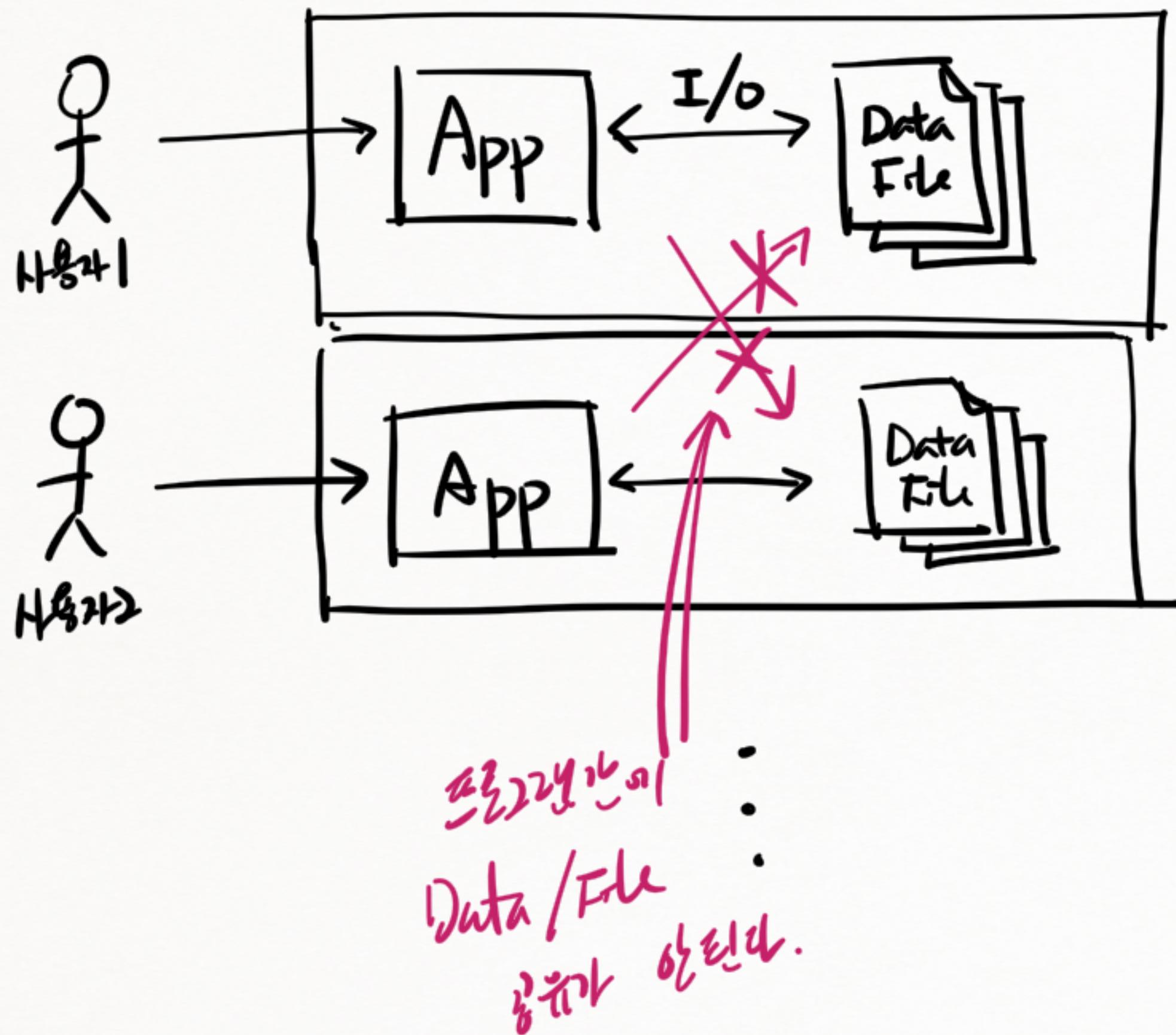


37. 바이오킹을 통한 데이터 공유

① 현황



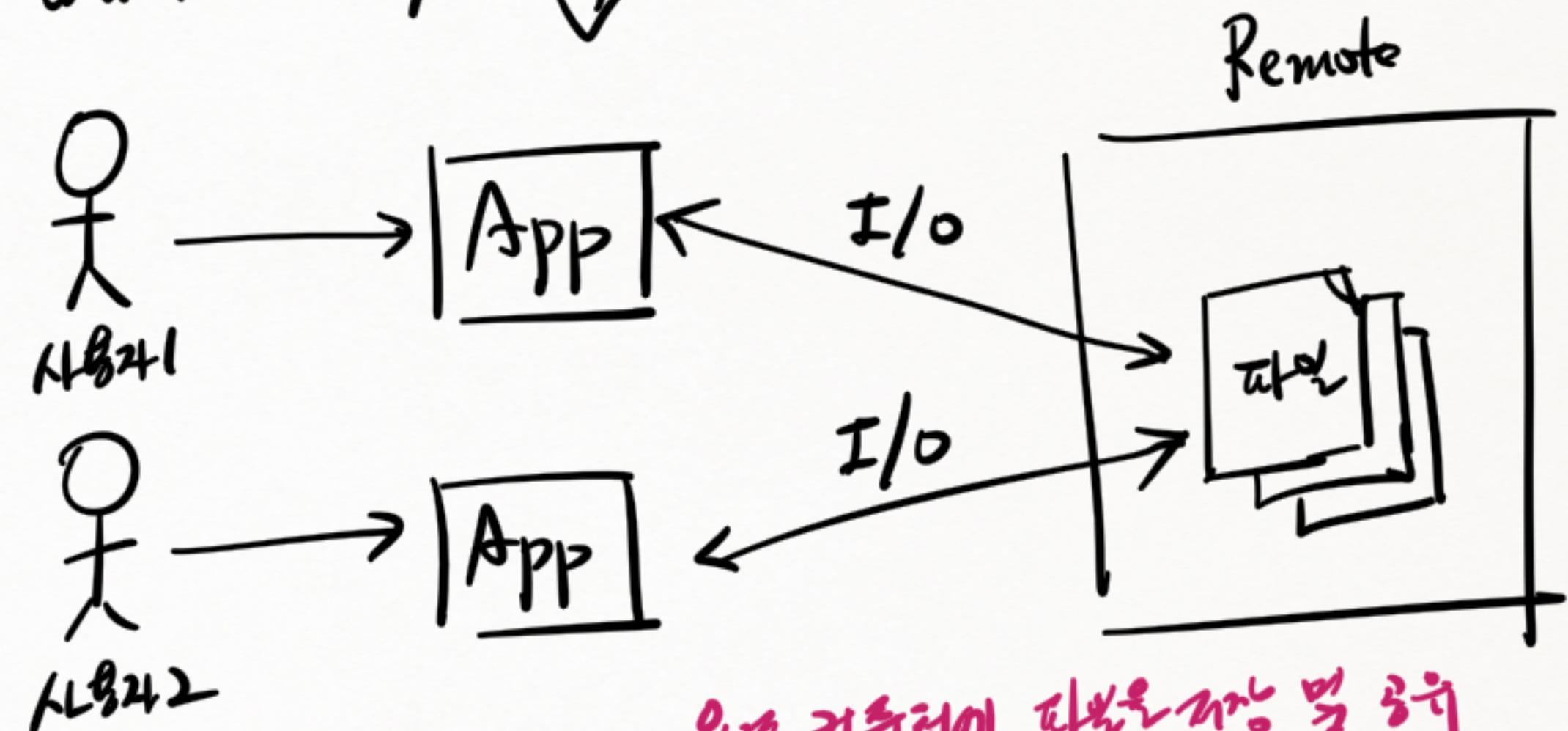
App 을 실행하는 컴퓨터

데이터 파일이 로컬에 저장된다.

↓

App 간에 데이터를 공유할 수 없다!

② 데이터 파일은
별도의 컴퓨터에 분리/공유 ↓ 가능할까?

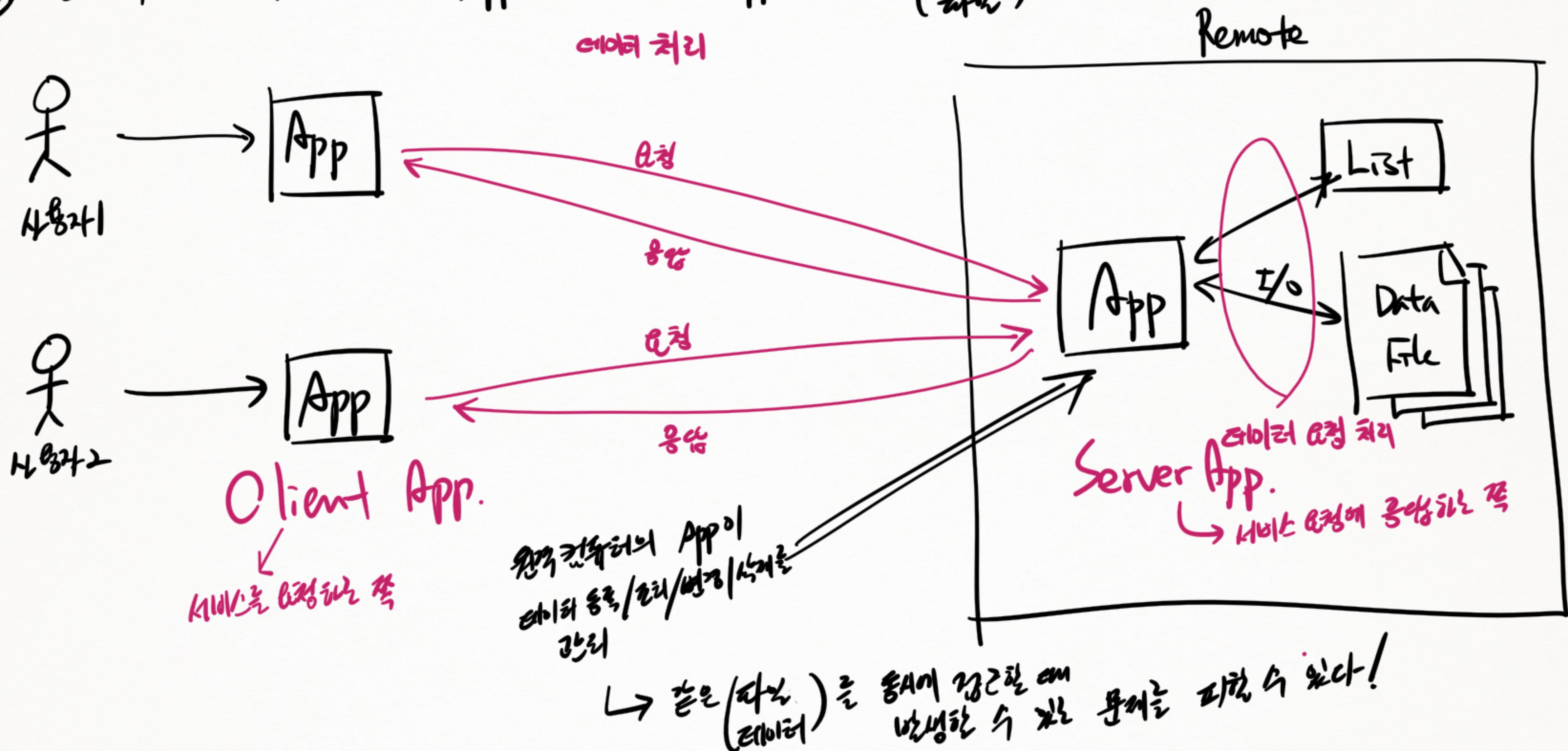


- 원격 컴퓨터에 파일을 저장 및 공유

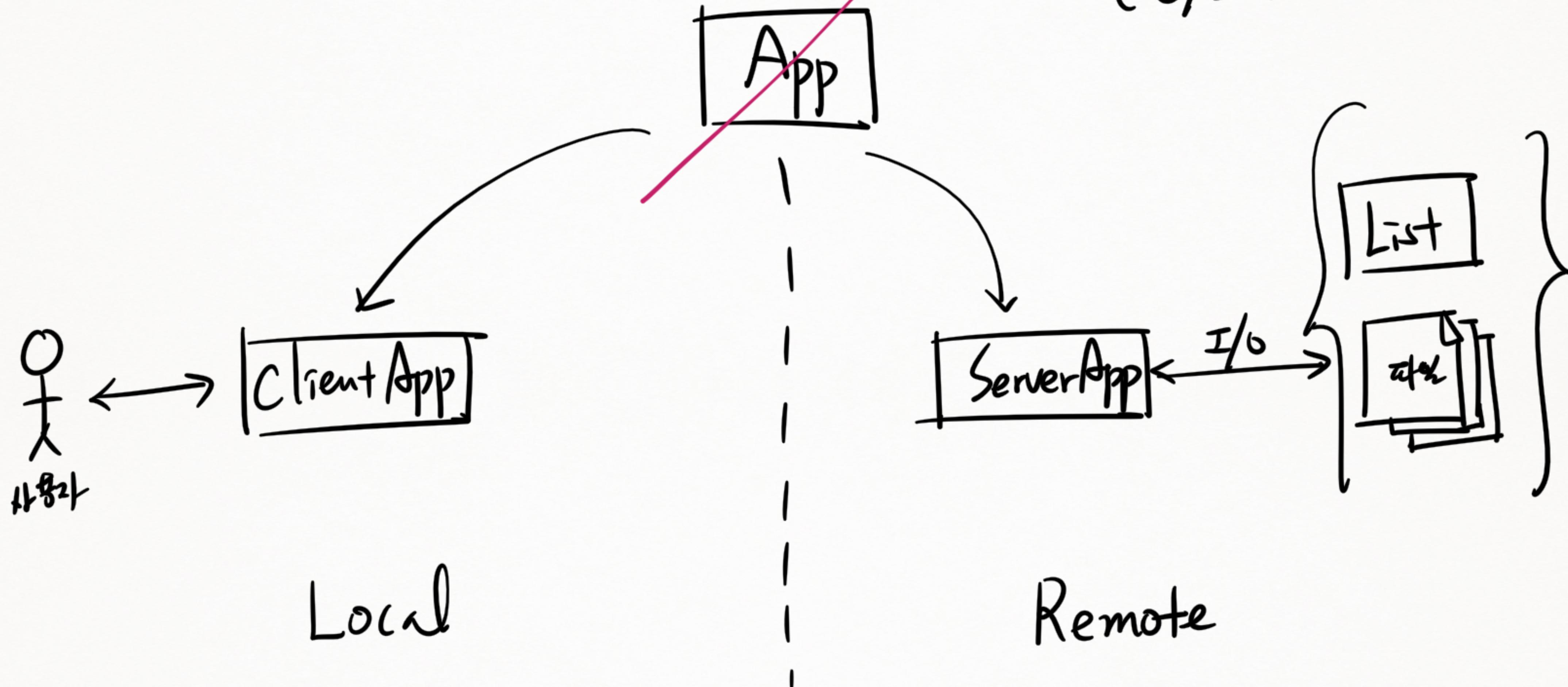
중재자

여기 App이 같은 파일을 편집하다 보면
각자의 데이터가 깨끗해졌다.

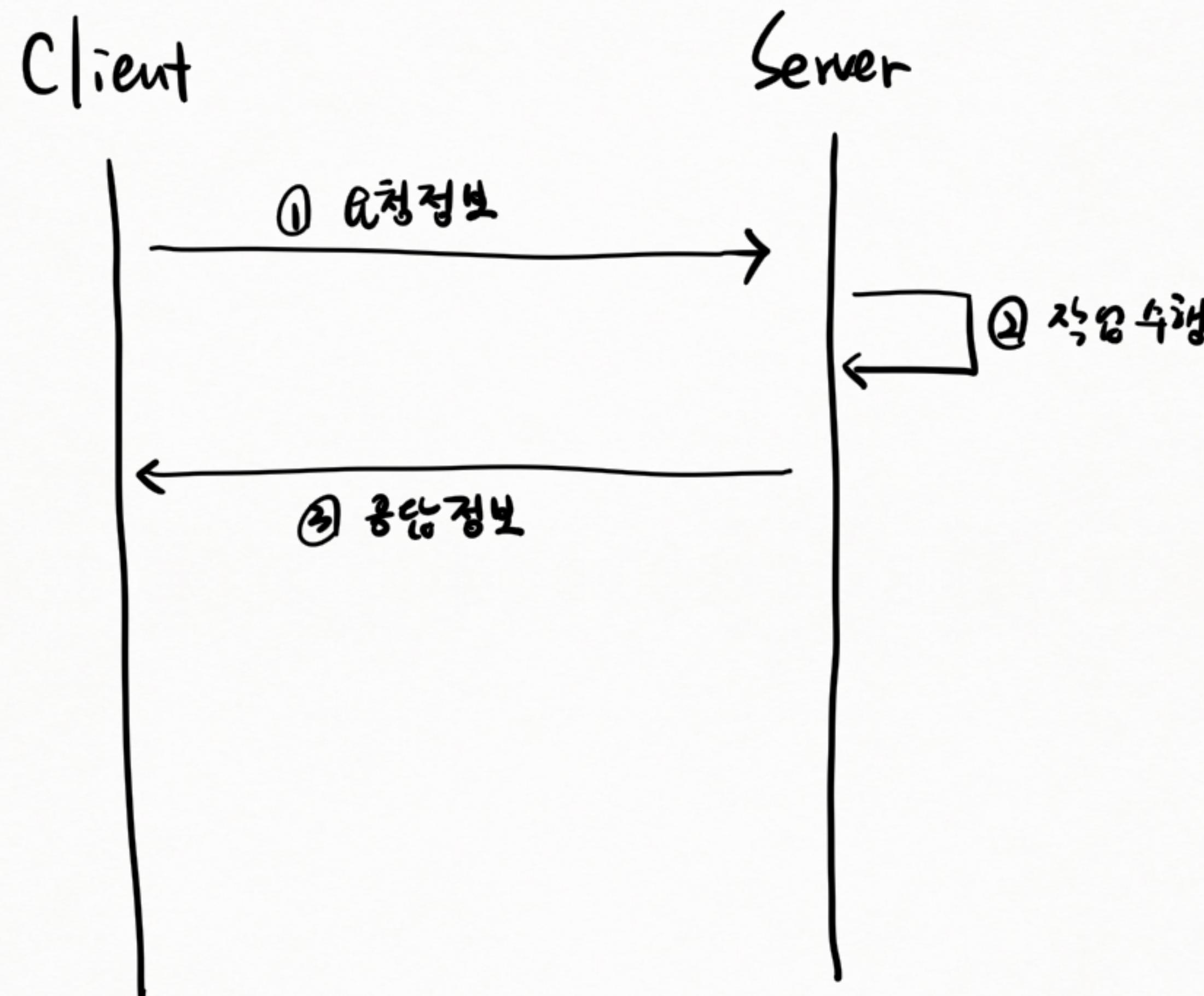
③ 데이터 관리 기능을 별도의 App. 으로 분리 - App이 직제 (데이터)
를 접근하는 것을 막는다.



