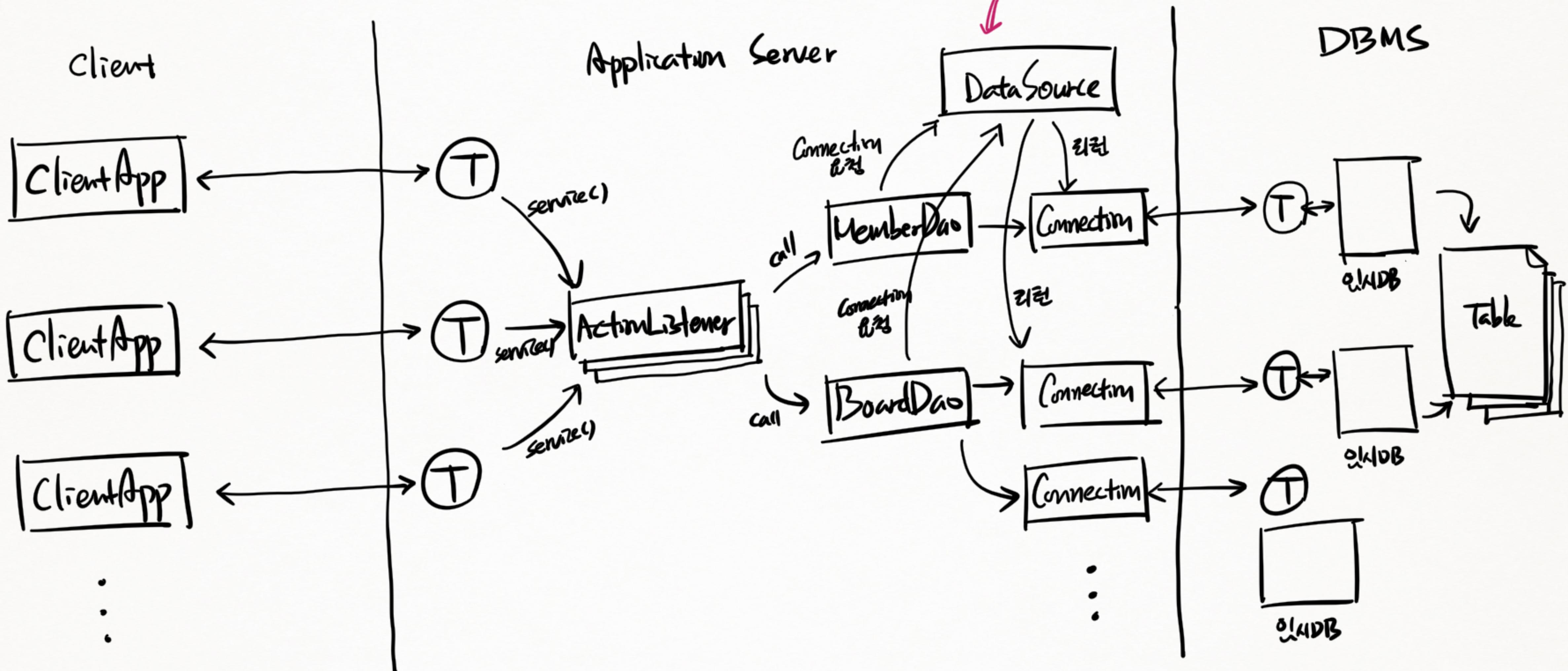
51. DB ConnectionPool 사용

② 혼합

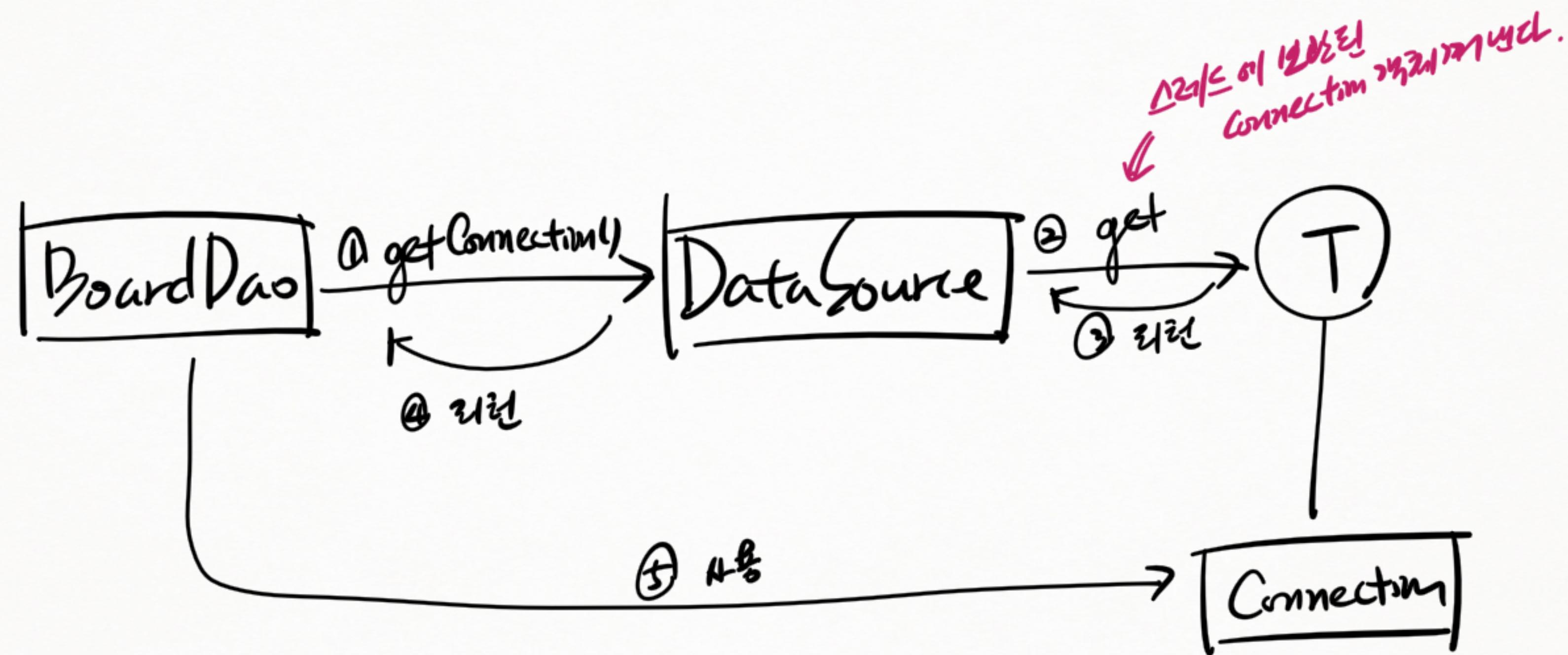


51. DB ConnectionPool 사용

③ 스레드별로 Connection 주분할 사용하기



