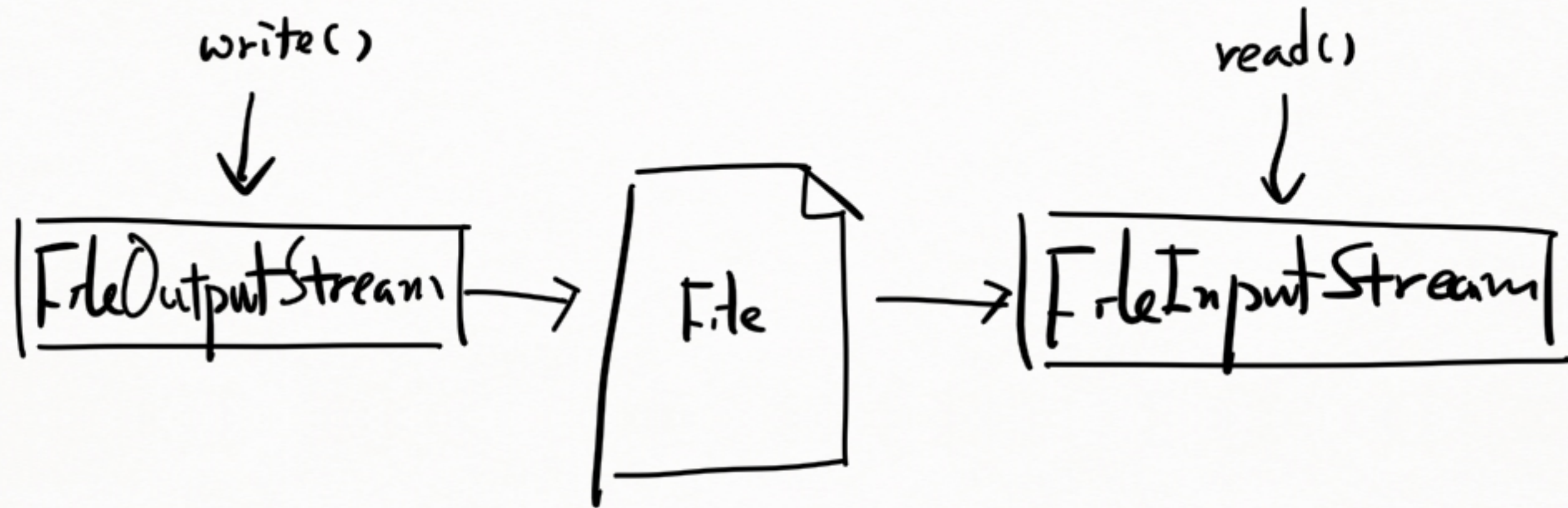
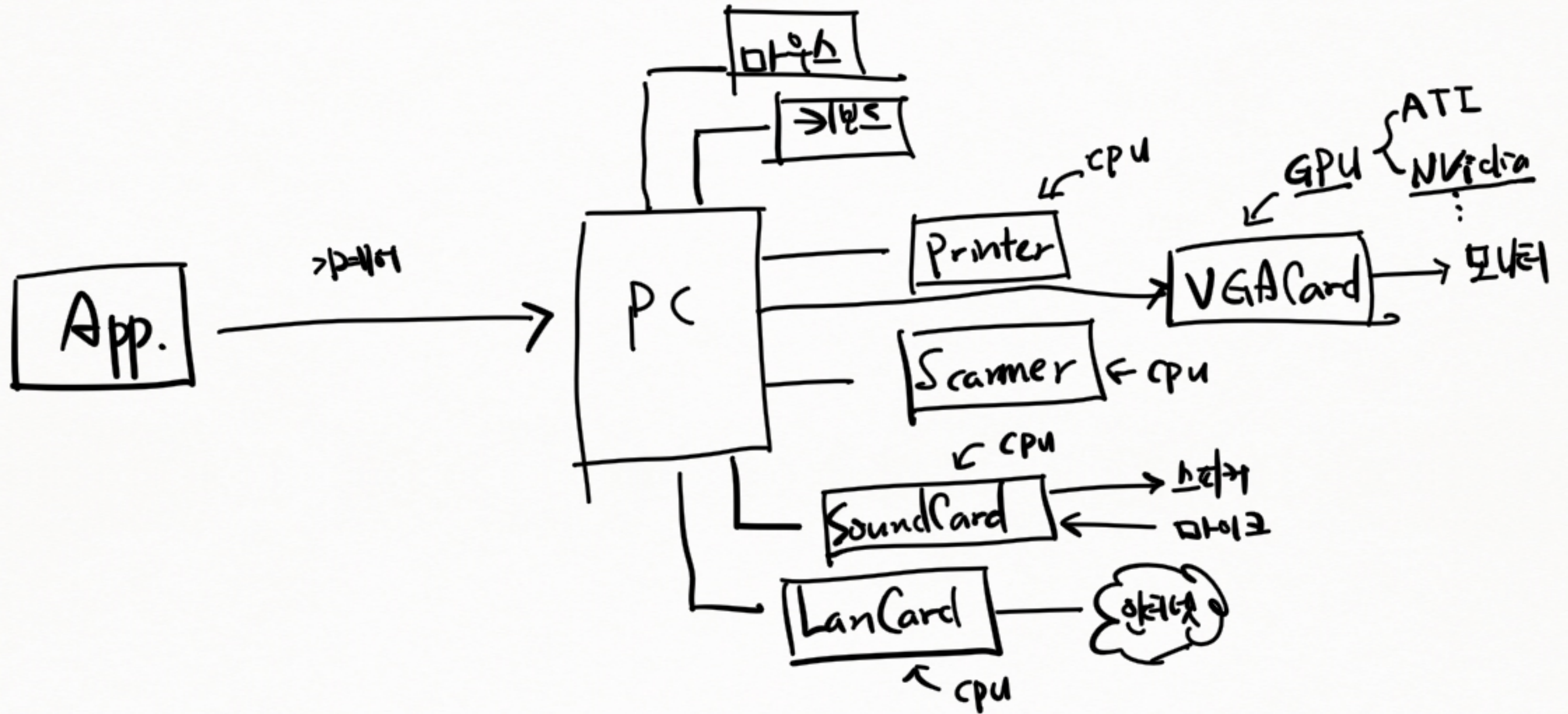
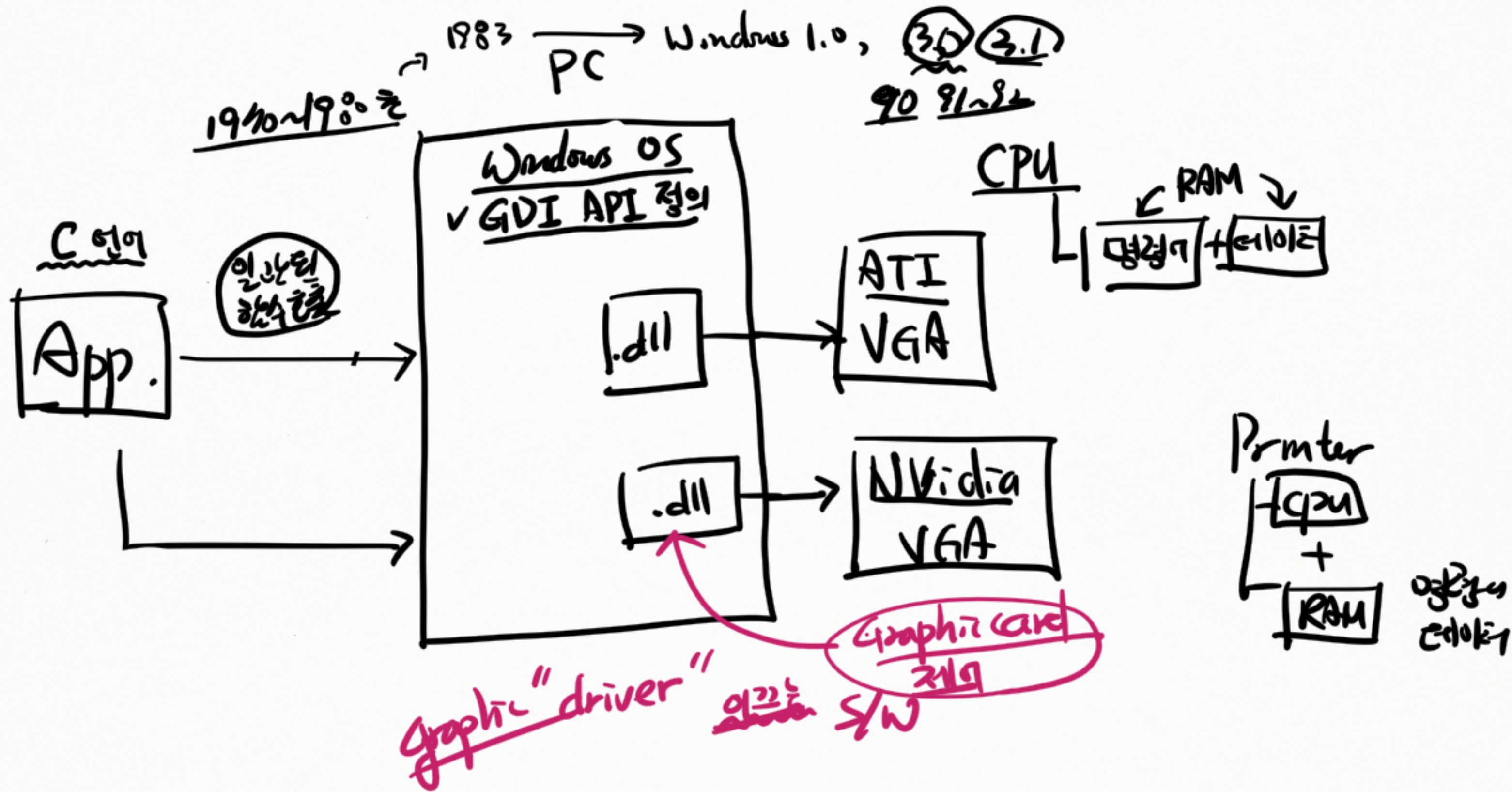
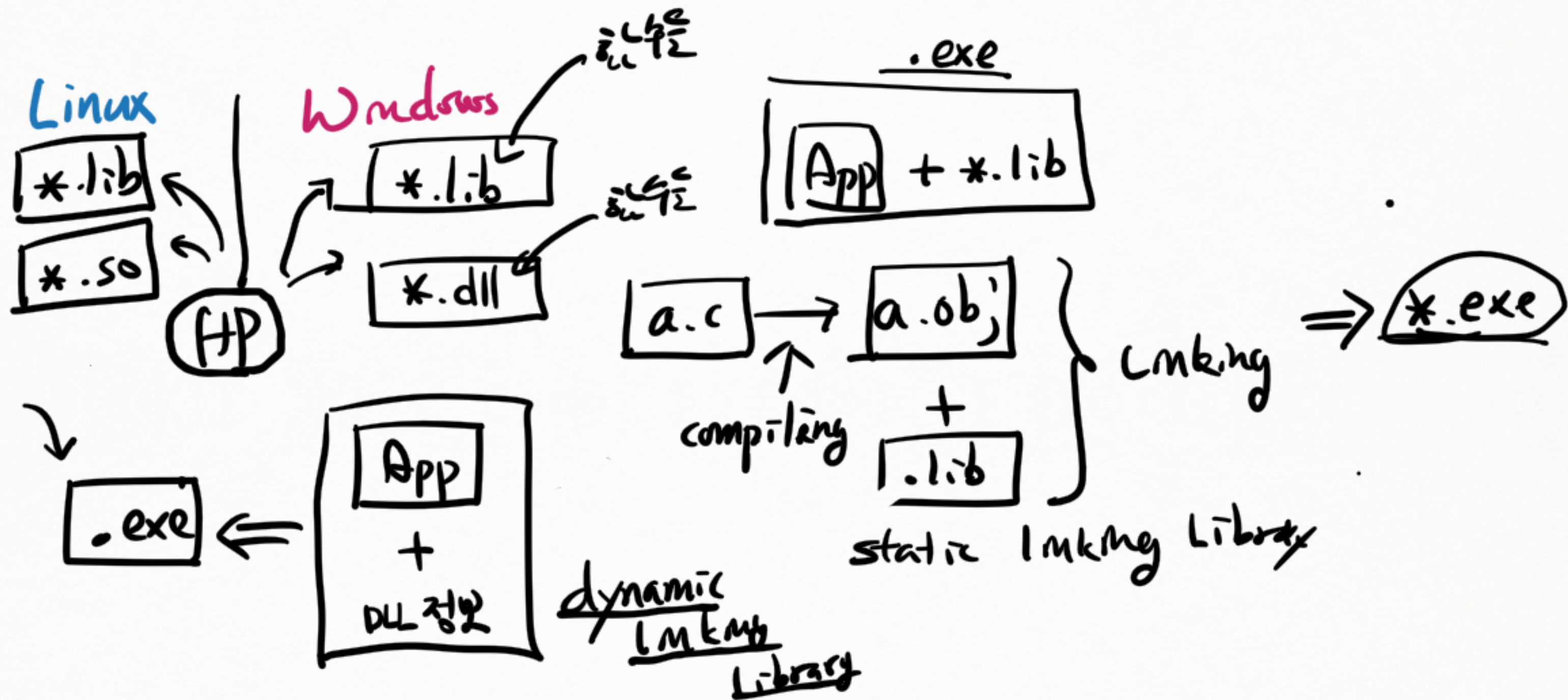
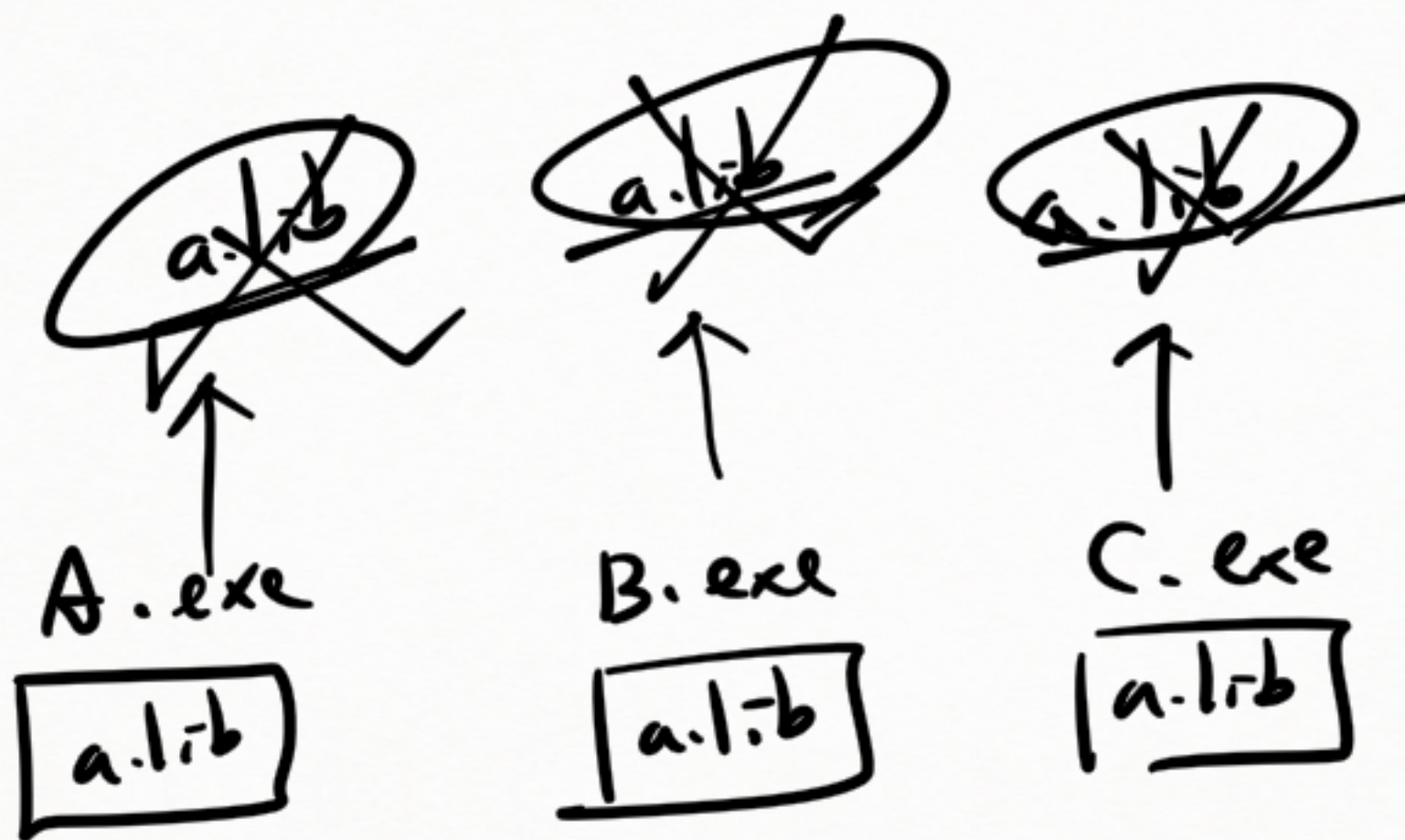


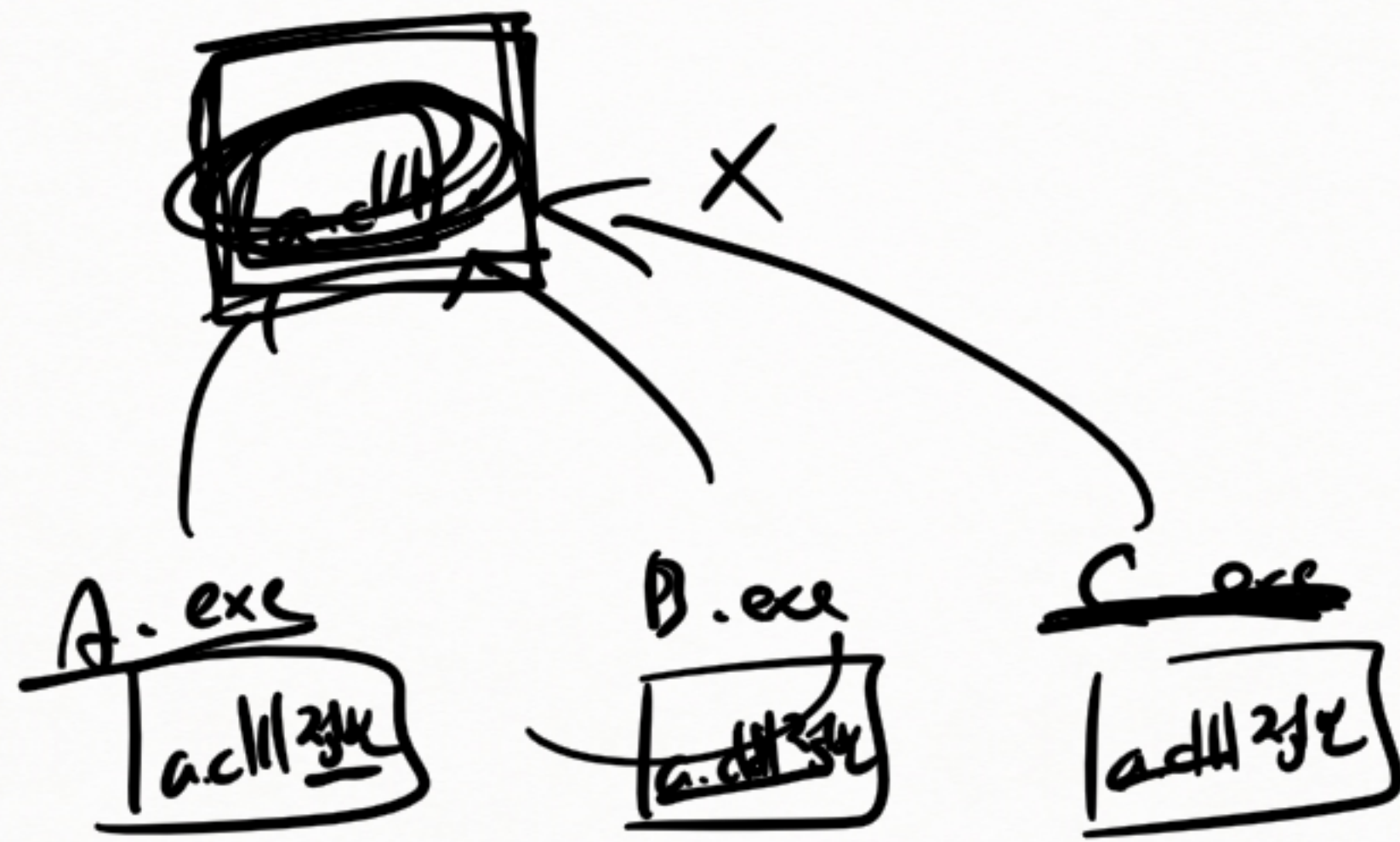




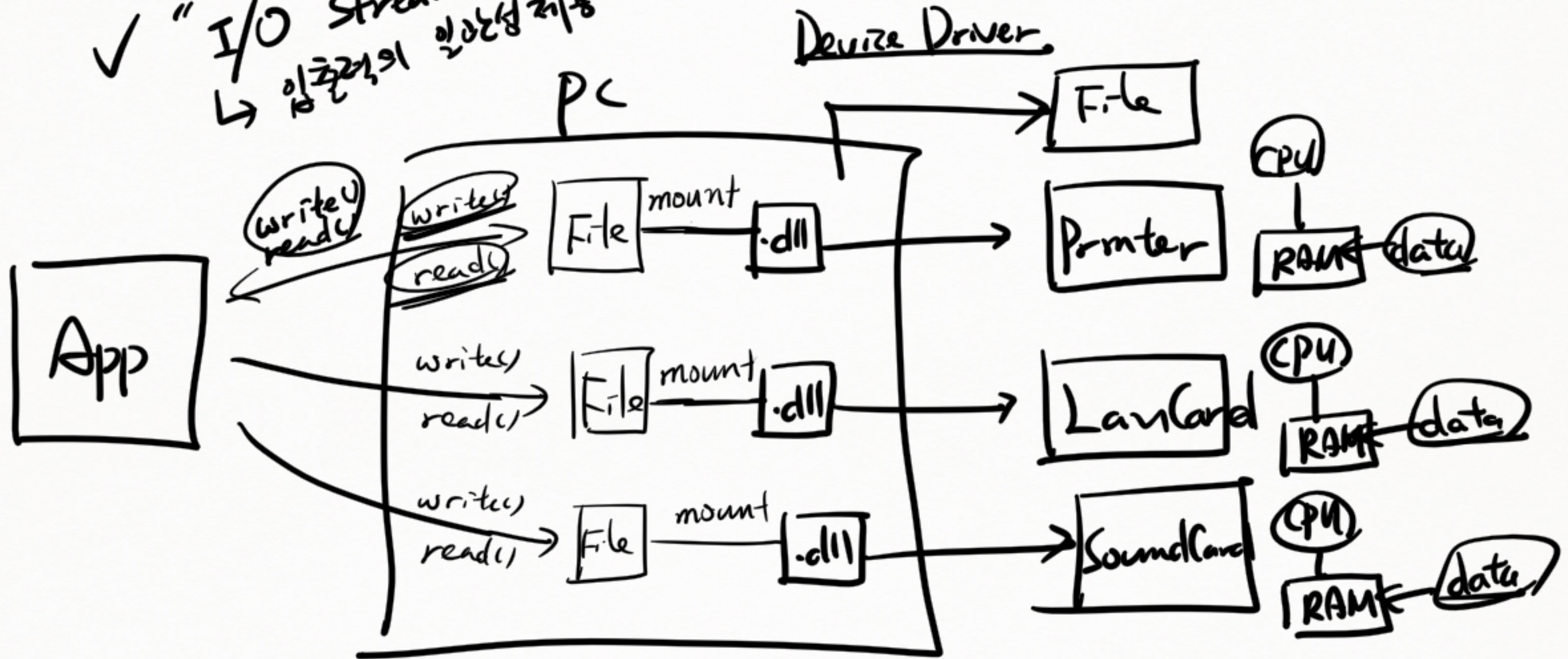








✓ " I/O Stream API "
↳ 입출력의 입출력 제어



Java I/O Stream API

byte stream

character stream

0/2=1
10

InputStream

Reader

2/2=1
10

OutputStream

Writer

Data Sink Stream Class

Input

Output

FileInputStream

FileOutputStream

ex) .gif .jpg .ppt
.avi .pdf

FileReader

FileWriter

ex) .csv .html .java
.js .css .txt

ByteArrayInputStream
CharArrayReader
StringReader

ByteArrayOutputStream
CharArrayWriter
StringWriter

PipedInputStream
PipedReader

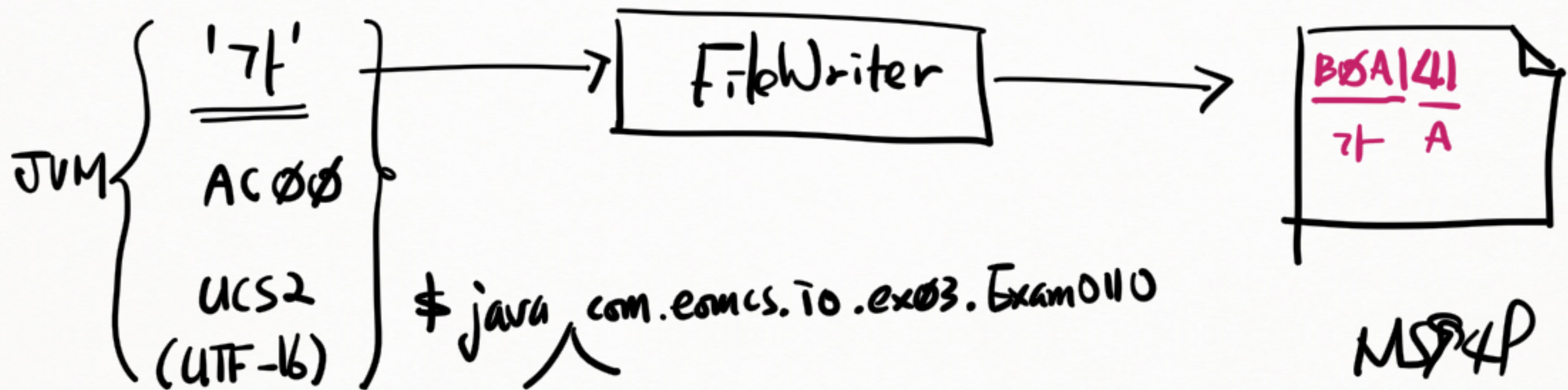
PipedOutputStream
PipedWriter

File

Memory

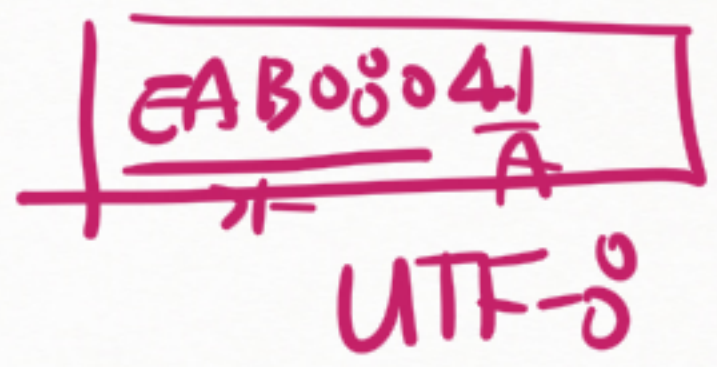
($\frac{d^2}{dt^2} \approx \frac{224}{24}$)
Process