* 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 서비스명.

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, ... }]"  
}
```

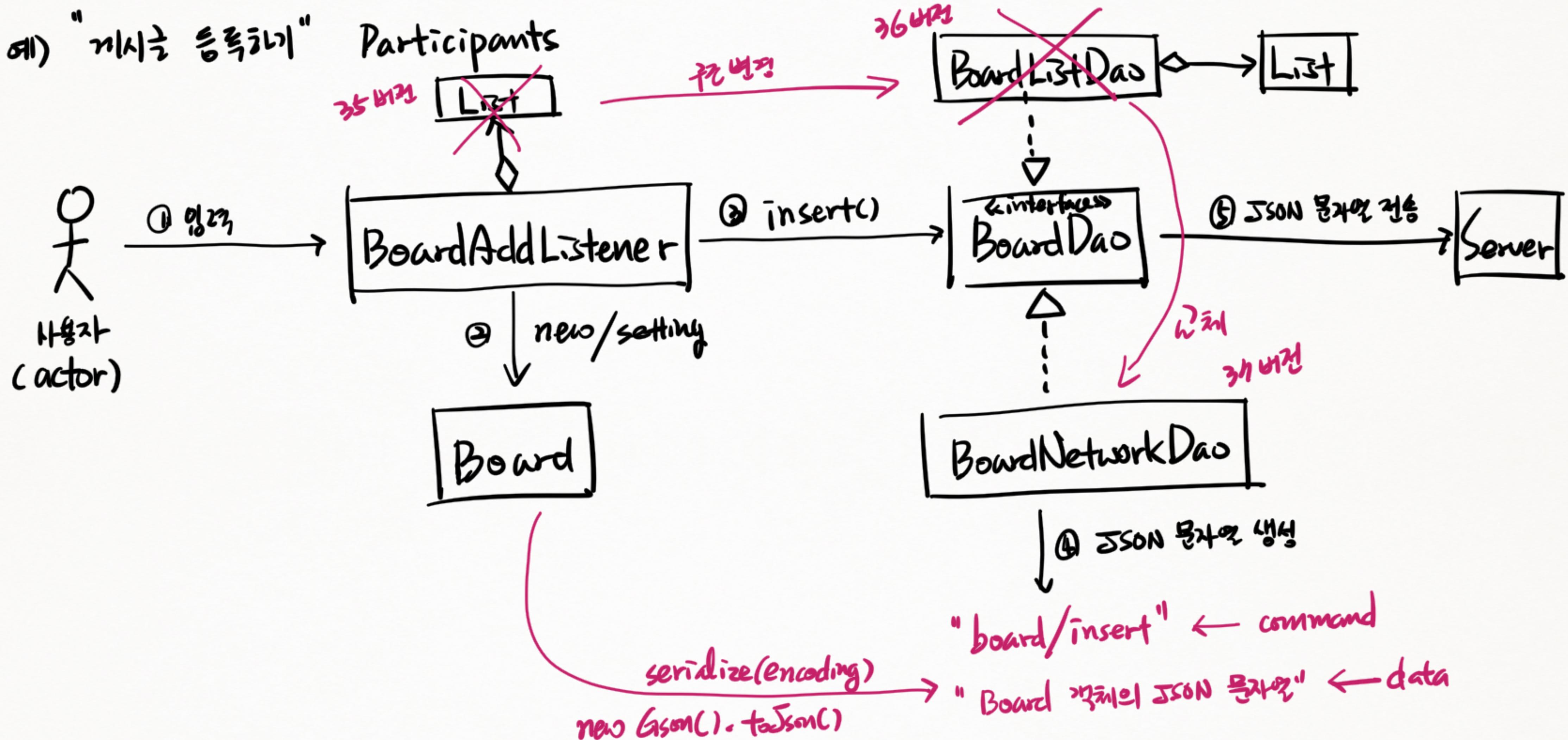
- * 실행 결과
"success": 성공
"failure": 실패

↓ 실패 예

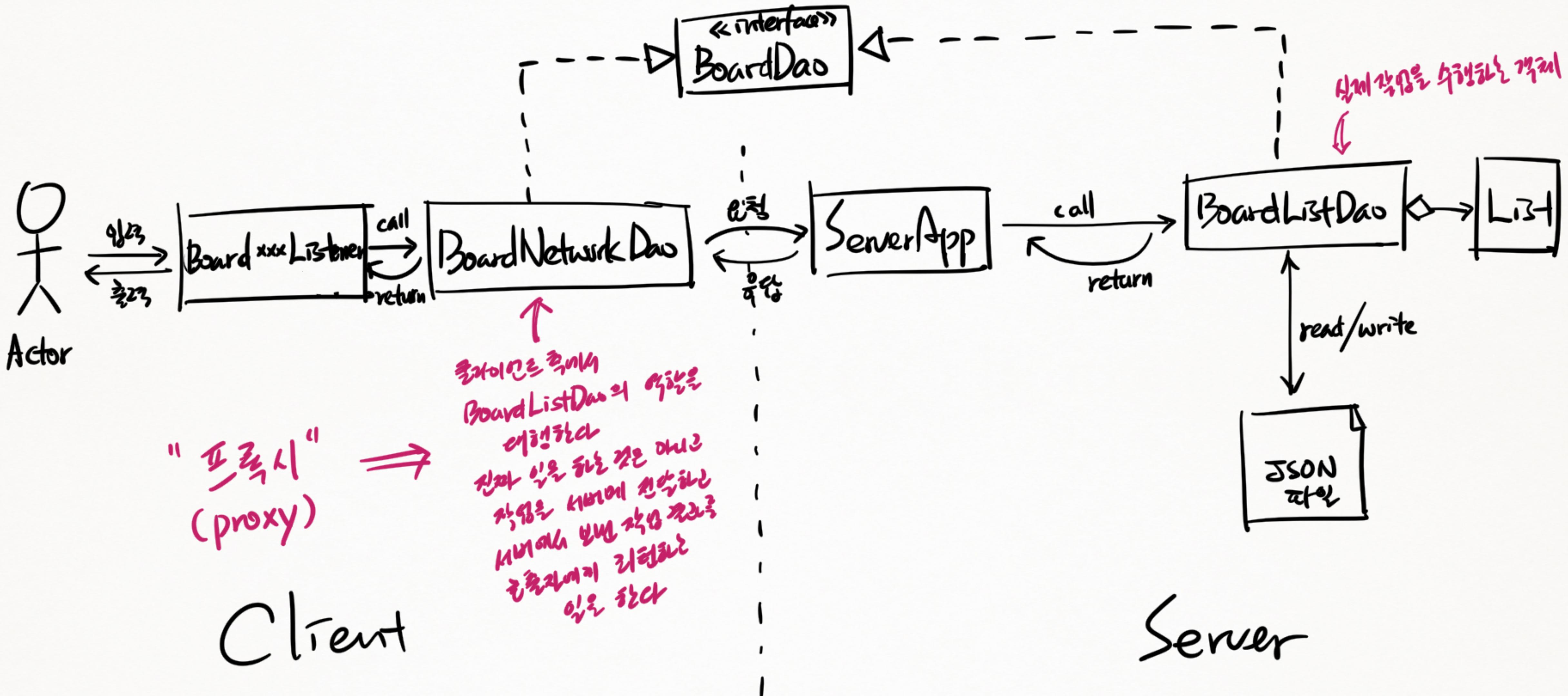
```
{  
    "status": "failure",  
    "result": "해당 번호의 데이터가 없습니다!"  
}
```

* 요청 정보 뷰티가 작업에 참여하는 객체들과 실행흐름

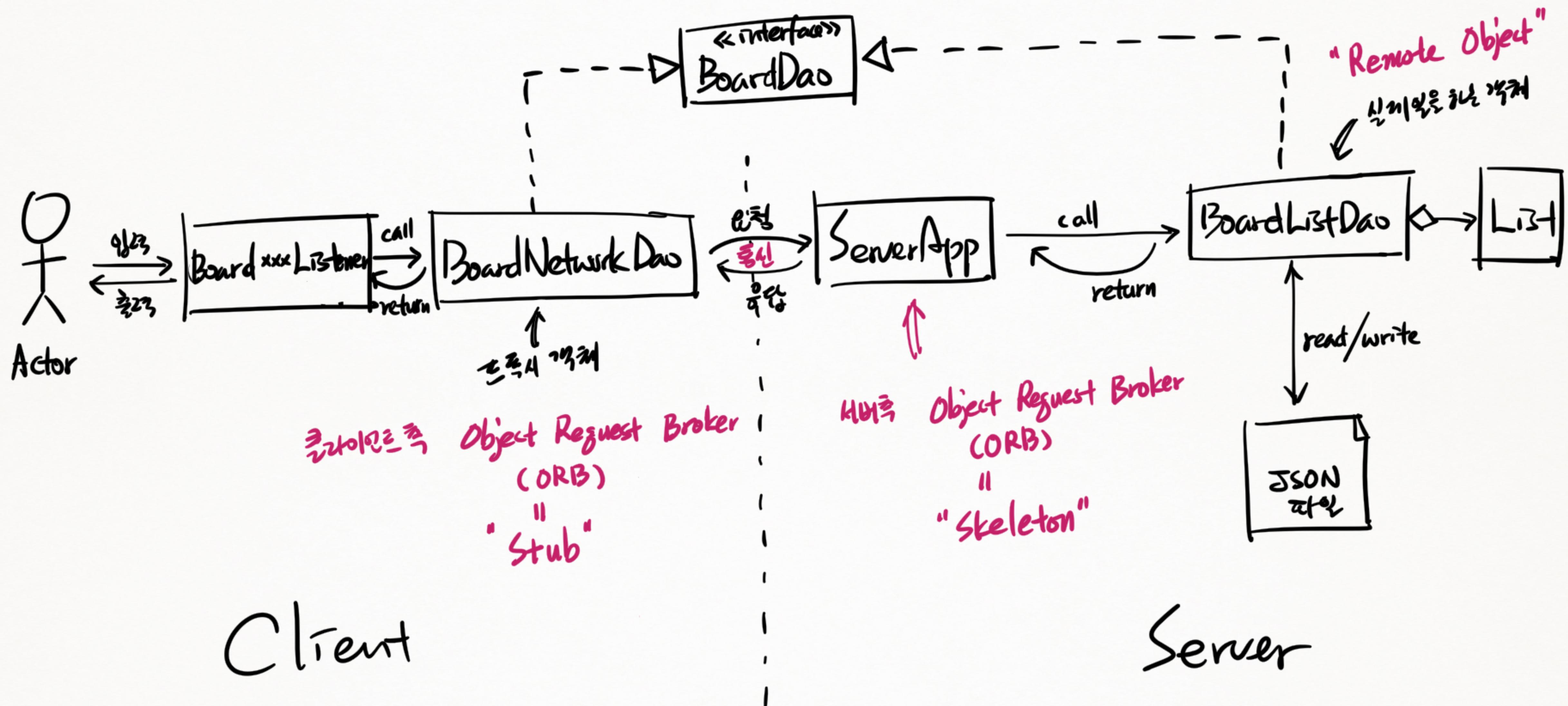
예) "게시글 등록하기" Participants



* DAO 와 Proxy 패턴 (GoF)

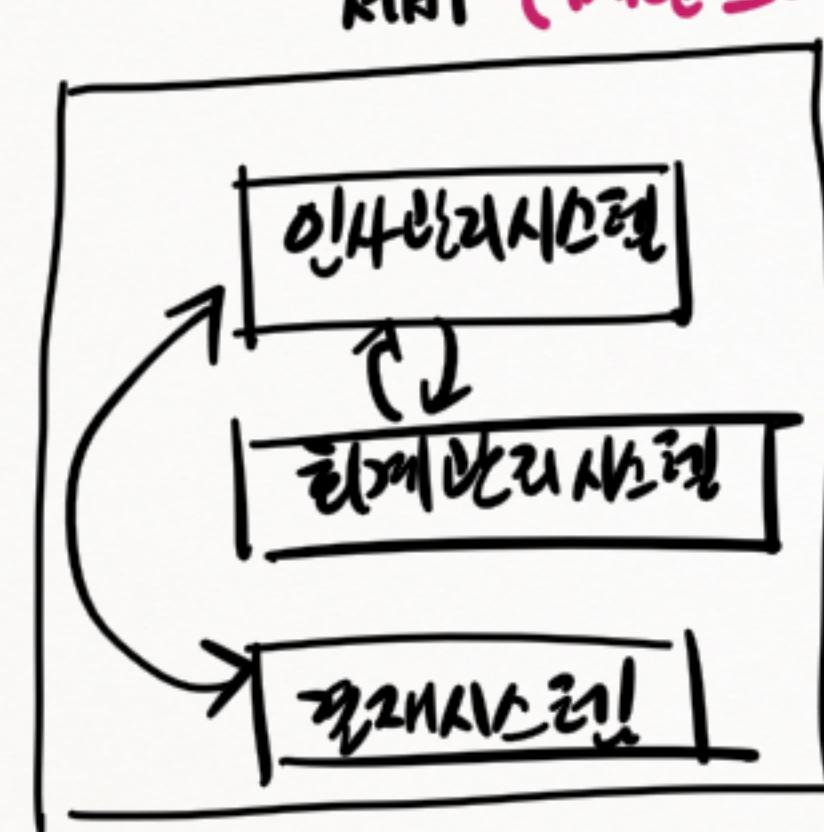


* DAO 와 Proxy 패턴 (GoF) ↗ 예술자의 명칭



* 온라인 컴퓨팅

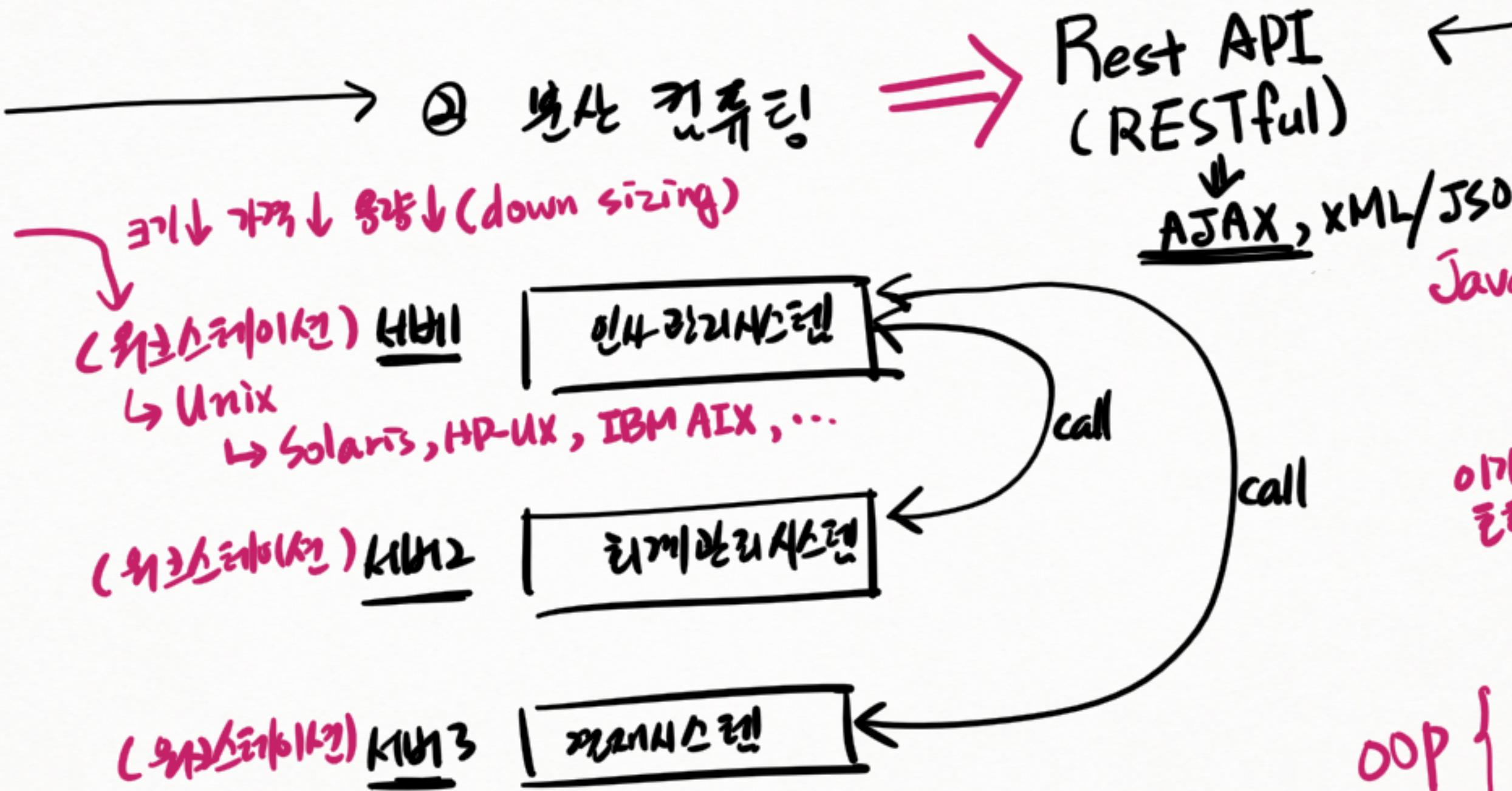
① 중앙 집중식 컴퓨팅



문제점

- 처리에 문제 발생
 - ↓
 - 모든 업무시스템이 멈춰
- 사용자 증가 → 서버 증설
(RAM, HDD, CPU)
 - ↓
 - 서버 장비는 고가
 - ↓
 - H/W 유지비가 높아졌다.

