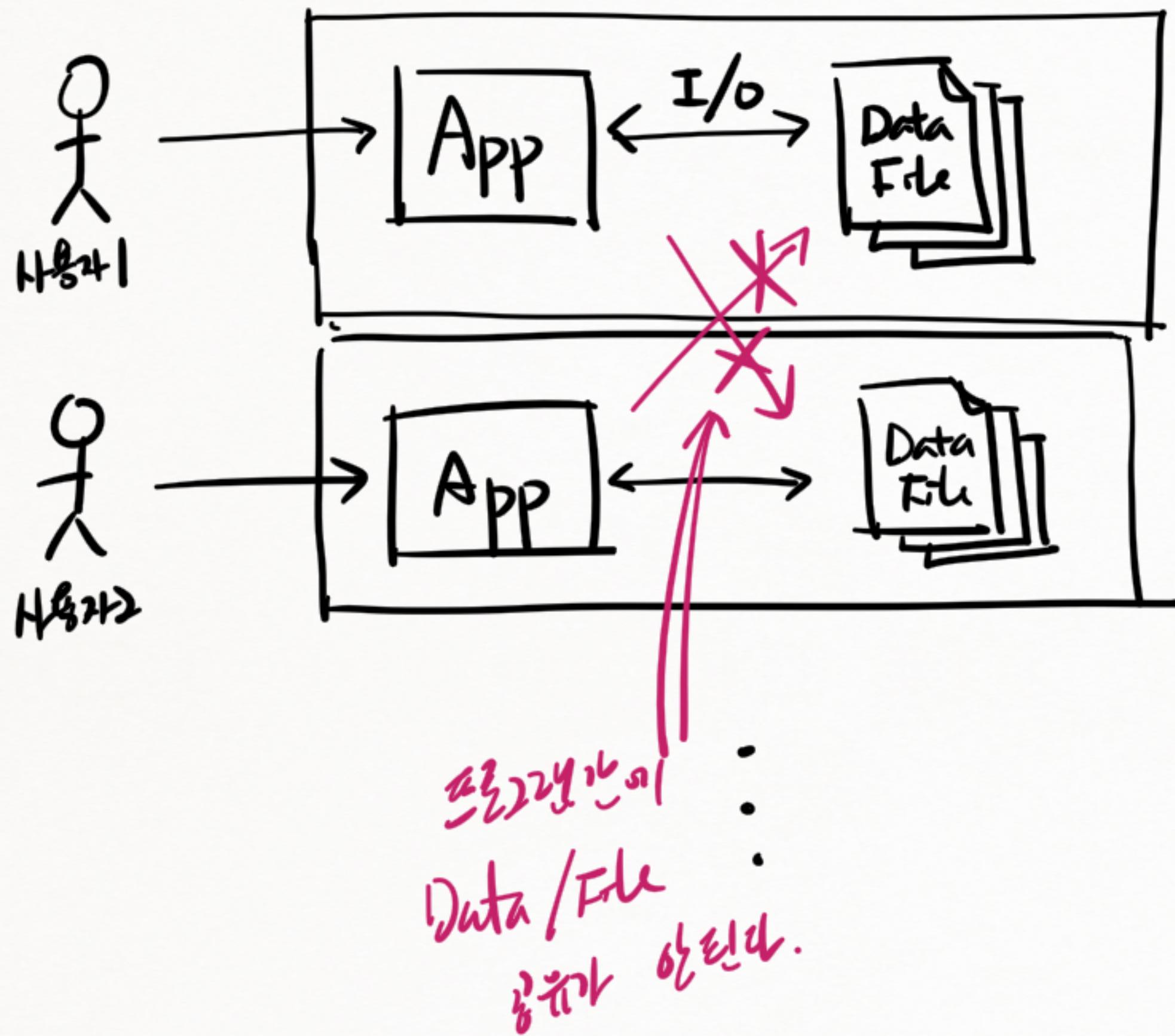


37. 바이오킹을 통한 데이터 공유

① 현황



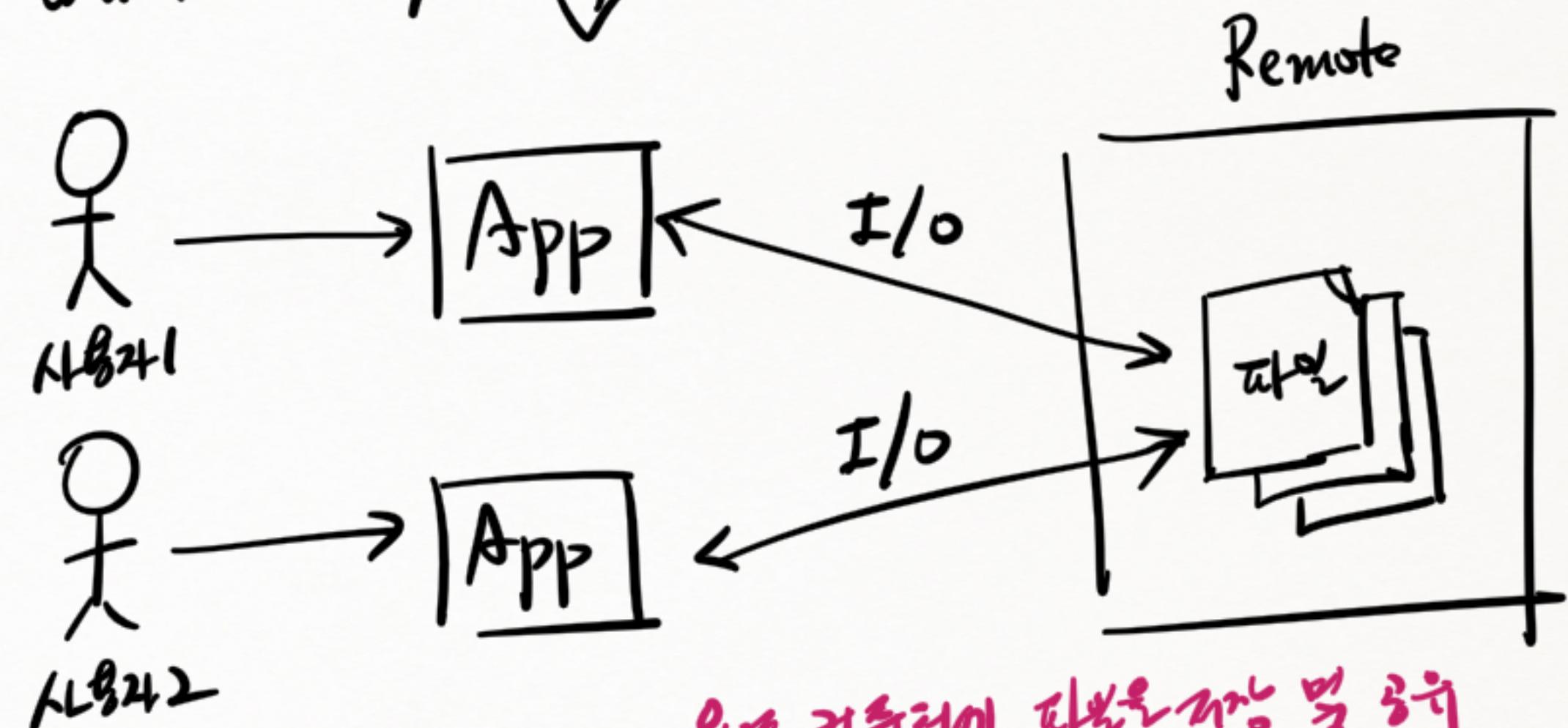
App 을 실행하는 컴퓨터

데이터 파일이 로컬에 저장된다.

↓

App 간에 데이터를 공유할 수 없다!

② 데이터 파일은
별도의 컴퓨터에 분리/공유 ↓ 가능할까?

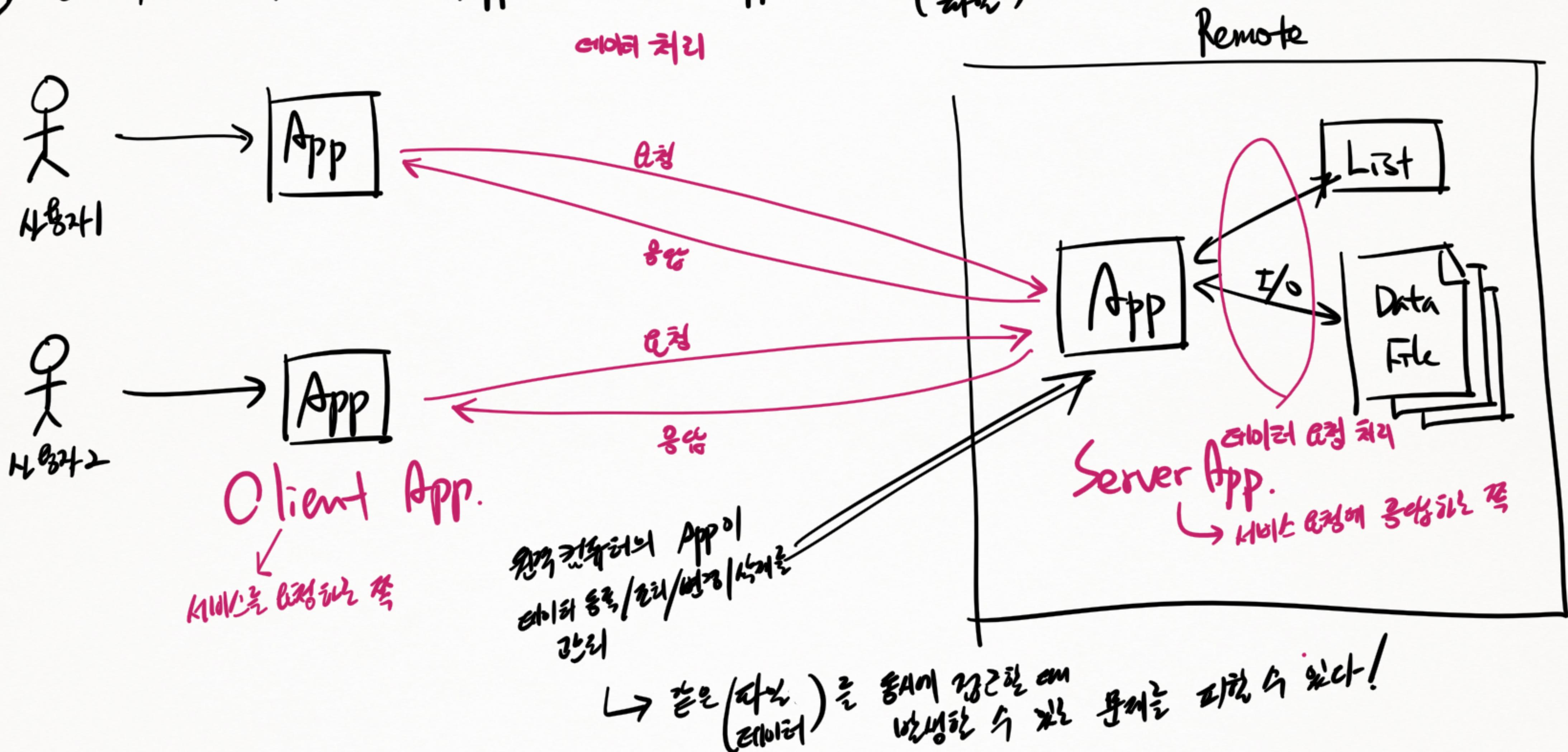


- 원격 컴퓨터에 파일을 저장 및 공유

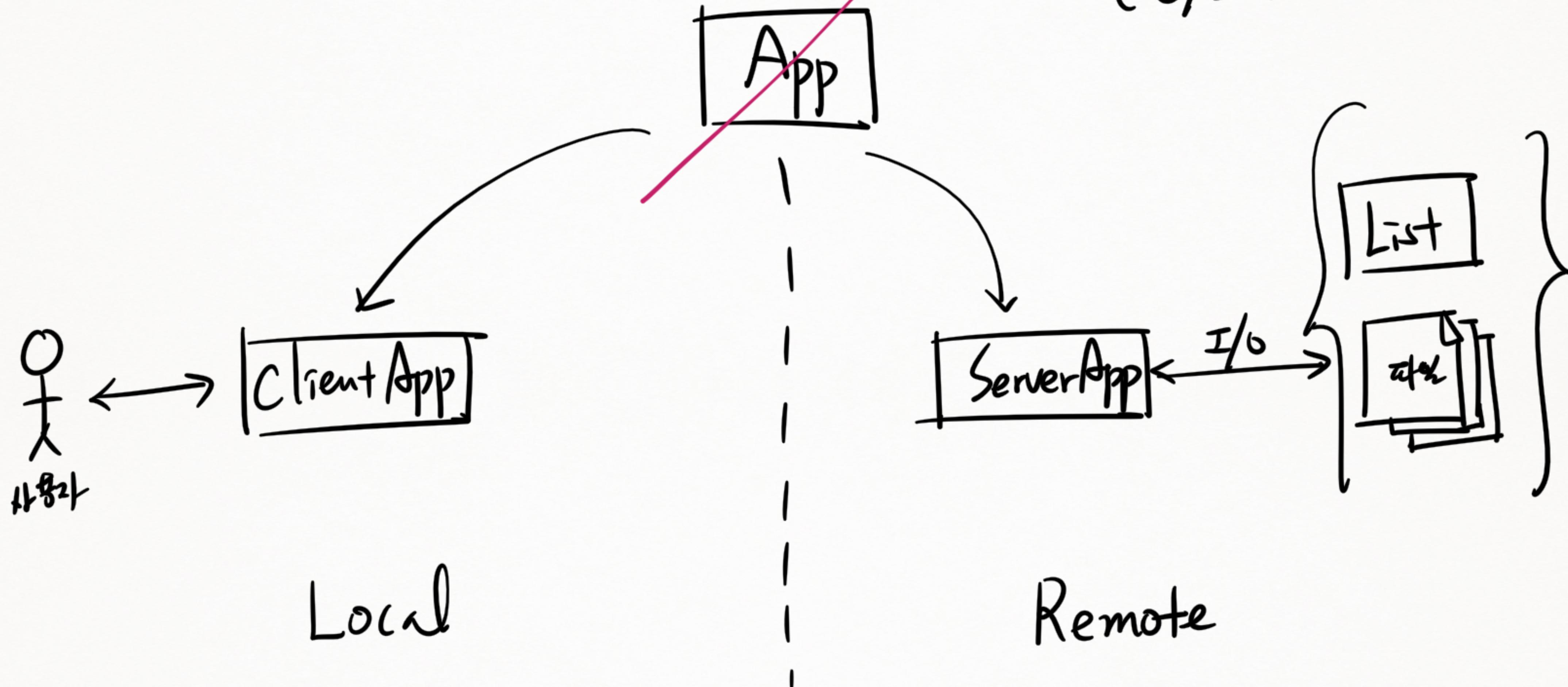
중재자