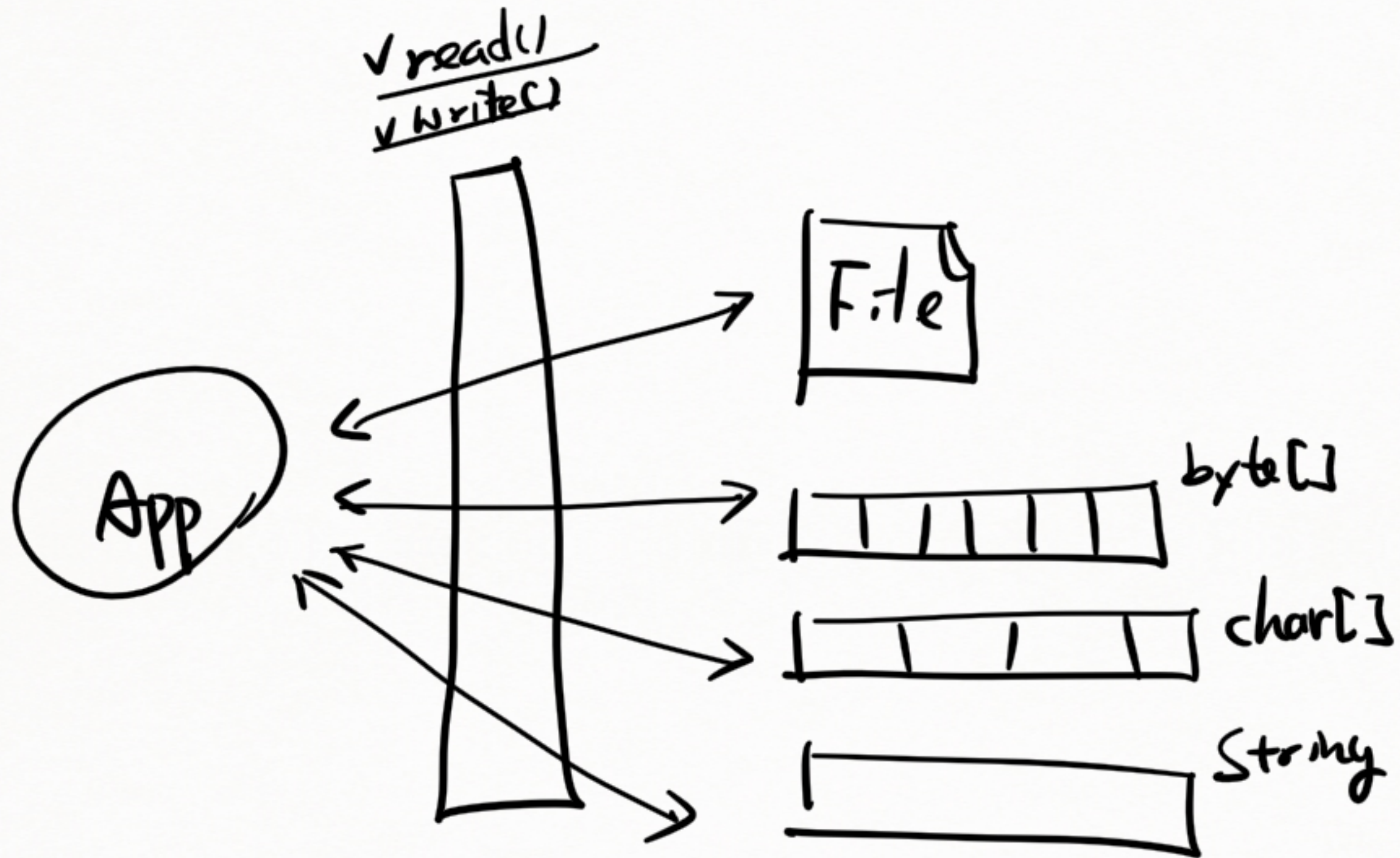




\$ java com.eomcs.io.ex03.Exam0110

x OS의 기본 문자셋
 ↓
 Windows → MS94P
 Linux/MacOS → UTF-8





MS949: 41 42 B0 A1 B0 A2

UTF-8: 41 42 EA B0 80 EA B0 81

UTF-16BE: 0041 0042 A000 A001

UTF-16LE: 4100 4200 00A0 00A1

JVM



char: UCS2 (UTF-16BE)

0041 0042 A000 A001
A B 7F 7F

new String(byte[], offset, length, charset)

File/Network

x MS94P
x EUC-KR
UTF-8



JVM



UCS2

||

UTF-16BE

||
char c;

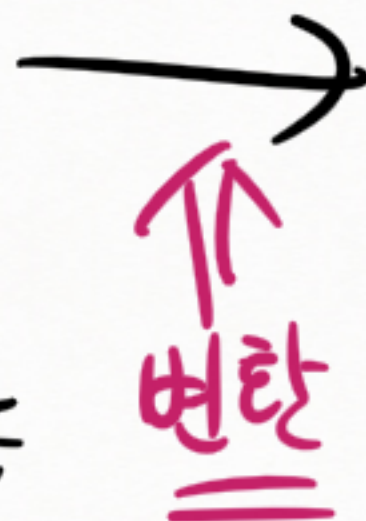


DBMS

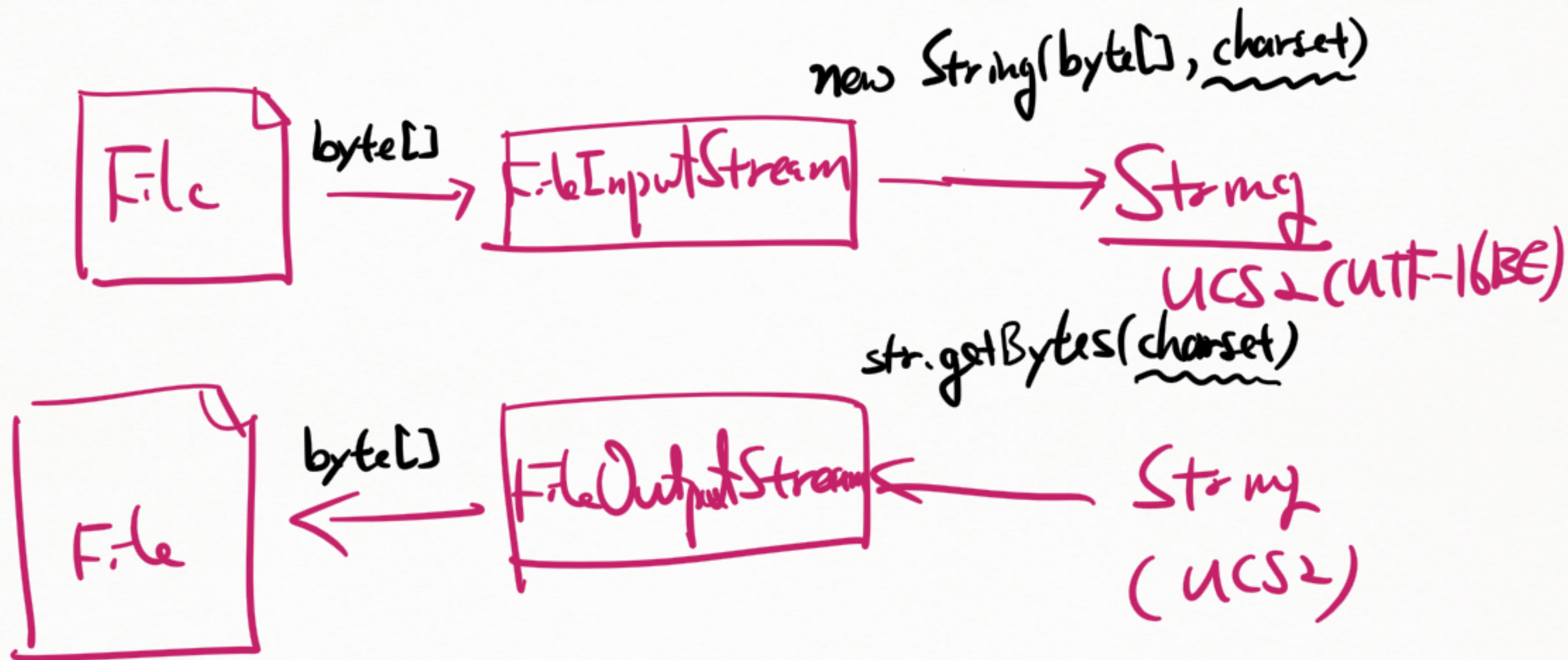
x

File/Network

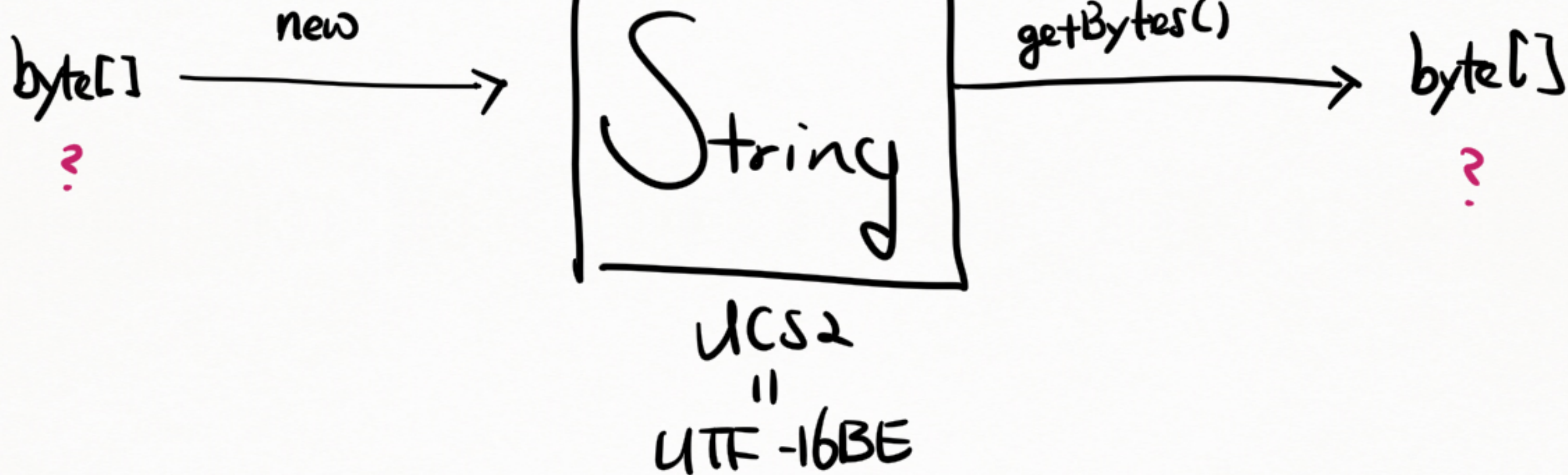
MS94P x
EUC-KR x
UTF-8



'A' → 0041
'가' → A000



JVM 환경 변수
file.encoding = ?



$$E \Rightarrow \underline{3017} - 3030 \Rightarrow \text{필수 } \boxed{?}$$

2x xx

중식

34 xx - 30 xx

한식

417 x - 30 xx - 30 xx

100 만 원

9999 + 30 xx - 12 xx - 30 xx

1억

✓ A $\Rightarrow$ 2134

✓ B $\Rightarrow$ 3478

✓ C $\Rightarrow$ 4179

✓ D $\Rightarrow$ 9999

- 3012

- 3033

- 1211

- 3088

file.encoding = UTF-8

"안녕"
"Halla"
"안녕"
"Halla"

"한글"

"영문"

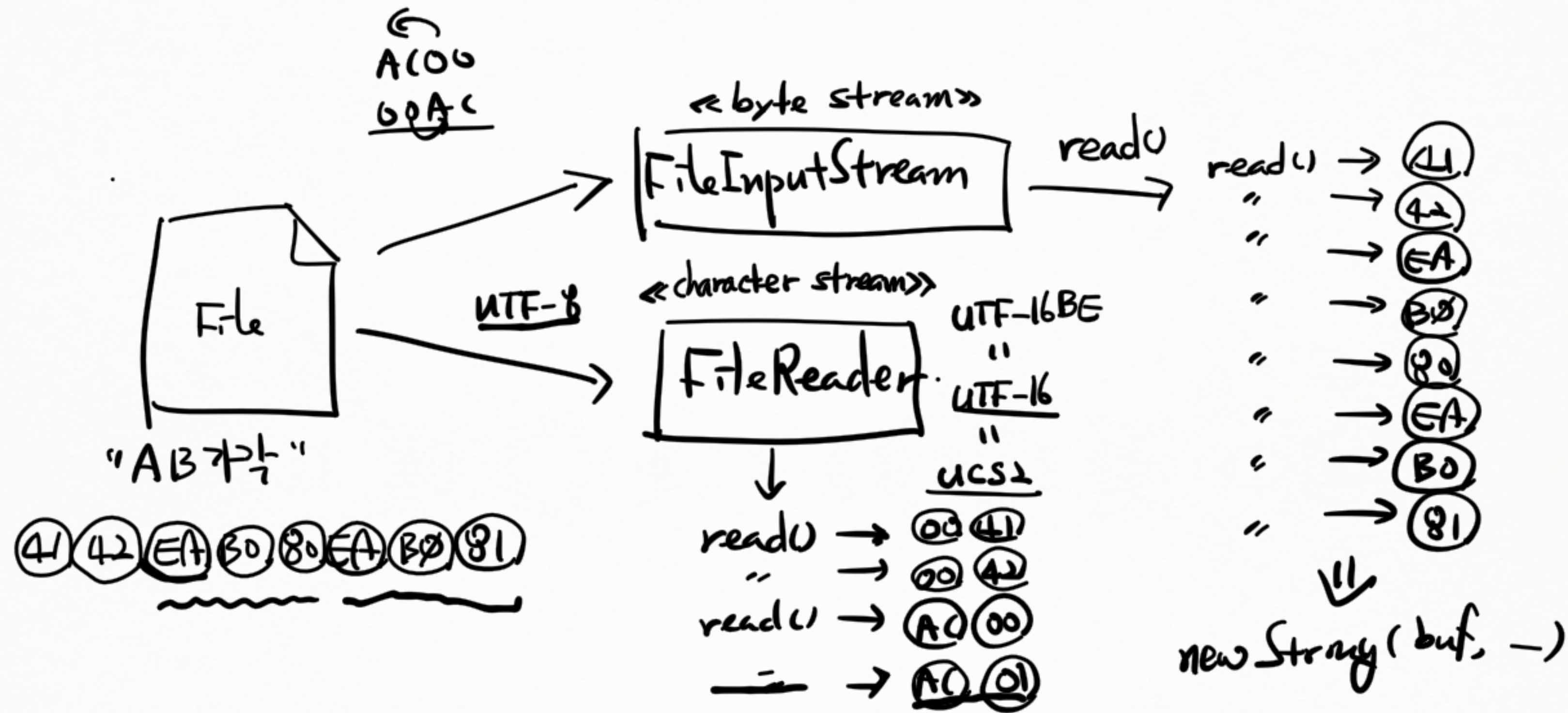
x

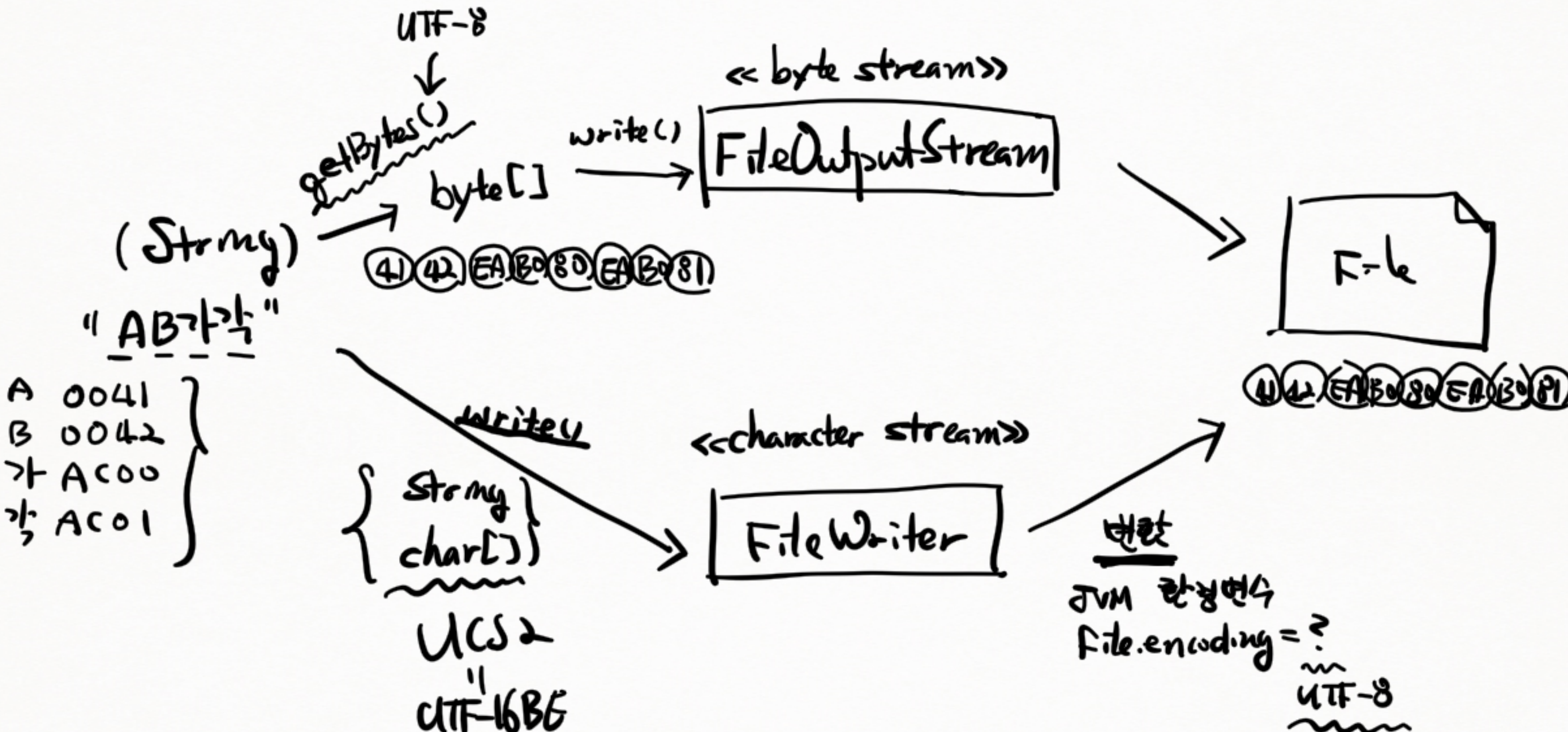
x

String

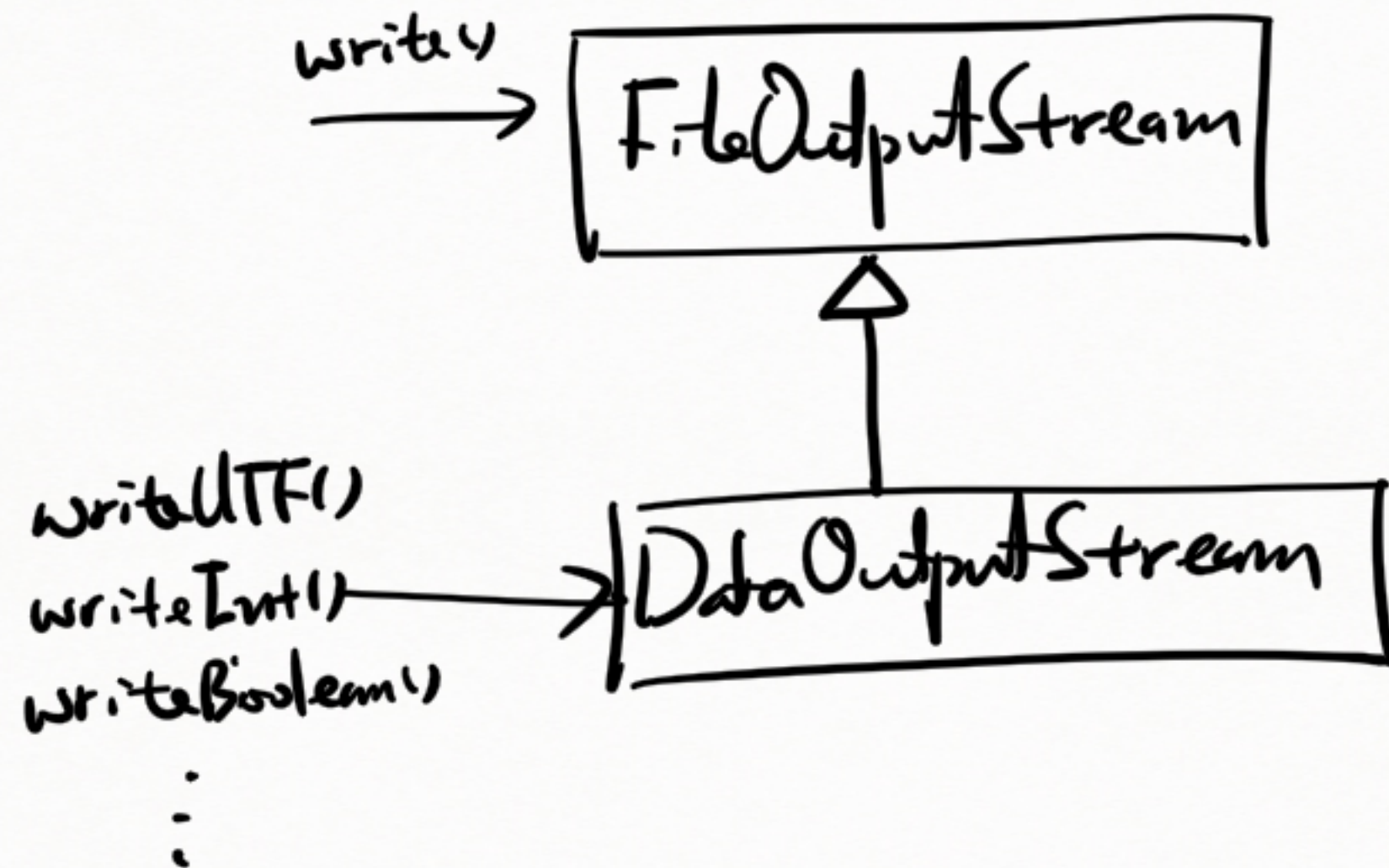
바이트 문자열

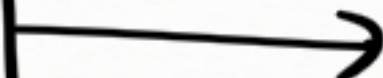
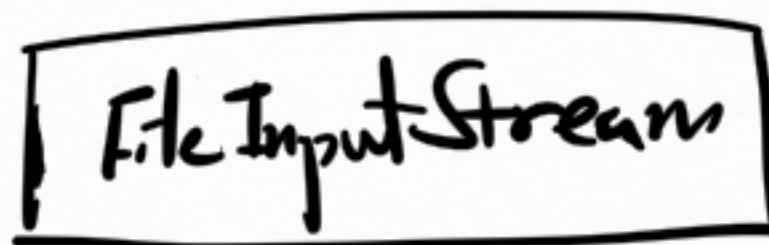
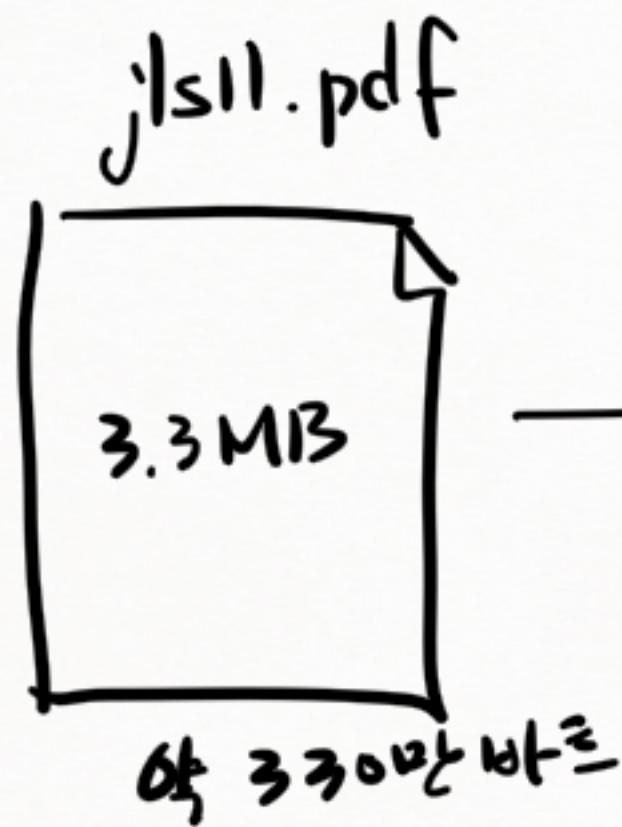
영문
[?]
Halla