② 온라인 컴퓨팅

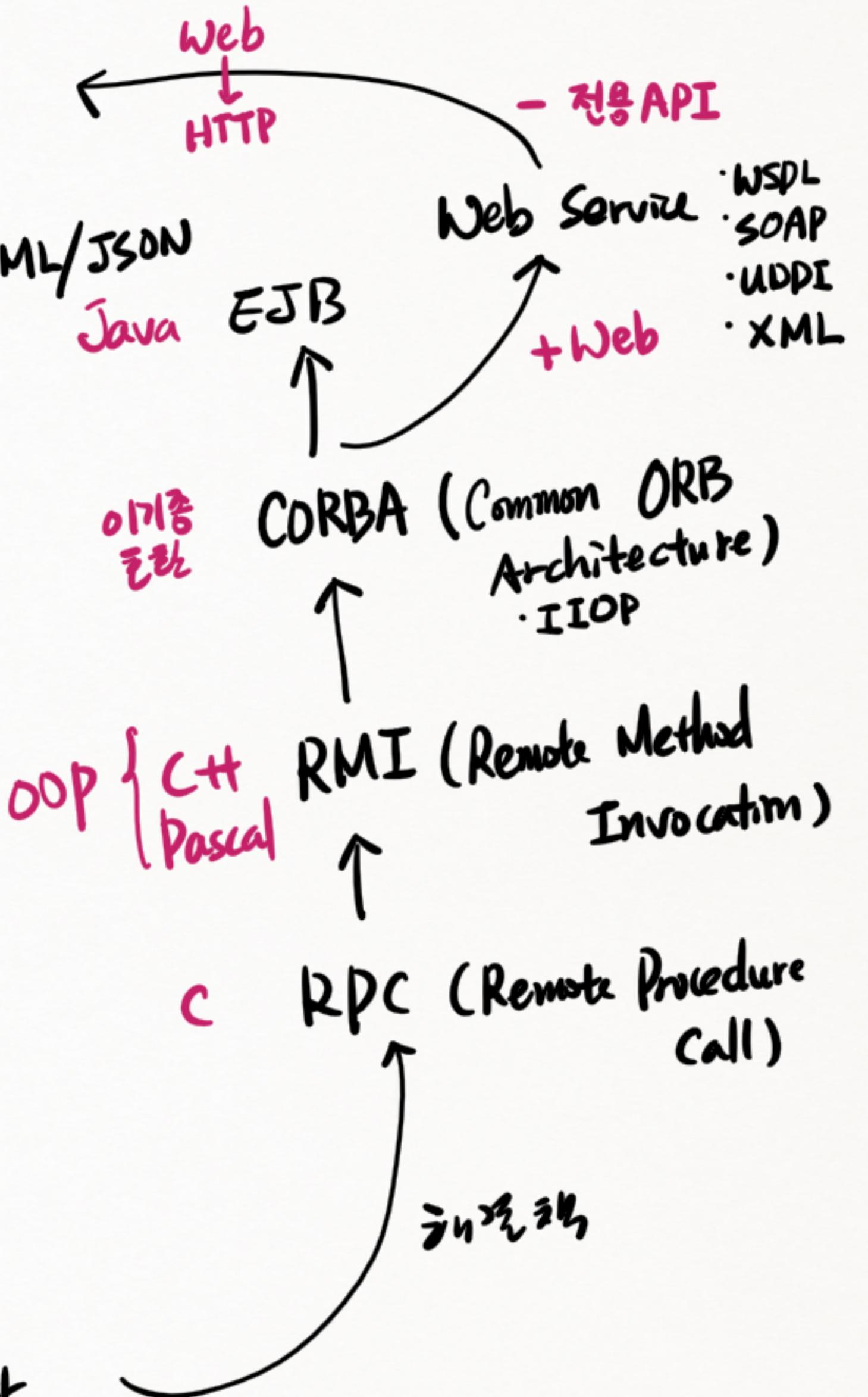


문제점

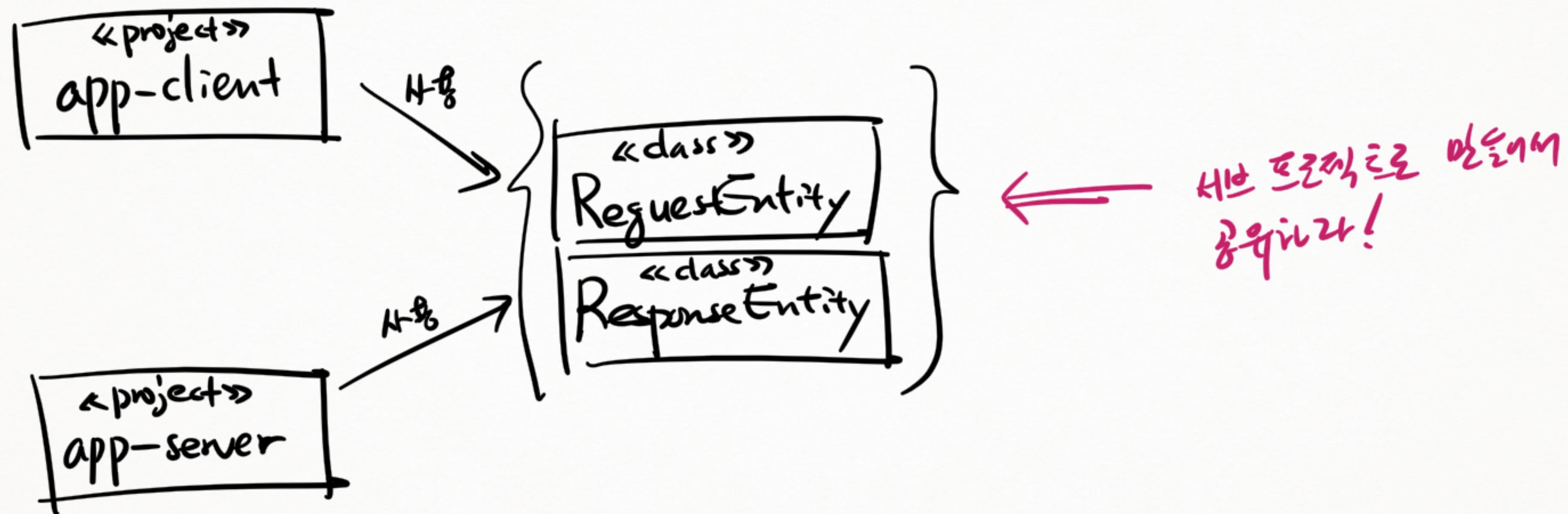
- 다른 시스템의 코드를 실행할 수 없다
 - ↓
 - 비즈위크를 이용하여 퍼블릭 혹은
 - ↑
 - 개발자/프로그래머 별로 구현
 - ↓
 - 복잡화 X → 유지보수 어렵다

Rest API (RESTful)

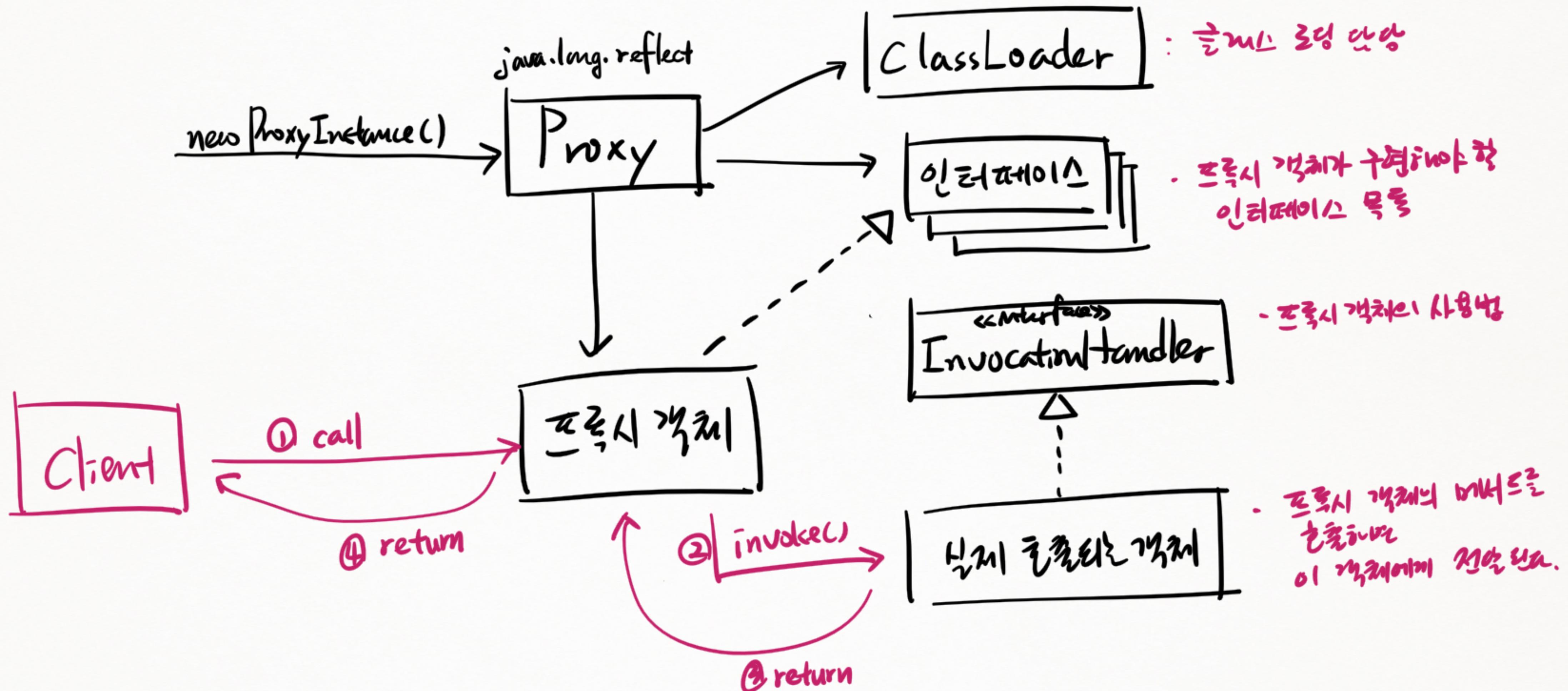
AJAX, XML/JSON



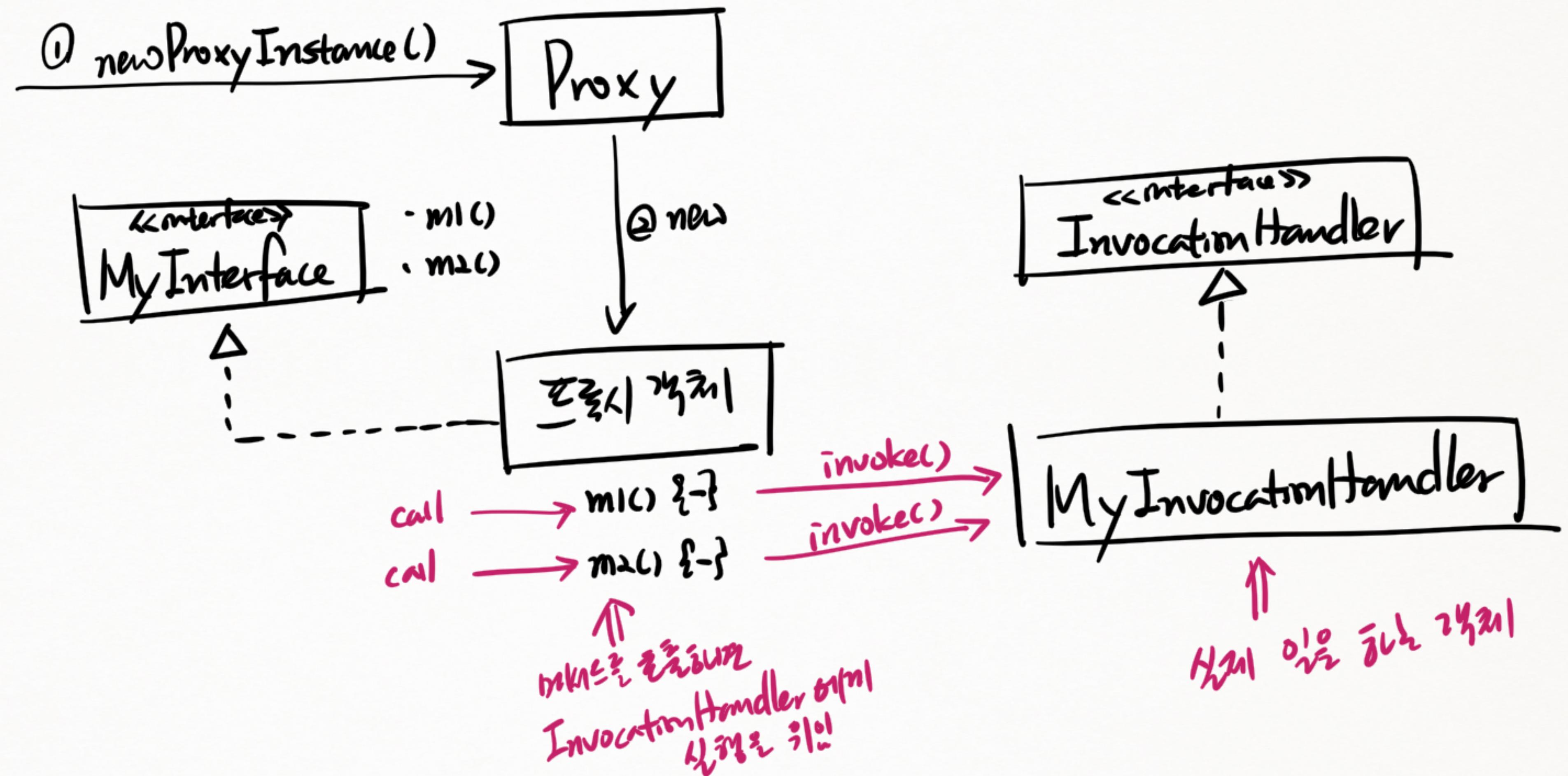
* Project 간에 헤더 공유하기



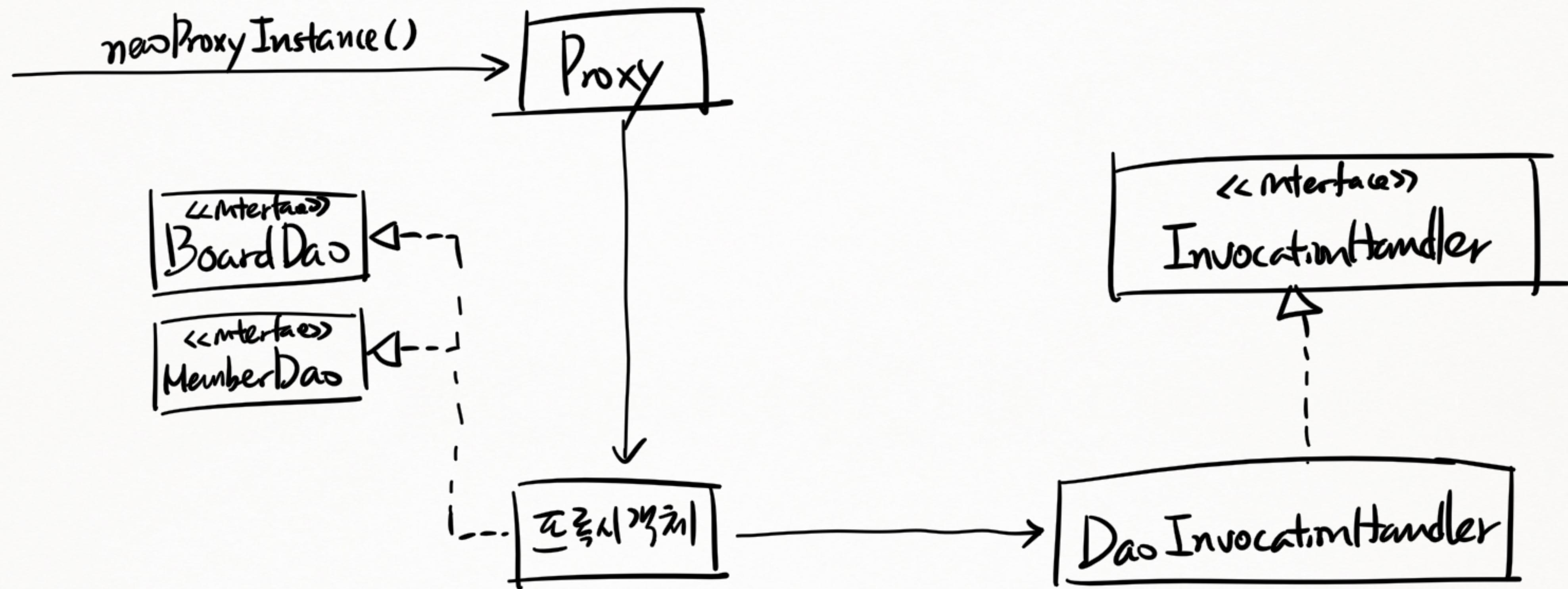
3.8. 프록시 객체 자동 생성



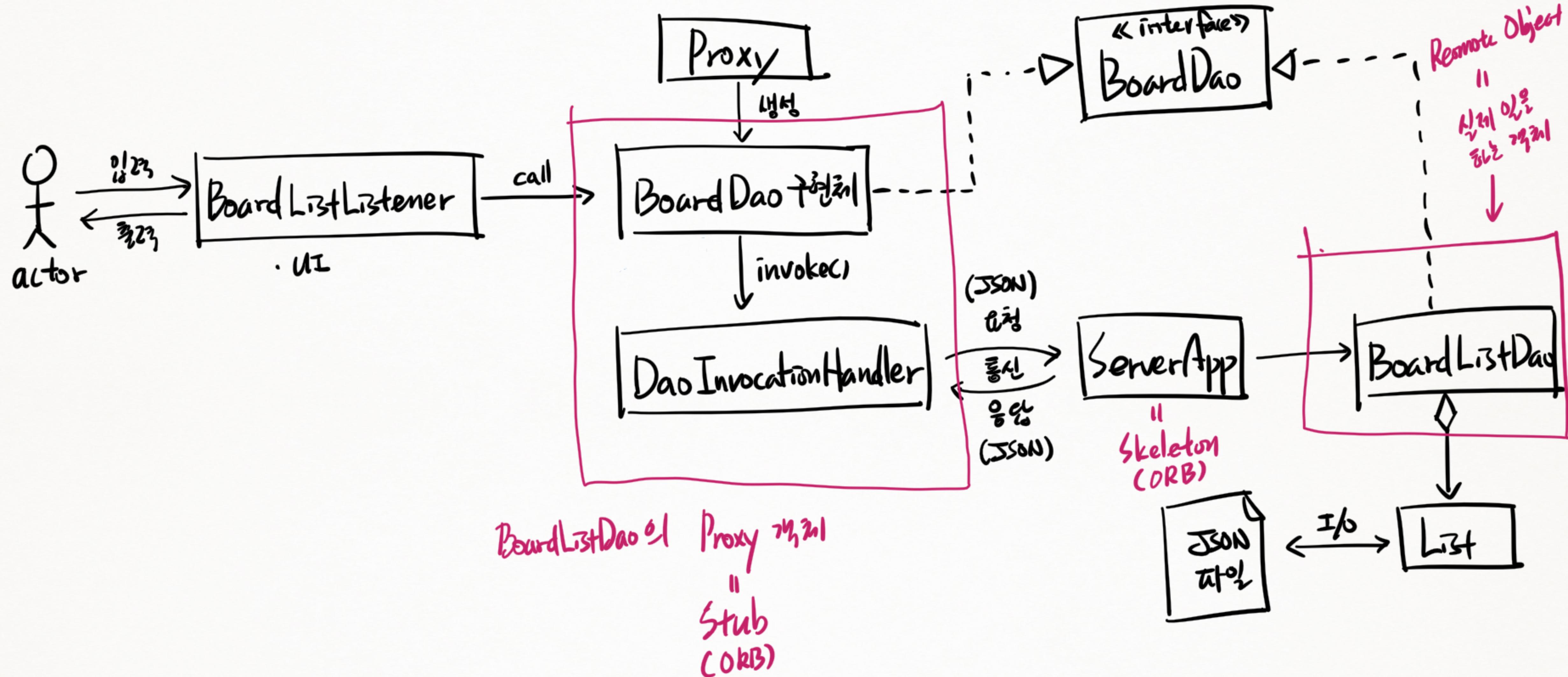