여기 App이 같은 파일을 편집하다 보면
각자의 데이터가 깨끗해졌다.

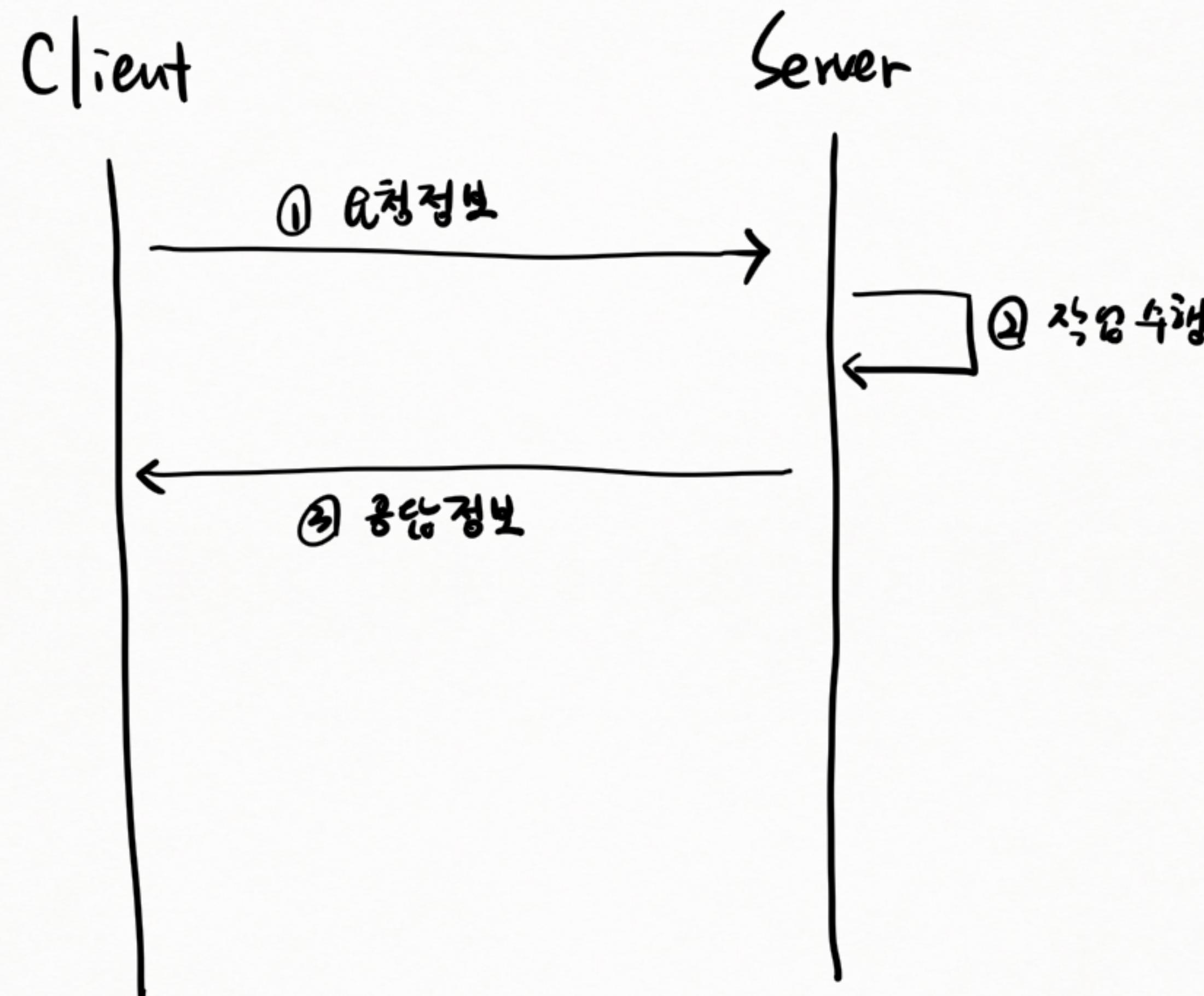
③ 데이터 관리 기능을 별도의 App. 으로 분리 - App이 직제 (데이터)
를 접근하는 것을 막는다.



* 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 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, ... }]"  
}
```

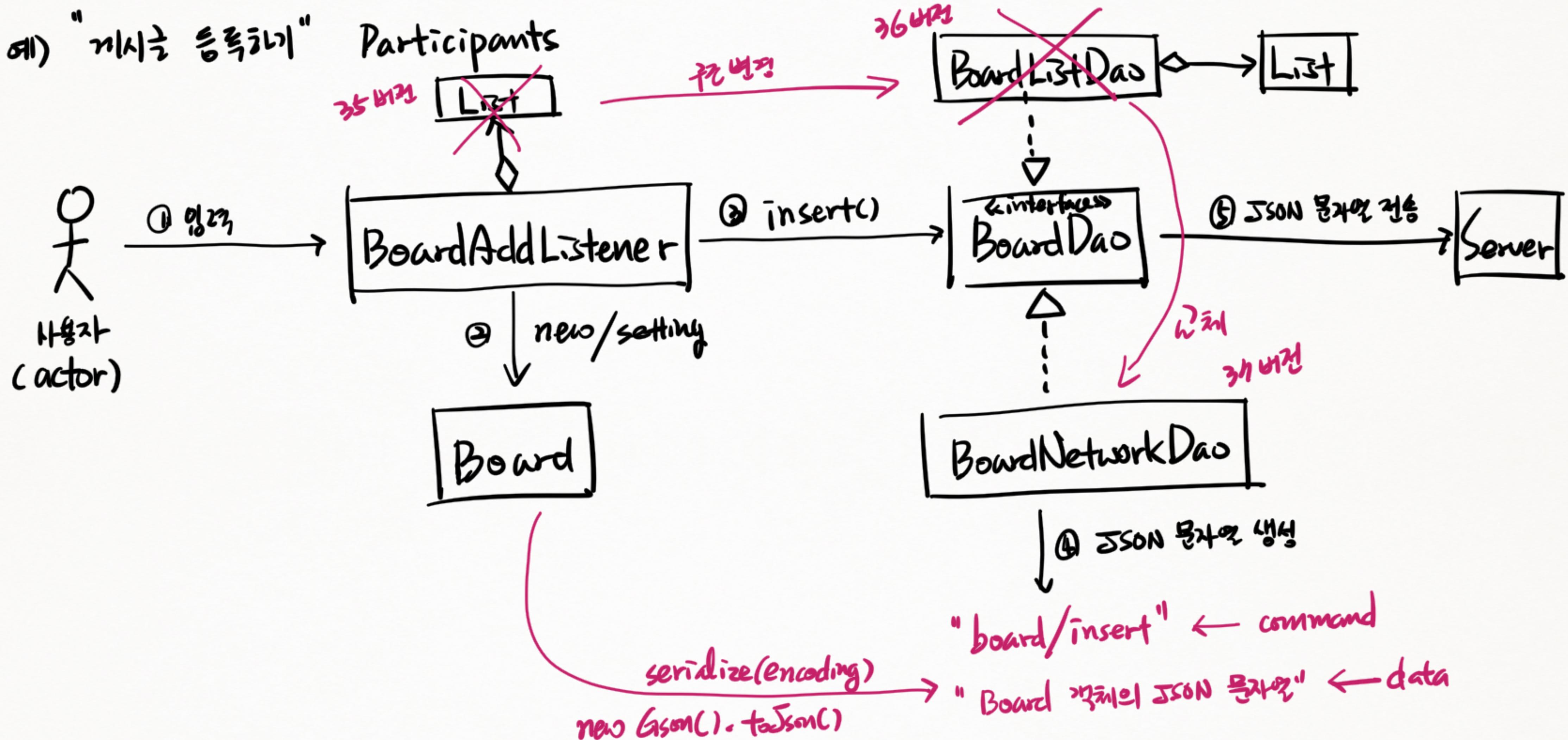