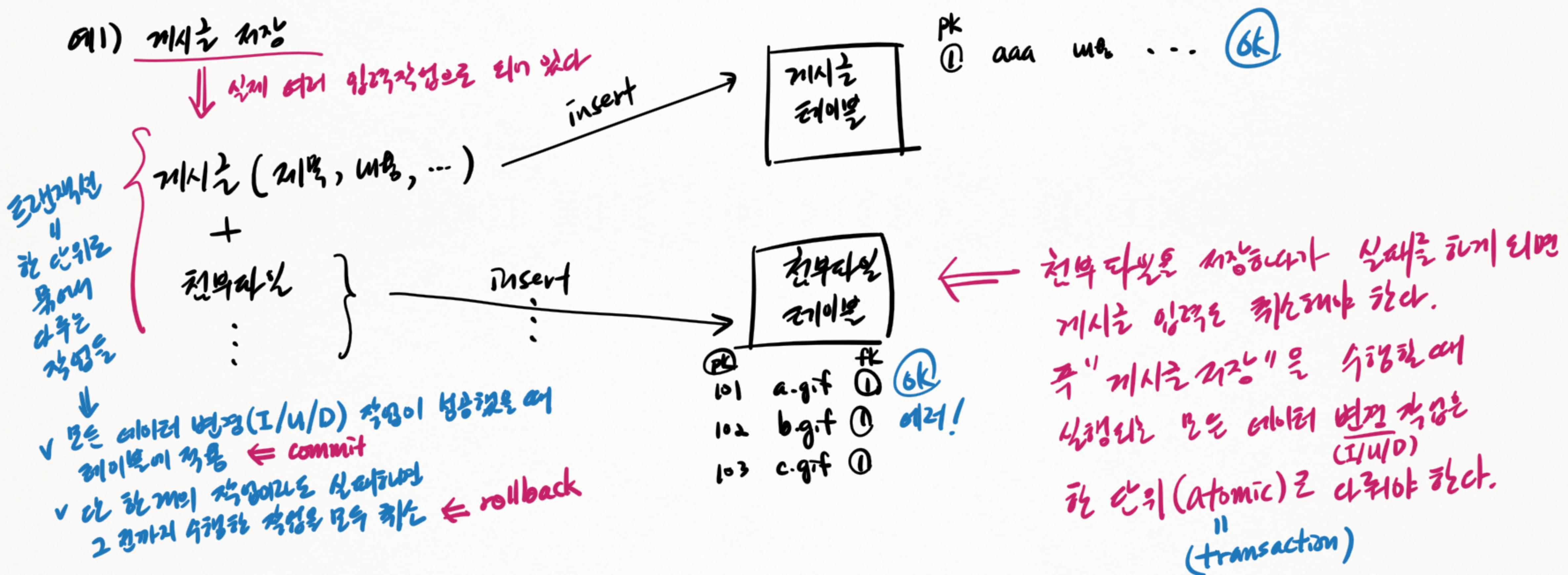
* DataSource 의 생애주기



- * auto commit 을 false로 설정해야 하는 이유?

11

- 실무에서는 데이터 입력/변경/삭제 작업을 수행할 때 여러 개의 테이블을 사용한다