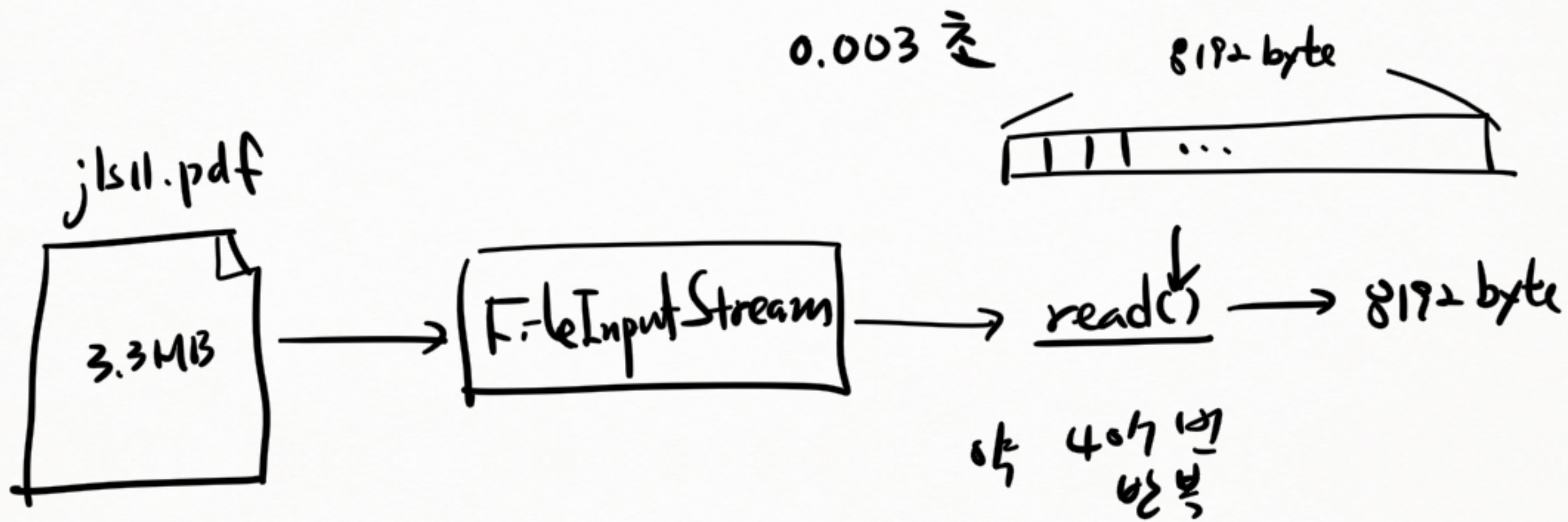




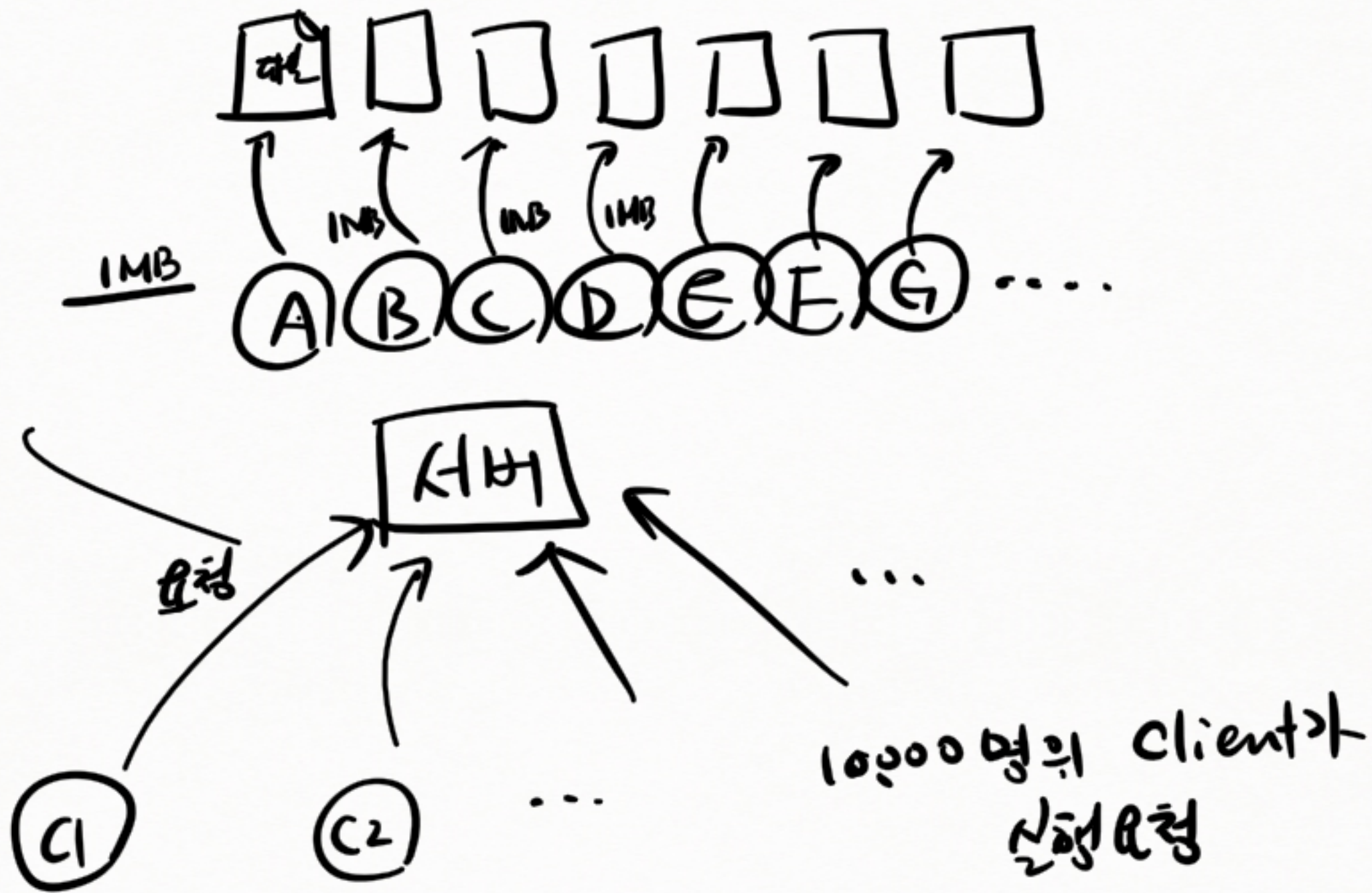
read() → 1 byte

3.3초
17.8초
36초

약 330만 번
반복



⑩. 000 MB
~~100MB~~



FileInputStream



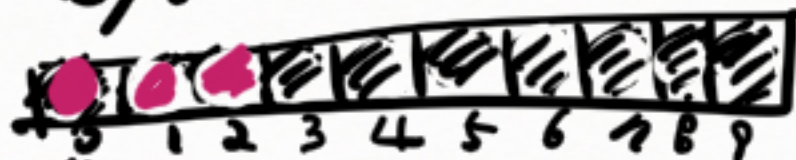
BufferedInputStream



read(byte[])

