Clinet / Server 리팩토링

순차적으로 여러 클라이언트의 요청을 처리한다.

ServerApp.java

```
ServerSocket ss = ...;
while (true) {
        Socket s = ss.accept();
}
```

ClientApp.java

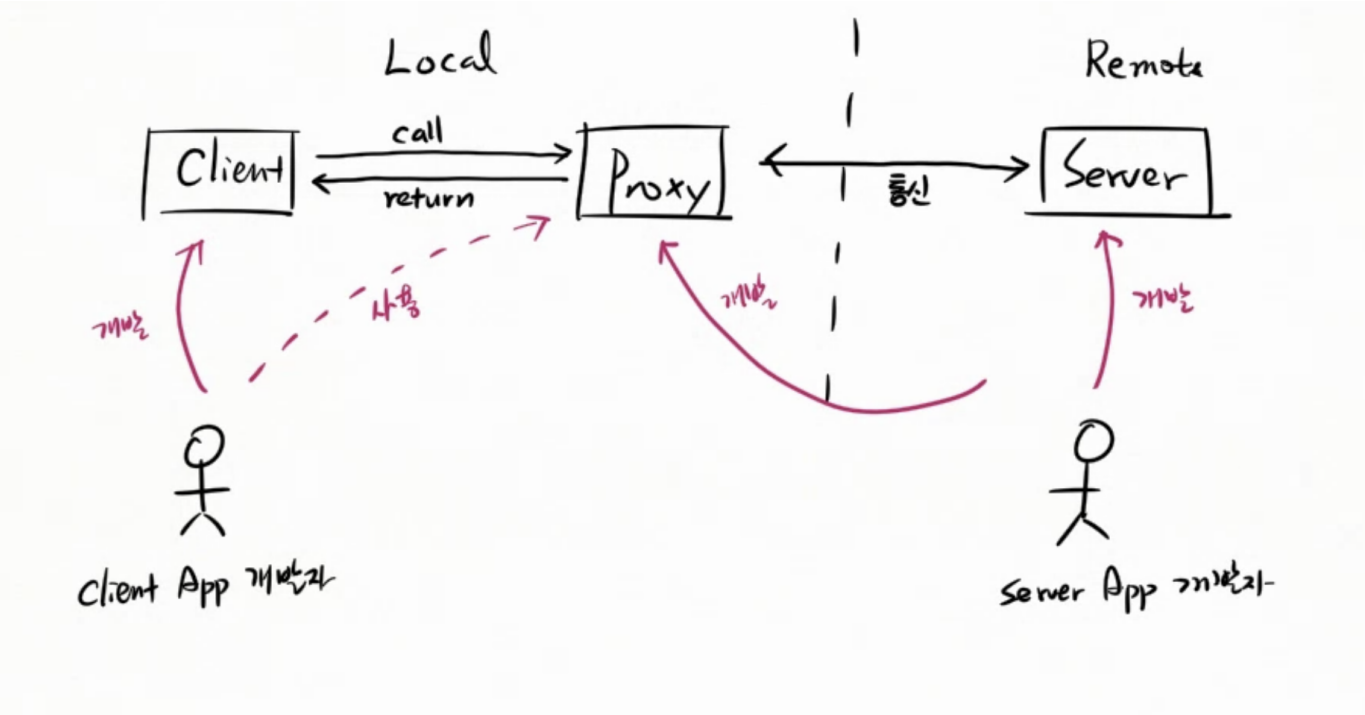
```
Socket s = new Socket(Ip,port);
... // 작업
s.close();
```