38. 프록시 객체 사용 예제 - ②



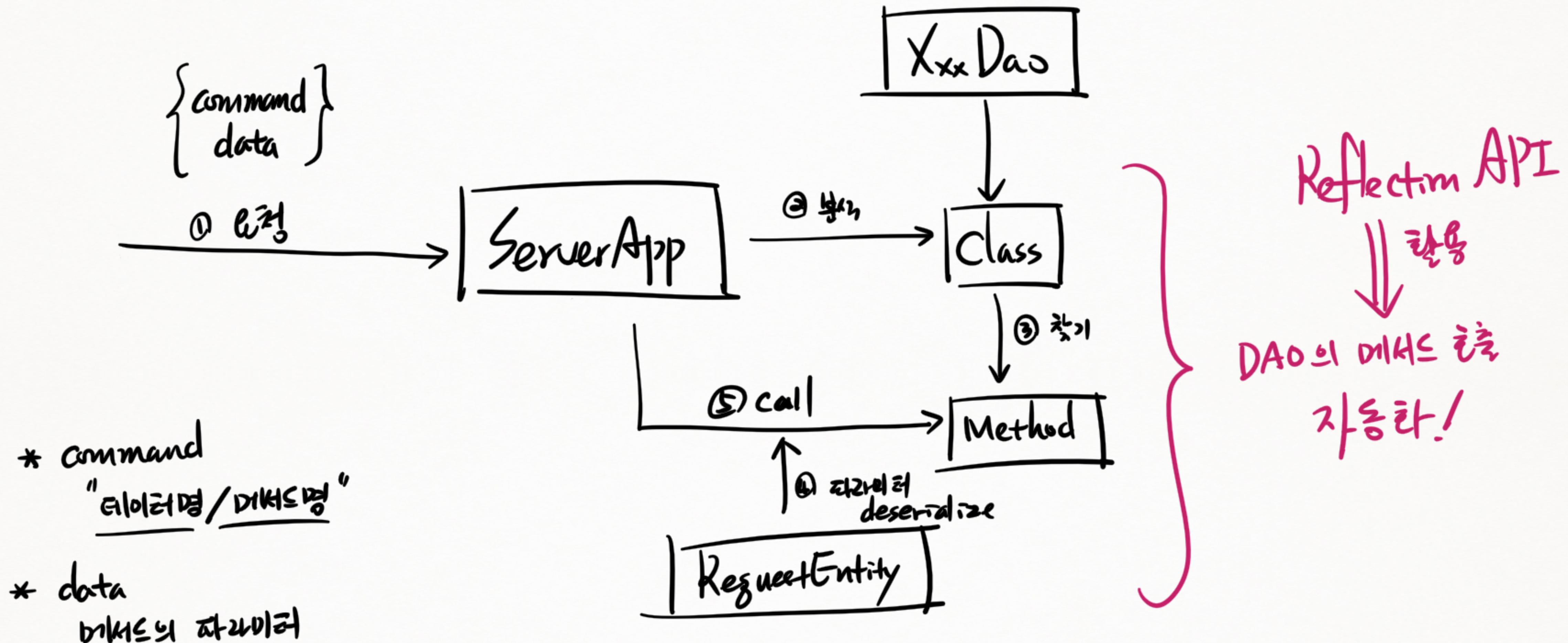
38. 프록시 패턴의 사용 예제 - Participants



38. 프록시 개념의 적용 예제 - Participants II



39. Reflection API를 사용하여 DAO 객체의 메서드 호출을 자동화하기



40. 예외 처리 예제

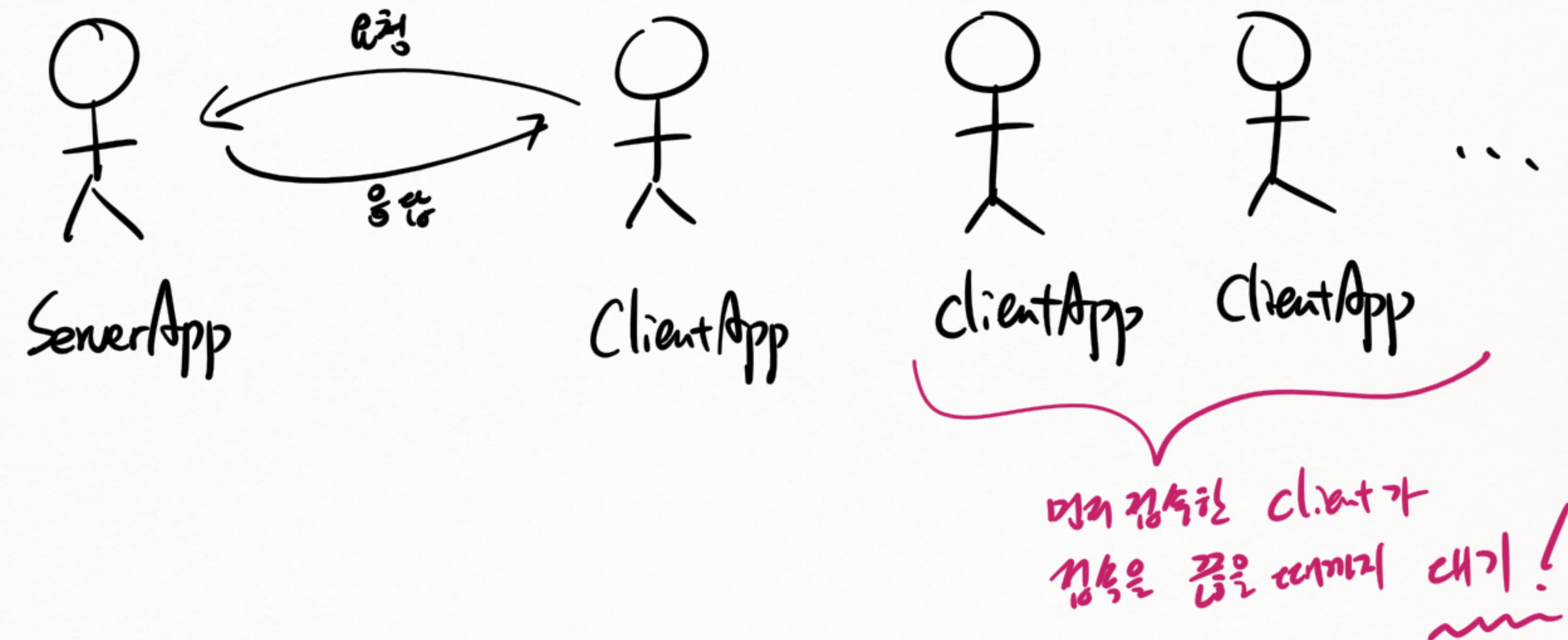
① client 편

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

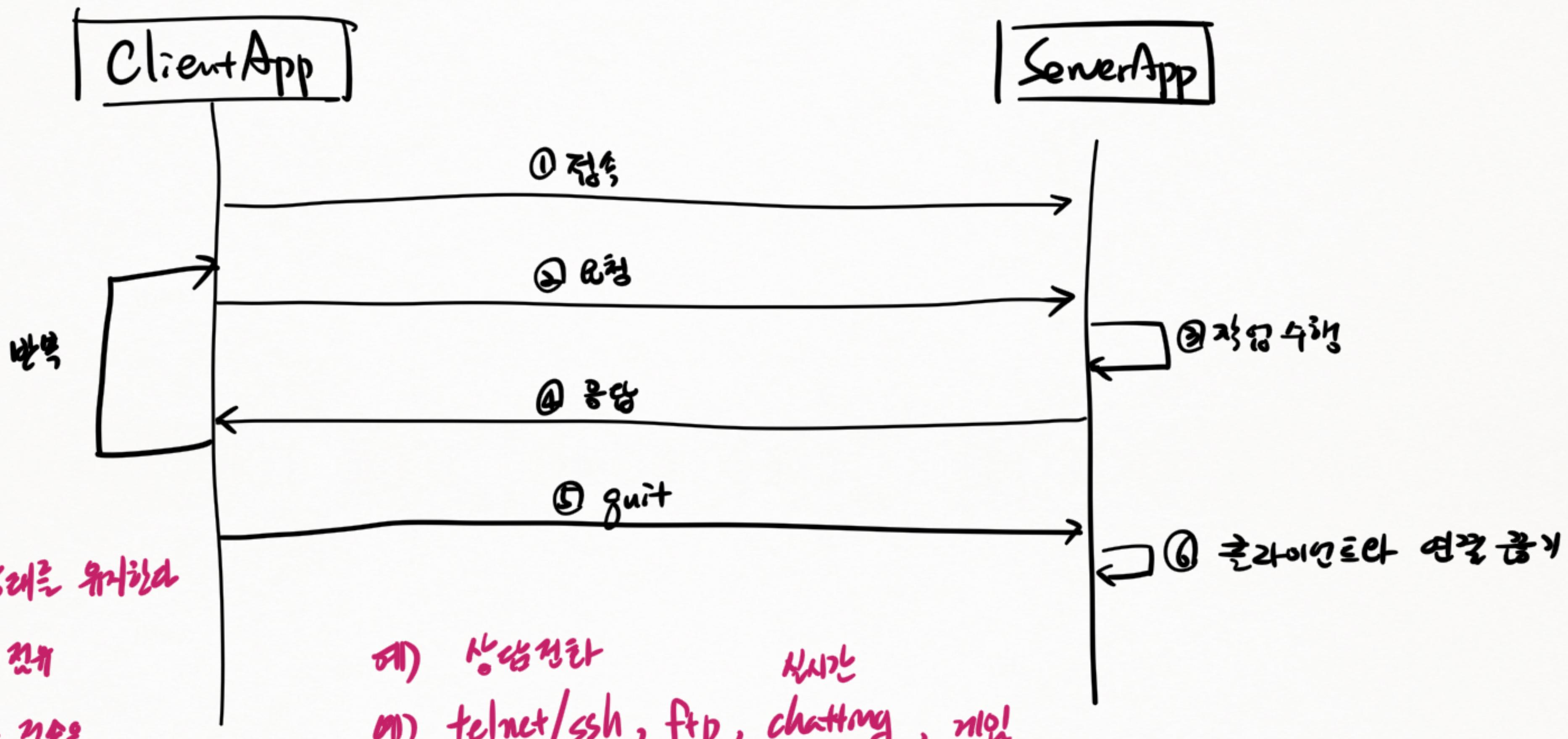
② server 편

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