- * 실행 결과
"success": 성공
"failure": 실패

↓ 실패 예

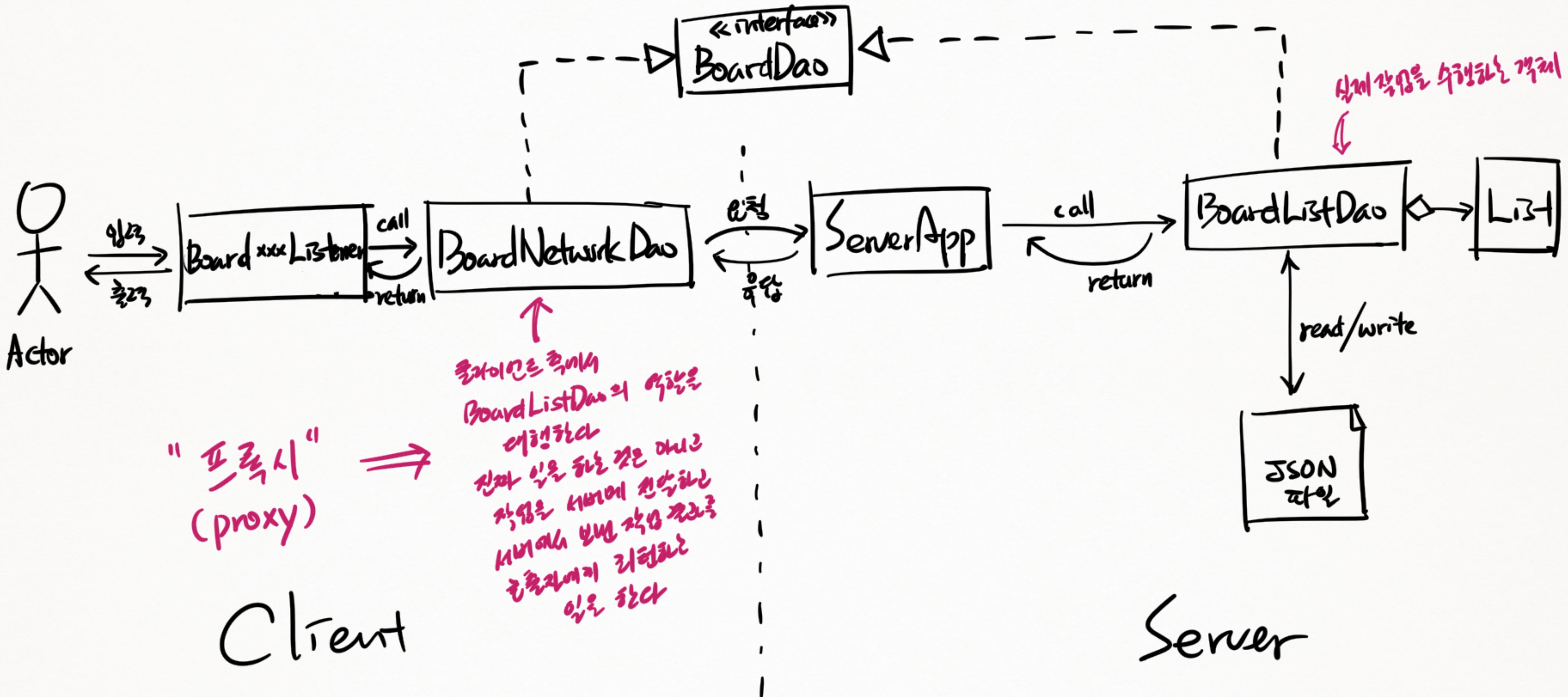
```
{  
    "status": "failure",  
    "result": "해당 번호의 데이터가 없습니다!"  
}
```

* 요청 정보 뷰티가 작업에 참여하는 객체들과 실행흐름

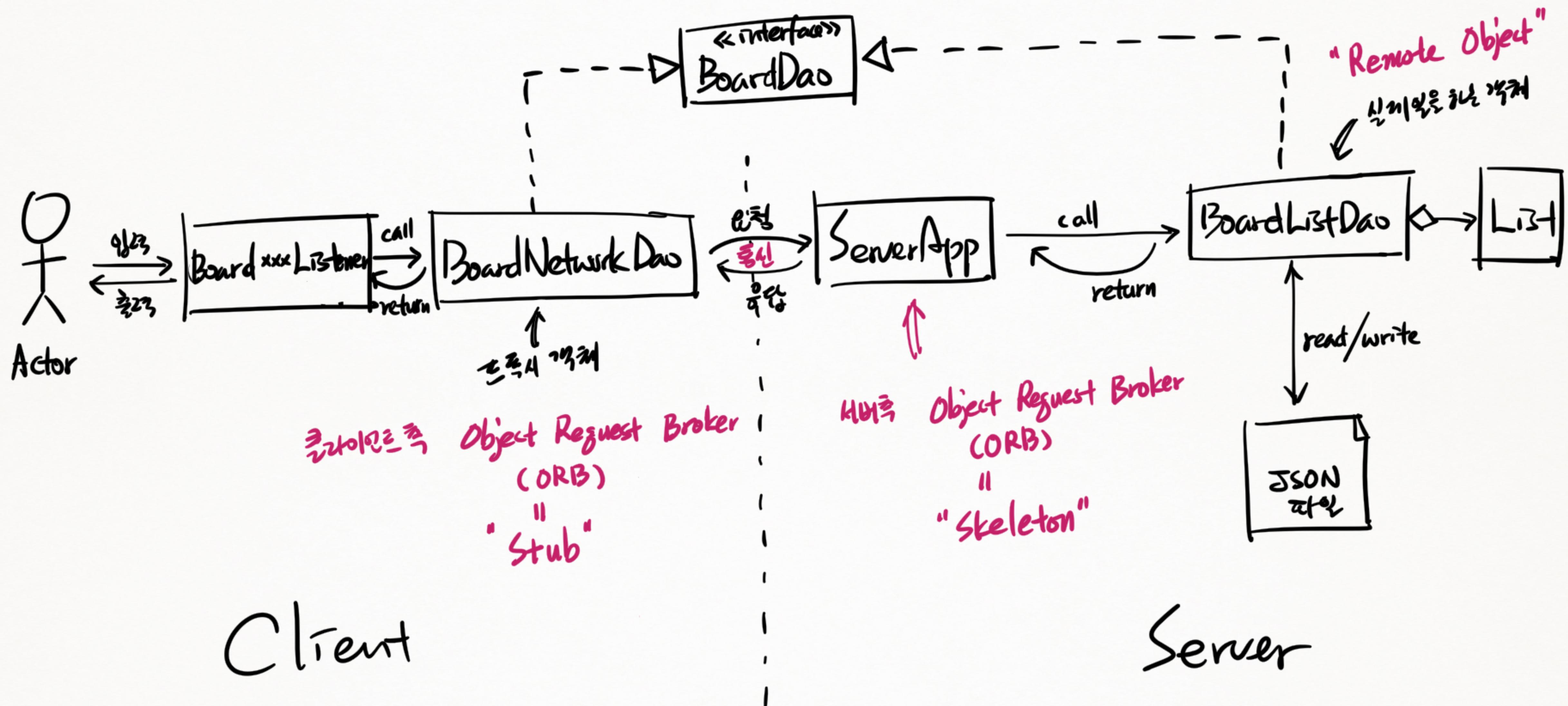
예) "게시글 등록하기" Participants



* DAO 와 Proxy 패턴 (GoF)

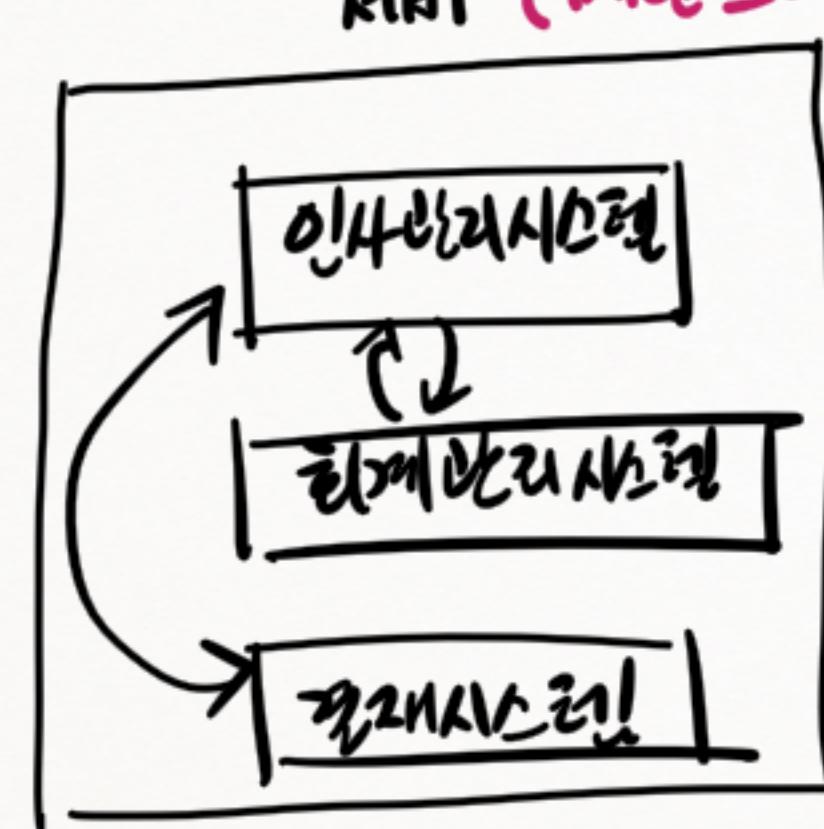