클라이언트가 계속 대기해야하는 구조. *41번 끝.*

Proxy 패턴을 이용한 네트워킹 코드 캡슐화하기

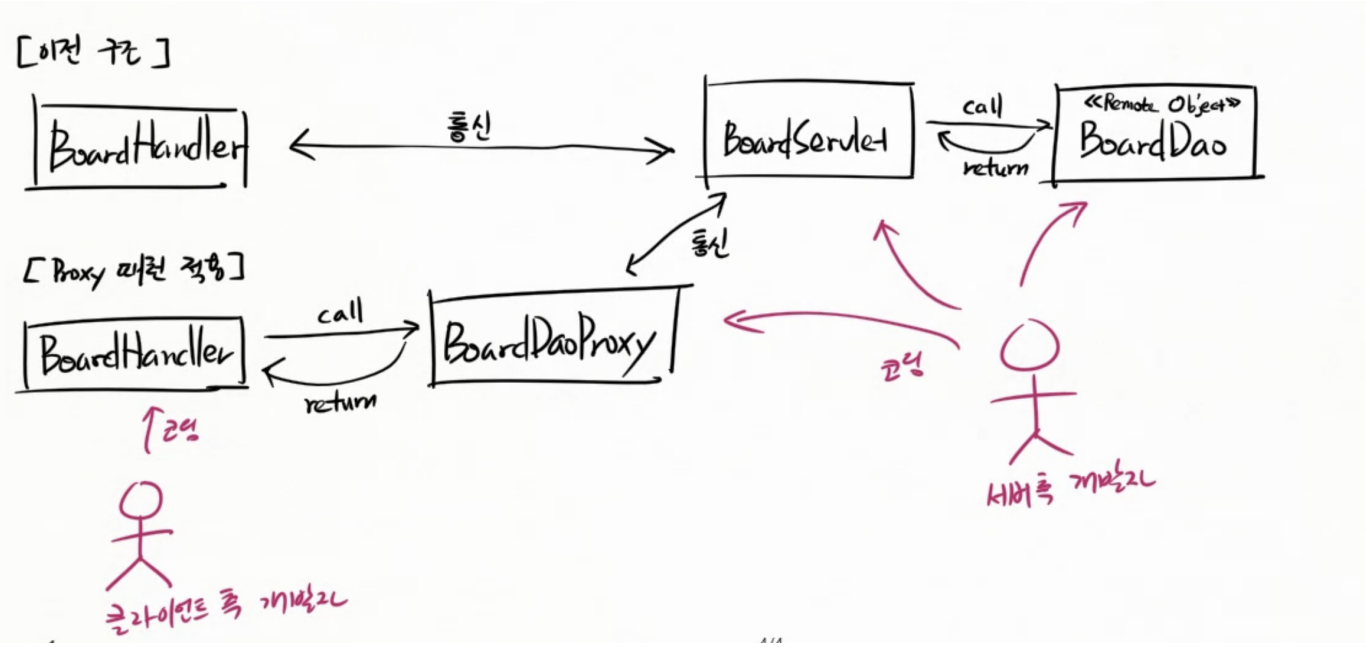
캡슐화 : 복잡한 코드가 안 보이도록 클래스 안의 메서드로 감춘다.
해당 기능이 필요한 경우 간단히 메서드 호출로 처리한다. *예) Collections.sort*

`사용자` (결과)<----------->(요구) `대행인: Proxy` (결과)----------->(지시) `노동자`
`Client` (리턴)<----------->(Call) `통신 대행자` (결과)<----------->(통신) `Server`
통신 대행자를 통해서 **통신 코드를 감춘다.**
Clinet는 통신 대행자의 메서드를 호출한다. 따라서 통신 코드를 알지 못해도 괜찮다.



Proxy는 대행자의 역할을 한다.