 - 학습용 예제는 한 개의 데이터를 한 개의 테이블에 저장하지만
실무에서는 많은 항목을 다뤄야 하기 때문에 데이터가 겹쳐지지 않도록 여러 테이블이 보관된다.

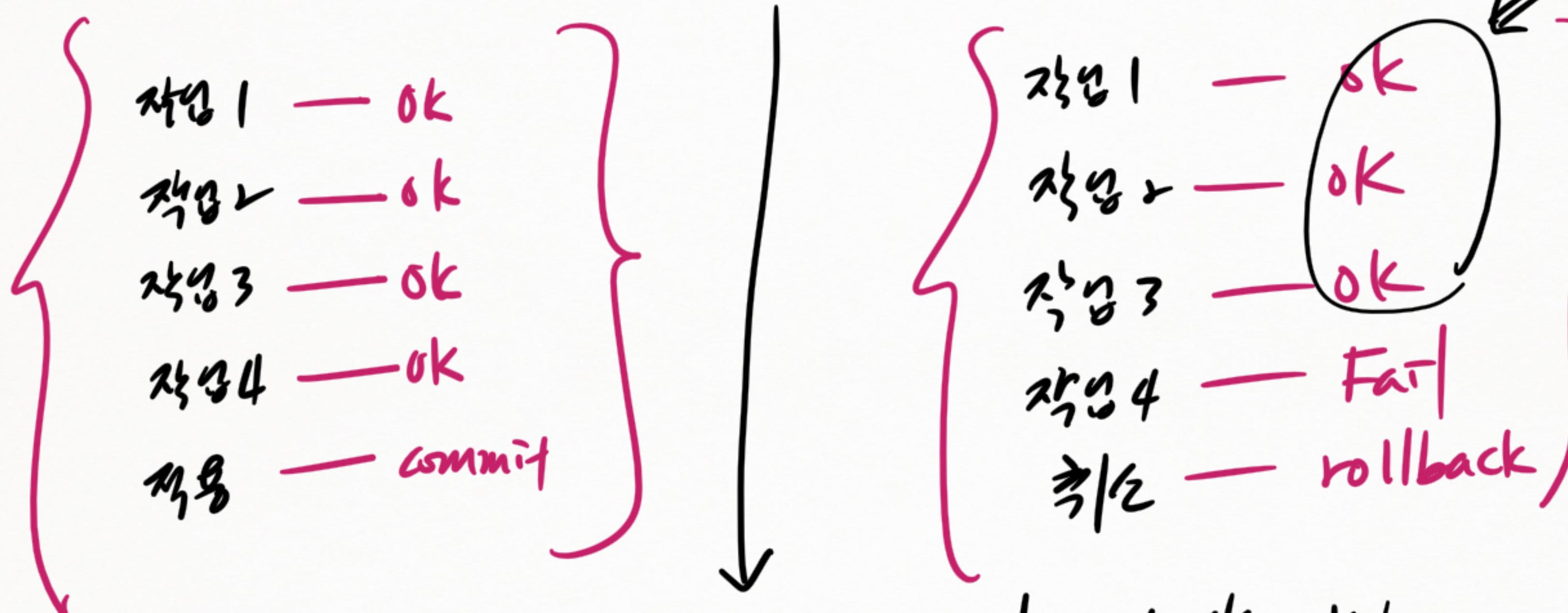


* 트랜잭션 (transaction)

↳ 여러 작업을 한 단위로 다루기 위한 높은 레벨의 것.