* DAO 와 Proxy 패턴 (GoF) ↗ 예술자의 명칭



* 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

↳

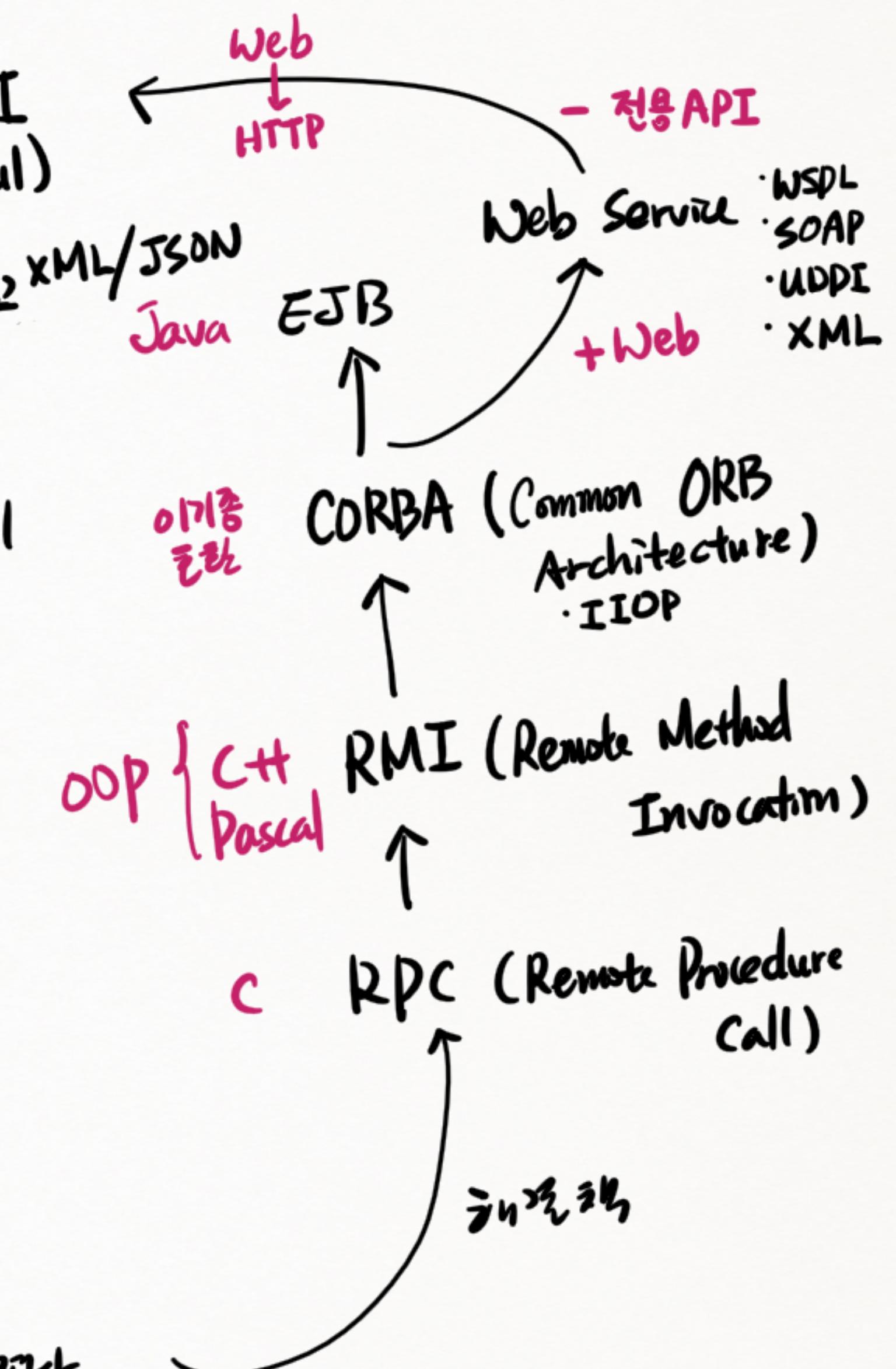
(원격스테이션) 터미널 3

인사관리 시스템

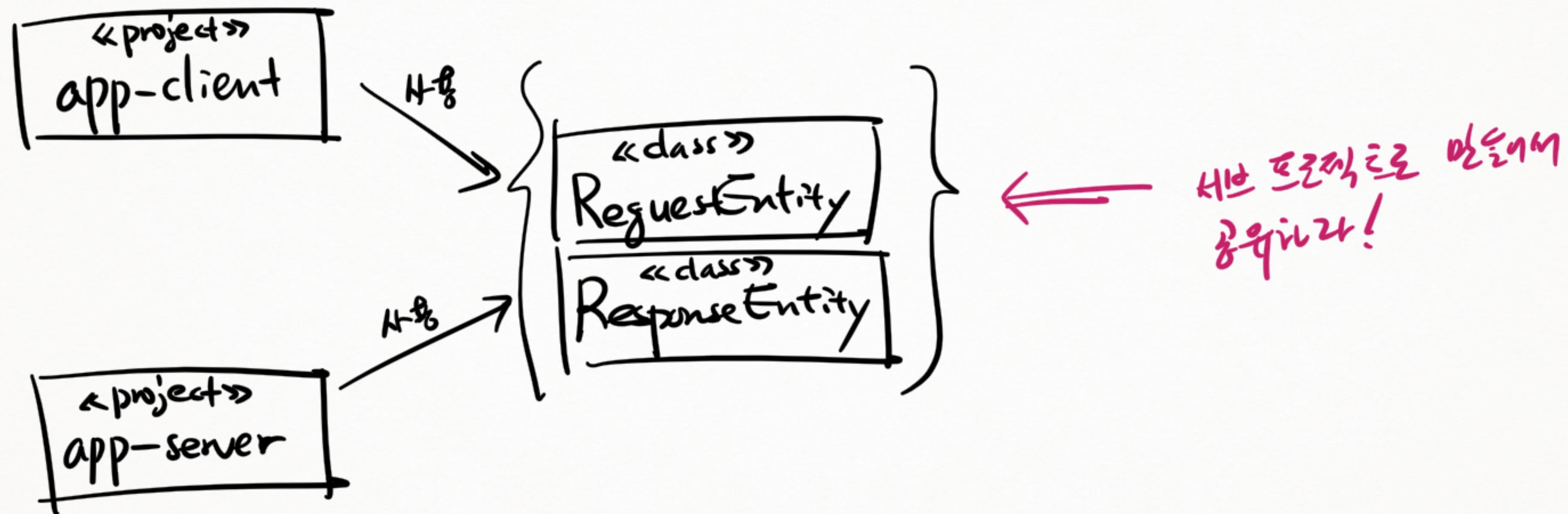
회계관리 시스템

회계 시스템

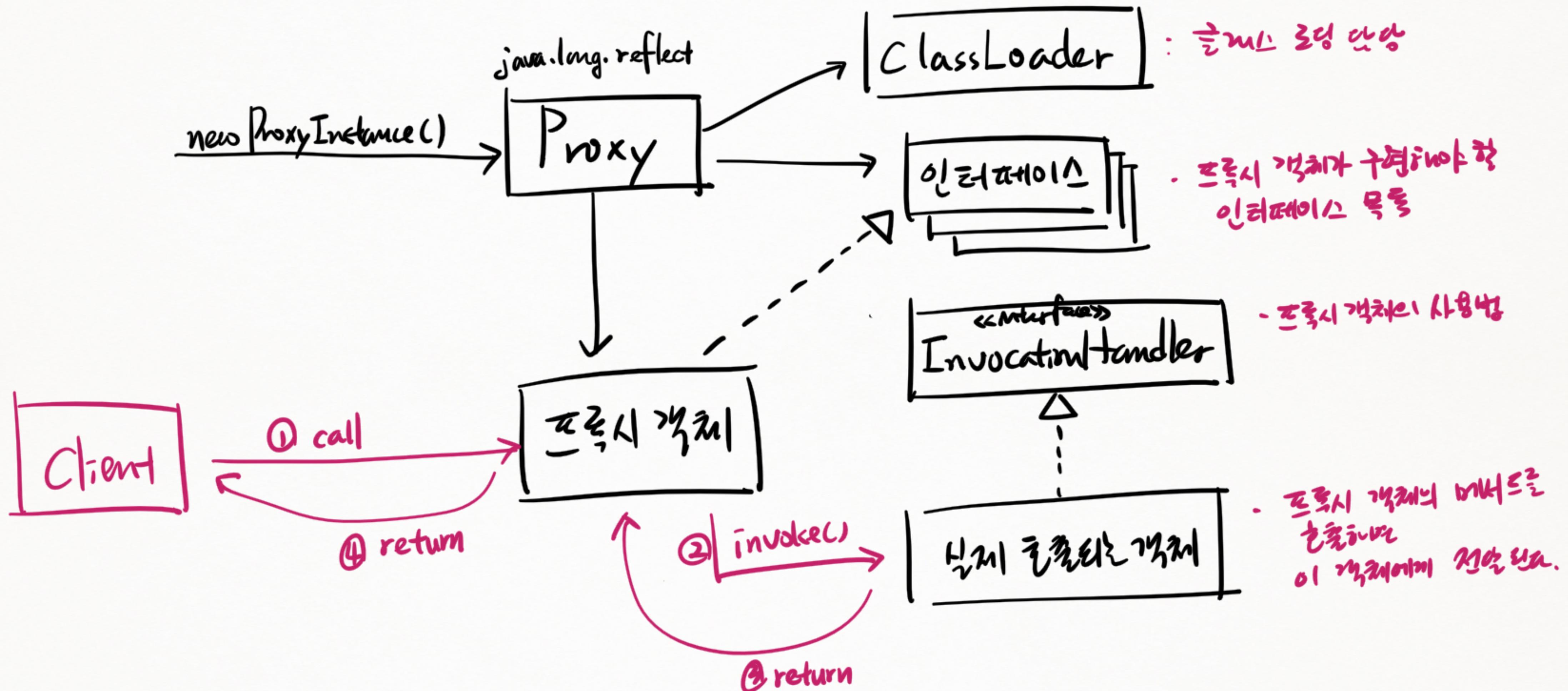
call
call



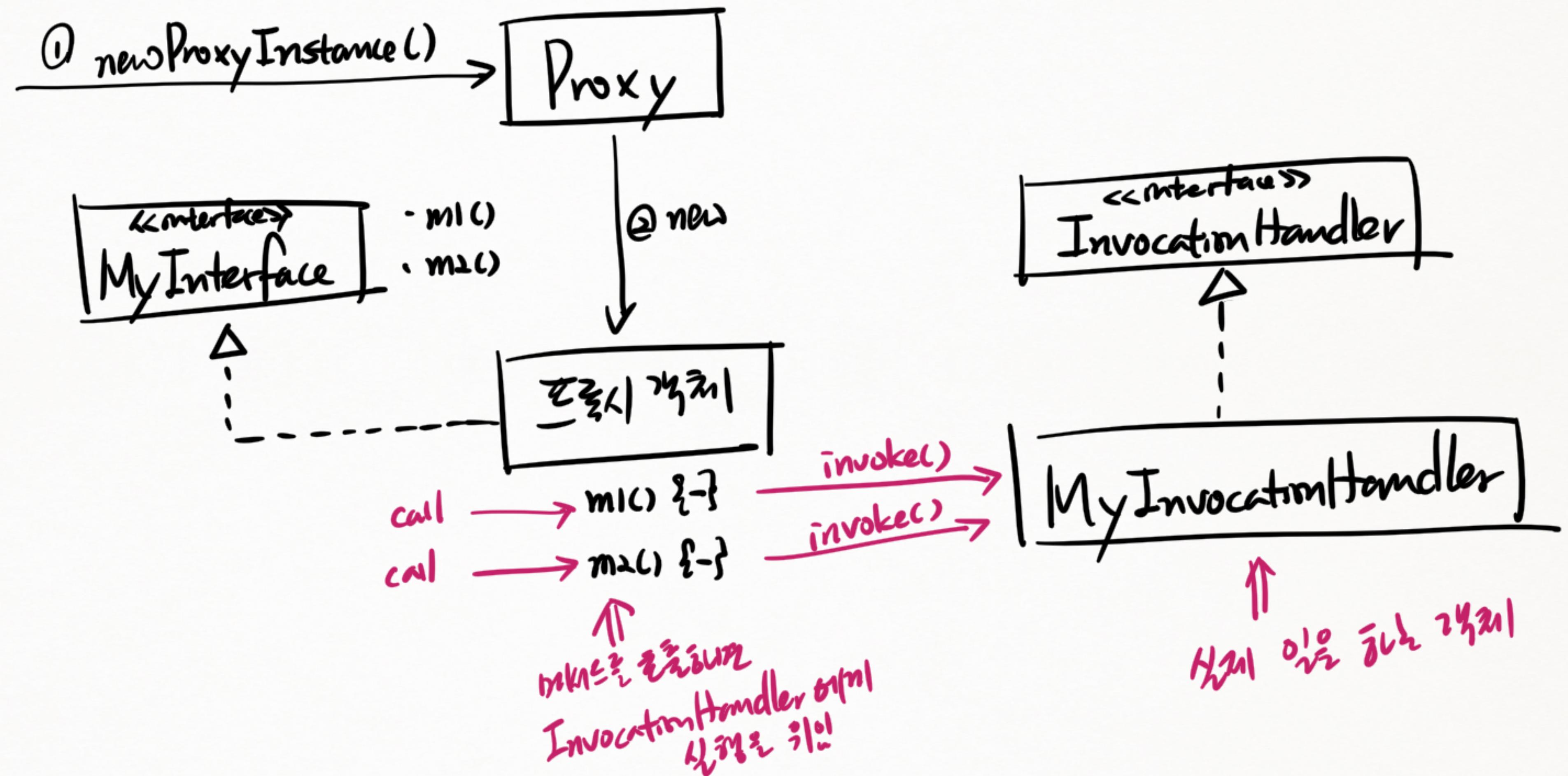