Proxy 패턴 적용 전, Proxy 패턴 적용 후 구조



BoardDaoProxy 클래스 생성

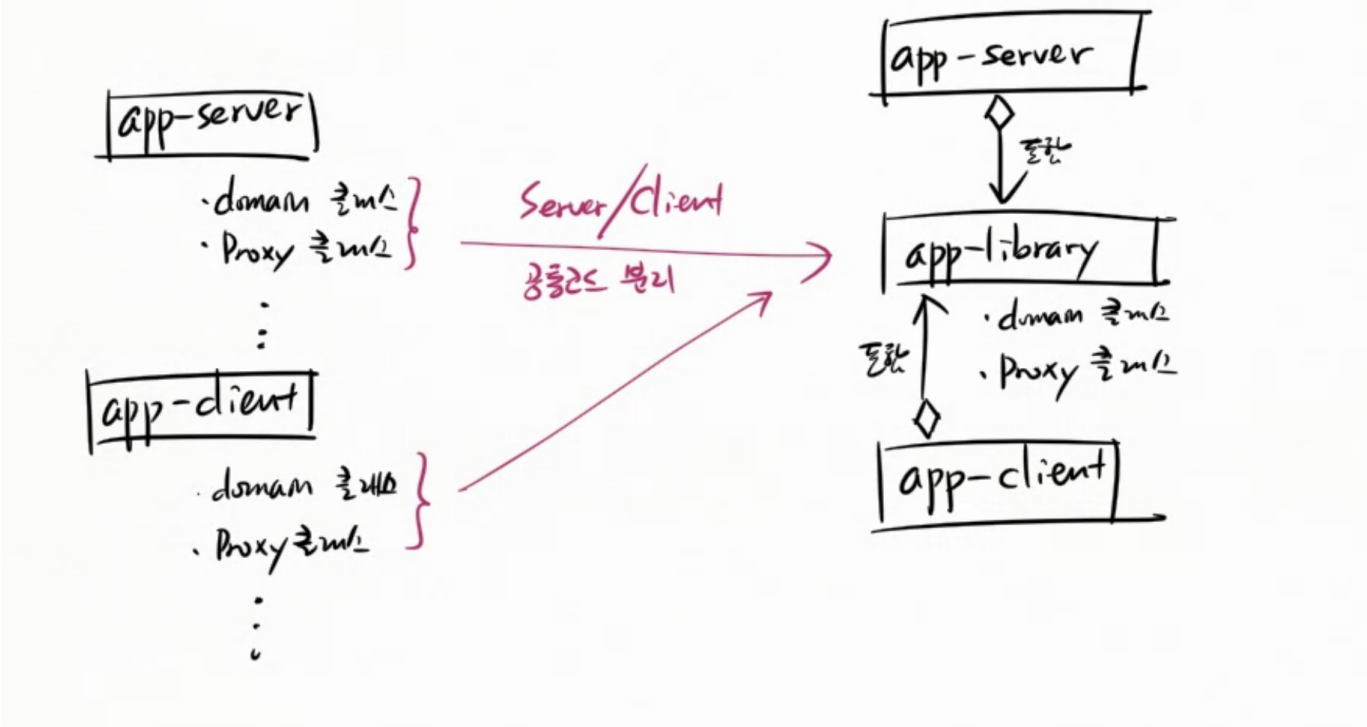
BoardDaoProxyTest 클래스로 BoardDaoProxy 테스트

그 후,

Client 쪽에서 BoardDaoProxy 클래스를 적용한다. -> *Clinet의 BoardHandler 수정*

Clinet / Server 공통 코드를 라이브러리 프로젝트로 분리하기

43. 서브 프로젝트 분리 및 공유



app-common은 공통 소스를 두는 프로젝트이다.

`implementation project(':app-common')` app-common 에 있는 클래스를 사용하겠다는 의미.

app-client를 복사해서 app-common을 만든다.

app-common에 있는 build.gradle을 다음과 같이 수정한다.

application 삭제, plugins에 id를 'java-library' 로 변경. project name = "board-app-common"

[app-common 에는 client와 server의 공통 코드를 둔다.]

dao 패키지

BoardDaoProxy.java

MemberDaoProxy.java

domain 패키지

Board.java

Member.java

서버 개발자는 app-server와 app-common을 가져와서 사용.

클라이언트 개발자는 app-client와 app-common을 가져와서 사용.

app-common 에 의존하므로 classpath도 변경. .bat, .sh 파일을 변경한다.

43 끝.

통신 방식을 Stateful에서 Stateless로 변경하기

Stateful 과 Stateless = Connection - Oriented(연결 지향)

Connection - Oriented(연결 지향), TCP

1. Stateful : 예) 은행 업무, FTP, POP3, 채팅, 게임
2. Stateless : 예) 114, ARS 인증, HTTP

   Connectionless, UDP

   예) 편지, 방송, ping