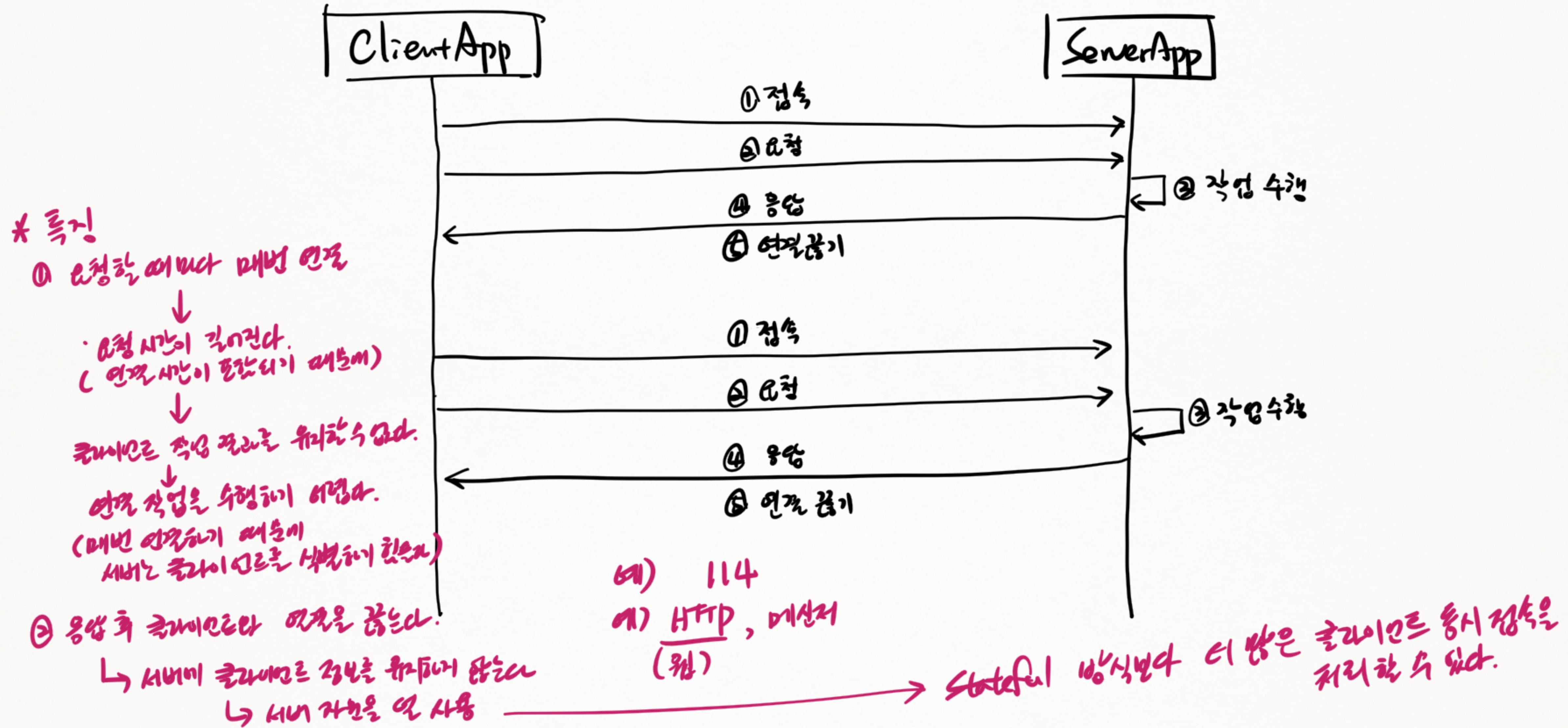
41. 예전 카카오로 페치를 스터디북으로 만들기 - Stateful 방식



41. 예제 클라이언트 페처를 소켓으로 처리하기 - Stateful 방식

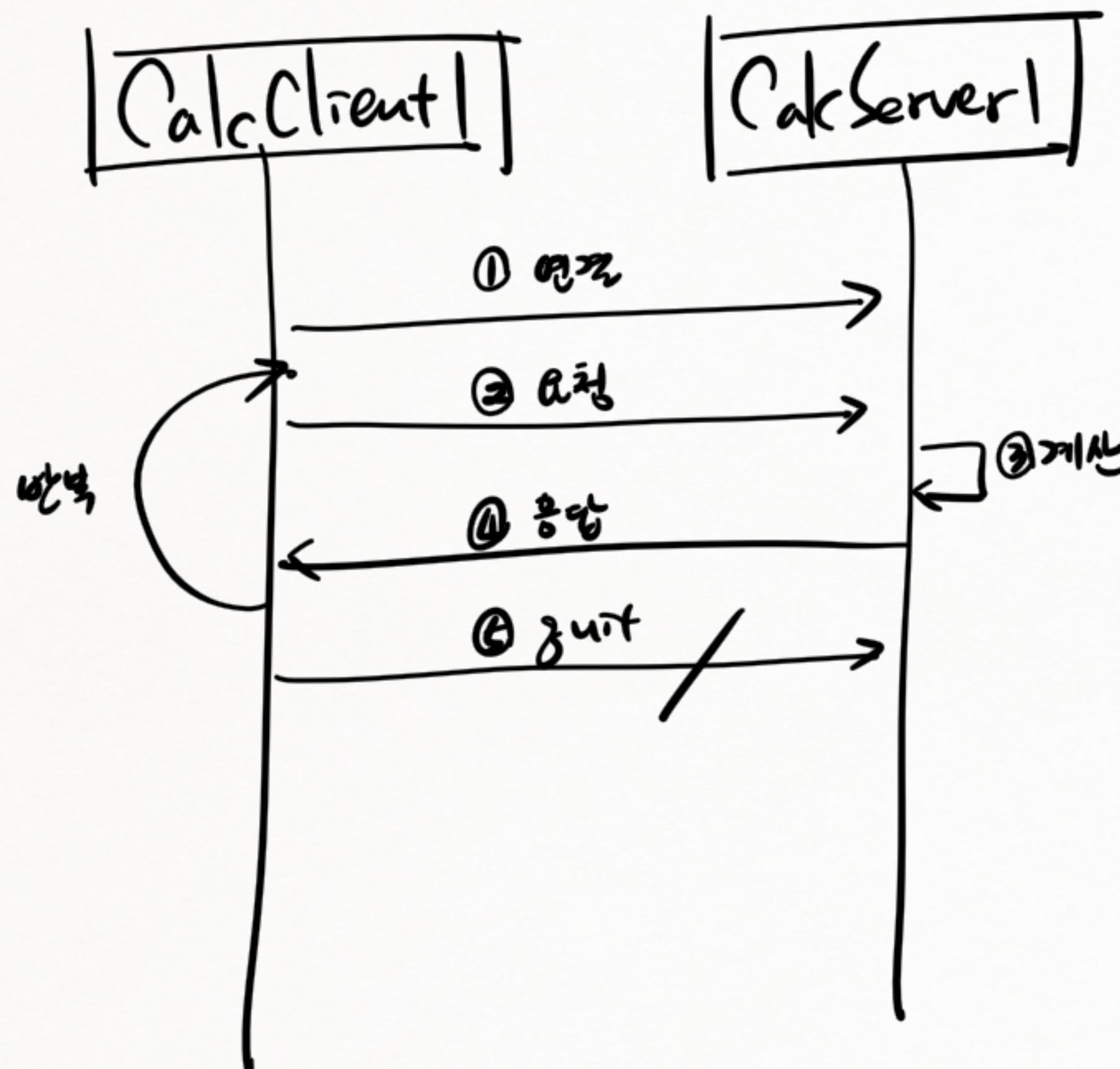


42. 예제 클라이언트 페처를 Stateless 방식

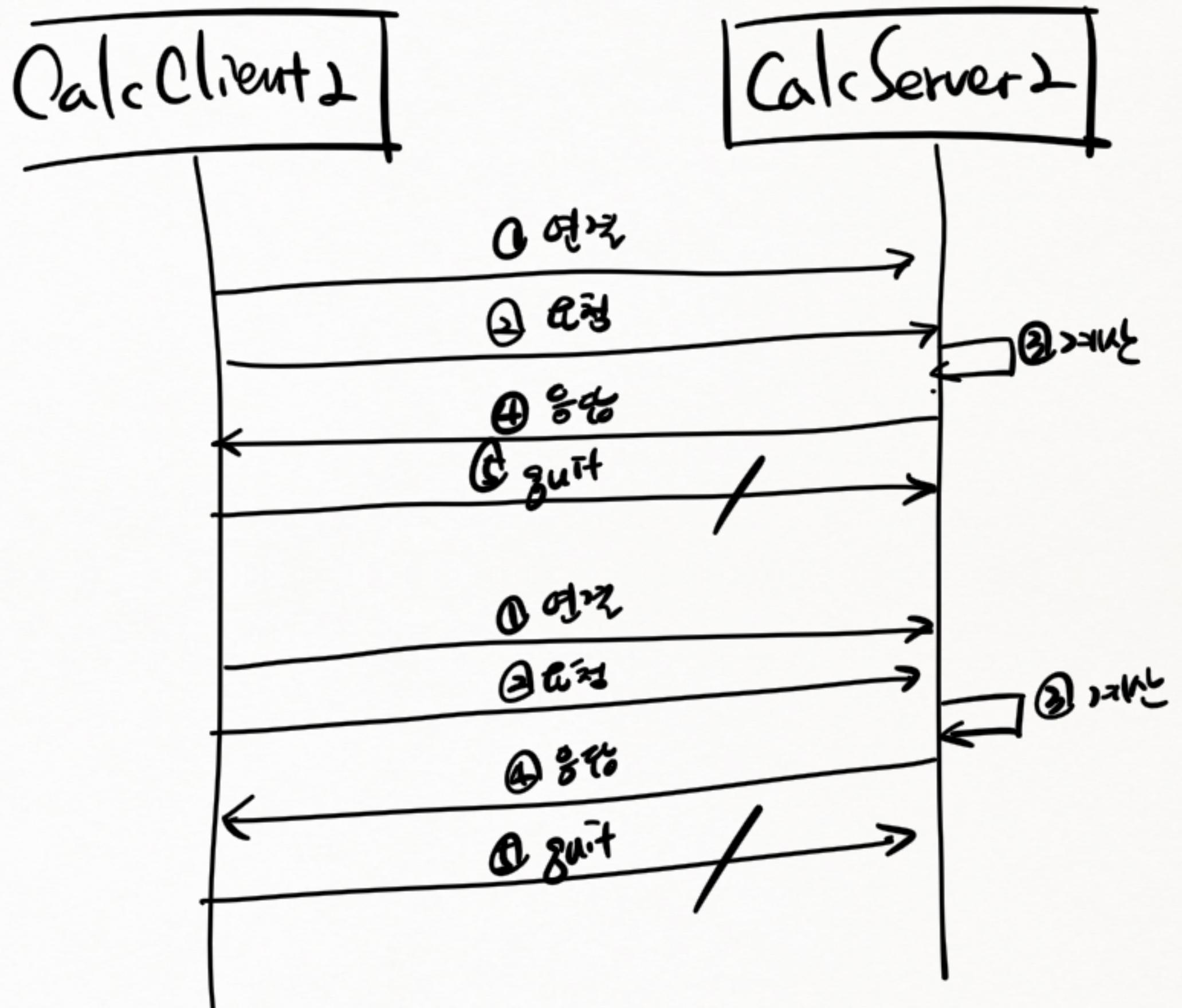


* Stateful vs Stateless

① Stateful

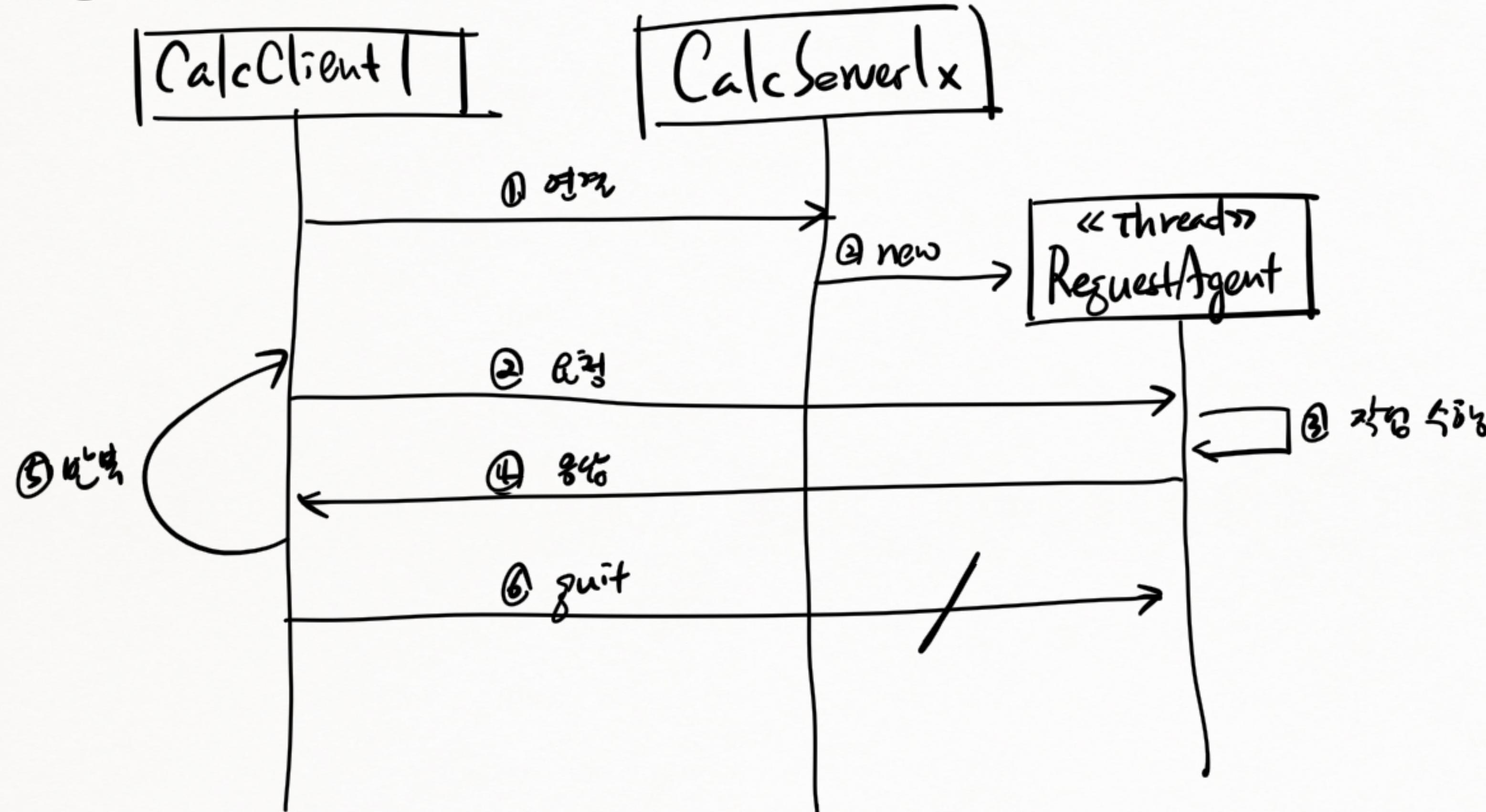


② Stateless



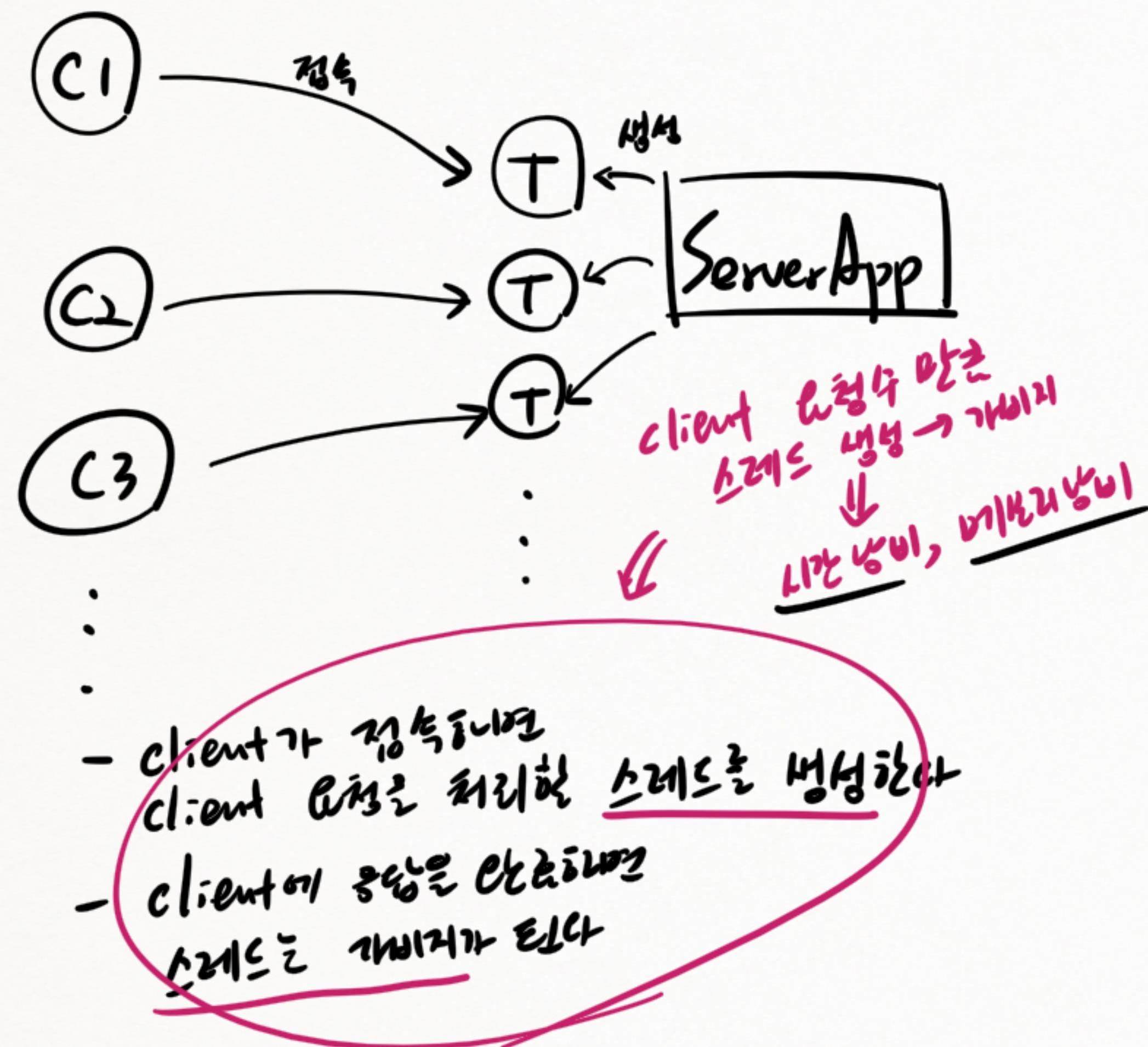
* Stateful vs Stateless

③ Stateful + Thread