* Project 간에 헤더 공유하기



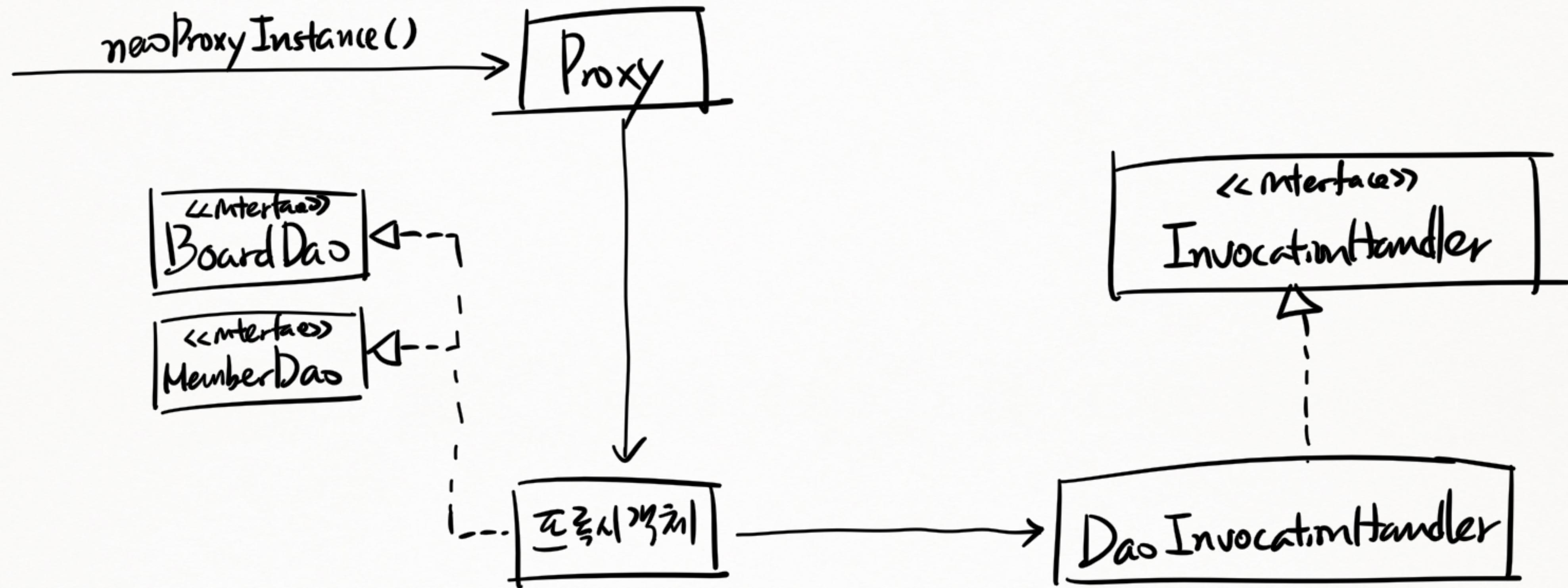
3.8. 프록시 객체 자동 생성



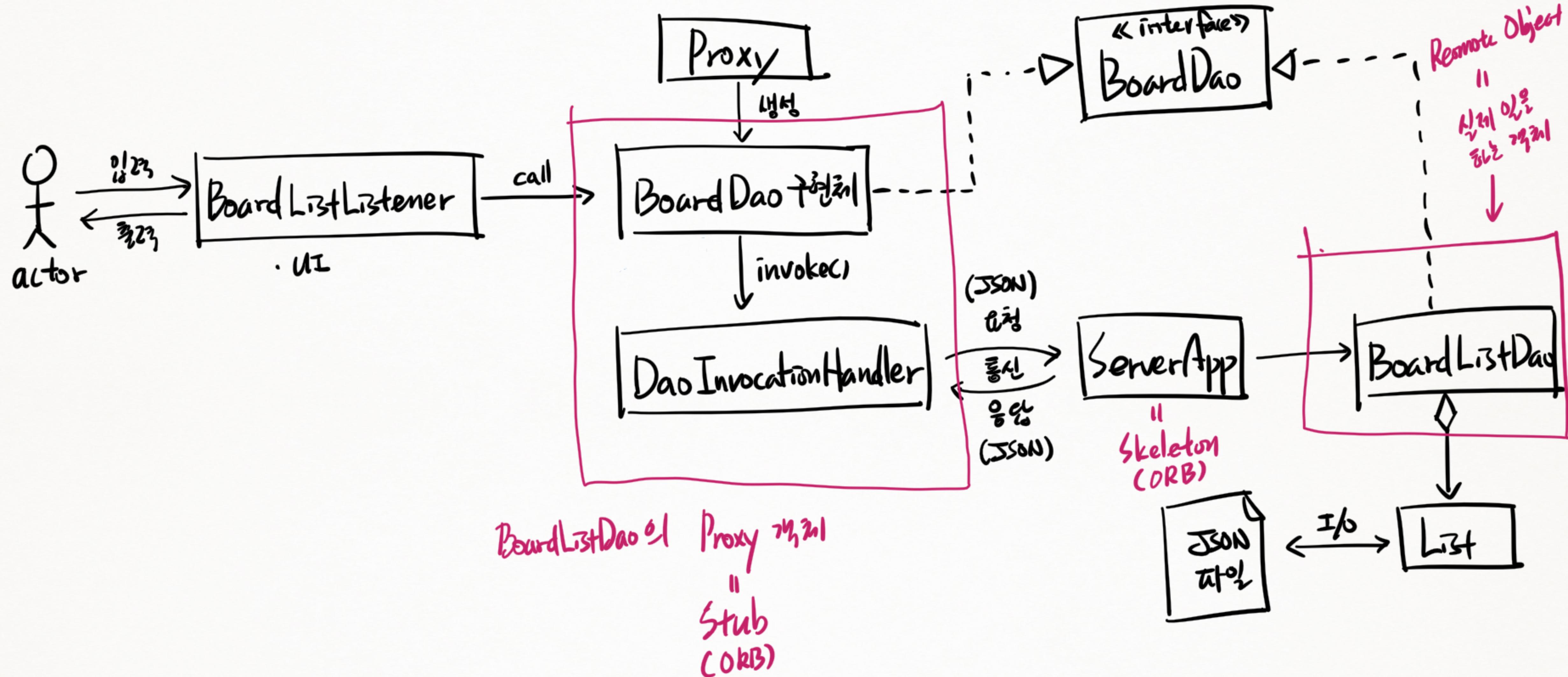
38. 프록시 객체 사용 예제 - ②



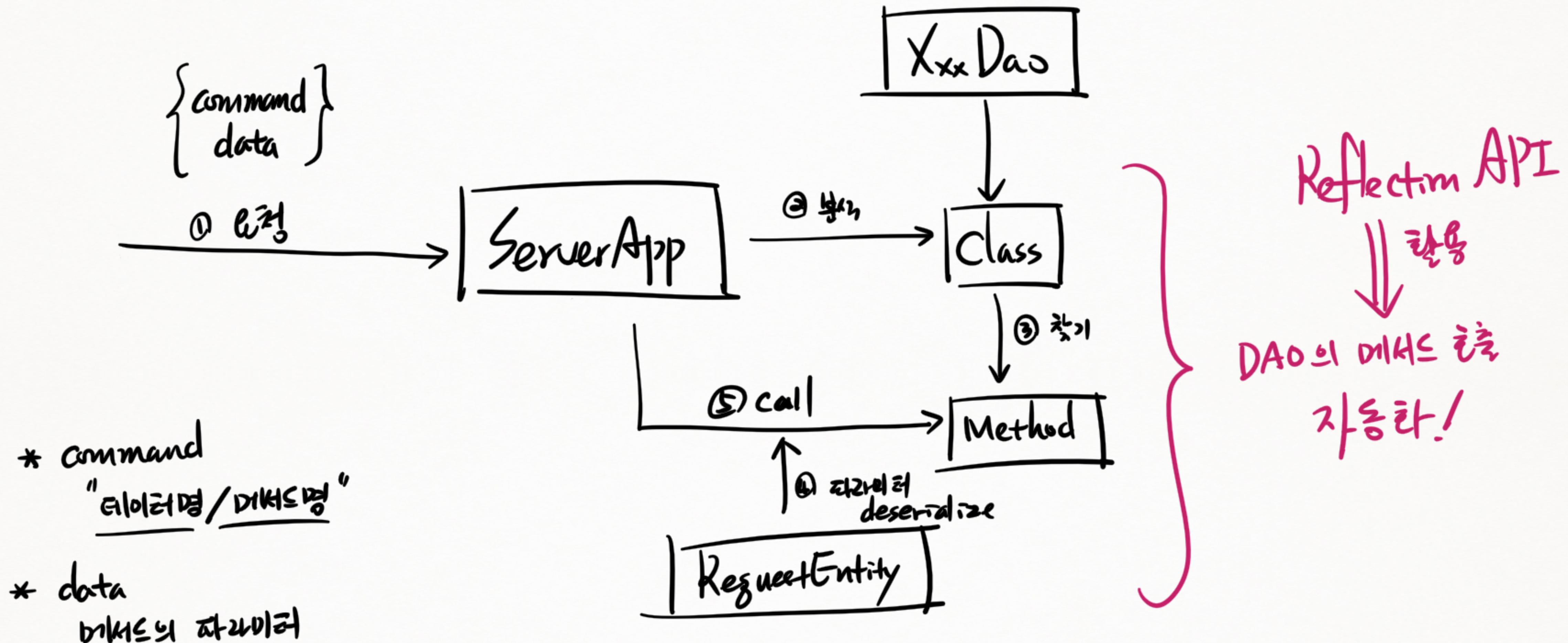
38. 프록시 패턴의 사용 예제 - Participants



38. 프록시 개념의 적용 예제 - Participants II



39. Reflection API를 사용하여 DAO 객체의 메서드 호출을 자동화하기