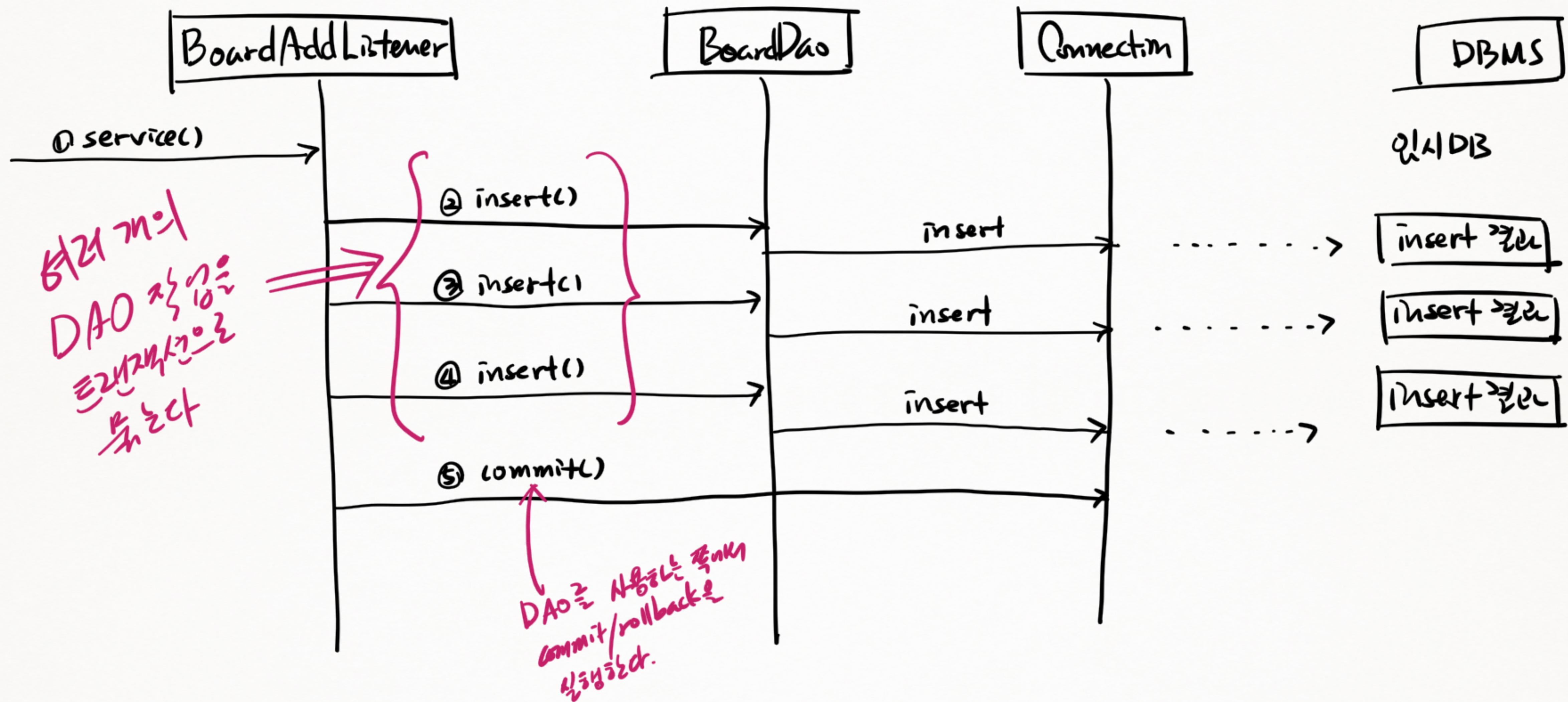
auto commit 을 false 로 설정해야 한다.

하지!



auto commit이 false 일 상황에서는
여러 번째 작업이 커넥션 빙글로 인해 DB에 연결된다
(I/U/D)

* commit / rollback 네 gì지?



* DAO or DataSource