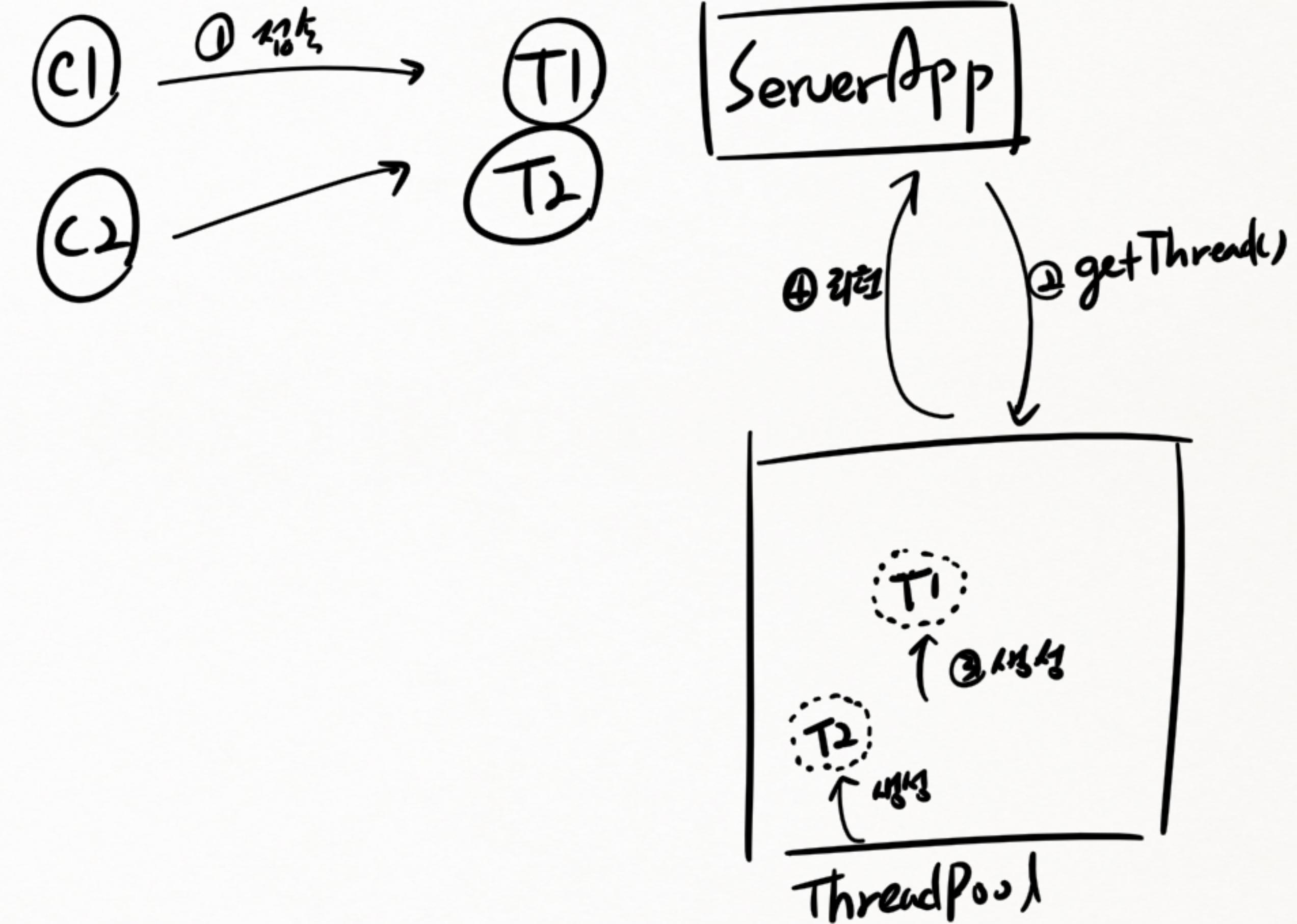


44. 스레드 재사용하기 : GOF의 Flyweight 패턴

① 스레드풀 적용 전

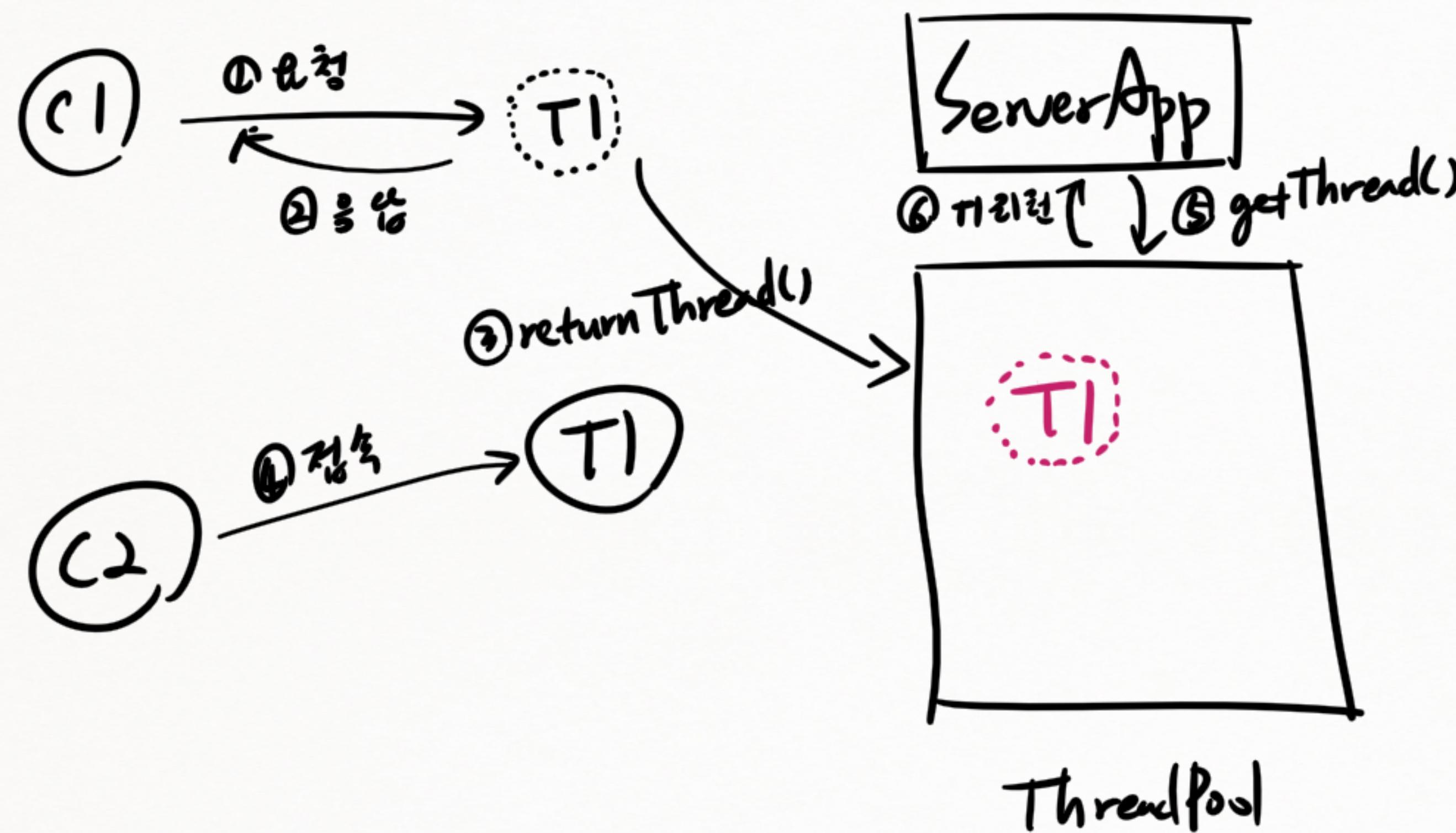


② 스레드풀 적용 후



44. 스레드 재사용하기 : GOF의 Flyweight 패턴

③ 스레드 재사용



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



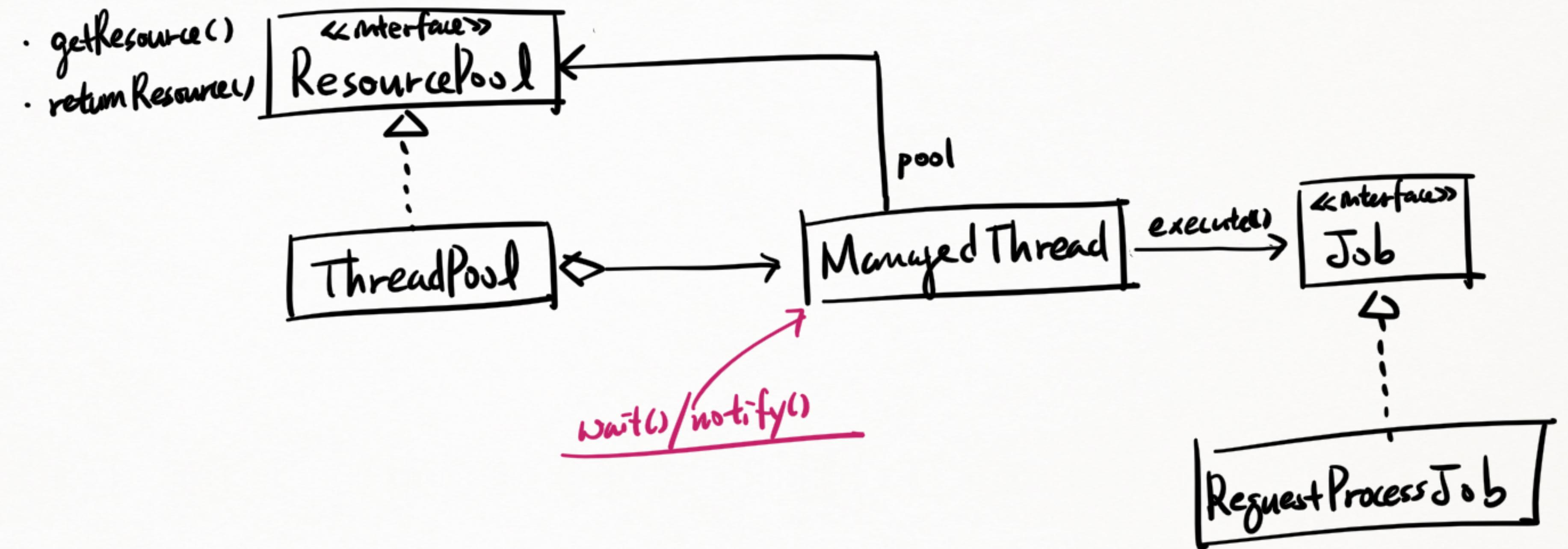
- 개체 사용후 가비지로 버리지 않고
 재활용성이 보완.
- 개체가 필요할 때,
 재활용이 필요한 개체를 미리 만들어
 "개체 재사용!"



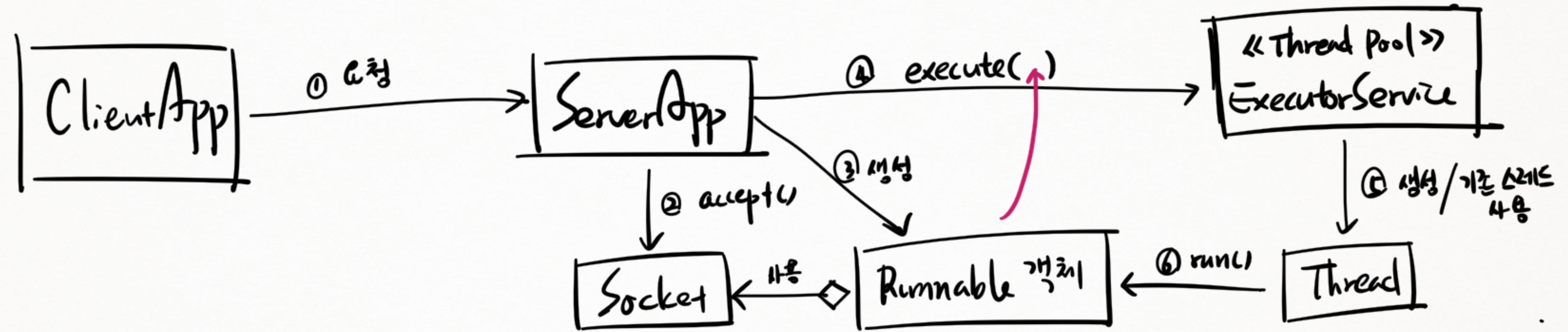
"Flyweight 패턴."