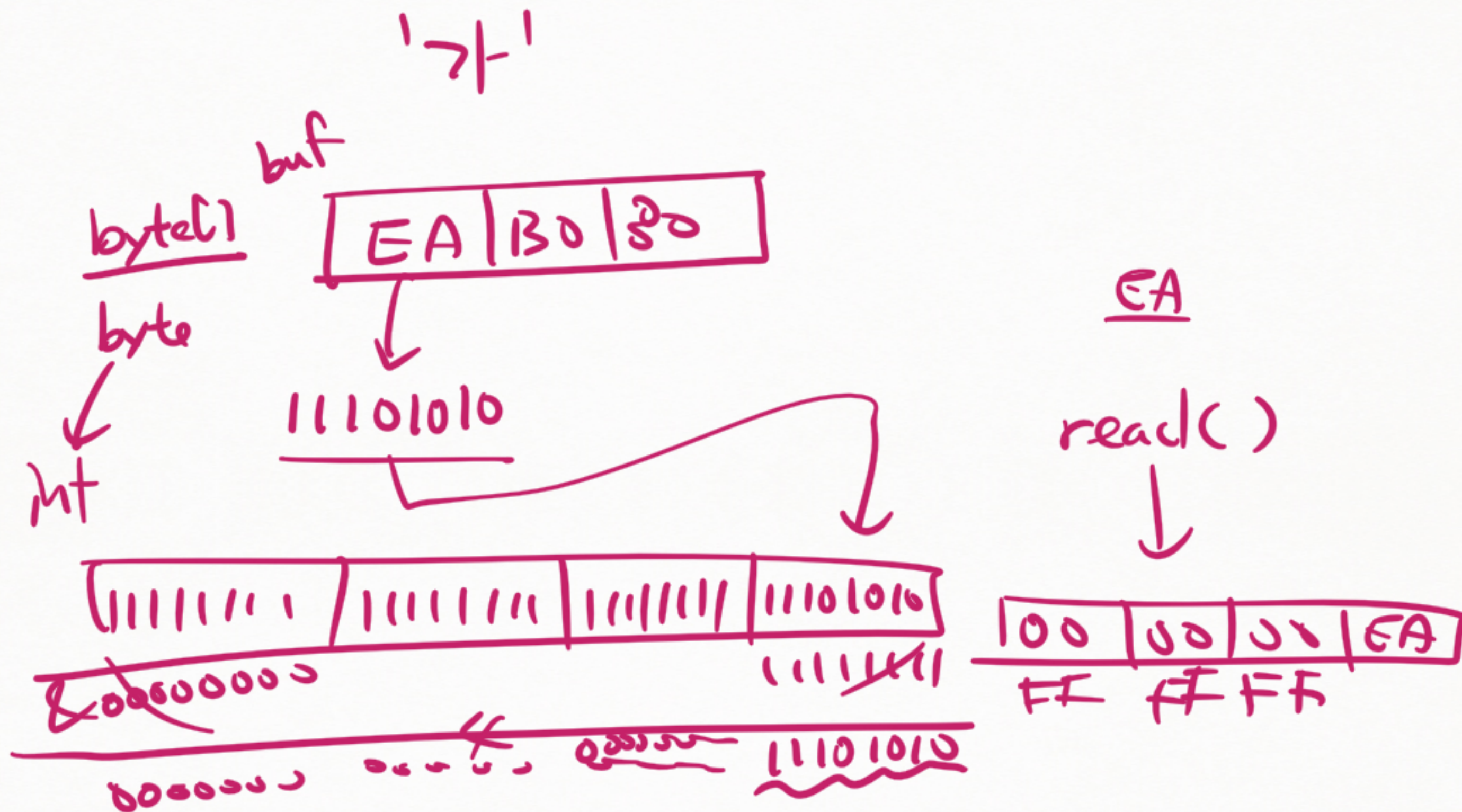
size = 3
cursor = 13

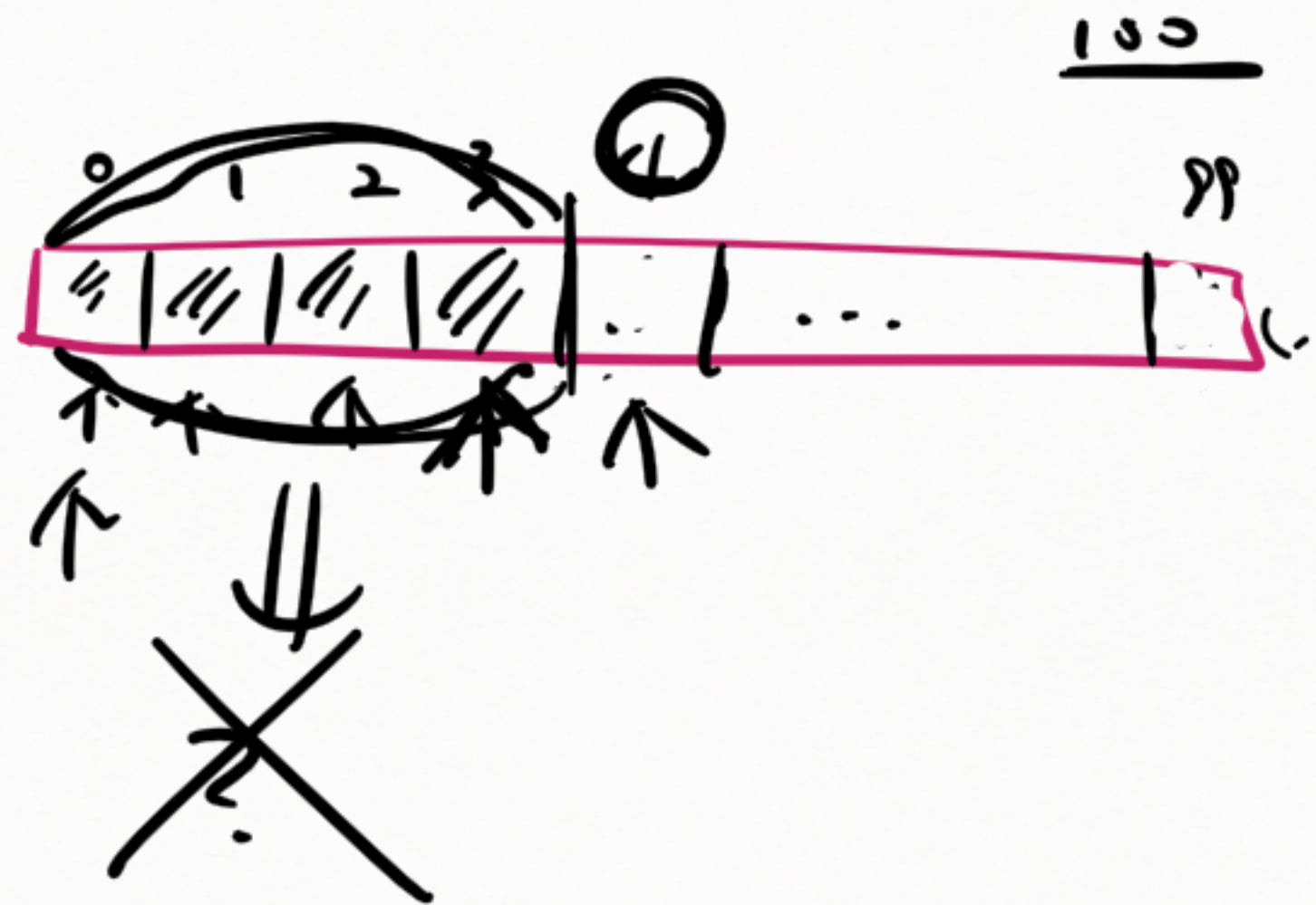
byte[] 10 byte

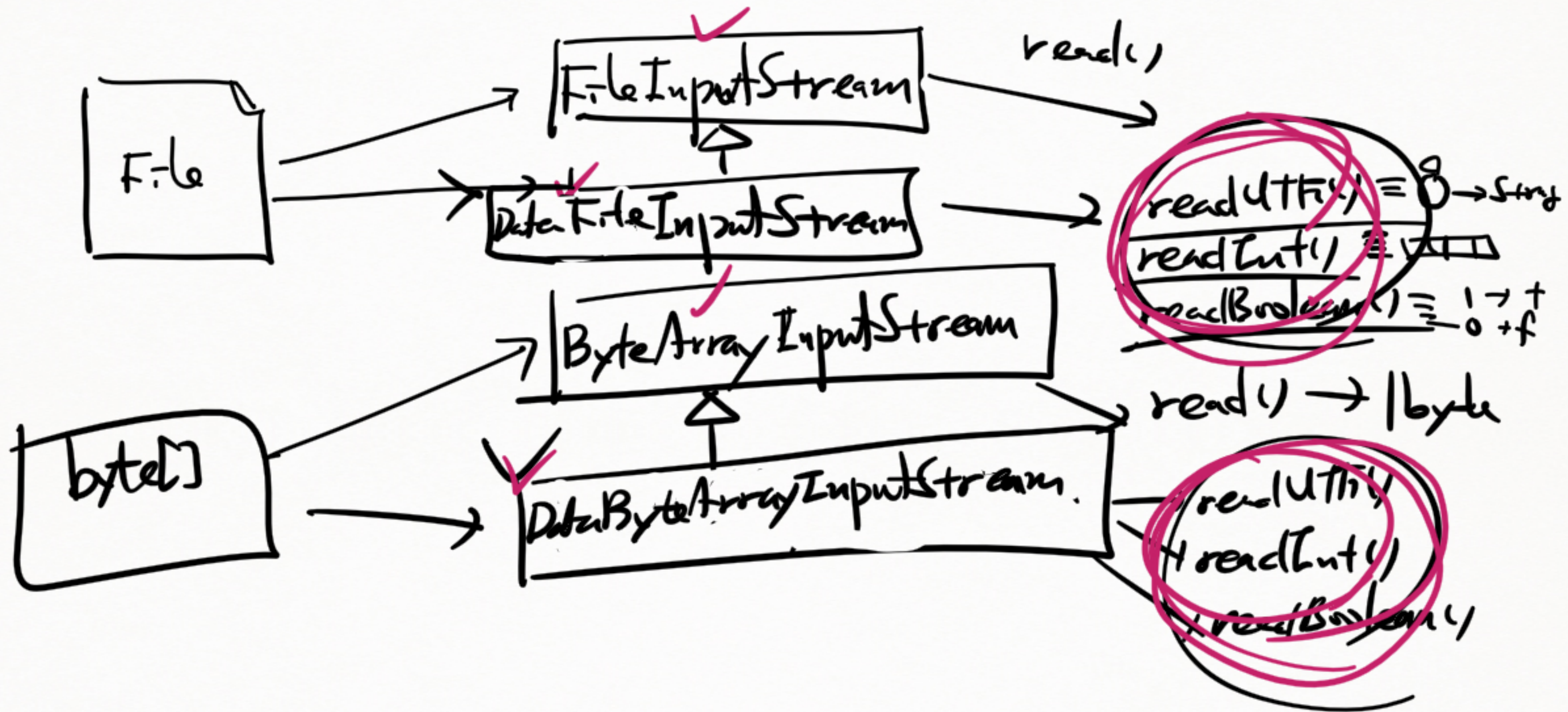


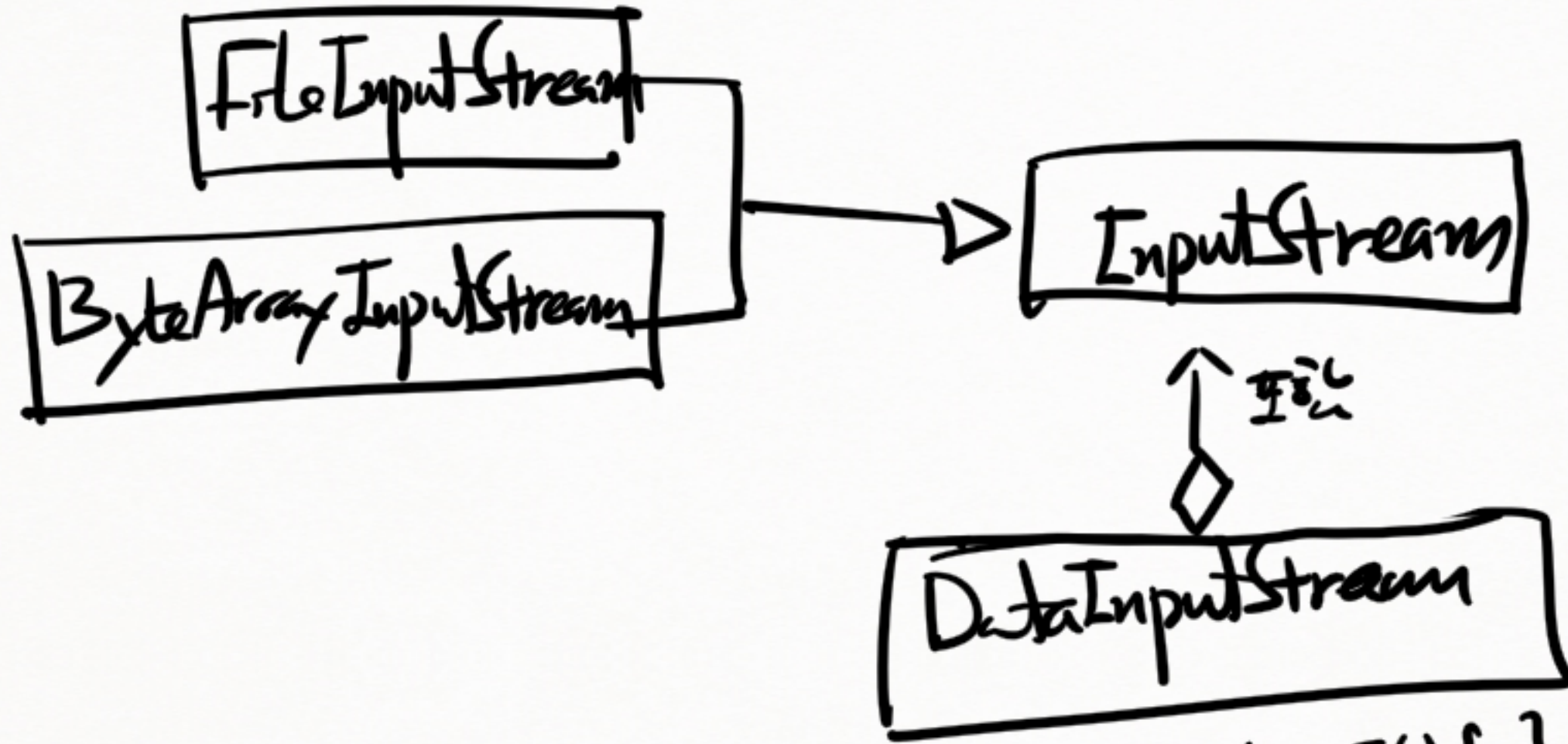
cursor

read() → 1 byte
⋮
read() → 1 byte
⋮
read() → =



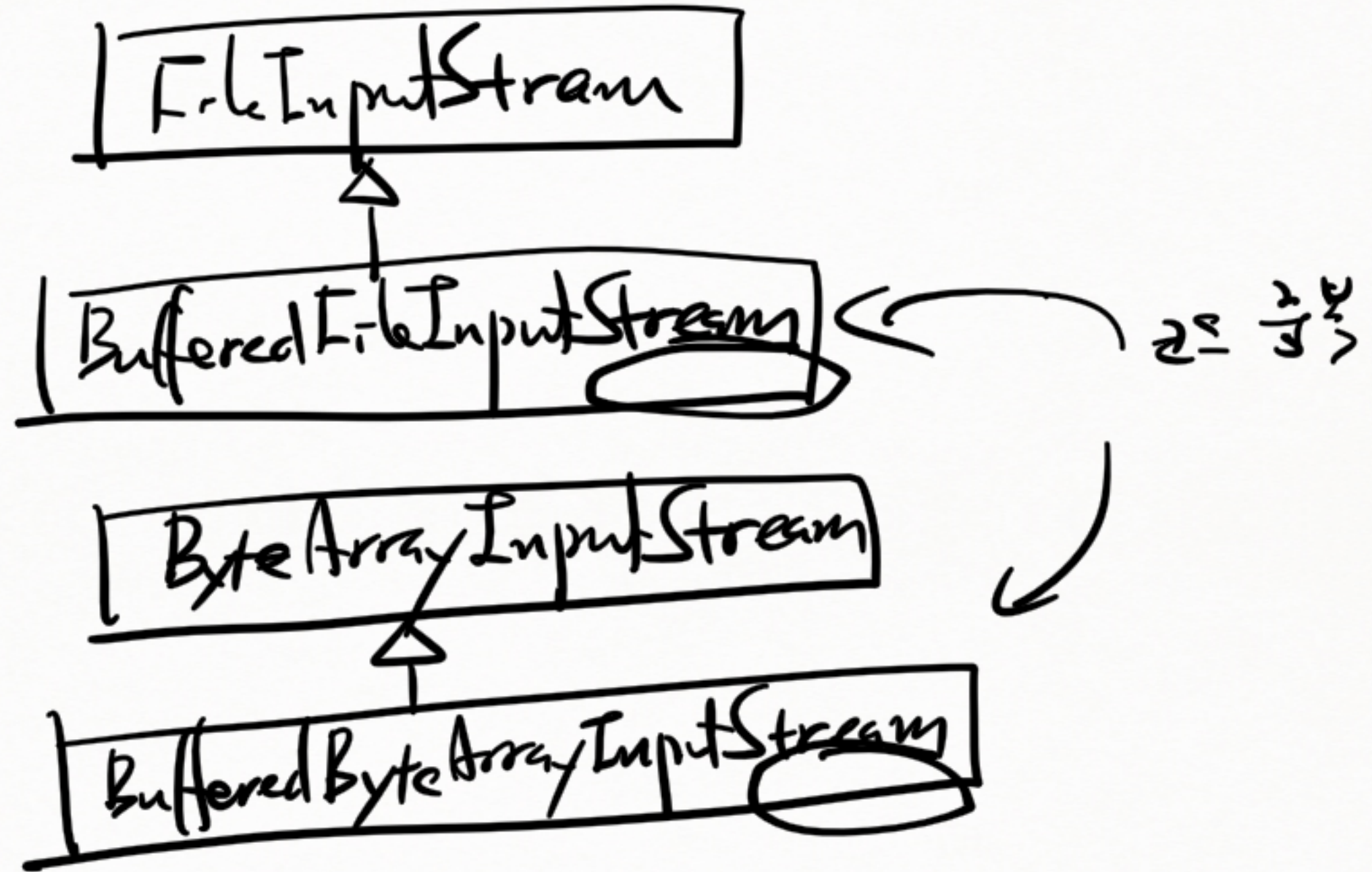


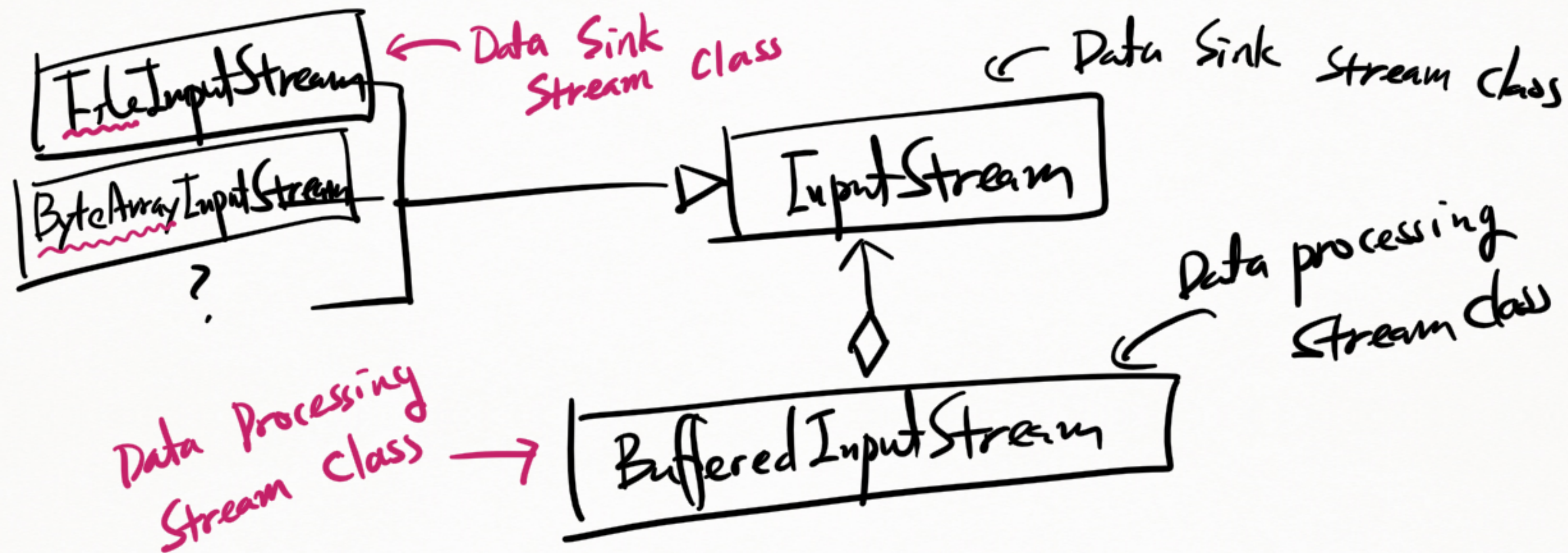




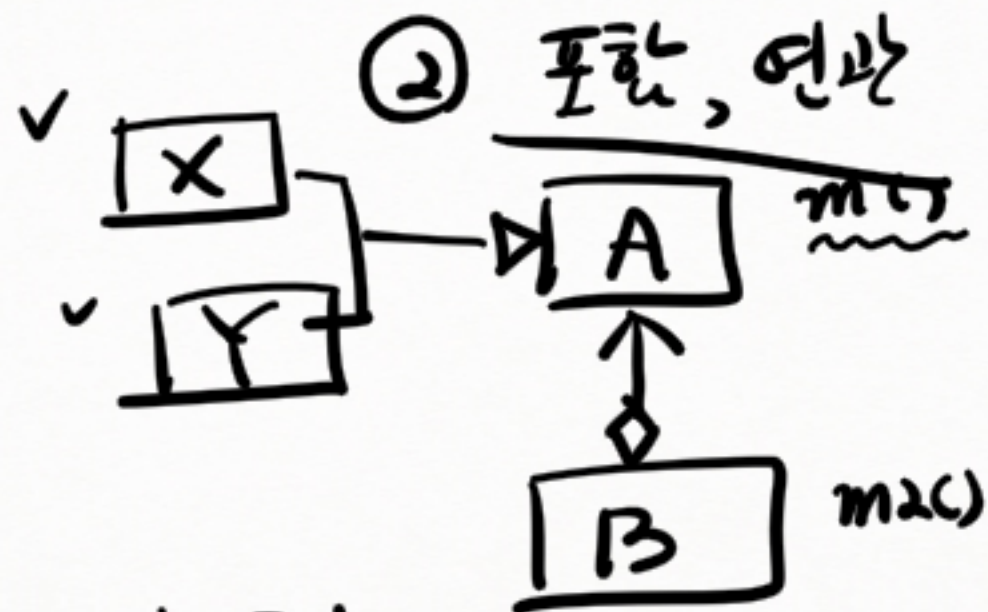
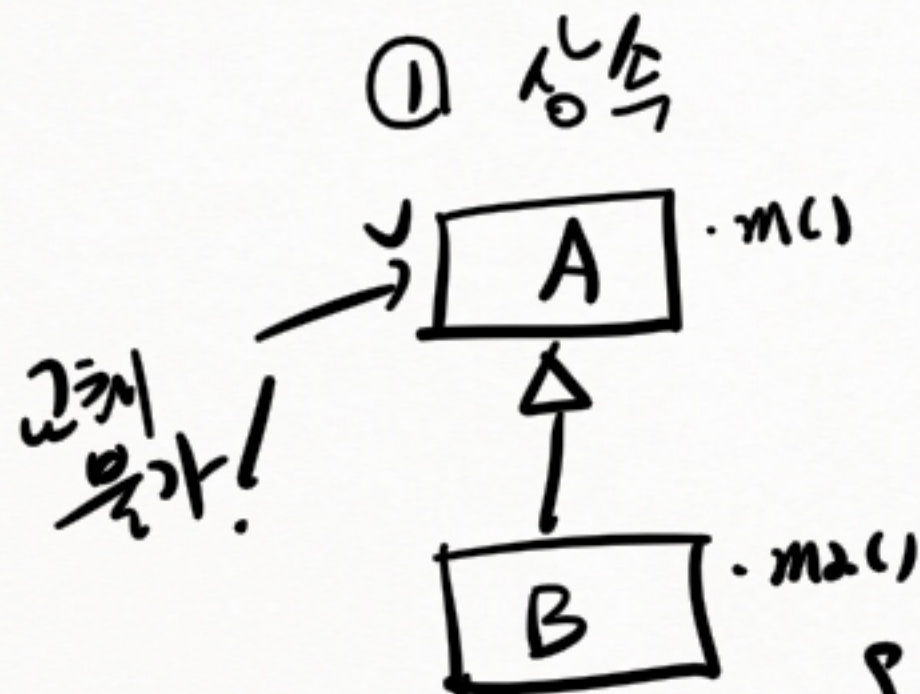
- readUTF() {-}
- readInt() {-}
- readLong() {-}
- readBoolean() {-}

7.4.2 (Decorator)





기능 확장



- ✓ 중복 최소화
- ✓ 재사용
- ✓ 1차가-용이
- ...

FileInputStream

+ 읽기 속도 높임



DataInputStream



BufferedInputStream

(문자열
자바 기본 타입
값으로
읽기)

- readUTF()
- readInt()
- readLong()
- readBoolean()

- byte[] buf = -
- int size
- int cursor

Data Sink Stream class

"Decorator" 패턴

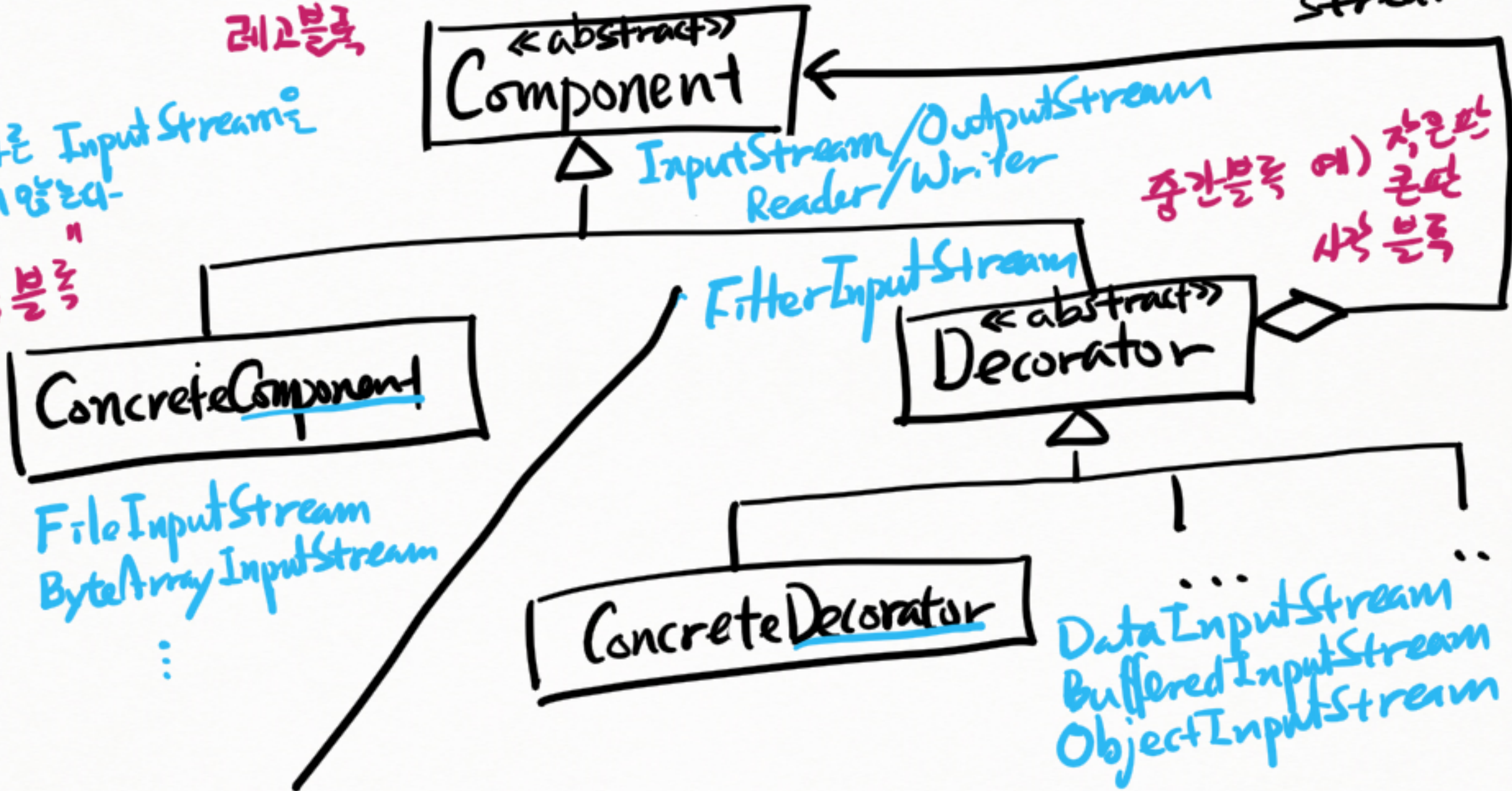
Data processing Stream class

레고블록

생성 시 다른 Input Stream을
받지 않는다.

"완성품 블록"

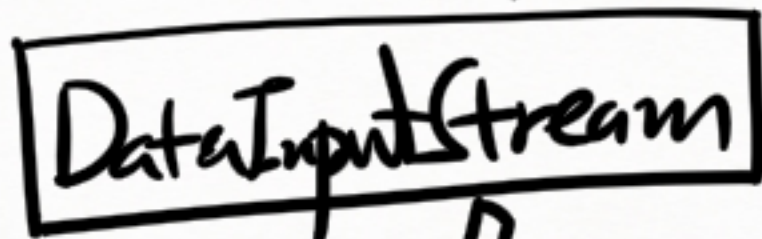
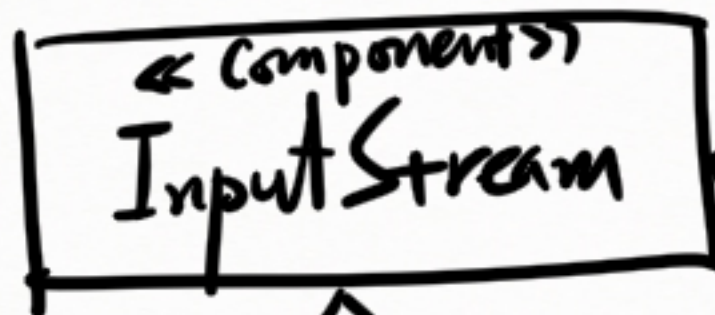
예) 인형
집
바퀴
성



중간블록 예) 자, 큰 판
사각 블록

...
DataStream
BufferedInputStream
ObjectInputStream

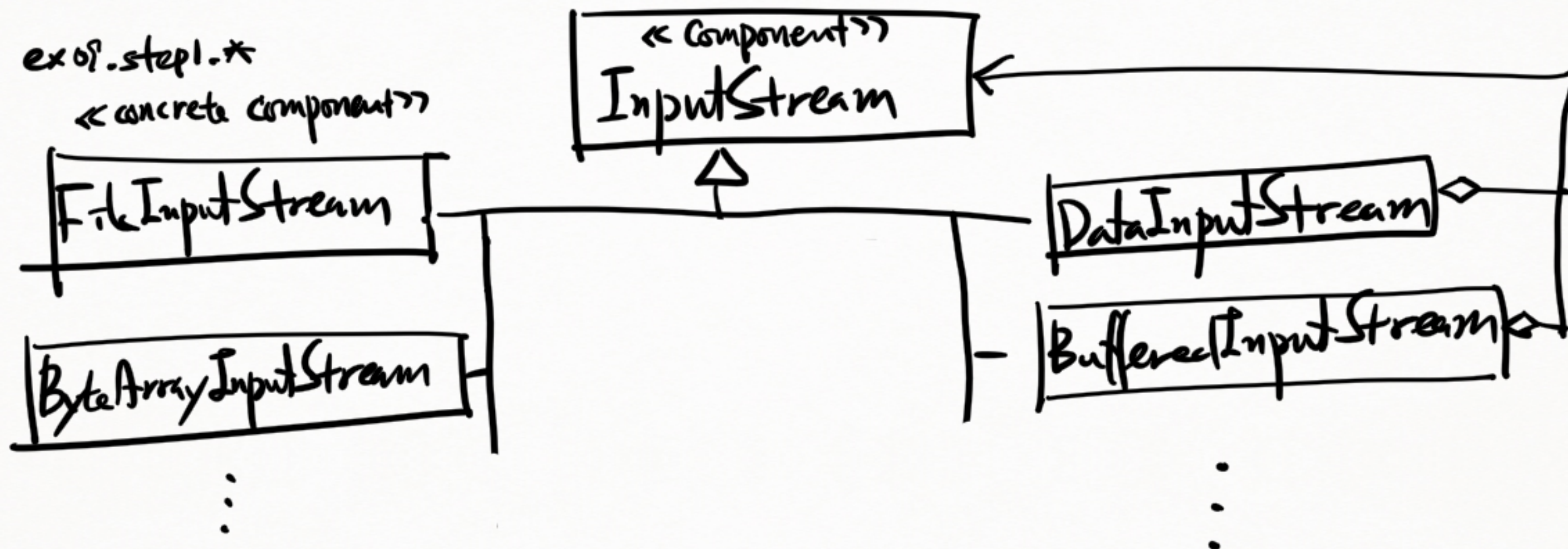
ex08.*



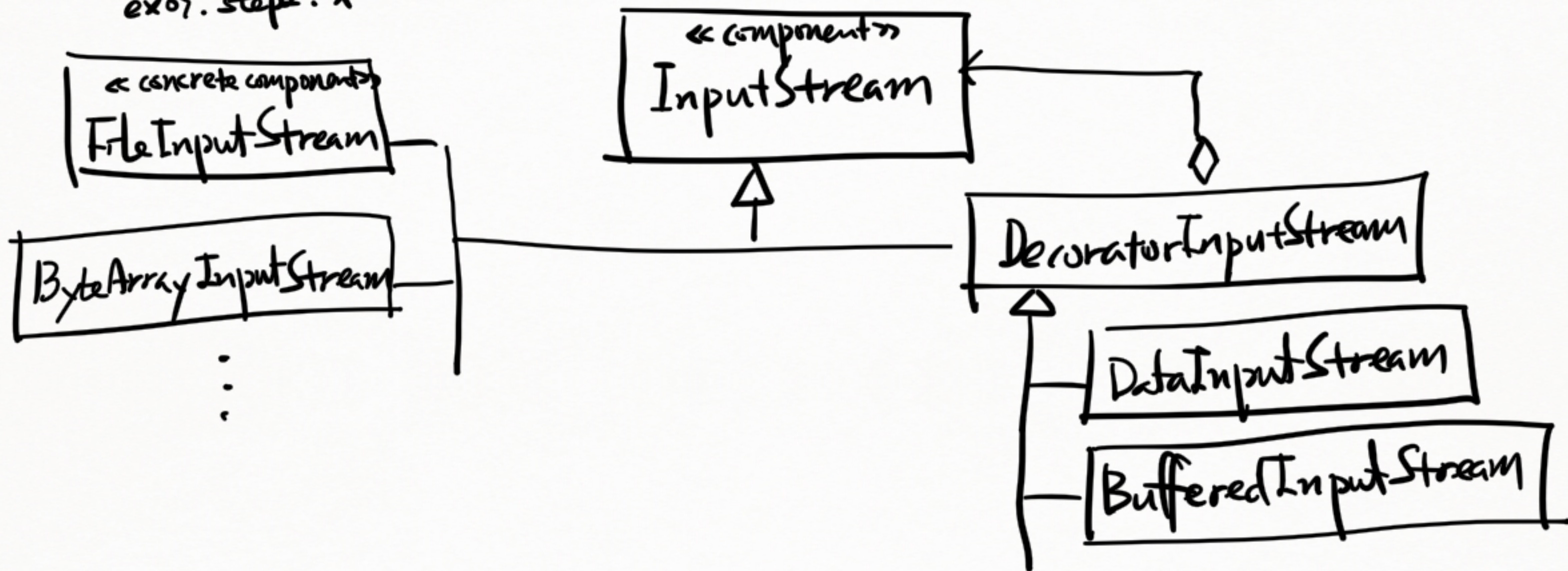
BufferedInputStream = InputStream 자원의
개체가 아님!
포함물! (포함물 아님!)