40. 예외 처리 예제

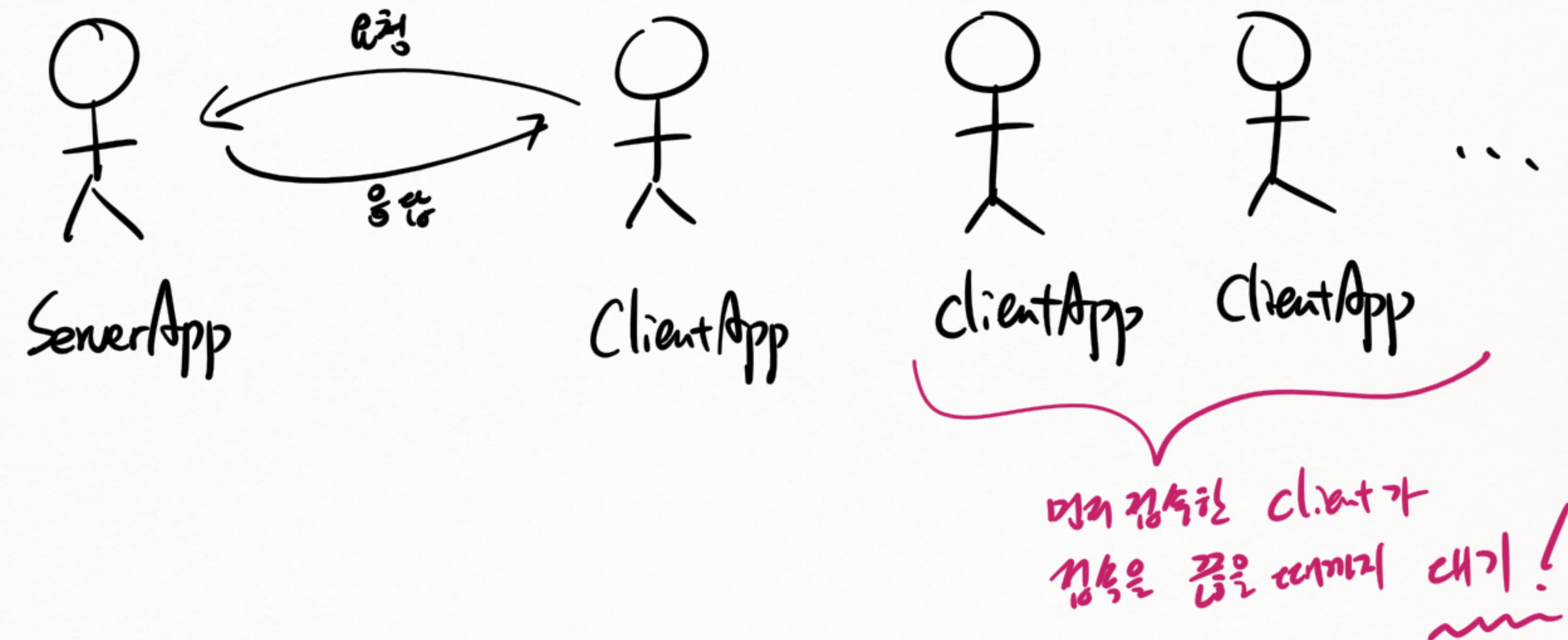
① client 편

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

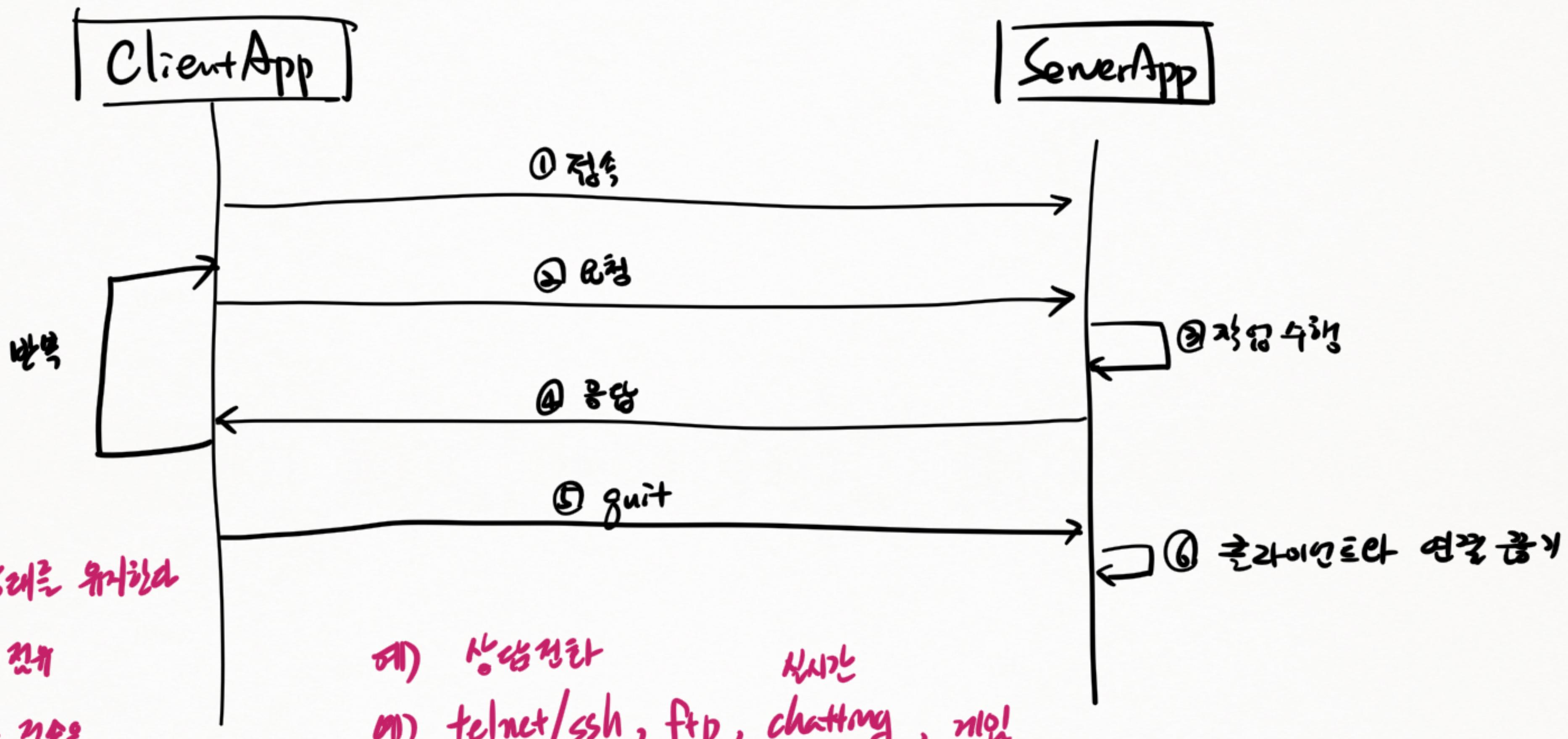
② server 편

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

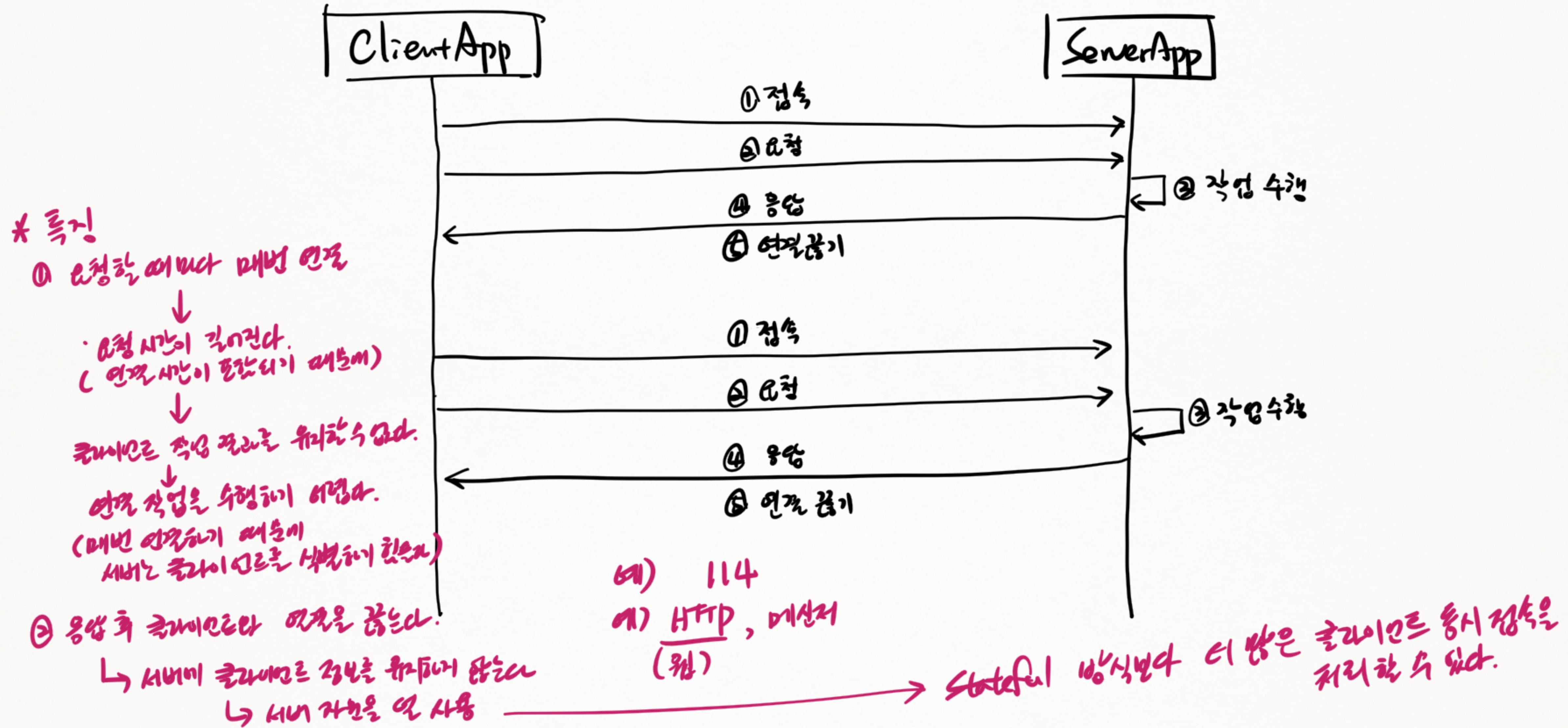
41. 예전 카카오로 페치를 스터디북으로 만들기 - Stateful 방식