"Pooling 기법"

* Thread Pool Class Diagram

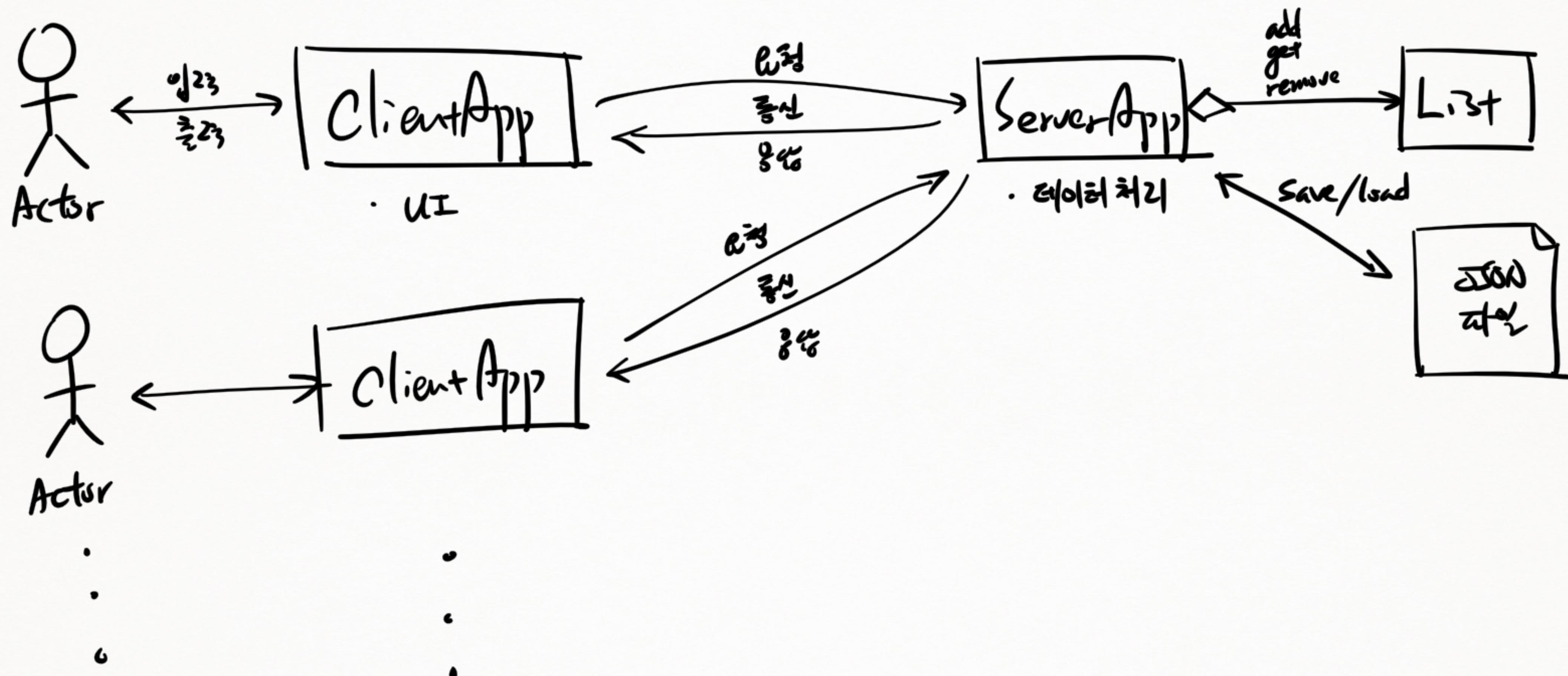


45.励志 스레드풀 구현하기 - Executors/ExecutorService 활용



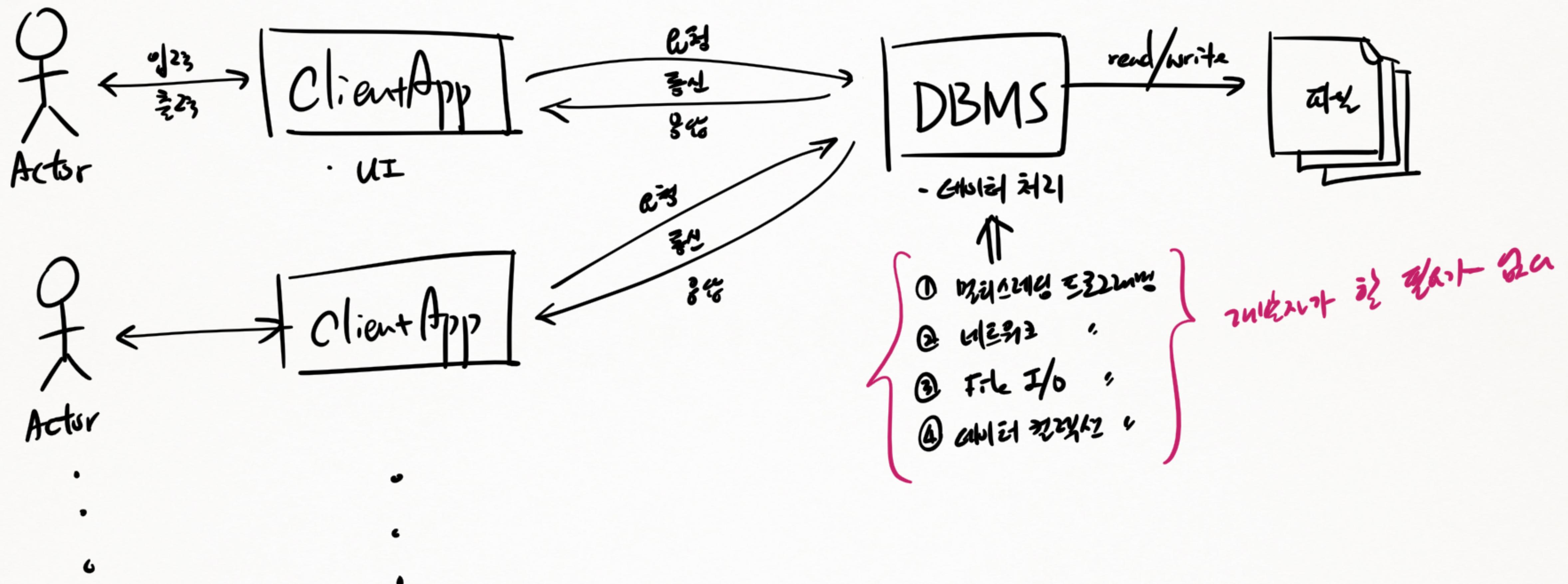
46. DBMS 도입하기

① 기본 구조

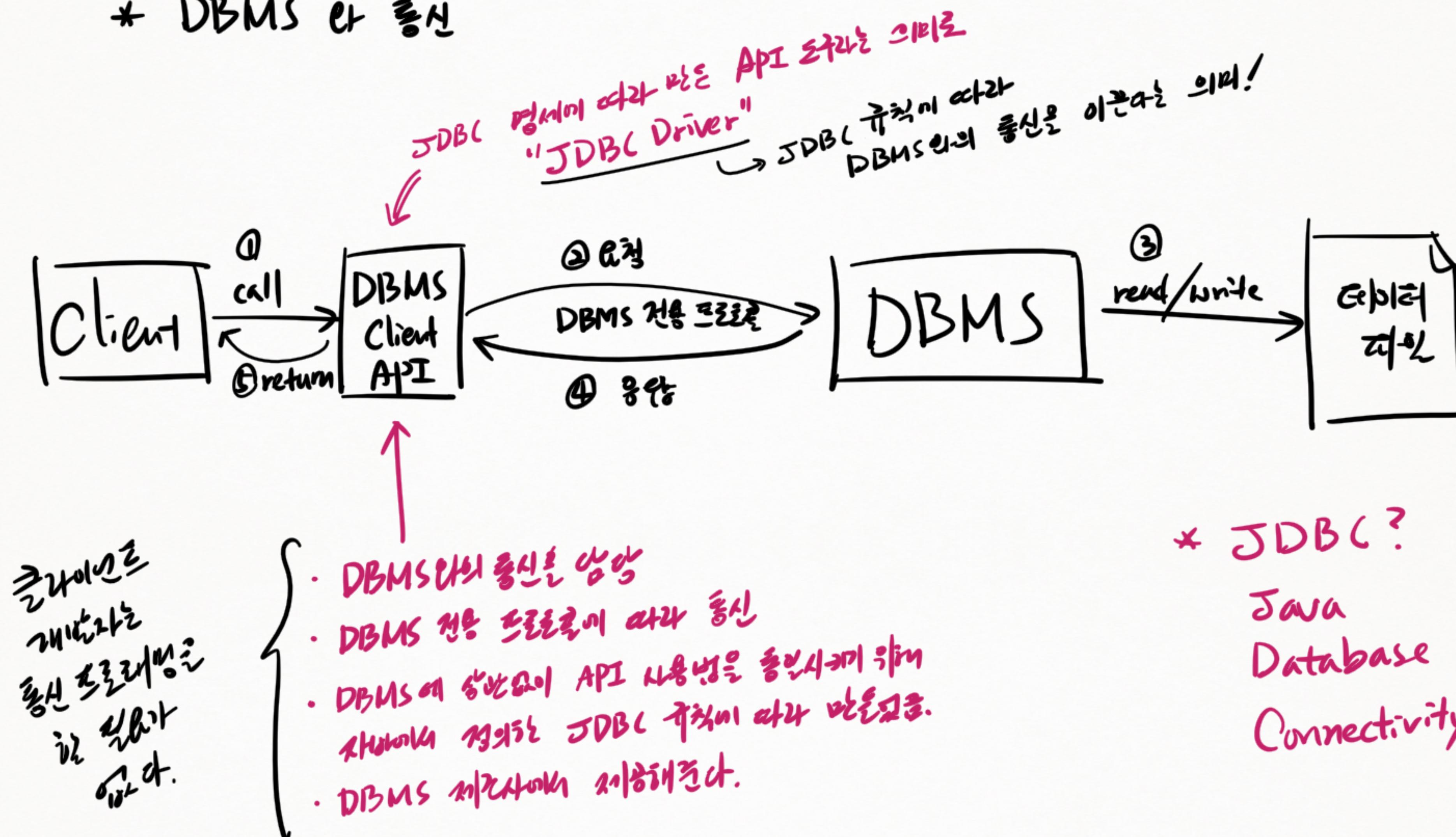


46. DBMS 도입하기

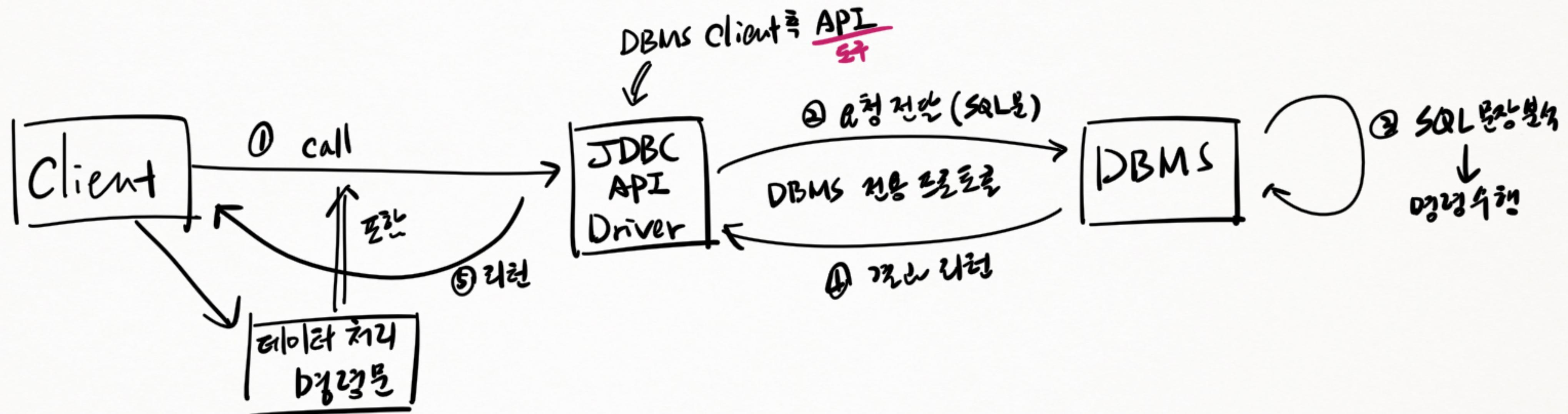
② 개발 단계



* DBMS 와 통신



* DBMS 에 데이터 처리를 요청하기



* SQL
 Structured
 Query
 Language

구조적인
 물체를 갖춘
 데이터 처리
 명령언어

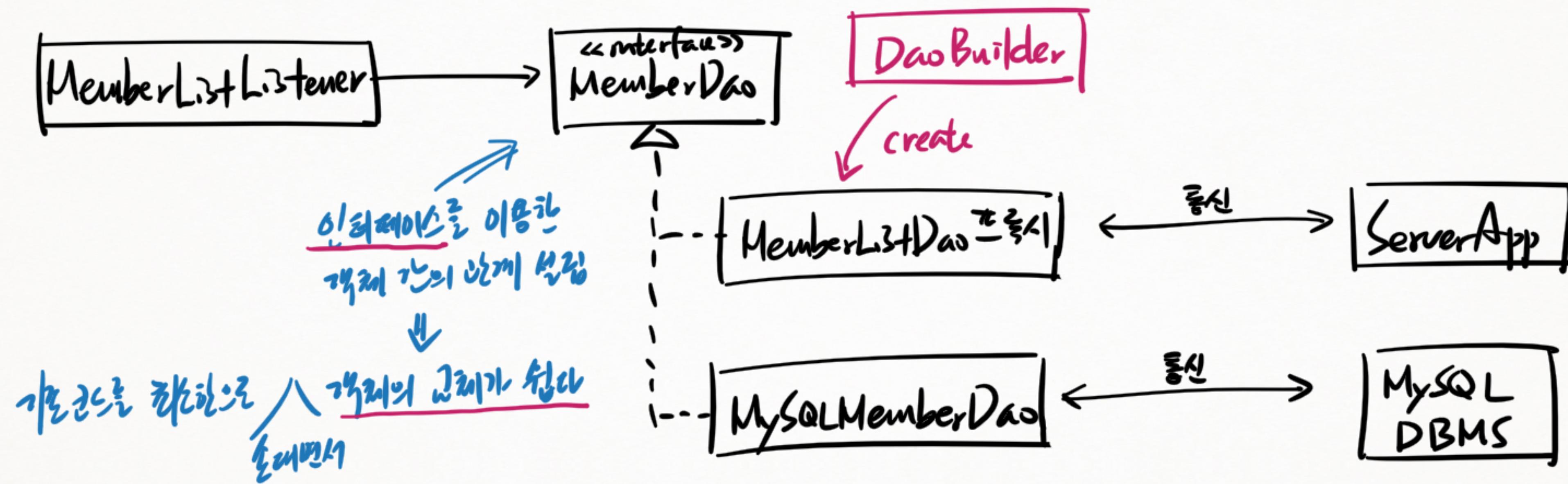
- 표준언어
 - DBMS 응용제작자.
 DBMS 와 상관없이
 사용할 수 있다.

SQL 문법에 맞춰
 작성한다.

→ 실수 :

준SQL + DBMS 적용 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 = '%s'

1111' or '1'=1'

"SQL 퀼 쿼리" " хр再也 퀼 쿼리 "

1810년 10월 20일
한국에서
한국에
한국에
한국에

SQL ပုဂ္ဂန်မှု
အသေးစိတ် ဖျက်ဆီး

120°

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

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);