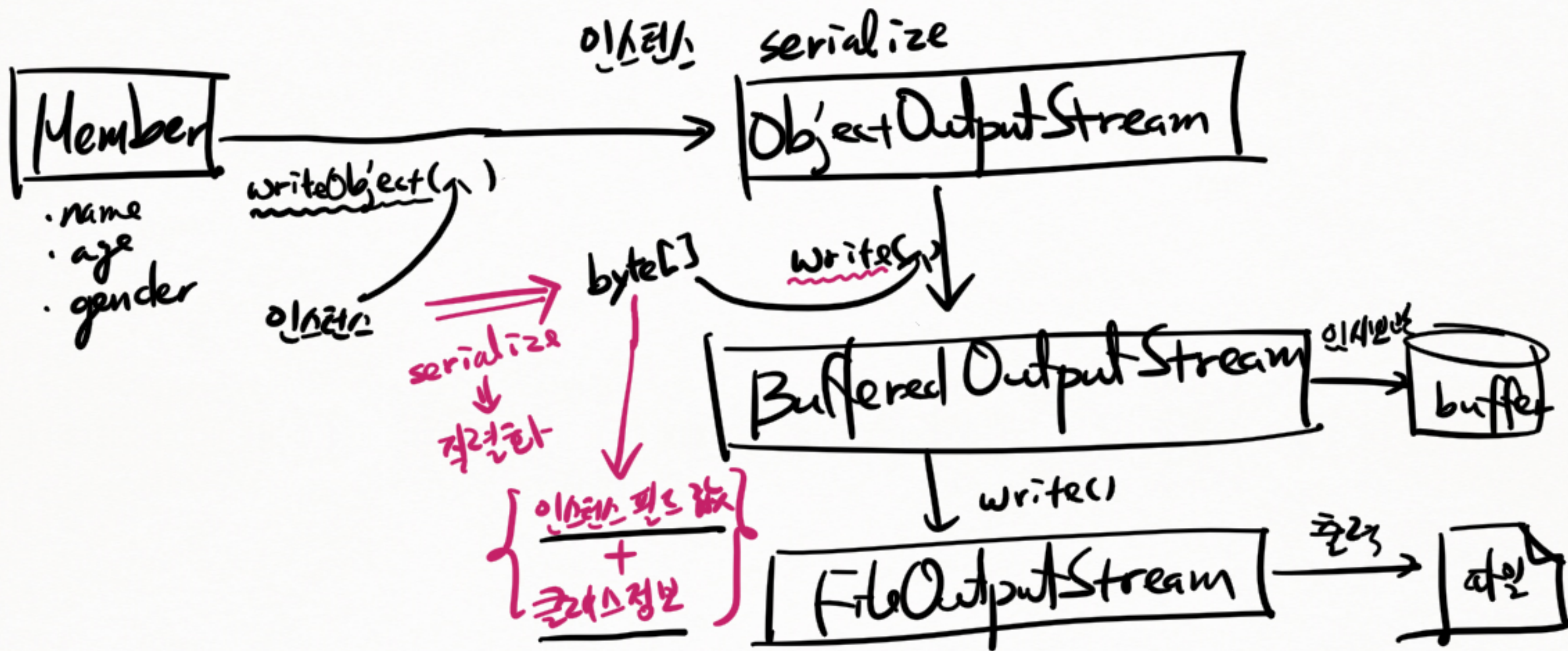
ex of. step 1. *

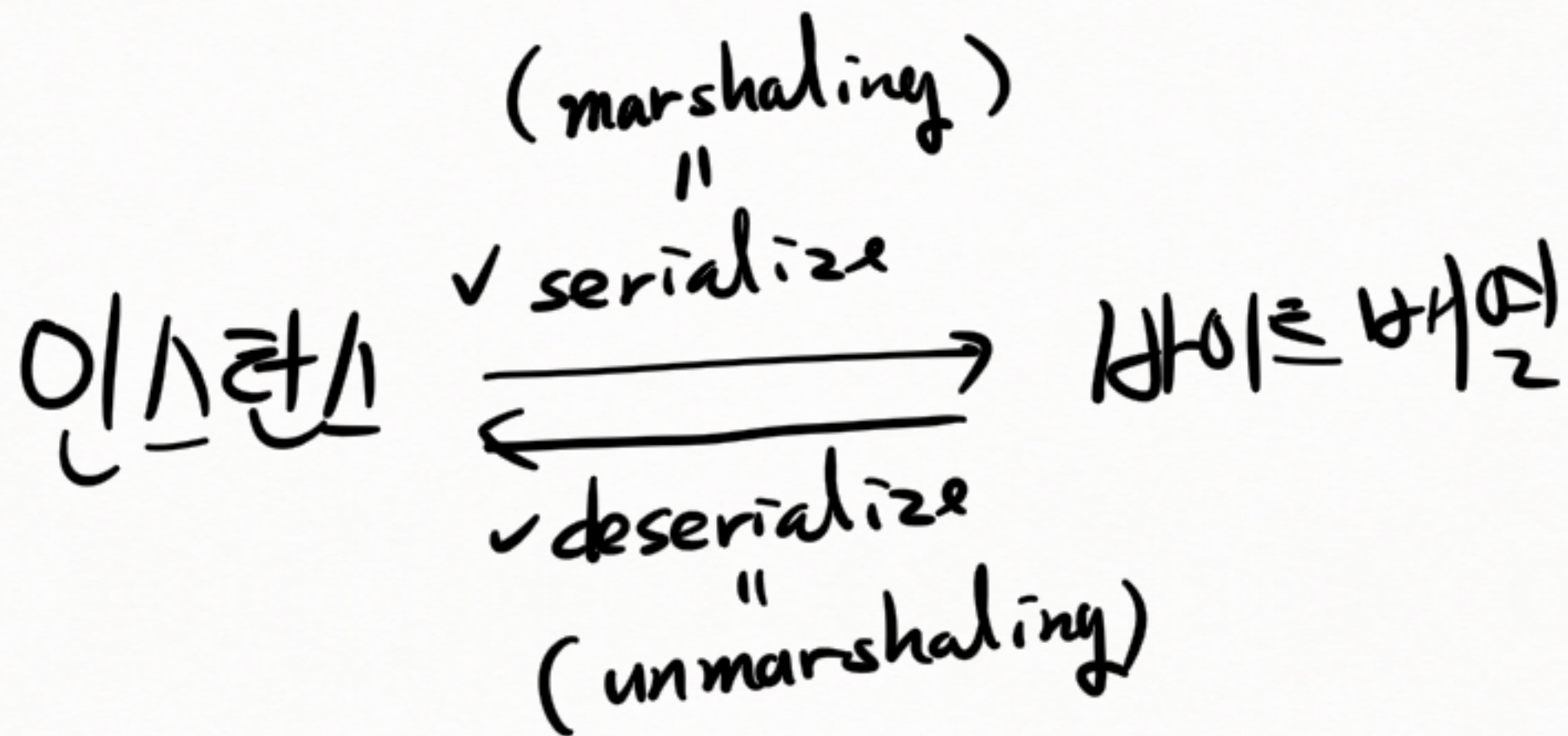
<< concrete component >>

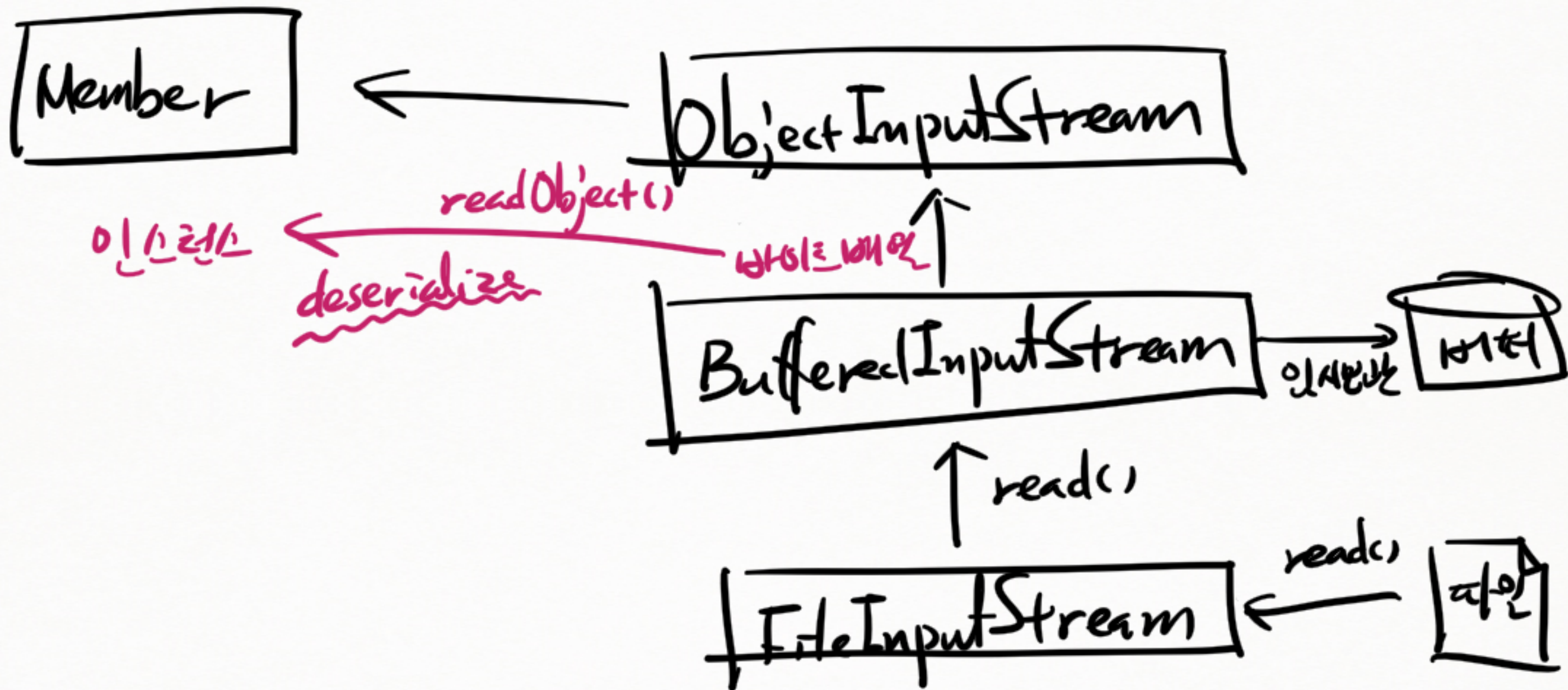


ex09. step1. *

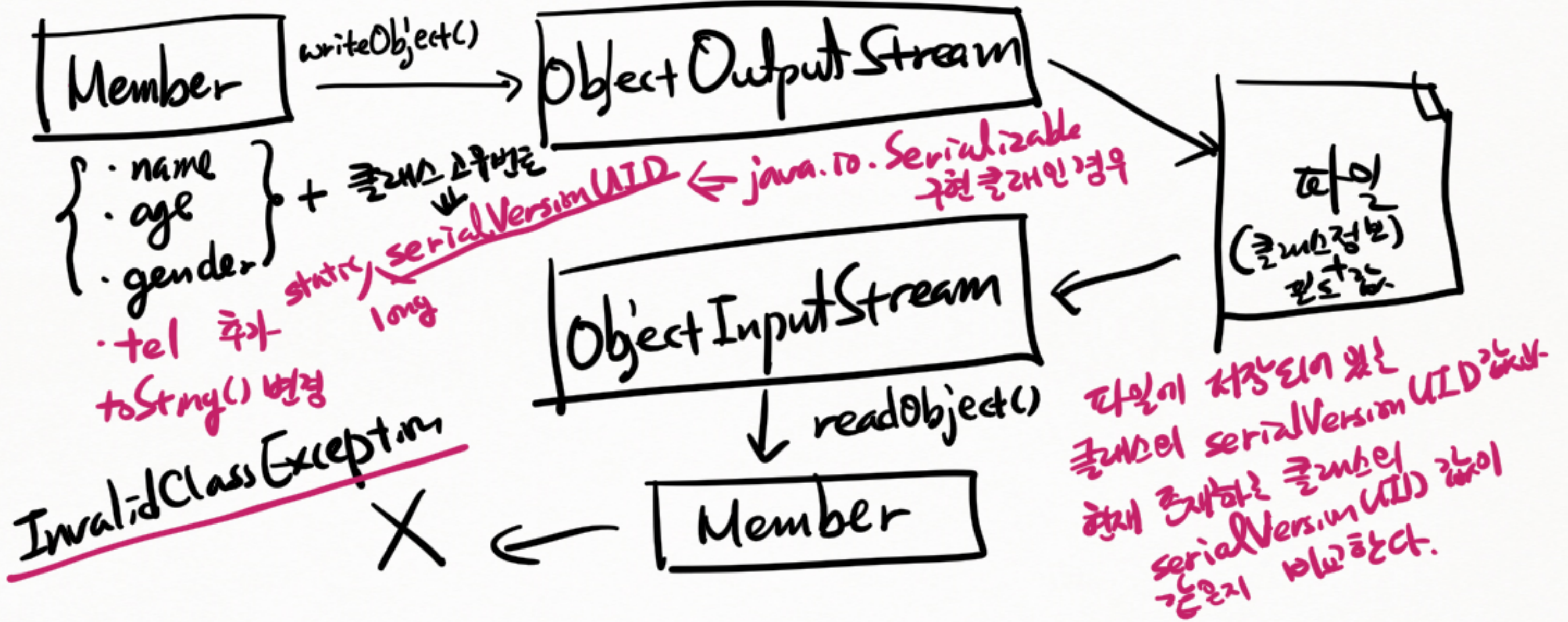


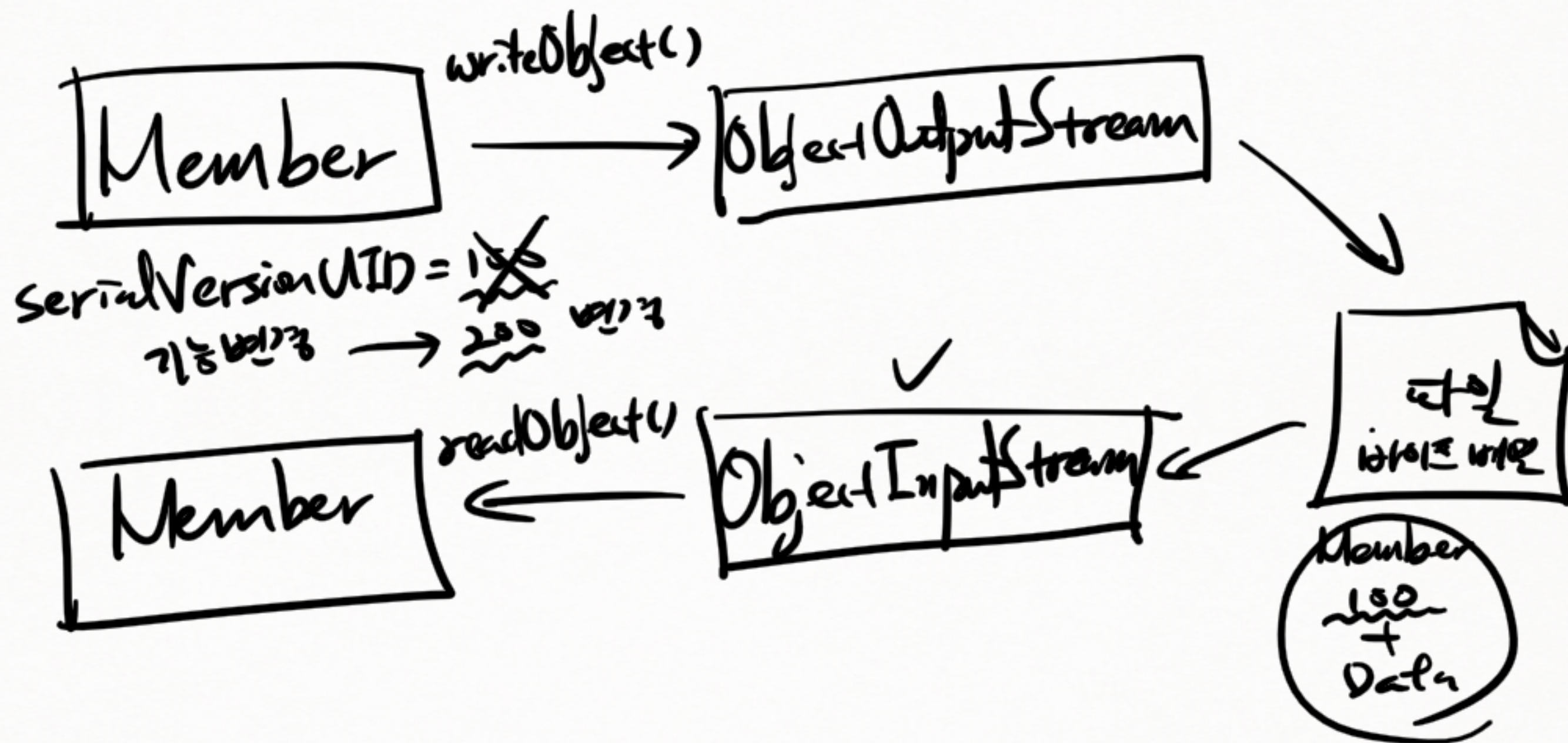






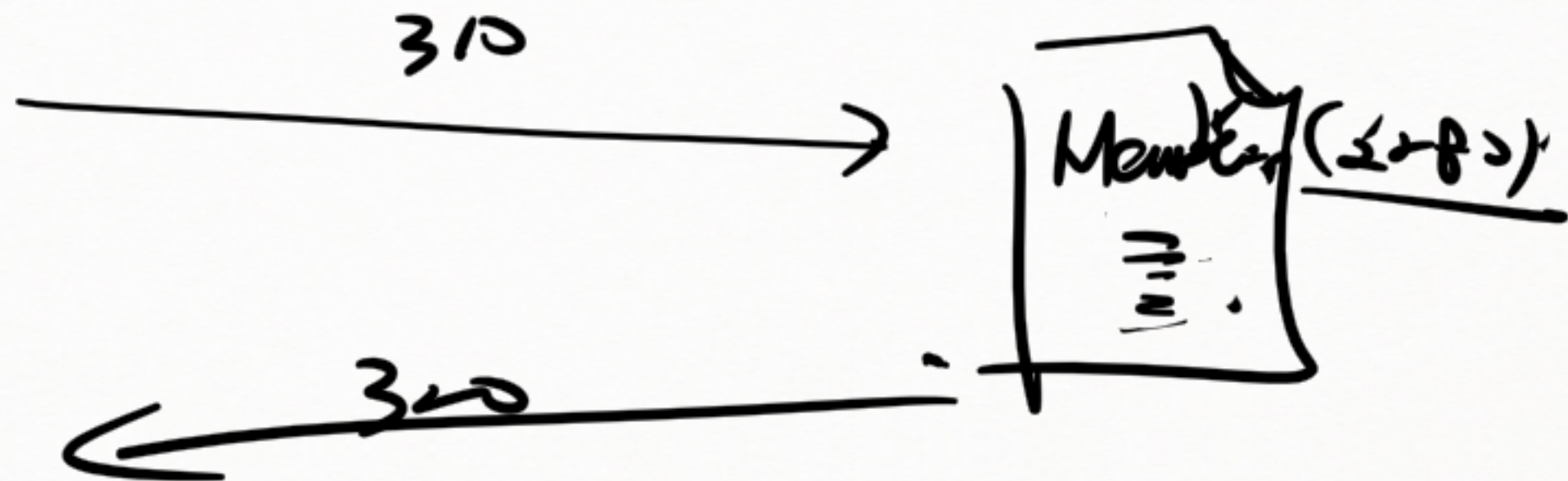
* serialVersionUID
변하는 순간마다 자동적으로 바뀌고, 이걸로 클래스가-변경되었는지 이걸로 바뀐다. Unique Identifire

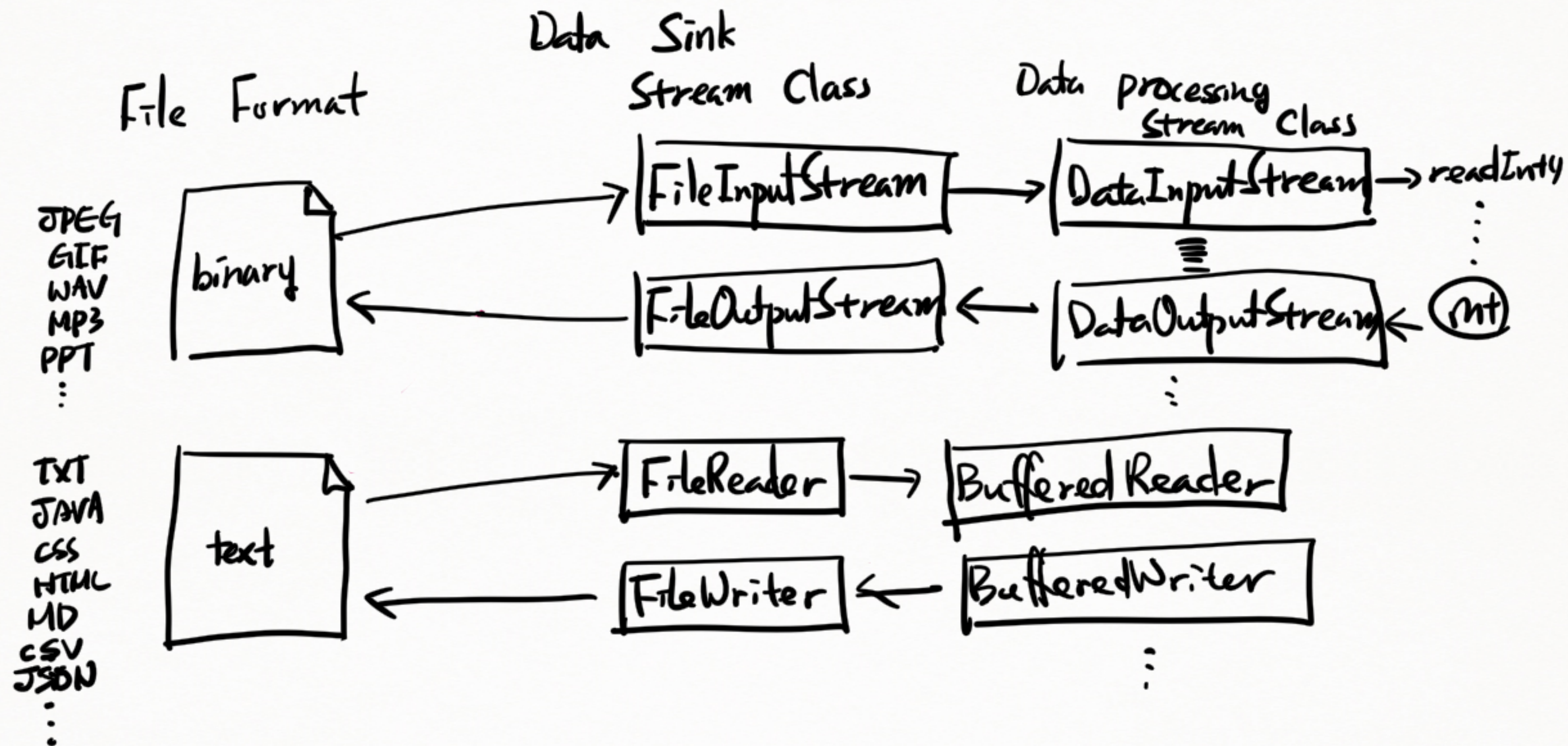




~~1048~~
2280 Member

- name
- age
- gender
- ~~sex~~





- 30-a : FileInputStream/FileOutputStream
 - 30-b : DataInputStream/DataOutputStream
 - 30-c : BufferedInputStream/BufferedOutputStream
 - 30-d : ObjectInputStream/ObjectOutputStream
- } file format은 알아야
 읽는 것과 쓰는 것
 ← 다른 언어와 호환이 안됨

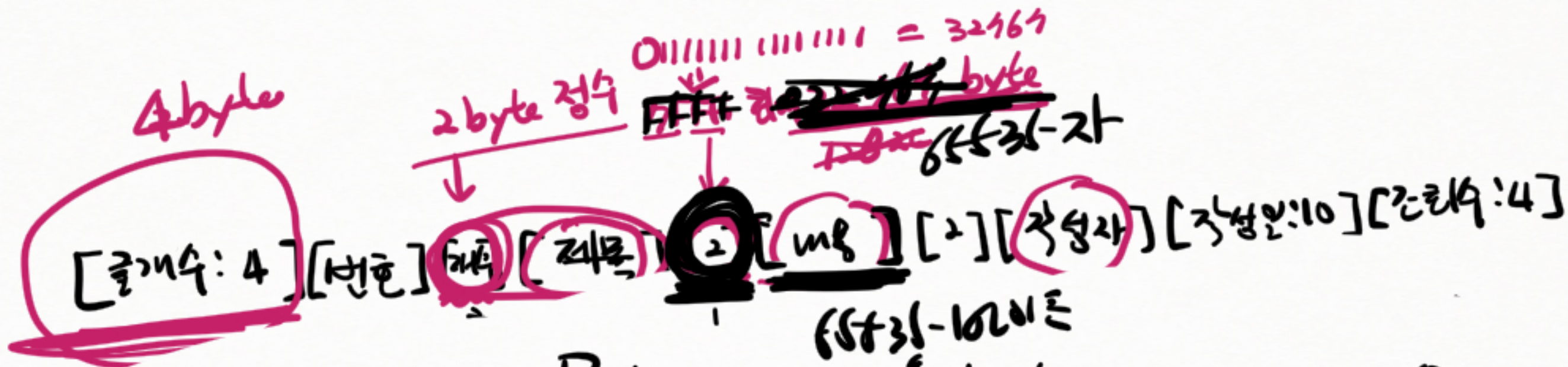
30-e : 리팩터링

31-a : FileReader/FileWriter text 파일 포맷

31-b : BufferedReader/BufferedWriter

31-c : 리팩토링 I

31-d : 리팩토링 II



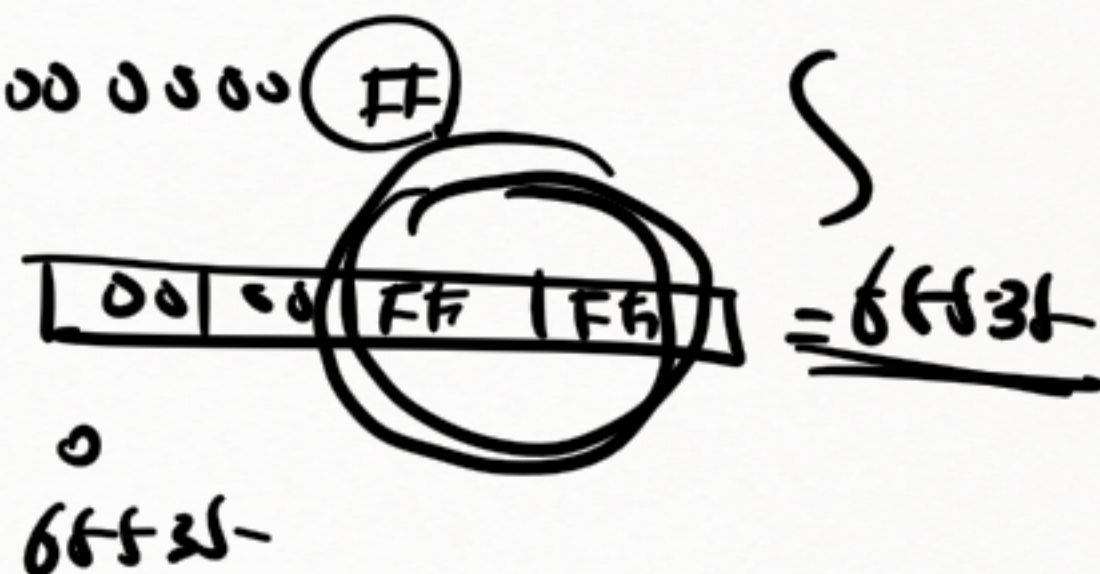
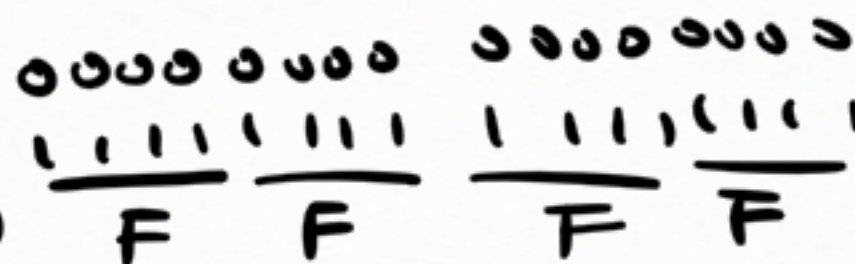
int ^{len} a = 65535;

write(a >> 8);

00 00 FF

write(a);

00 00 FF



XML
HTML

(일반적인) tag = markup (표판기)
data를 식별 (meta-data)

시작태그

<no> data </no>

↑ 콘텐츠

필요한 Data

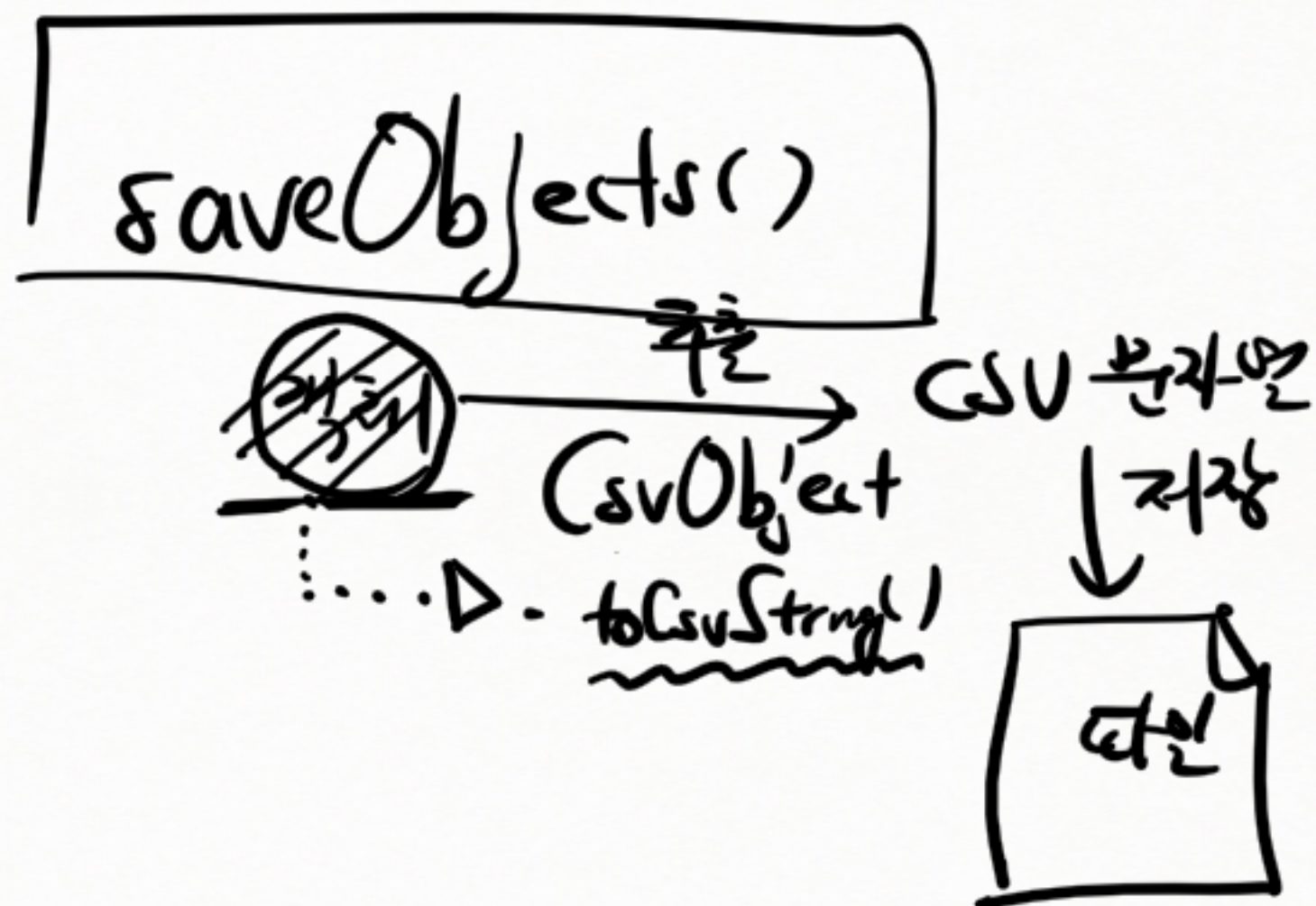
element (개별자)

끝태그

<h1> 제목 </h1>

 aaa

`saveBoards()`
`saveMembers()`
`saveProjects()`
`saveTasks()`



• Data 나열

- Text
- Data 형



✓ XML

✓ JSON

JavaScript
Object
Notation

<수업> 00 <수업>
<이름> 나 <이름>

{
수업: 00,
이름: 나
}

Android

Java/Kotlin

iOS

Objective-C
Swift

Windows

JavaScript

Linux

macOS

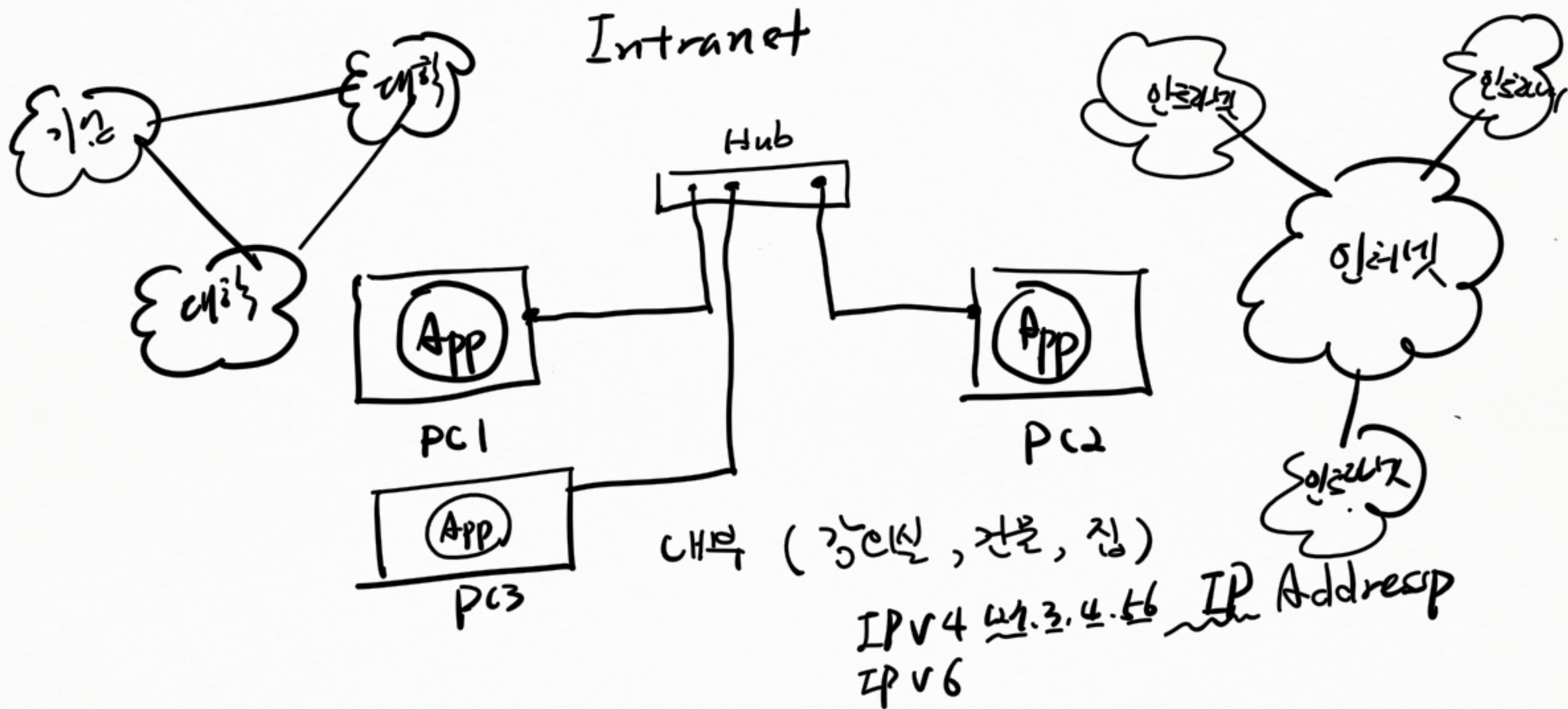
Data (예: 기성 데이터)

기성 데이터

• Java
• PHP
• Go ...