41. 예제 클라이언트 페처를 소켓으로 처리하기 - Stateful 방식

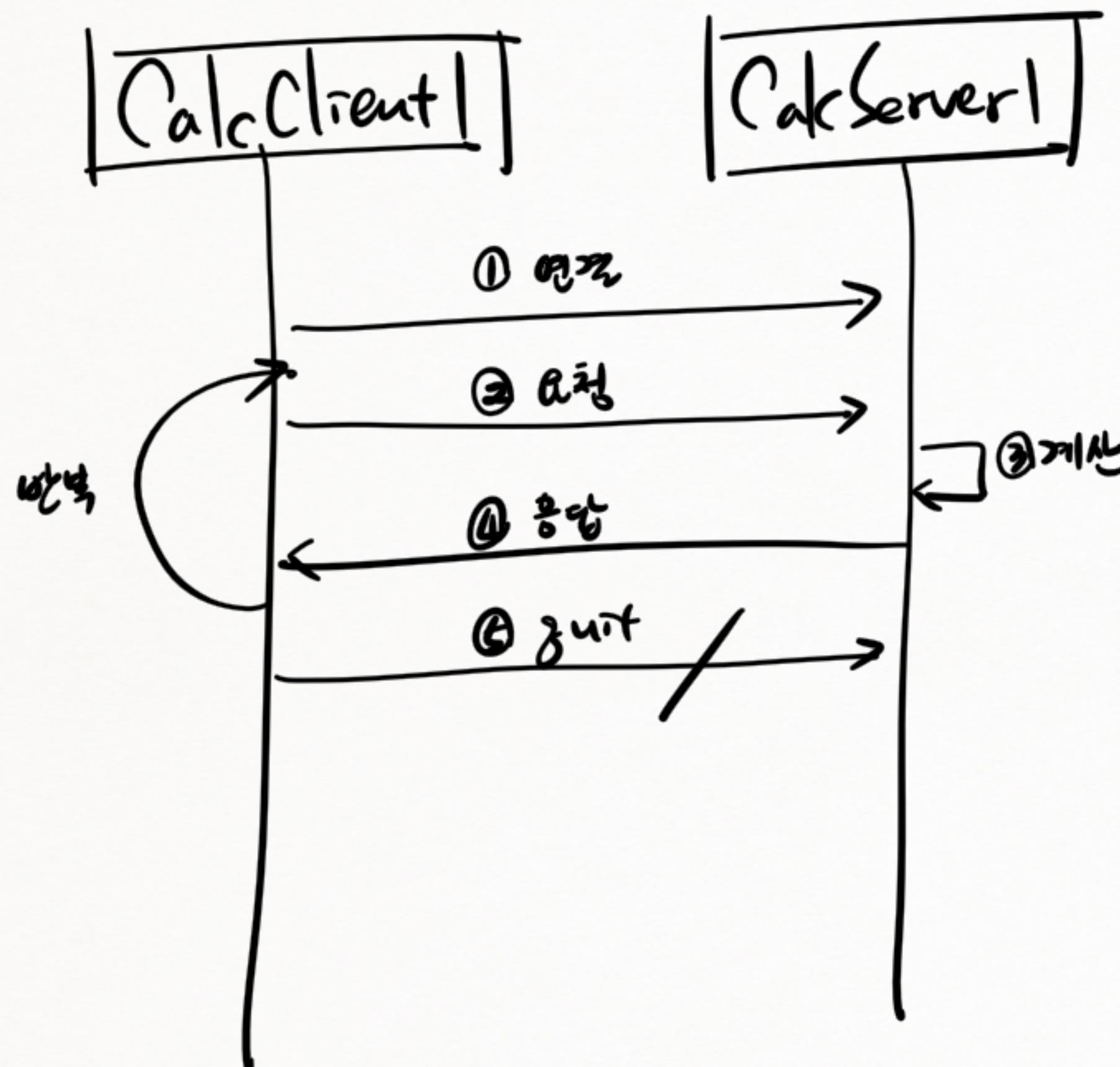


42. 예제 클라이언트 페처를 Stateless 방식

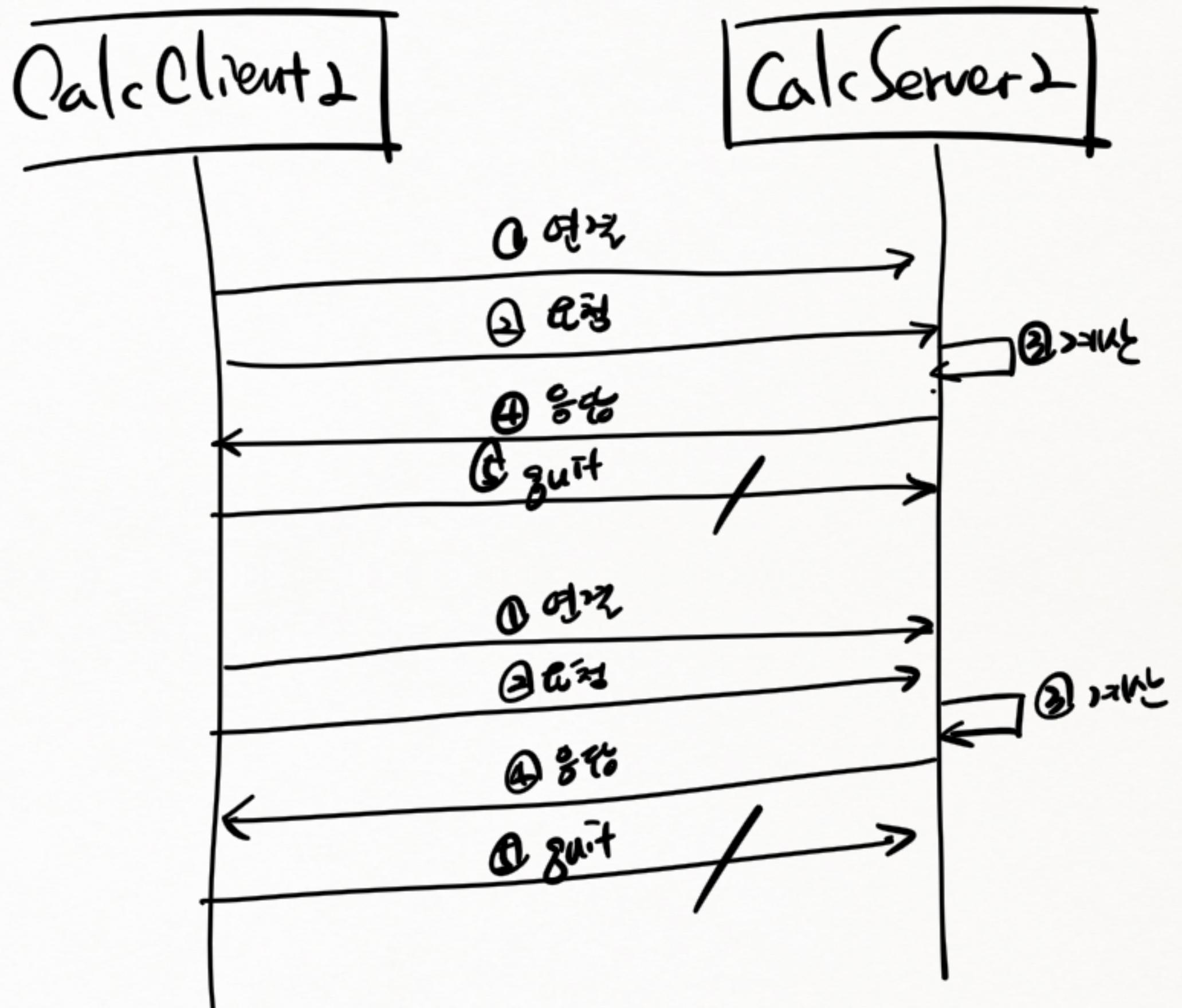


* Stateful vs Stateless

① Stateful

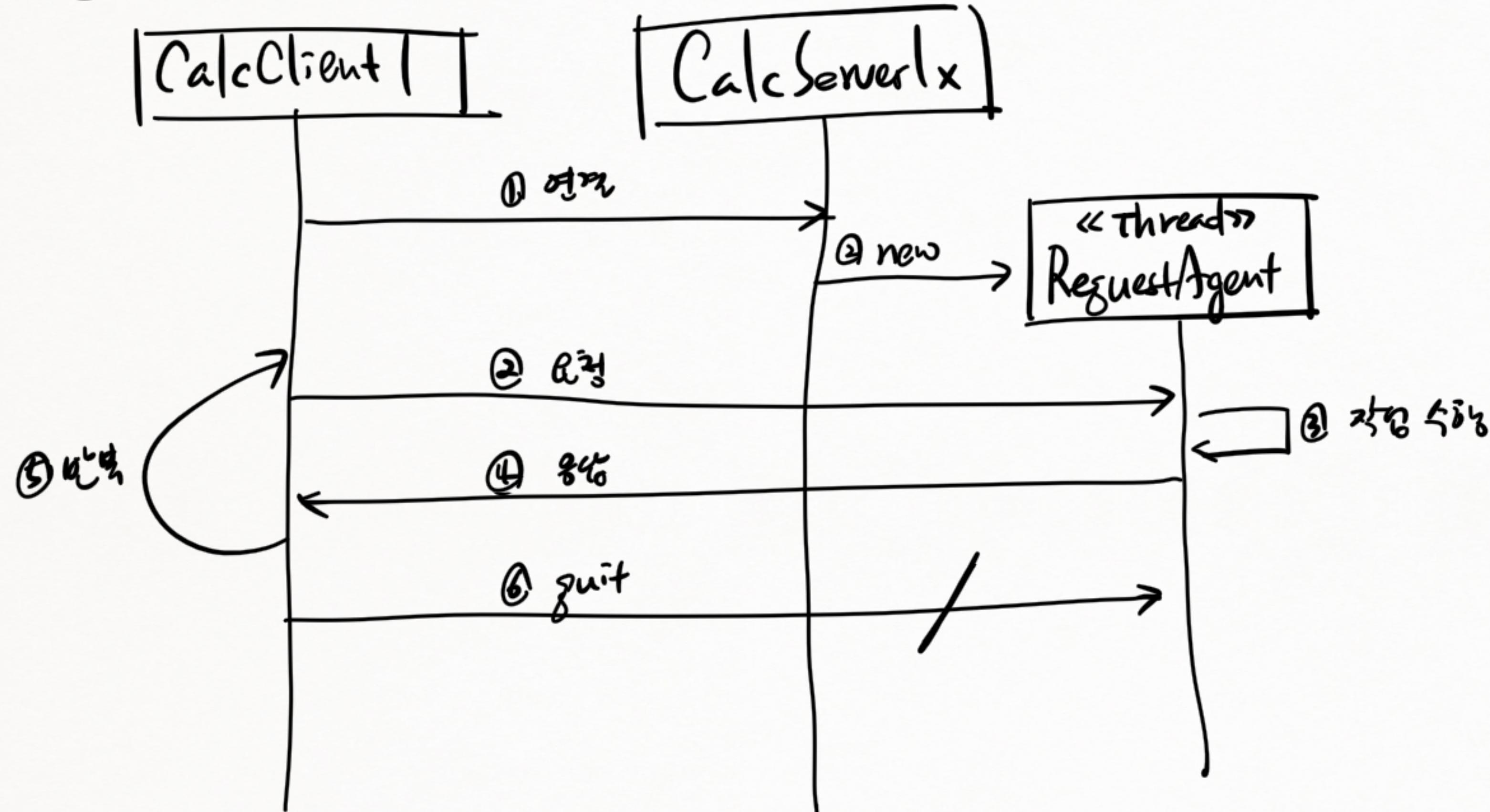


② Stateless



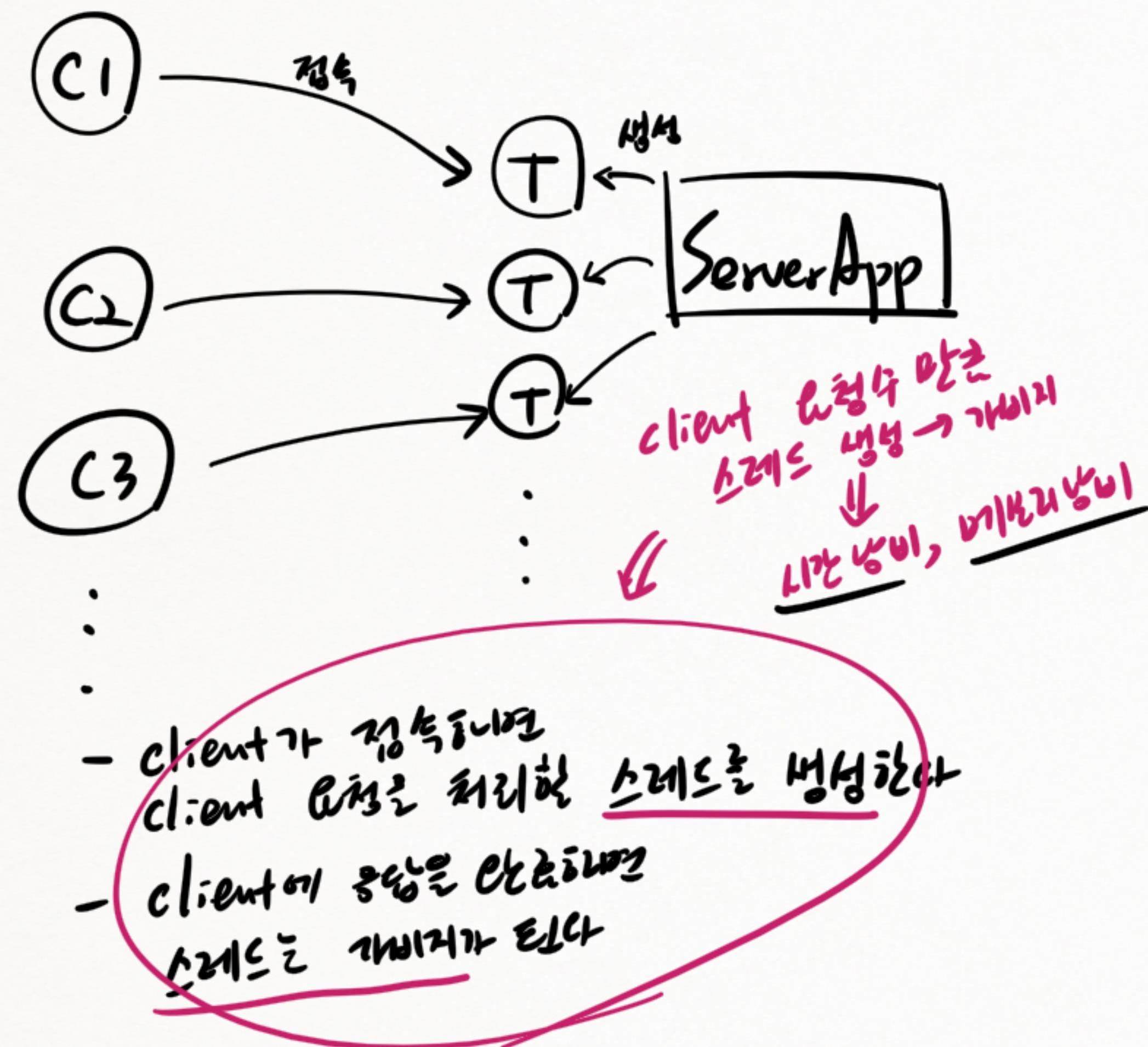
* Stateful vs Stateless