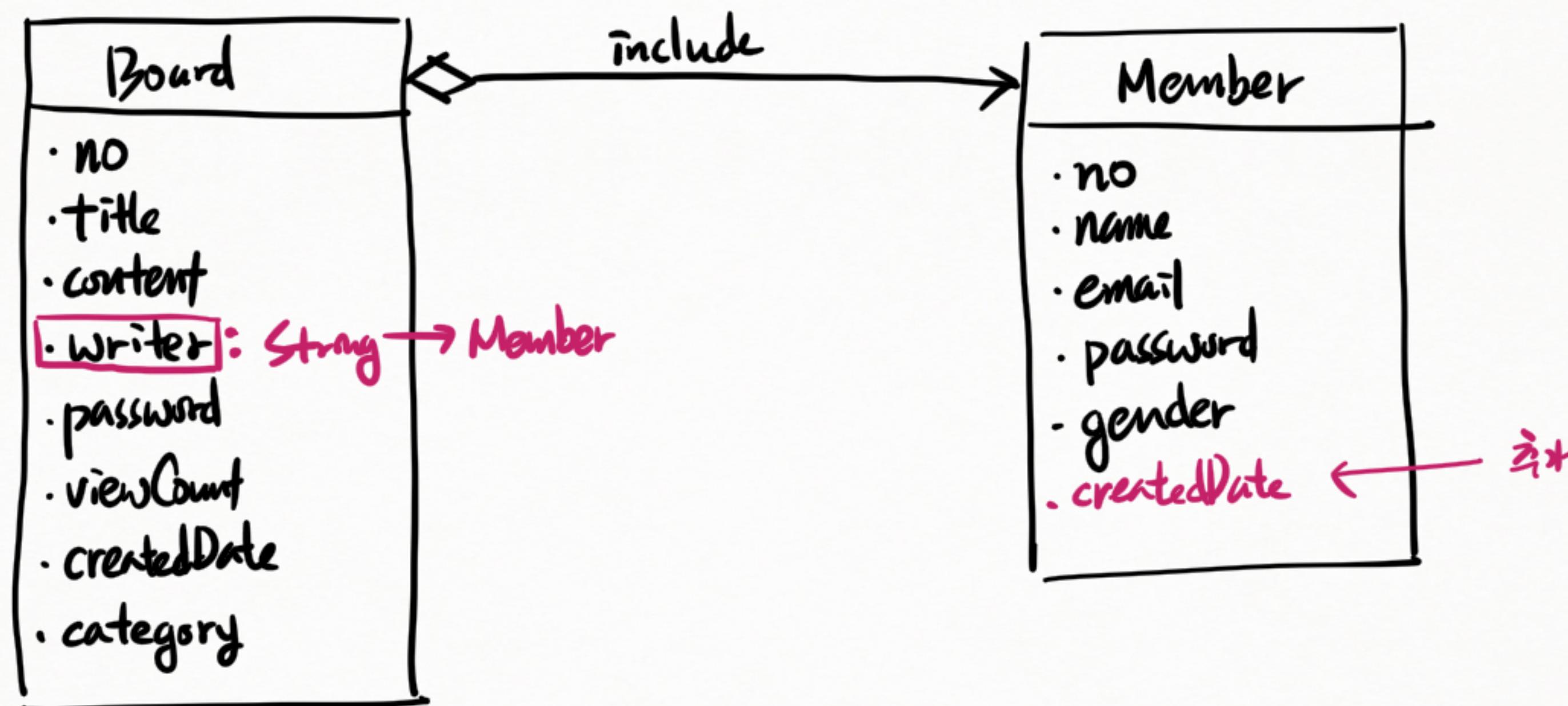
pstmt.set~~xxx~~(1, ?);

~~in type~~

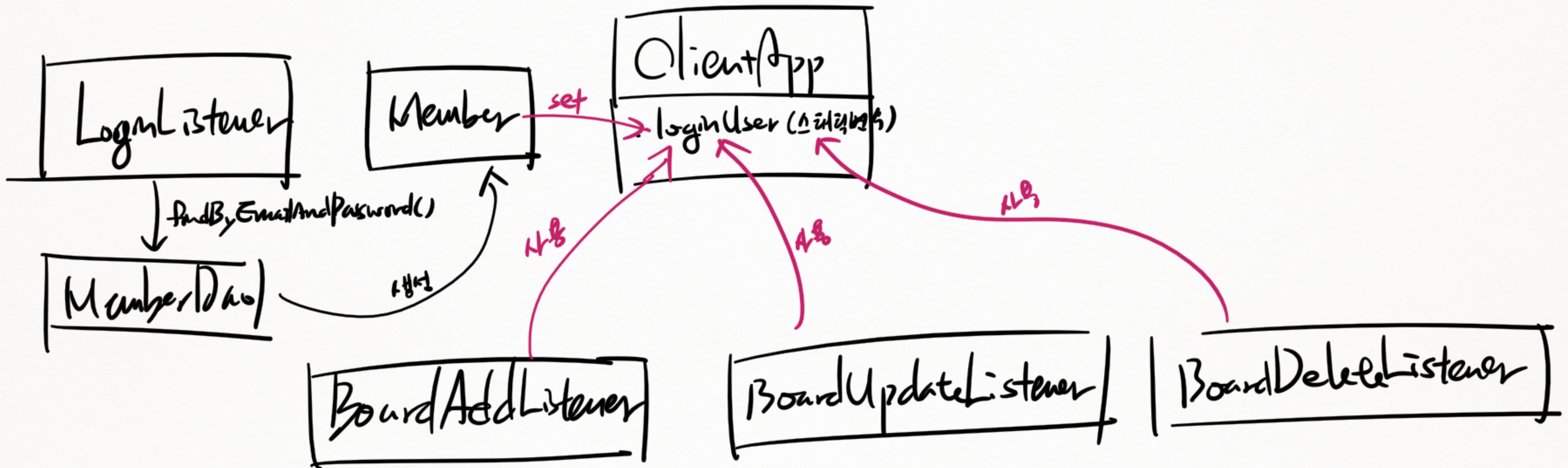
in-parameter인 줄 알았던

in-parameter인 줄 알았던

48. Entity Relationship

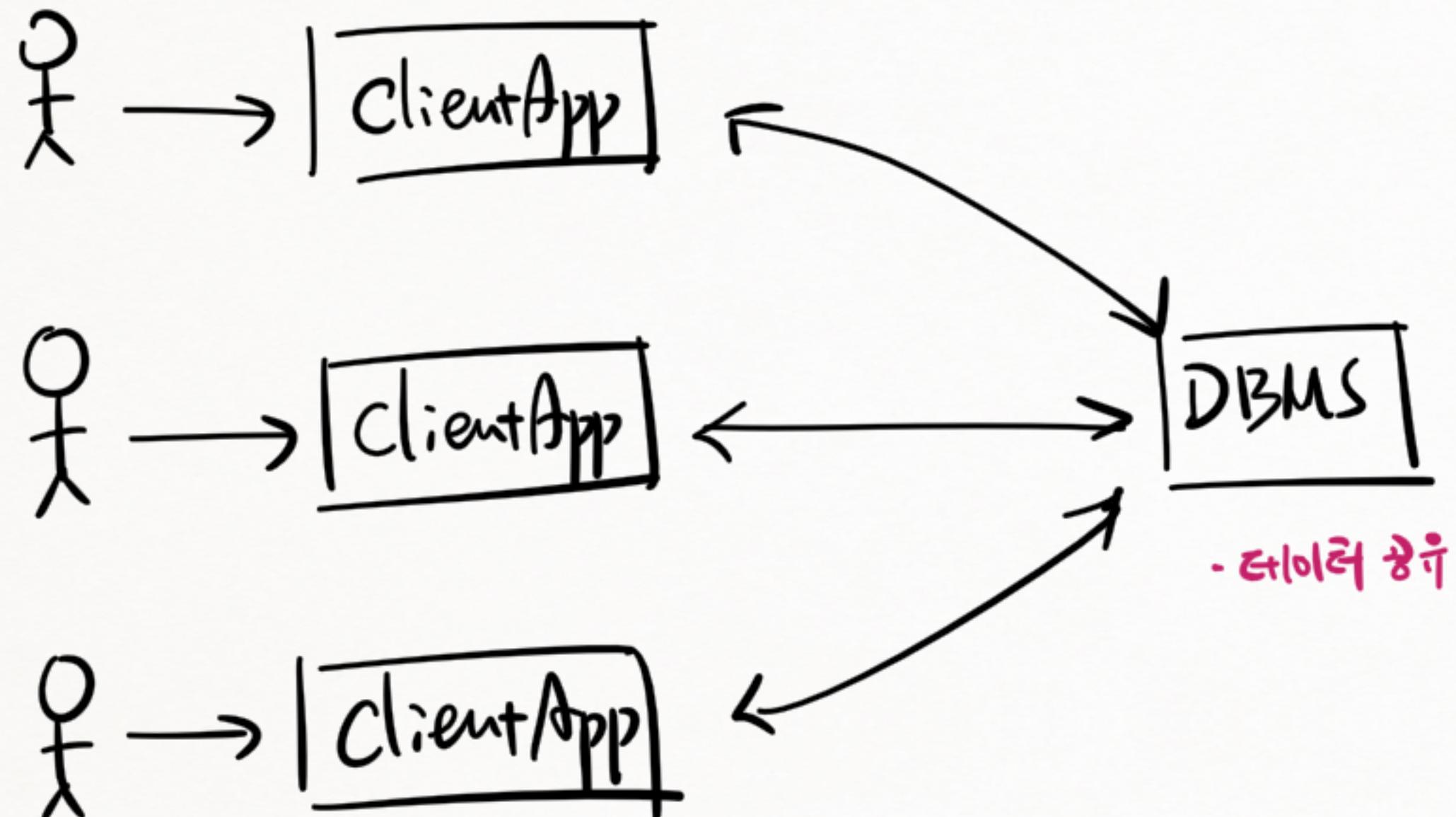


49. 로그인 흐름도



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 라이즈 사이클이 짧아짐

업무 조직의 변화가 빨아짐

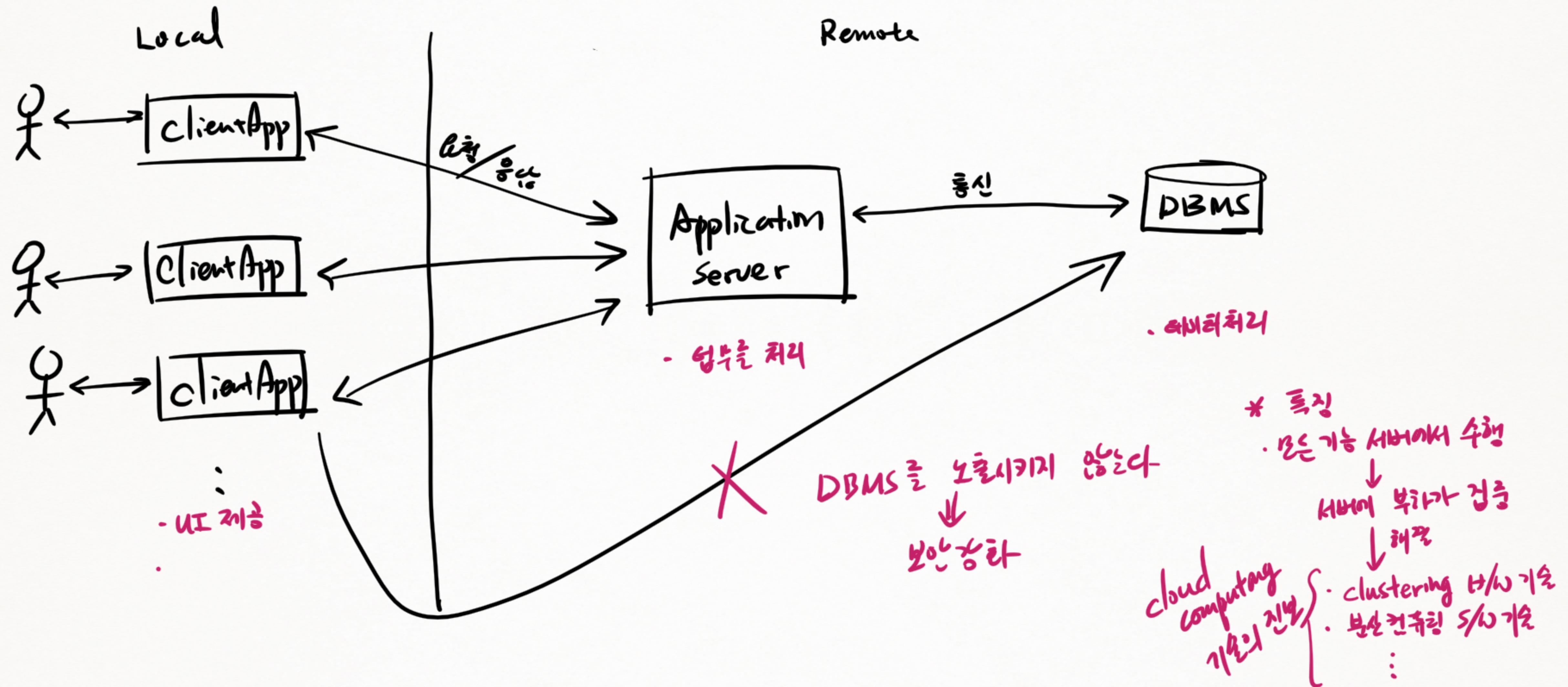
업무 변경이 빨아짐 → 업무 시스템 변화가 빨아짐

유지보수
비용 증가

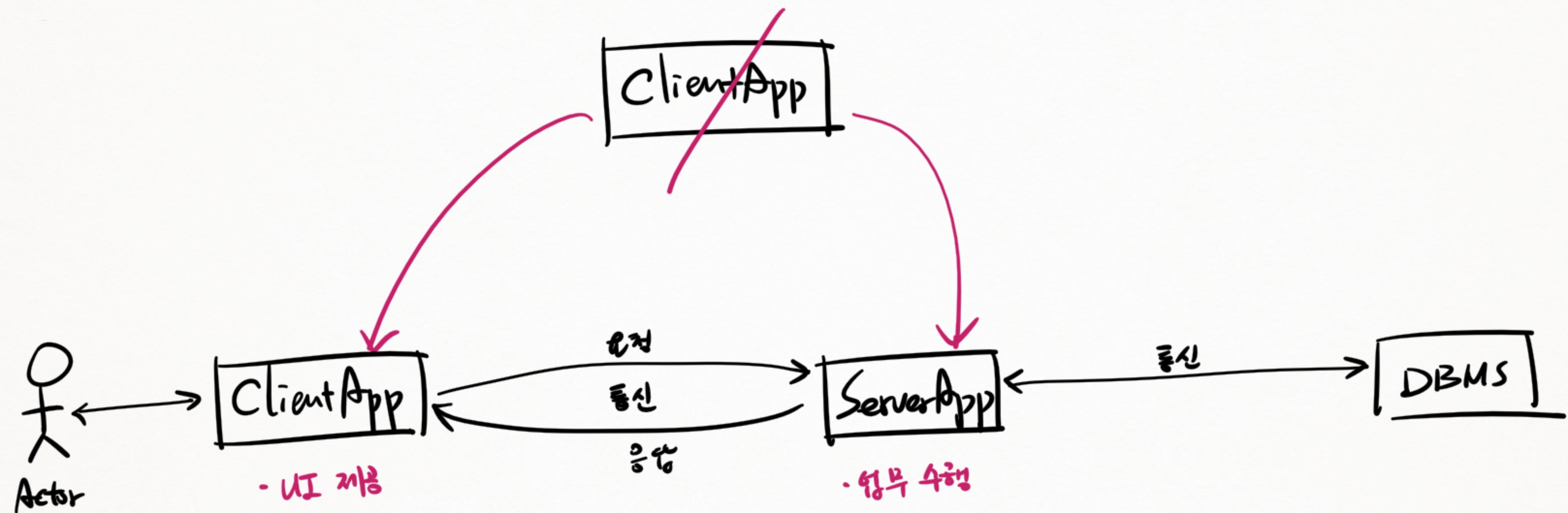
제작비가 빨아짐

50. Application Server Architecture

② Application Server Architecture



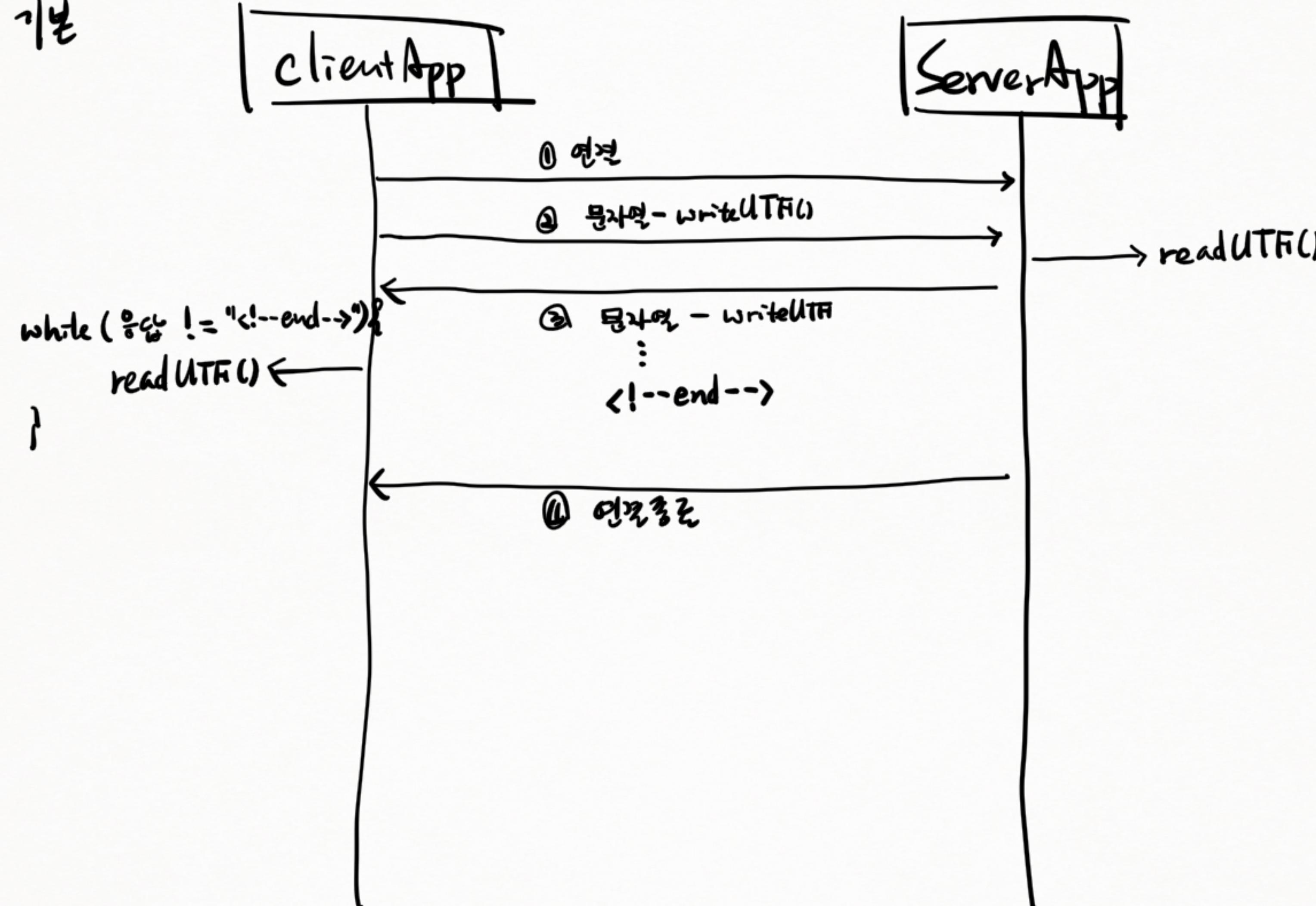
50. Application Server Architecture



50. Application Server Architecture - Protocol

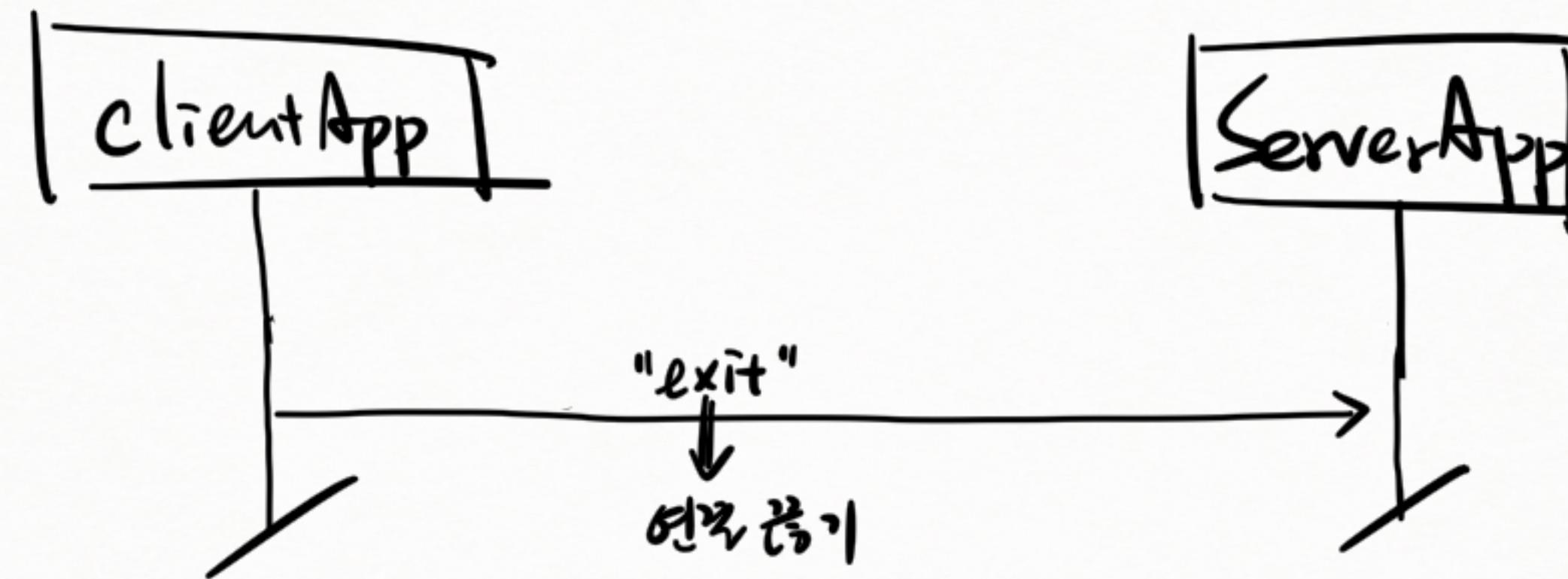
① 요청/응답 기본

Pseudo code
기호적 표현



50. Application Server Architecture - Protocol

② exit



③ prompt