Web Server

• Go
• Java
• Python
• PHP
• ASP
• ...



0
0/4202

0.0.0.0 = 4 byte

1111.1111.1111.1111

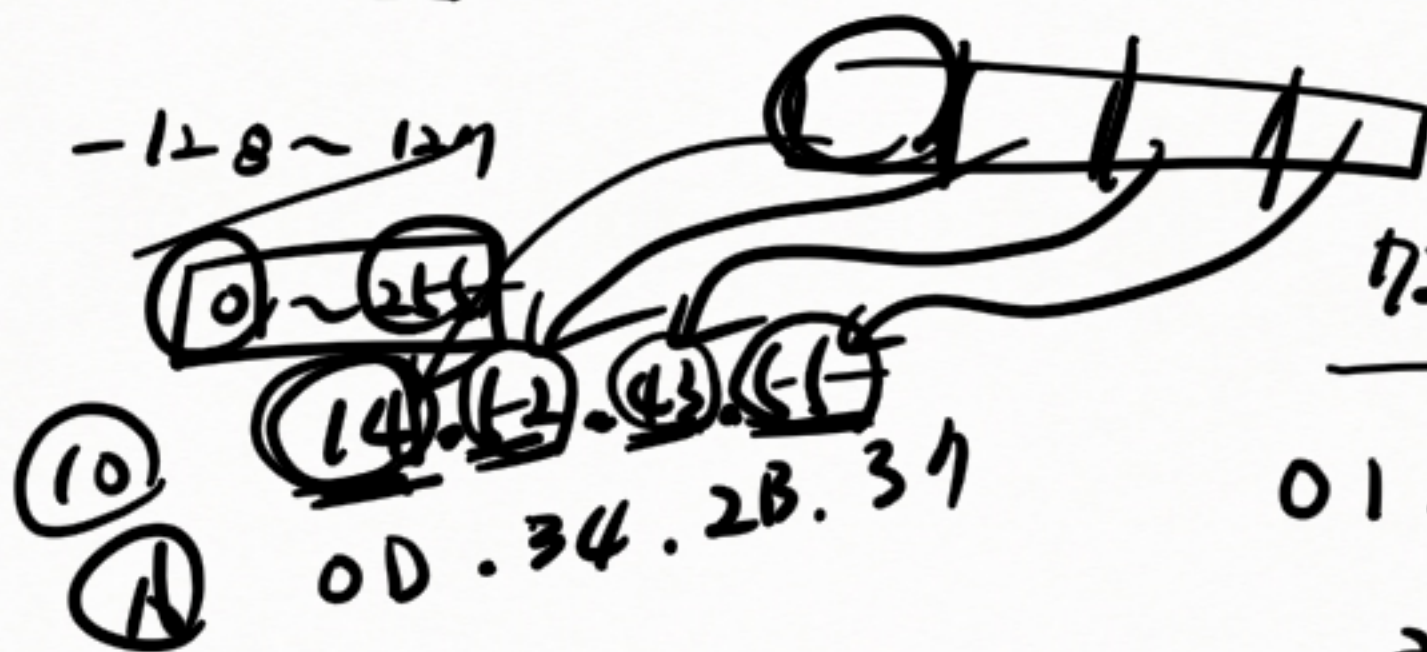
127.354.117

010-1234-4444

1 byte 1 byte

221,522,743

23
23
172



12001-1221234

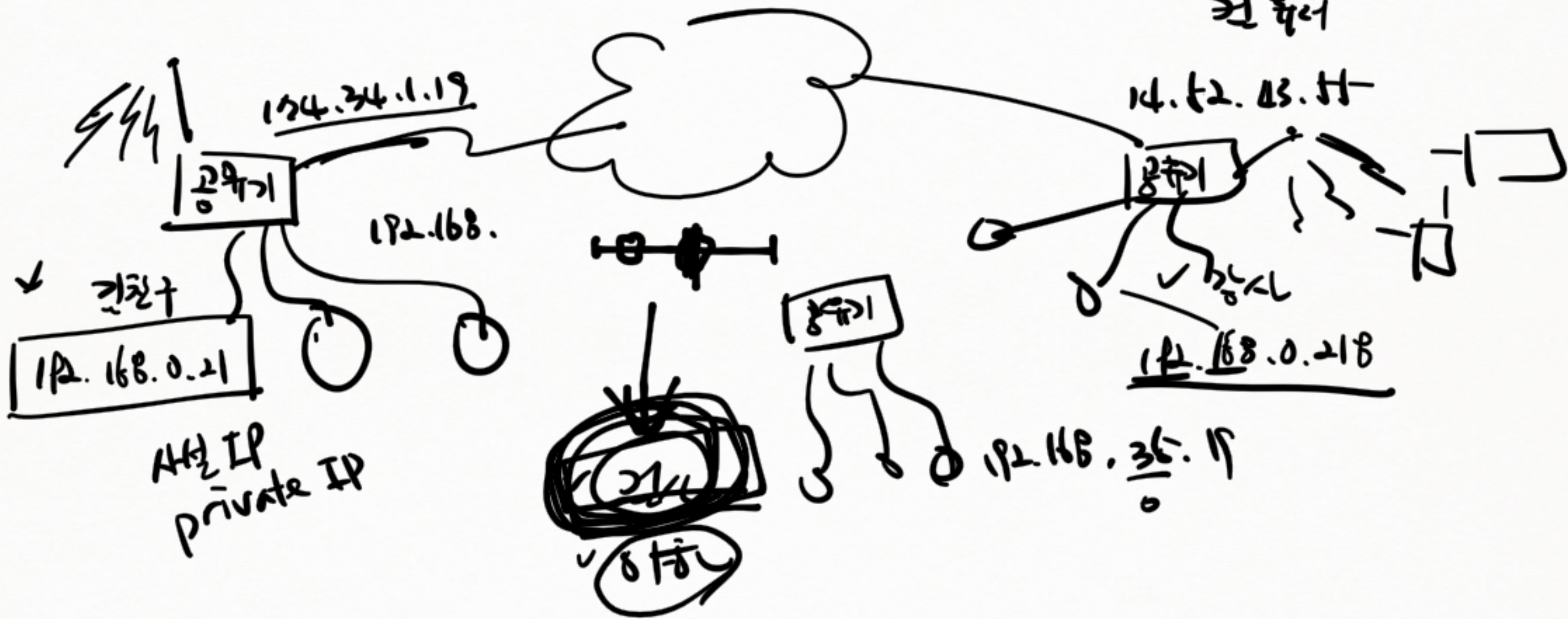
01234...

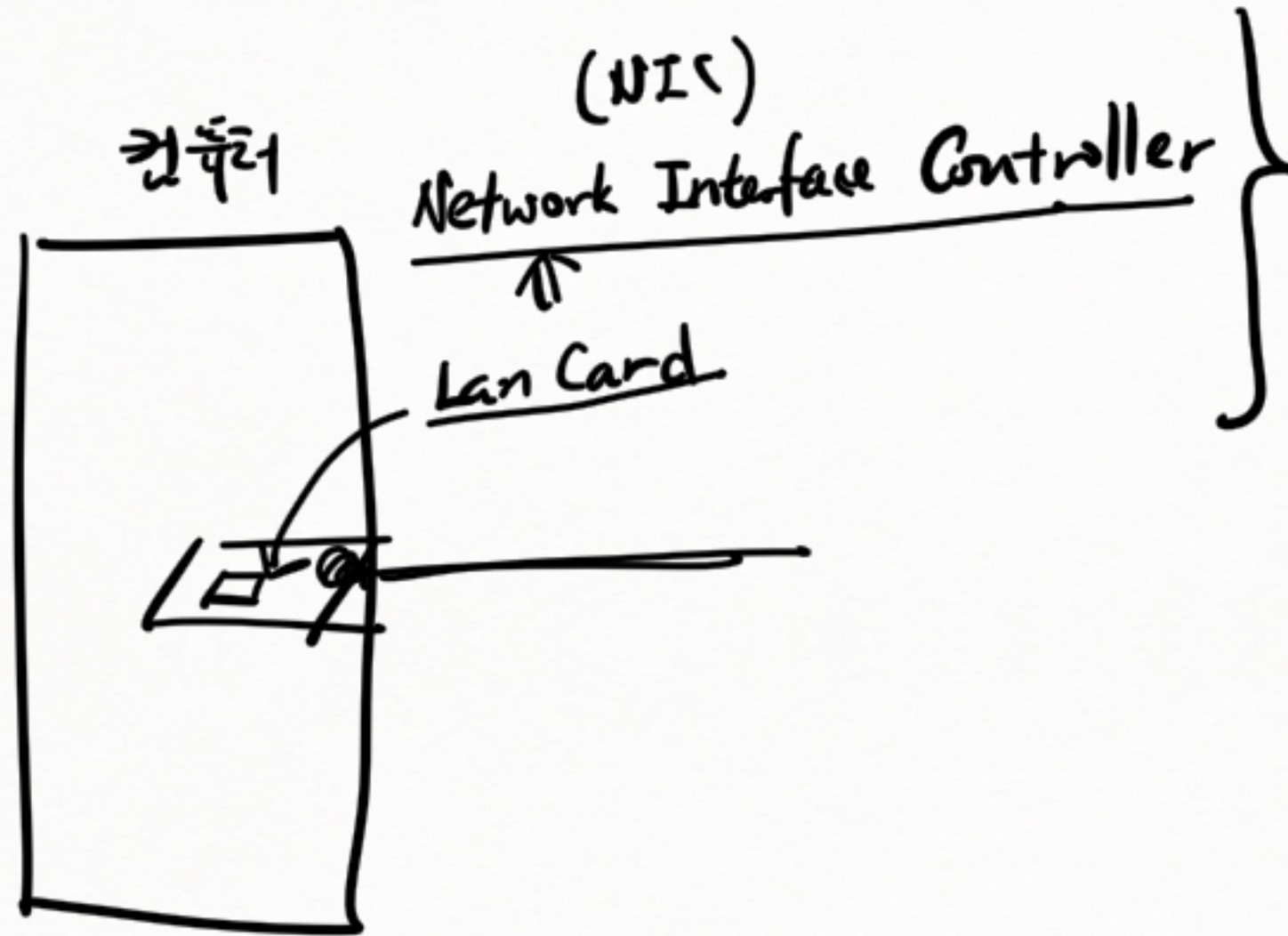
221522743. 429-

v4

IP Address

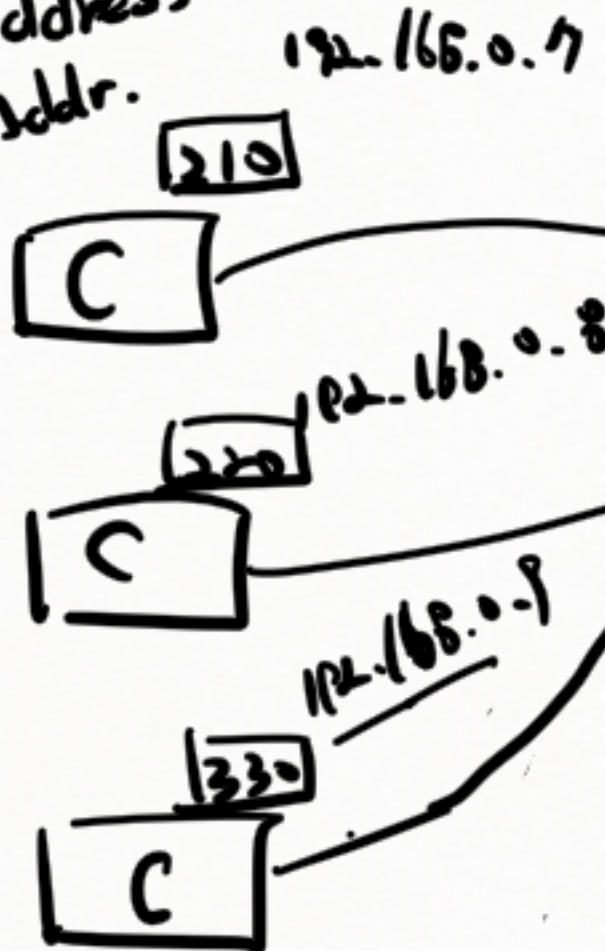
4 Byte = 4바이트
32비트





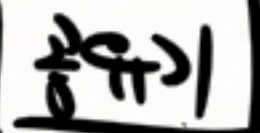
Lan Card 구분
||
NIC Address
⇓
MAC Address
{
Media
Access
Control