③ Stateful + Thread

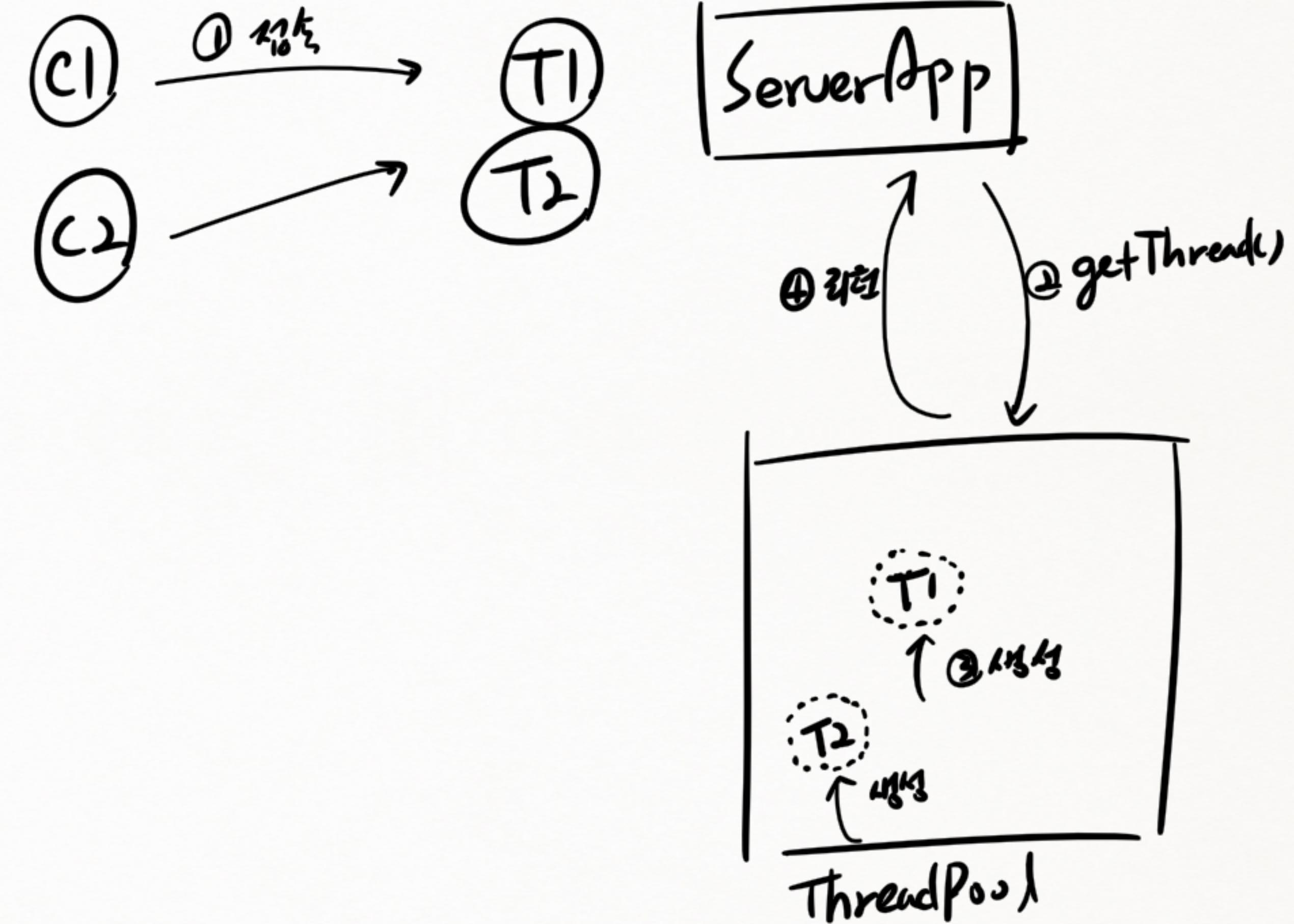


44. 스레드 재사용하기 : GOF의 Flyweight 패턴

① 스레드풀 적용 전

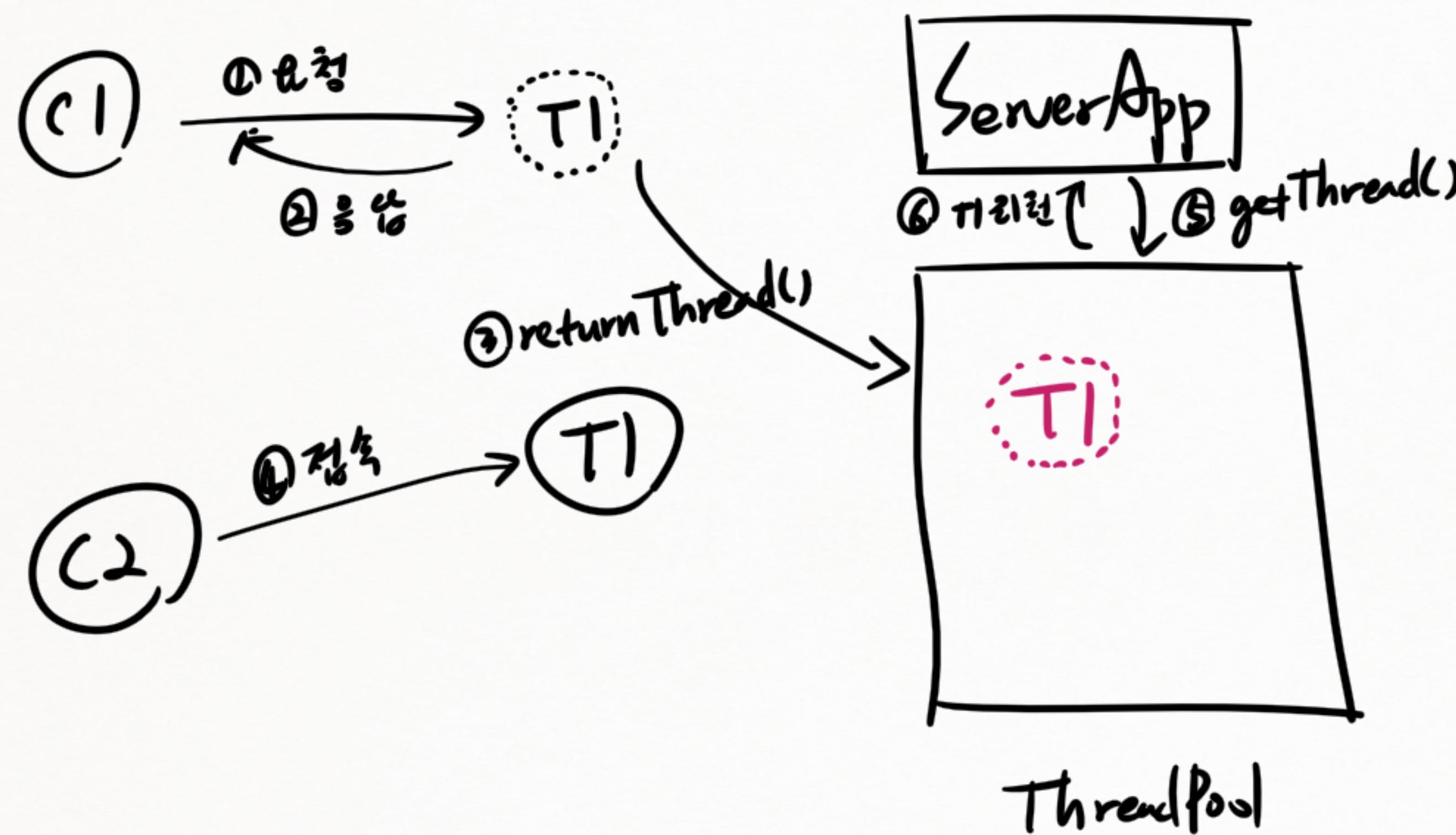


② 스레드풀 적용 후



44. 스레드 재사용하기 : GOF의 Flyweight 패턴

③ 스레드 재사용



{
· 개체 별 생성 시간이 오래 걸리는 경우
· 동일한 객체를 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