MAC Address
NIC " Addr.
Lan Card
네트워크 카드

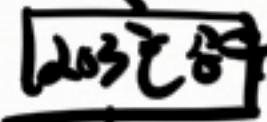


스위치
+

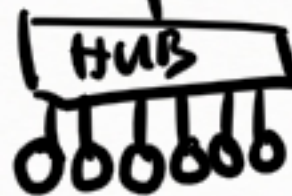
192.168.0.x



210 - 192.168.0.7
220 - 192.168.0.8
330 - 192.168.0.9

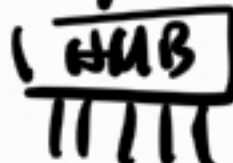


192.168.1.*



203H

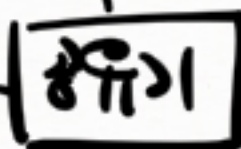
192.168.1.24



204H

192.168.1.*

AP
+

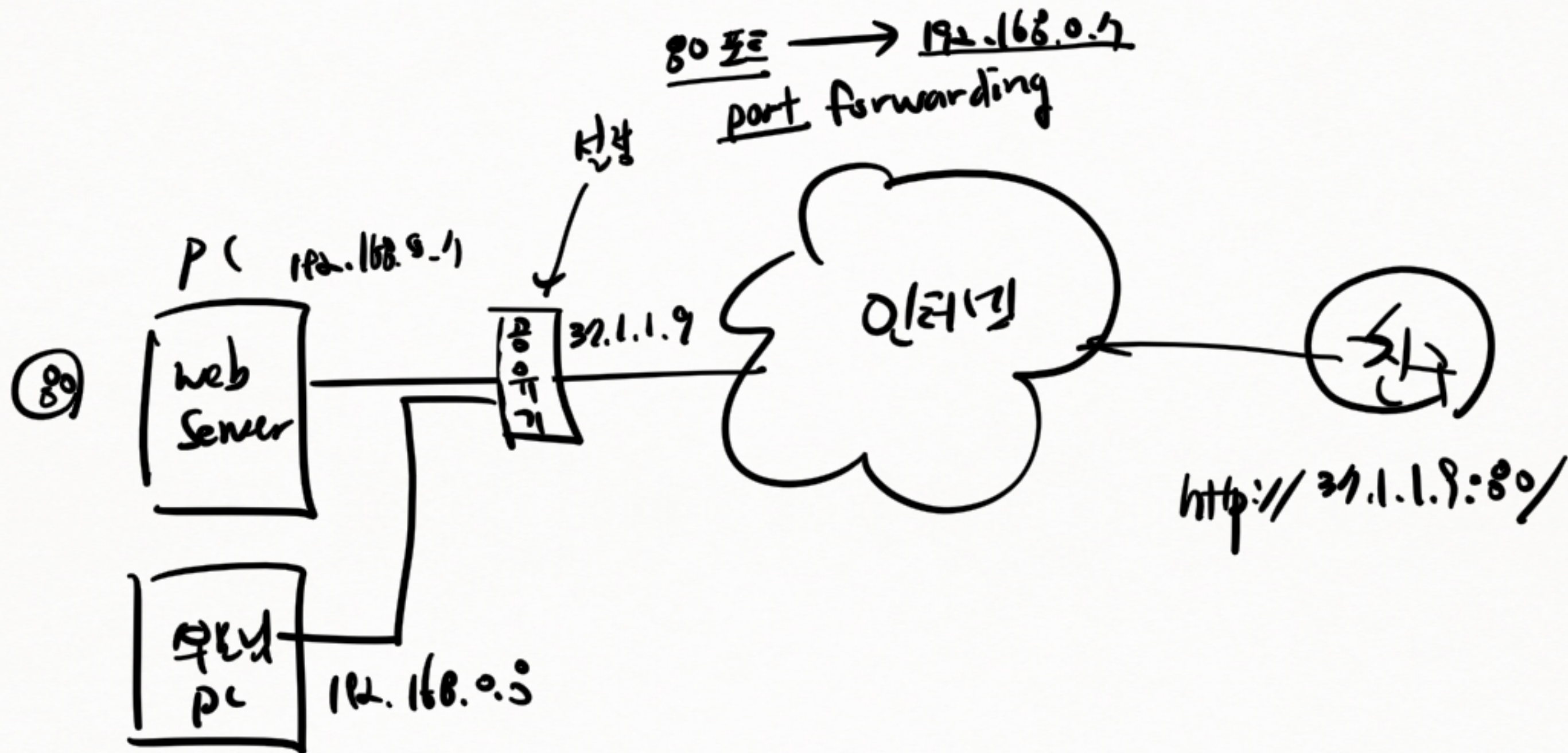


노드
번호

315-2

192.168.1.22

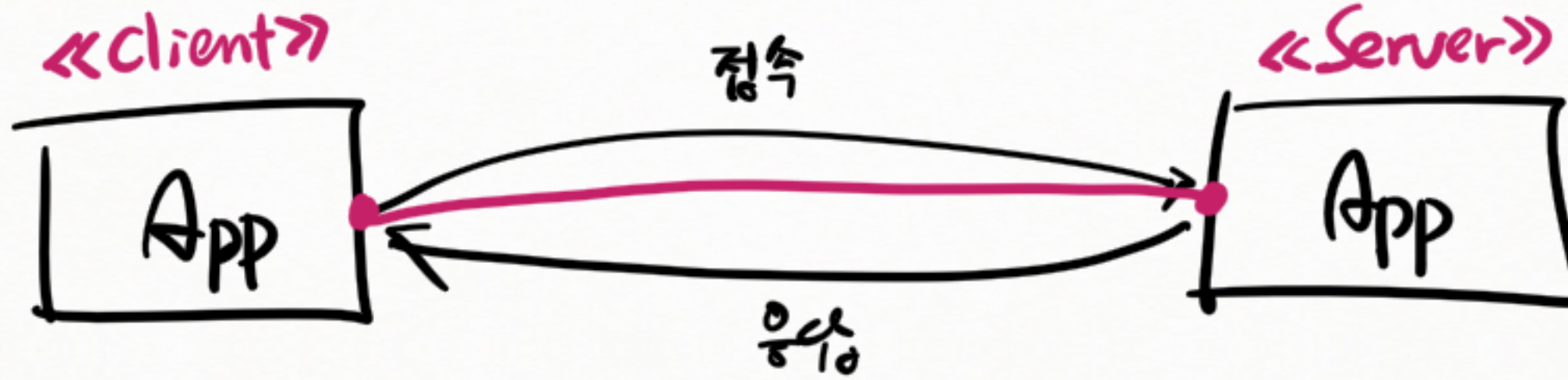


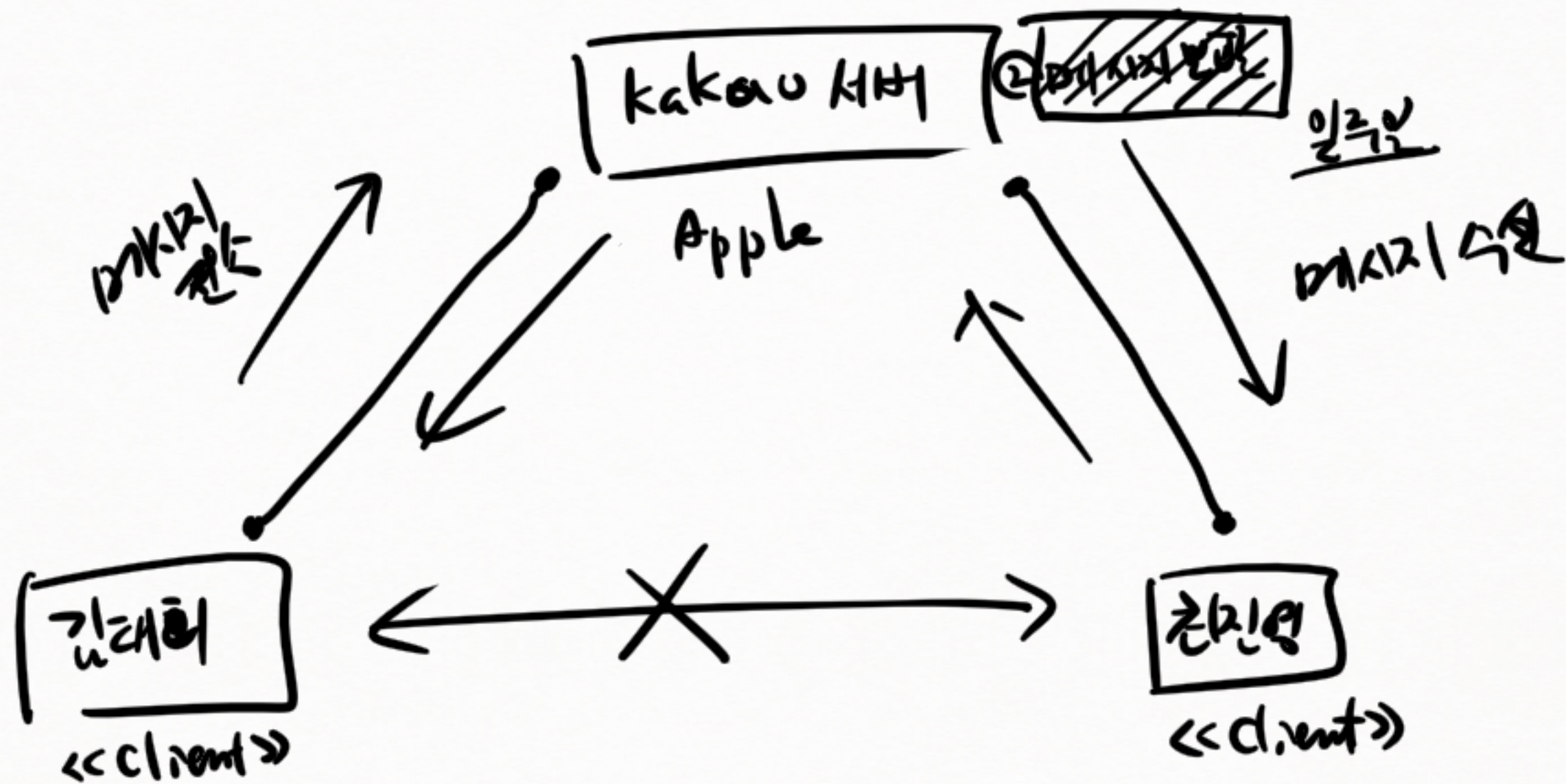


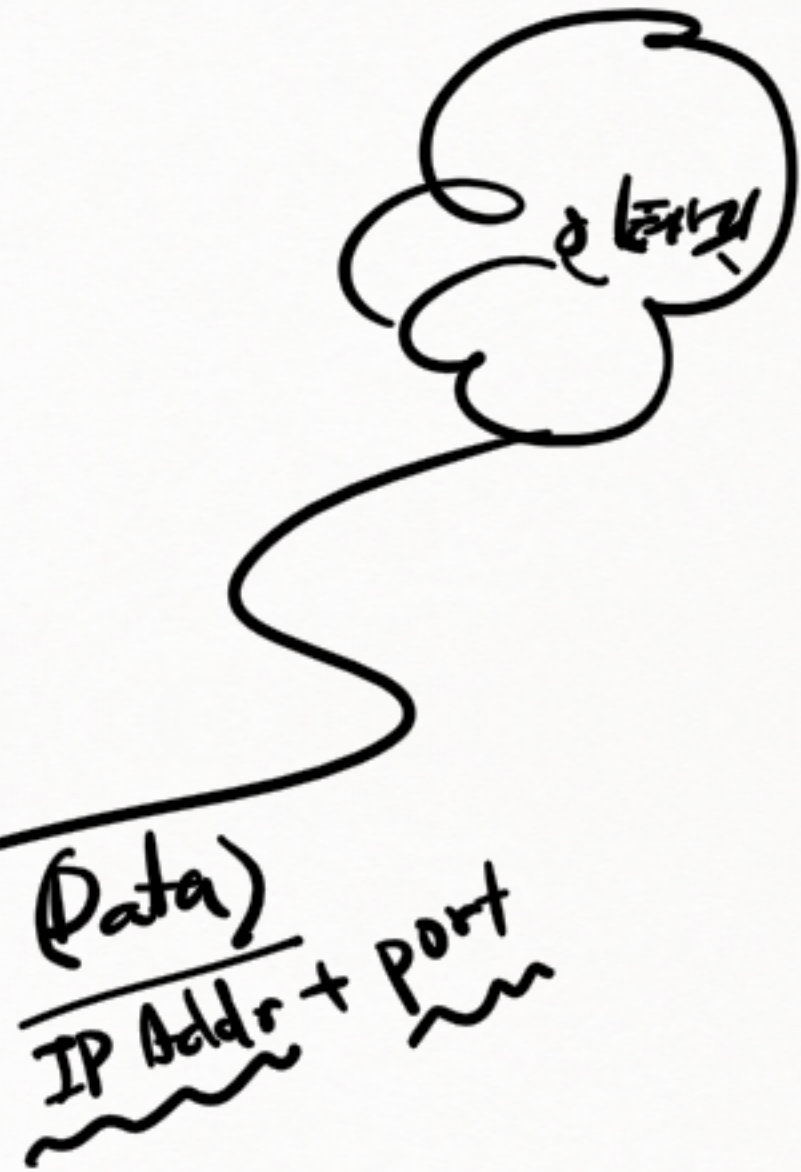
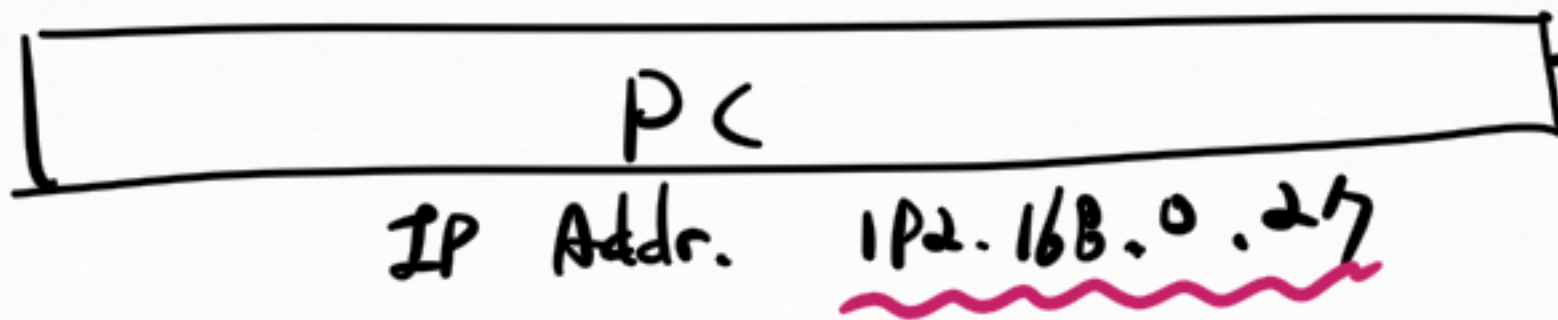
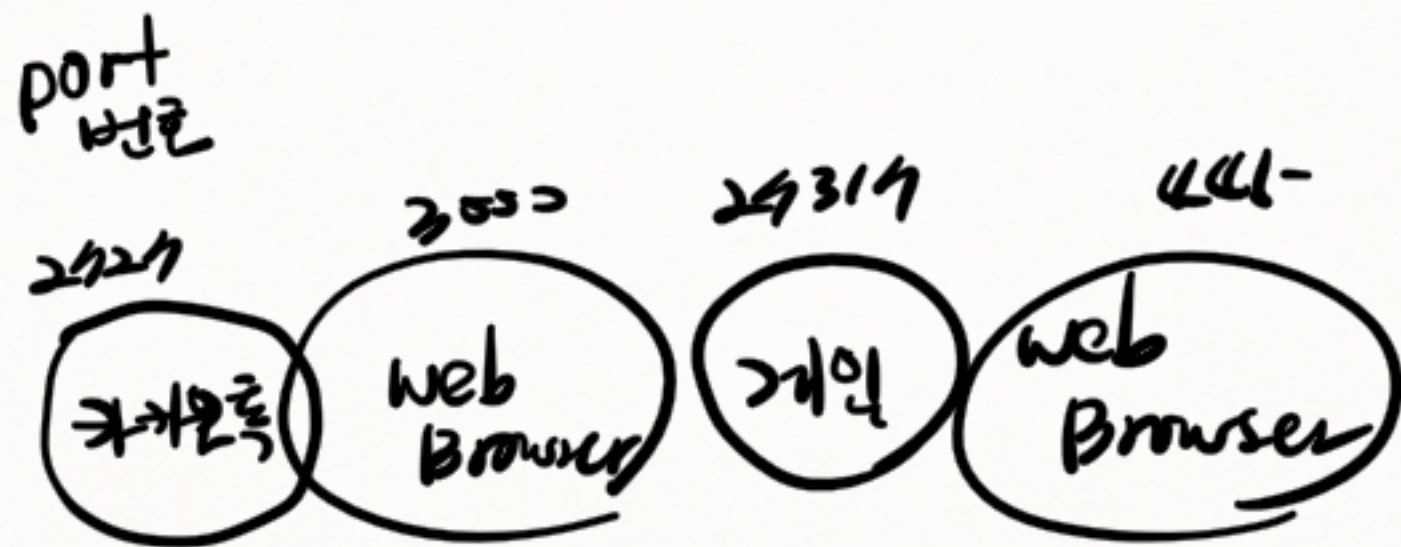
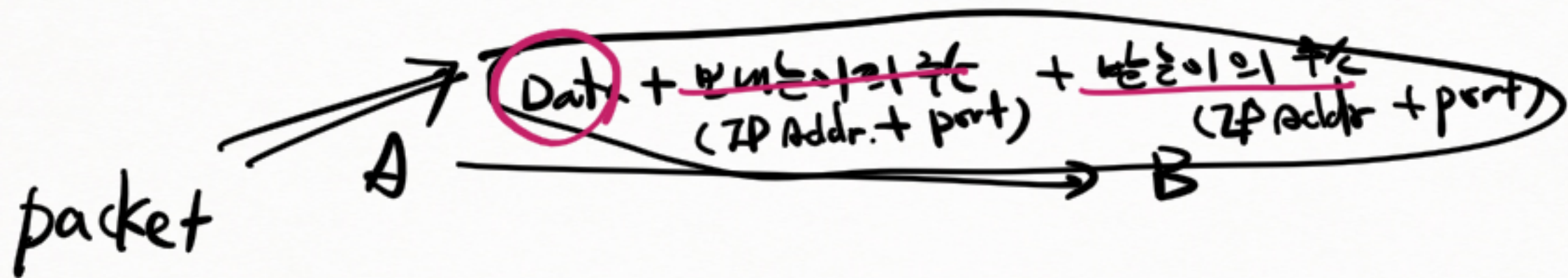
PC

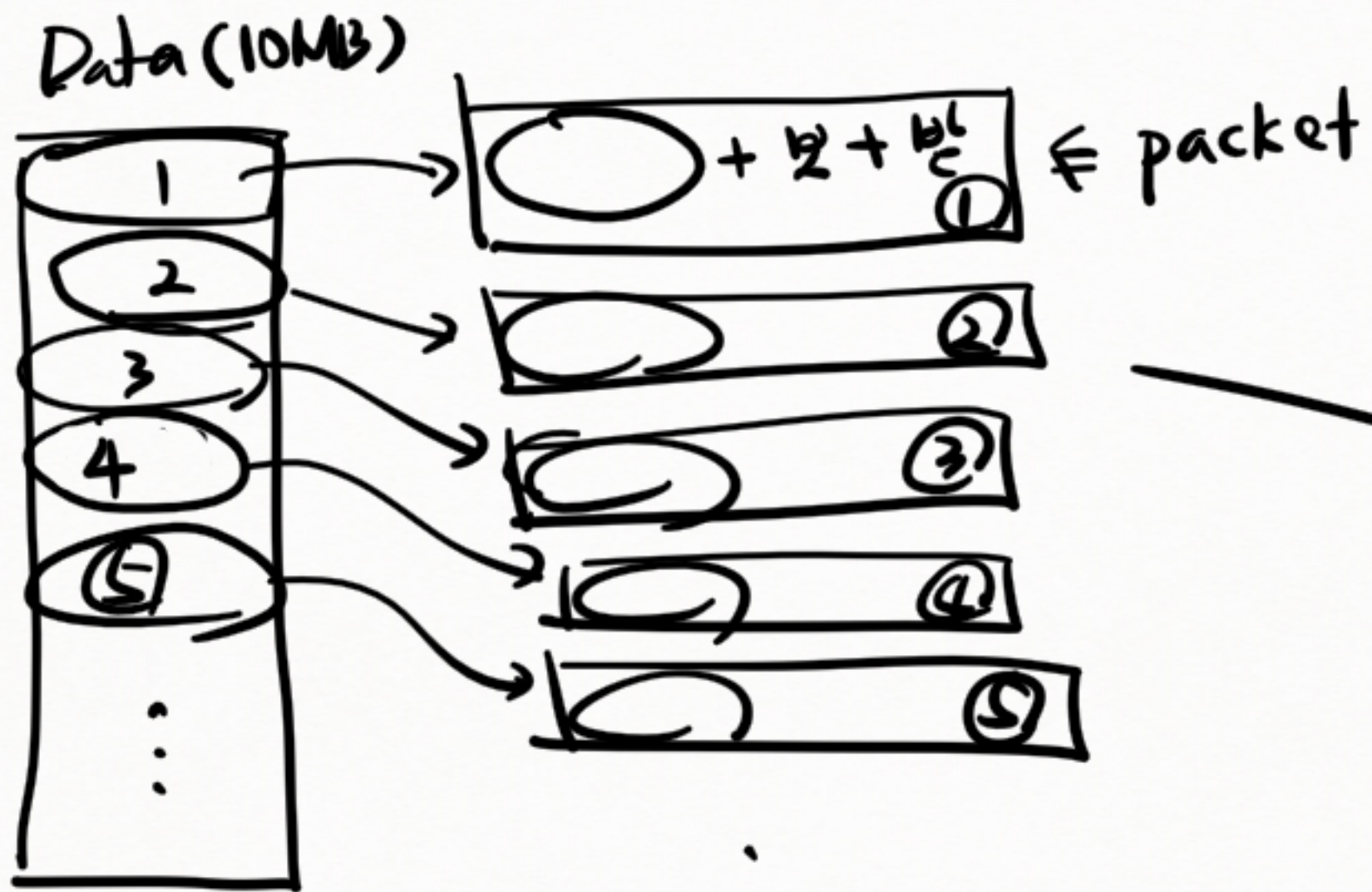
고객 = 사용자

제공자

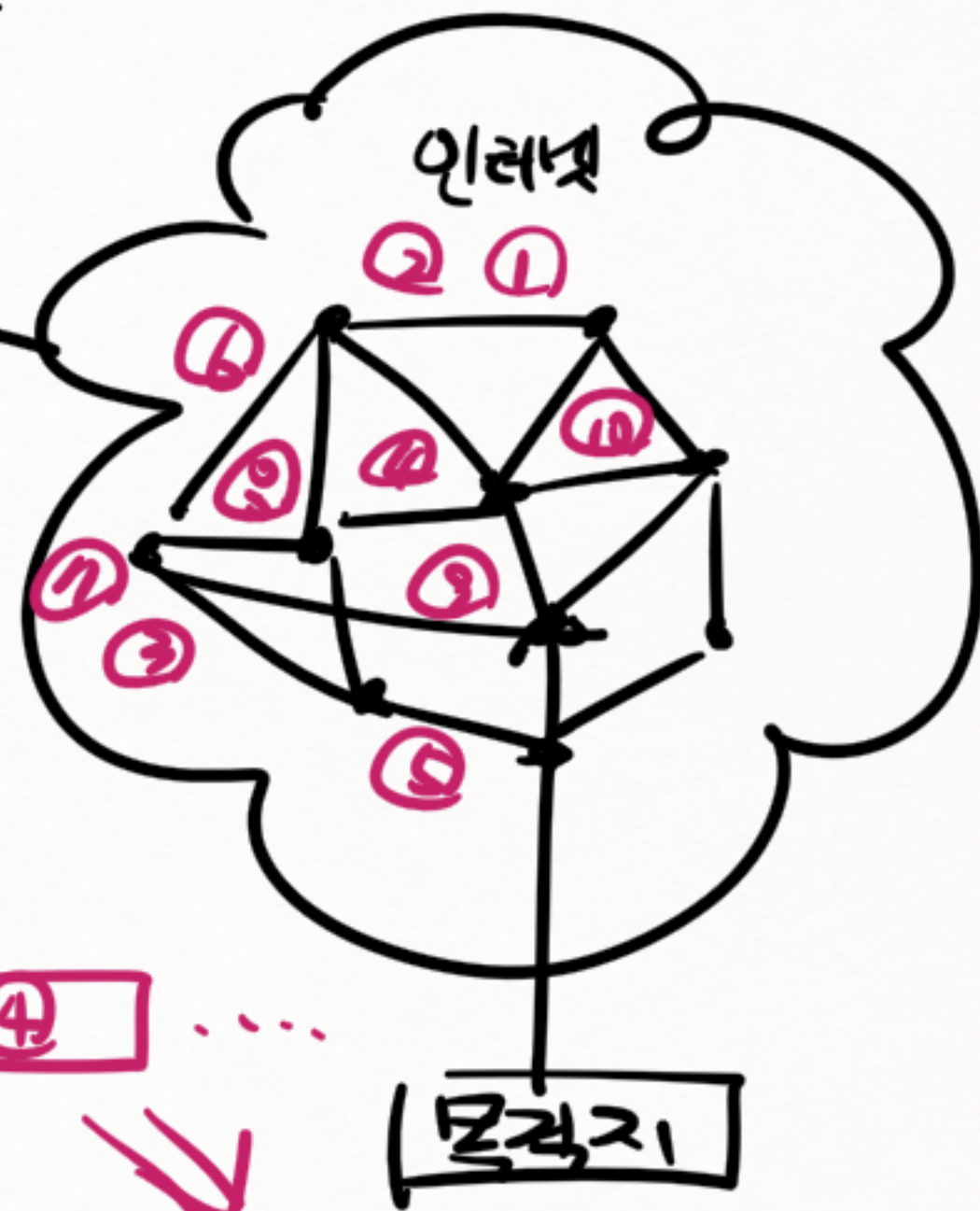




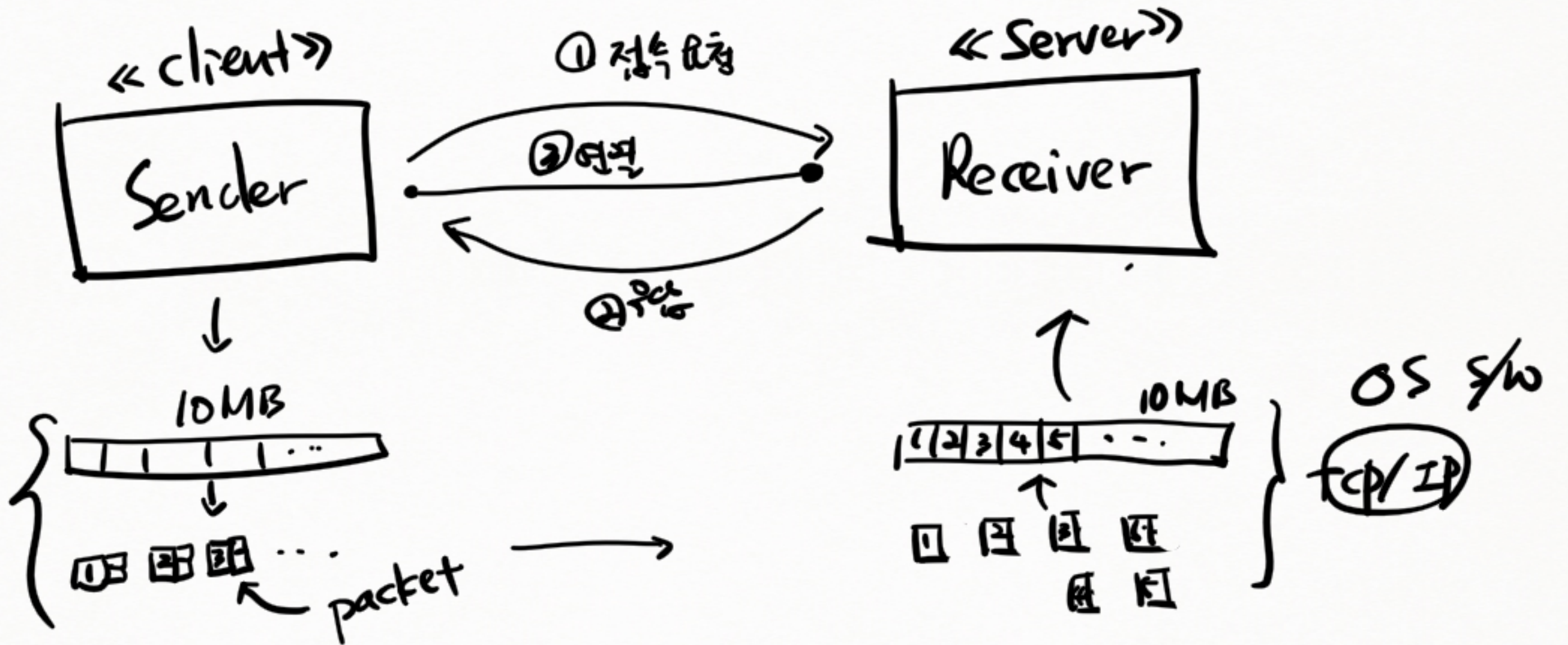




패킷은
순서대로 전송

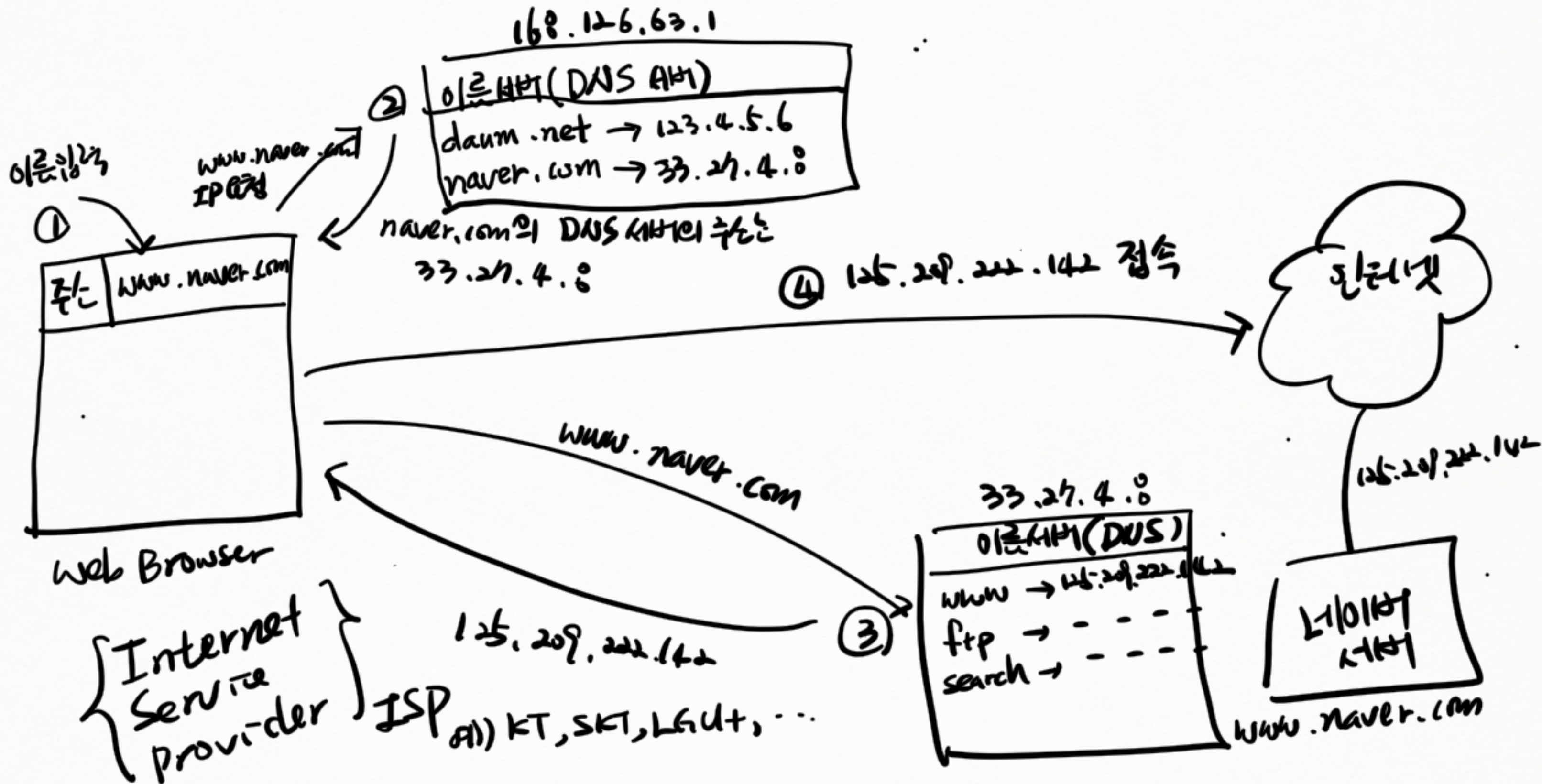


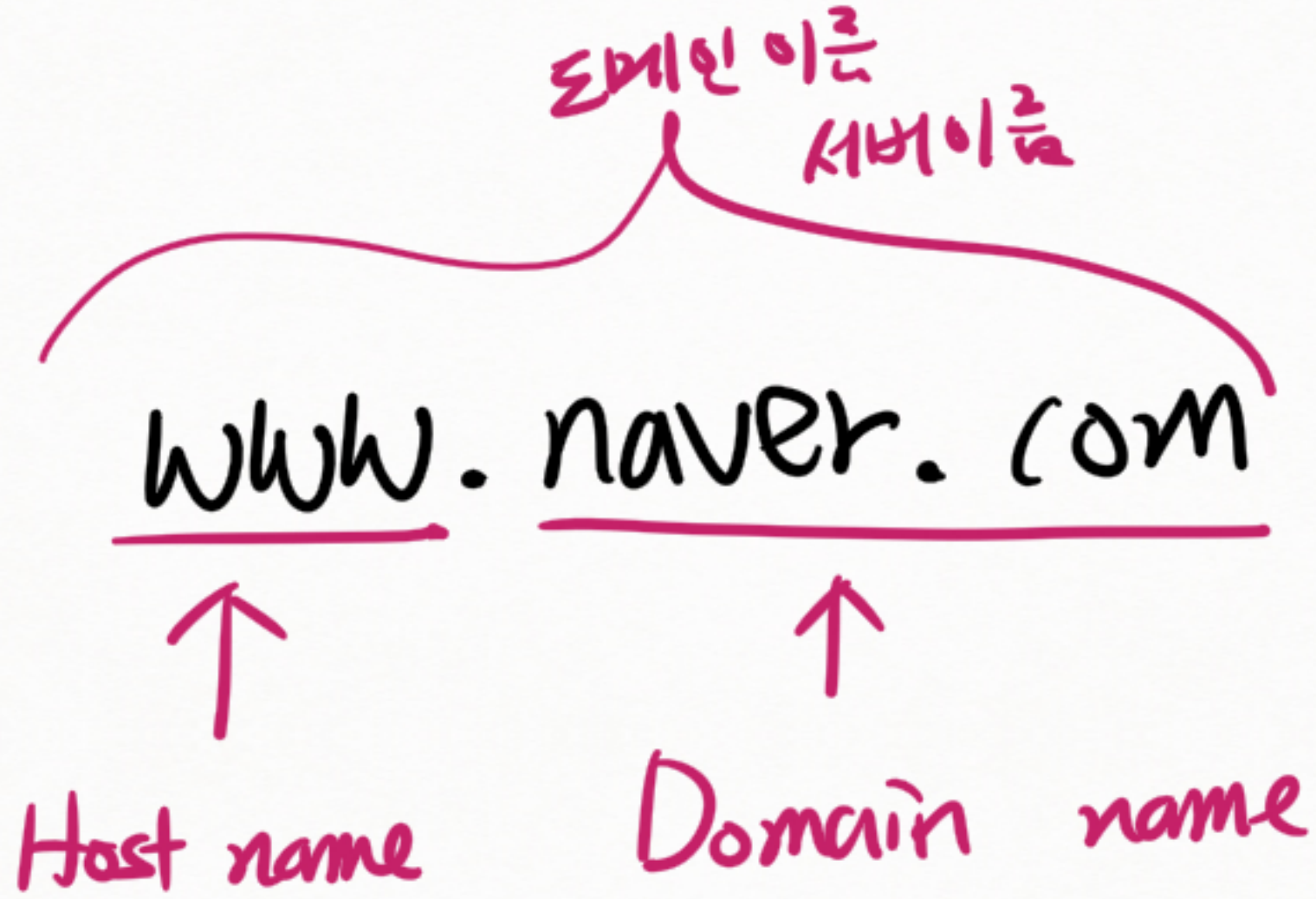
s/w
TCP/IP



new Socket (IP Address , port)
 ↑
 Domain Address
 ↓
localhost ⇒ 현재 컴퓨터

127.0.0.1 ⇒ 현재 컴퓨터

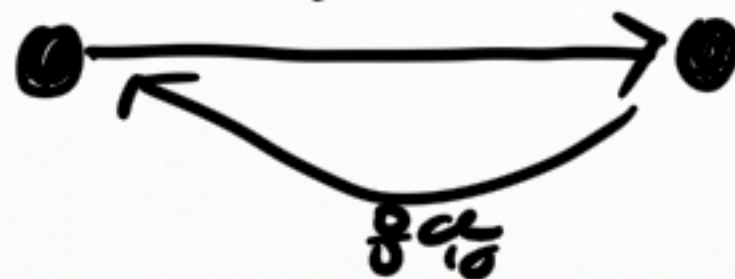




new Socket("localhost", 8888)

Client 0110

접속요청



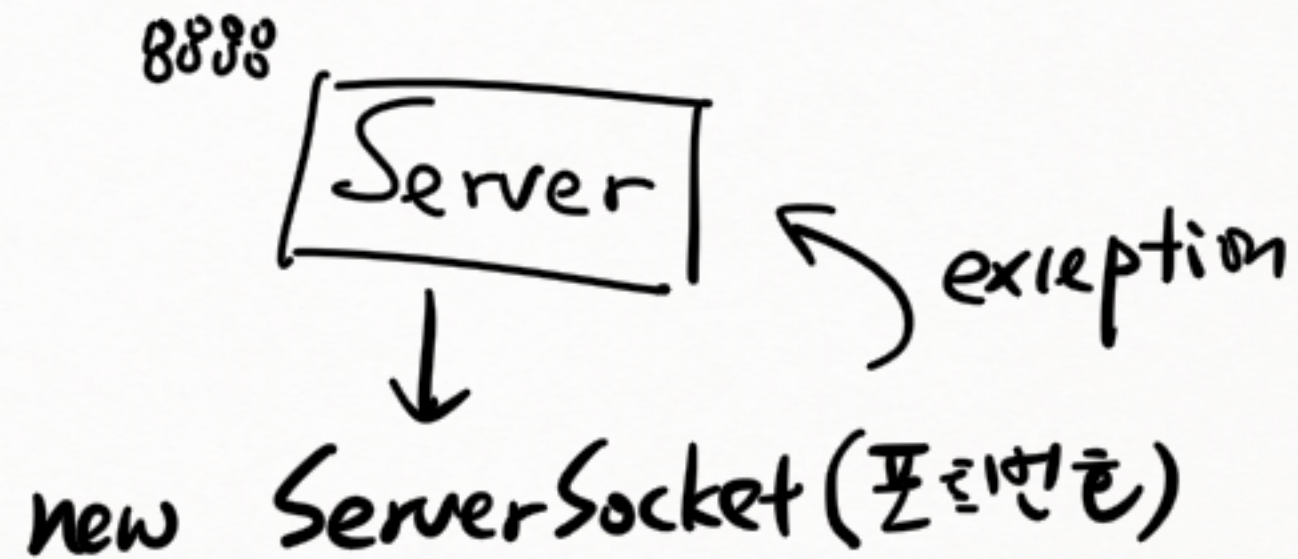
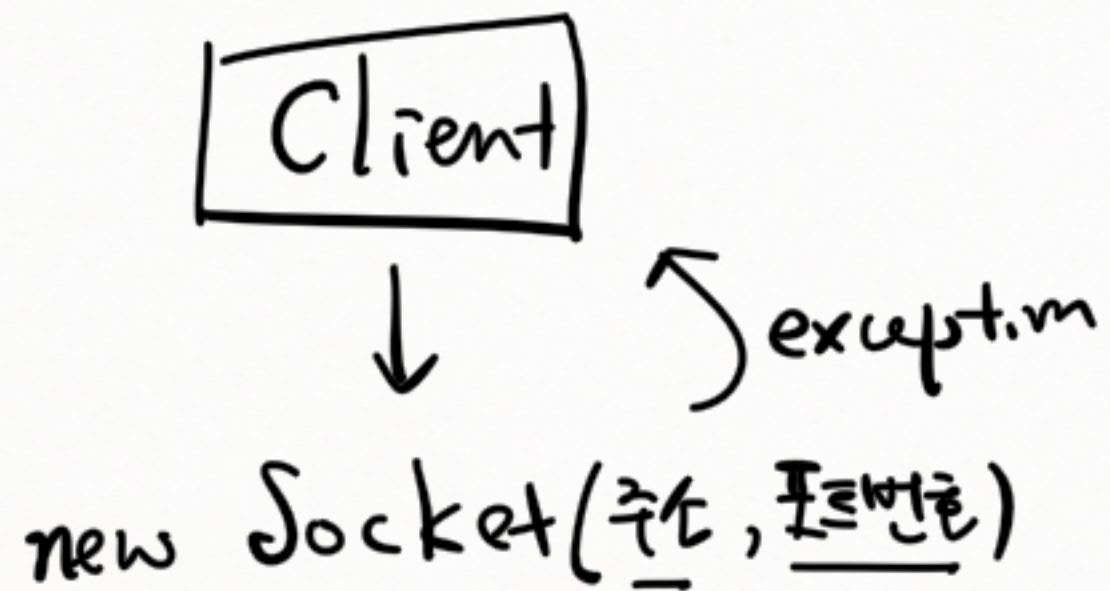
Server 0110

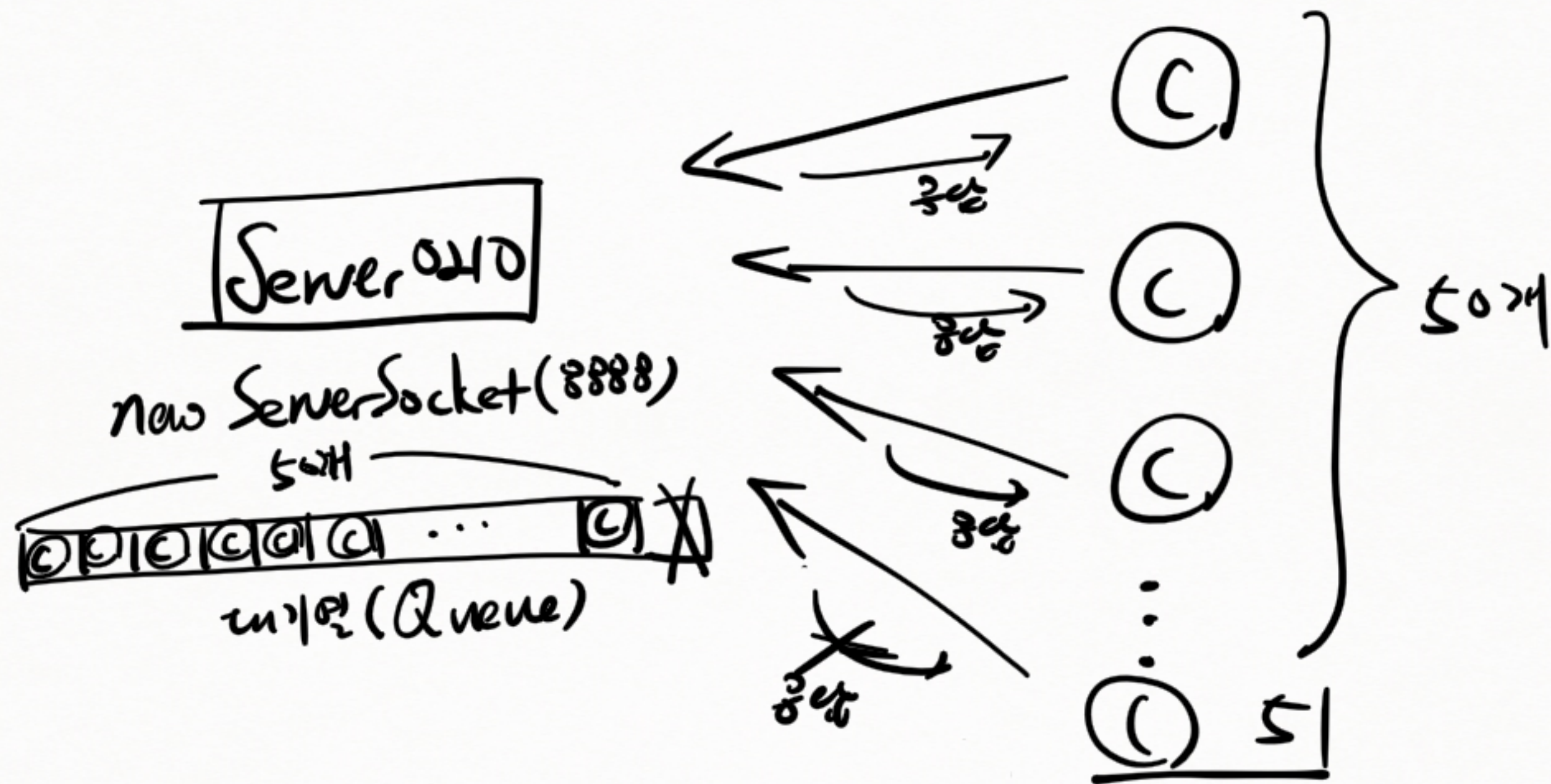
127.0.0.1
8888

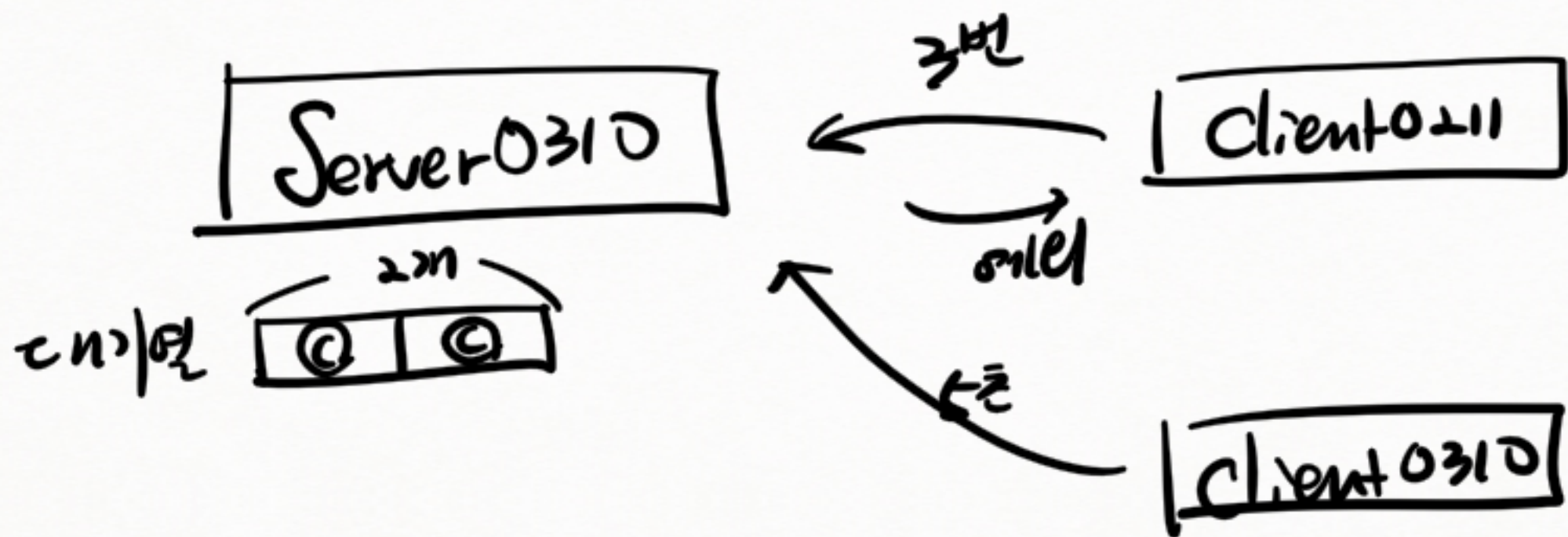
127.0.0.1
있을의 포트번호를 부여

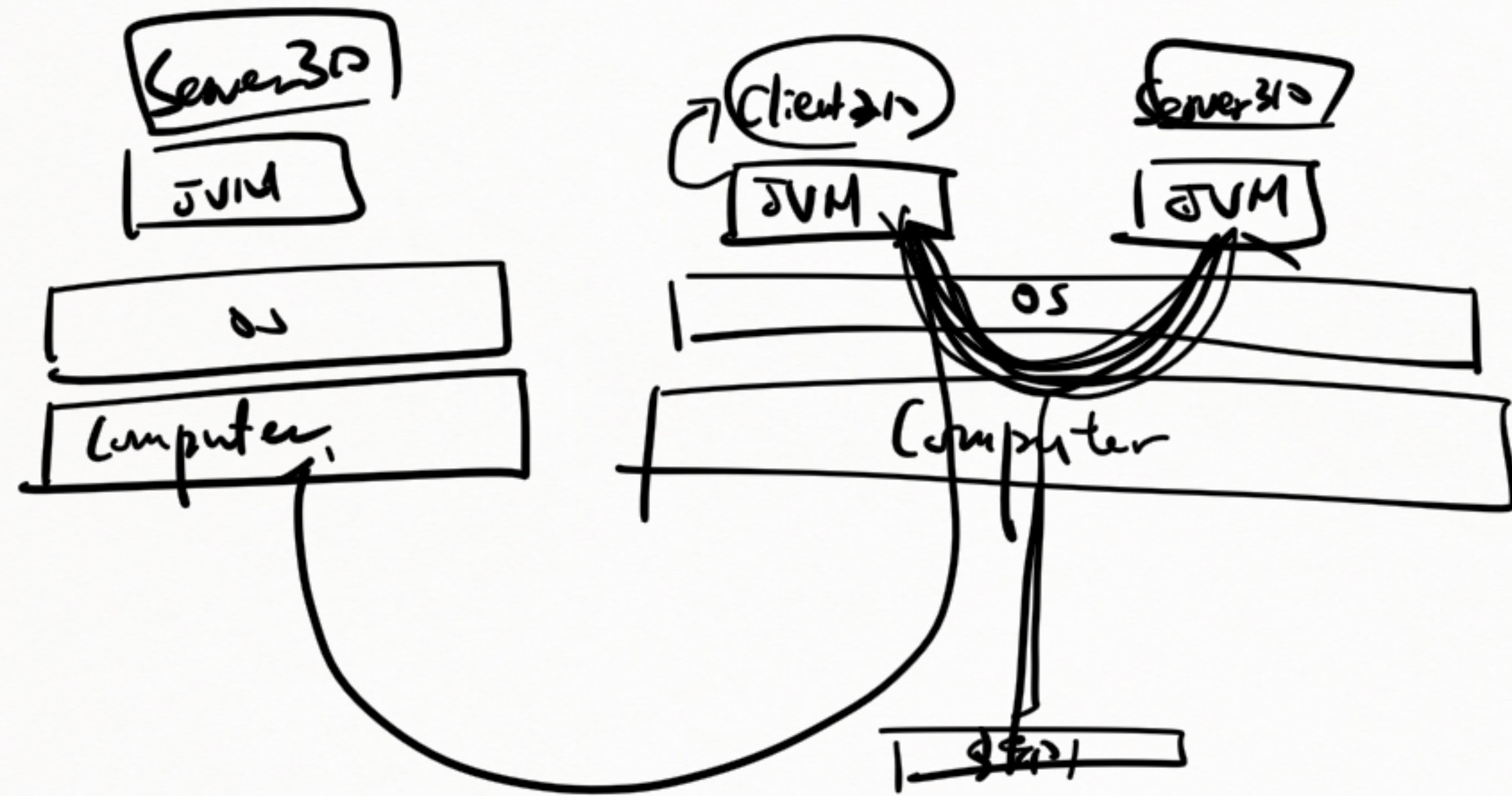
OS

Local









localhost
127.0.0.1
212121

대기열
(Queue)

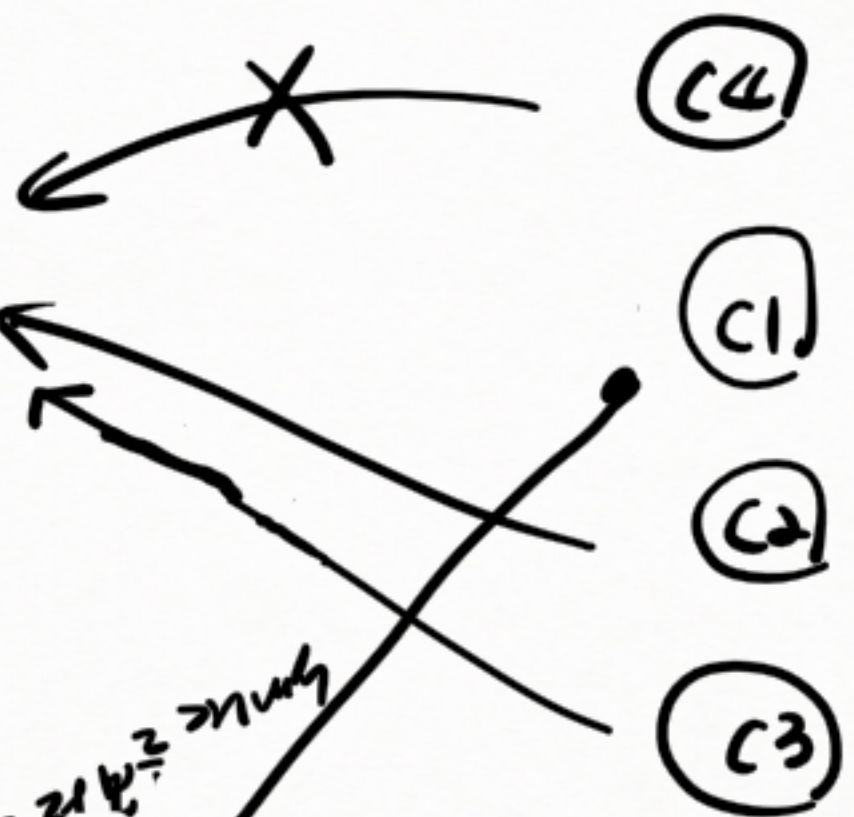
Server 0410

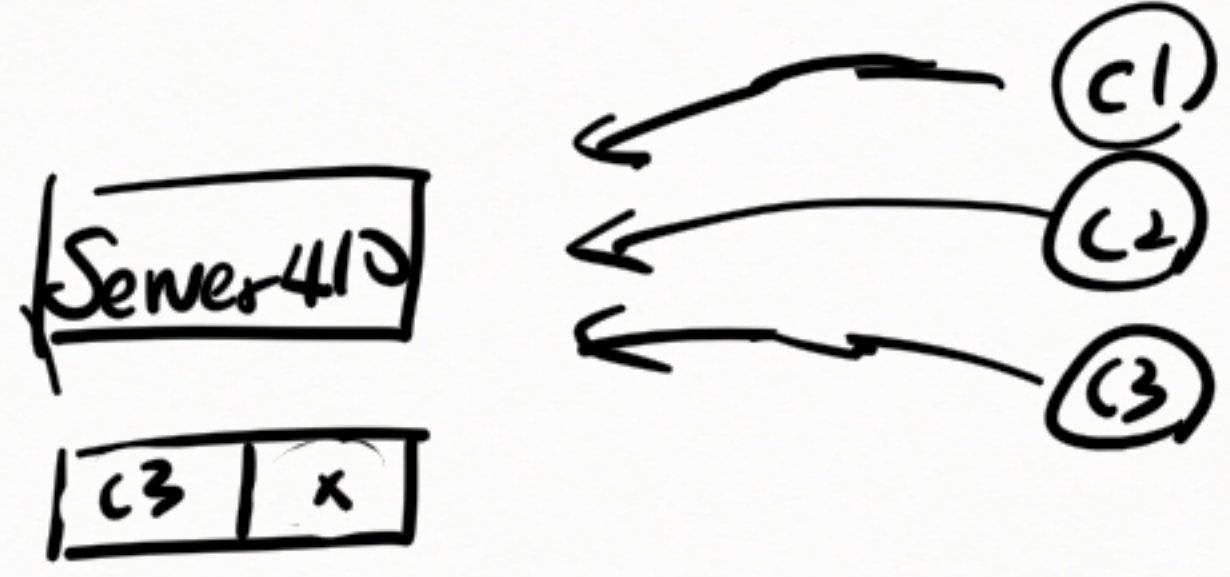
C2 C3

~~.accept()~~

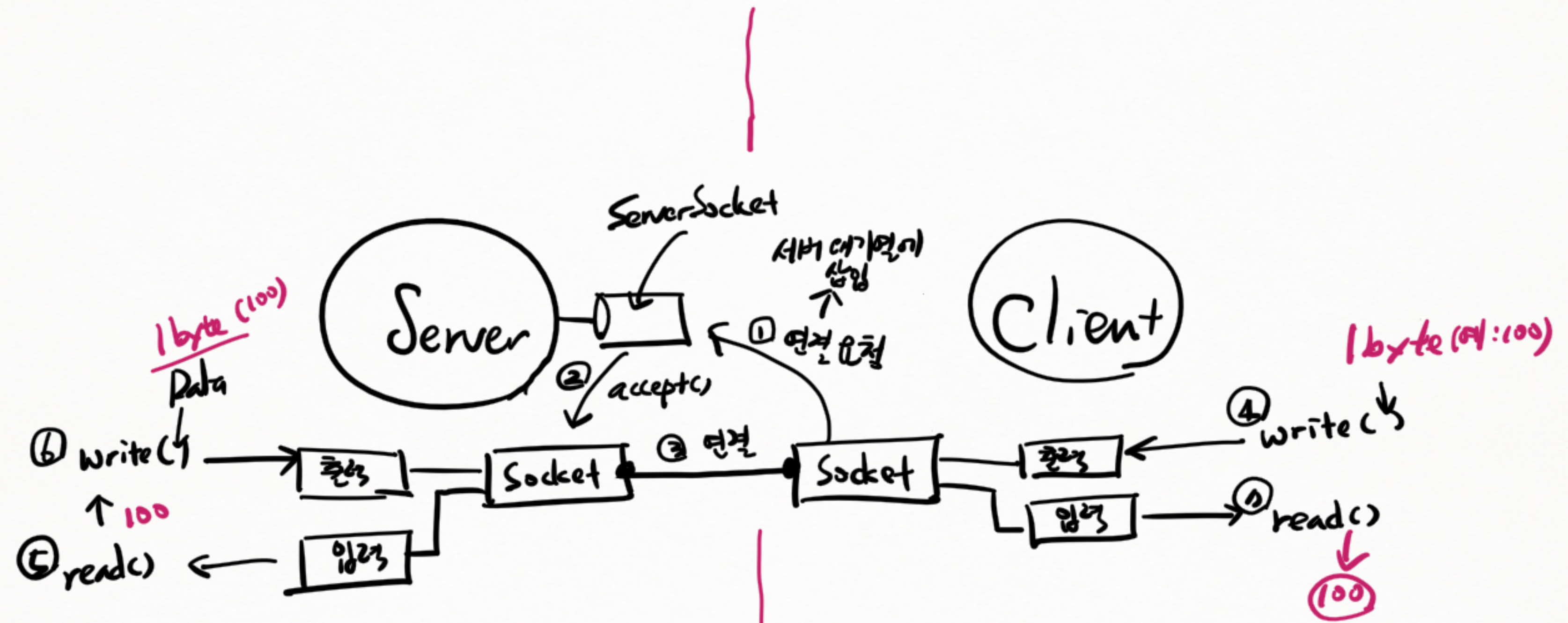
대기열에서 클라이언트 리소스를
비트맵으로 연결

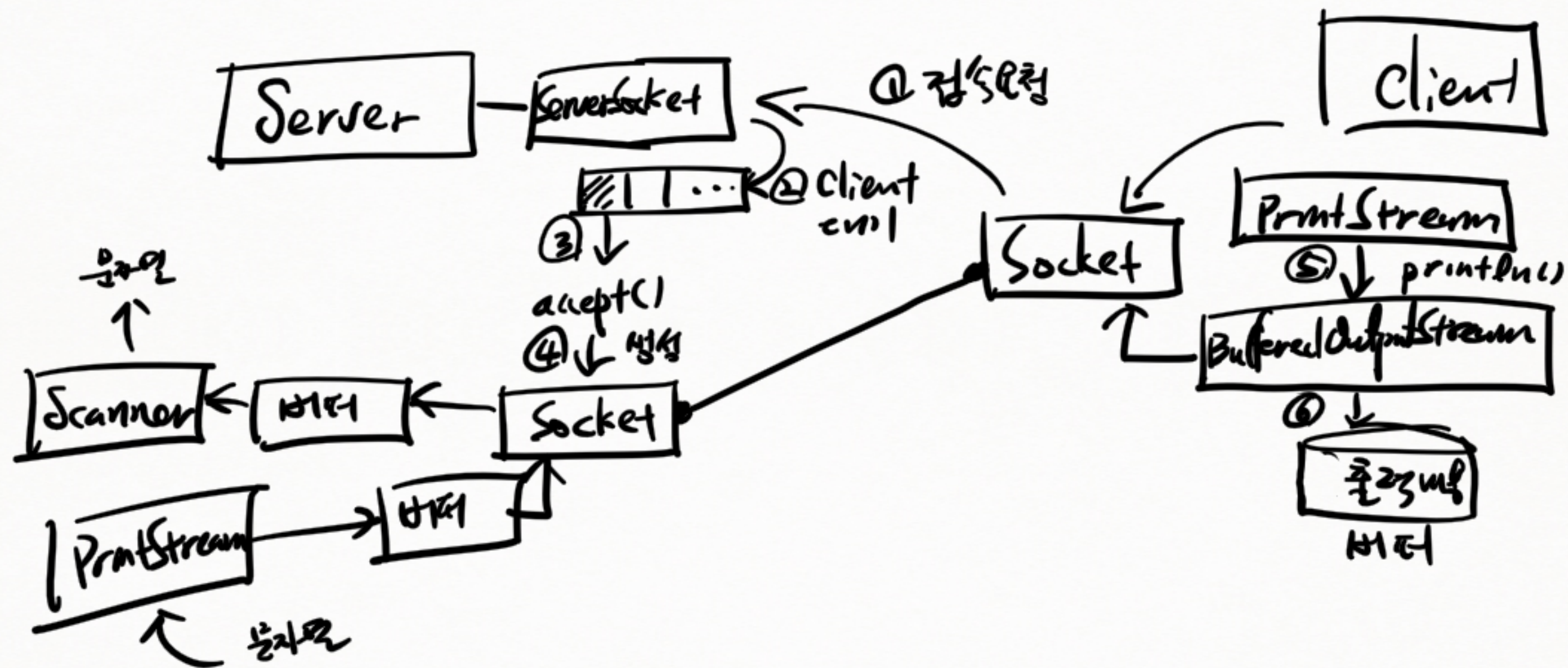
new Socket()

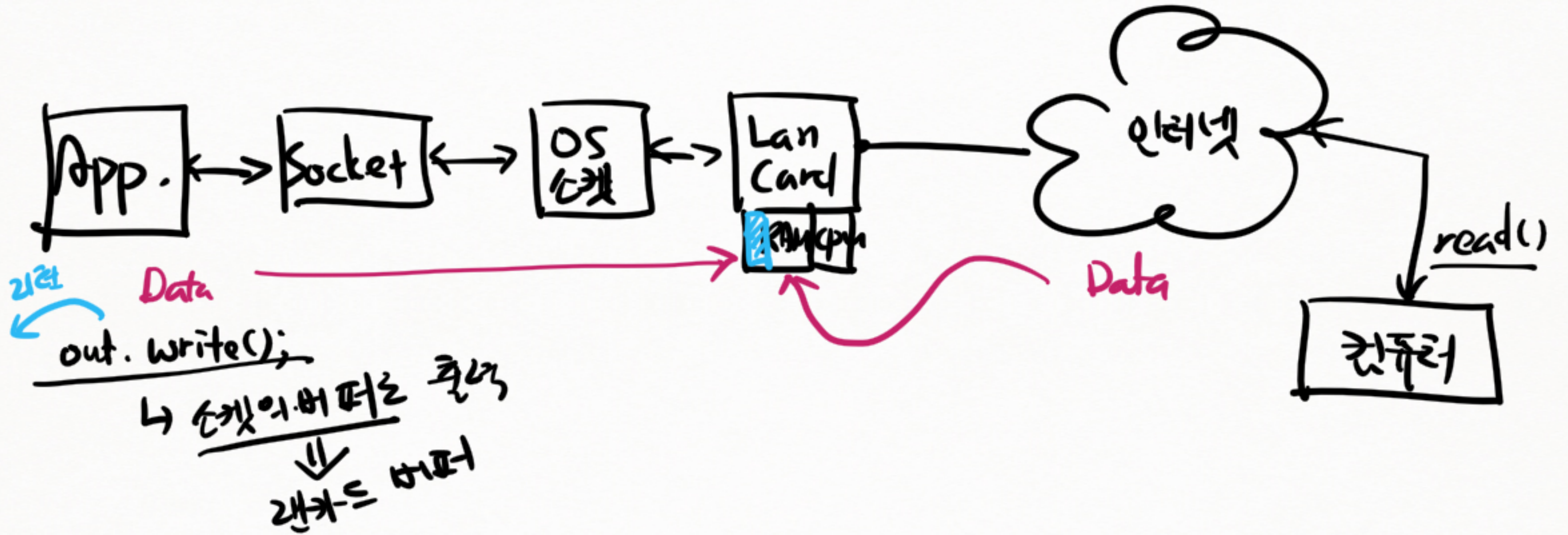


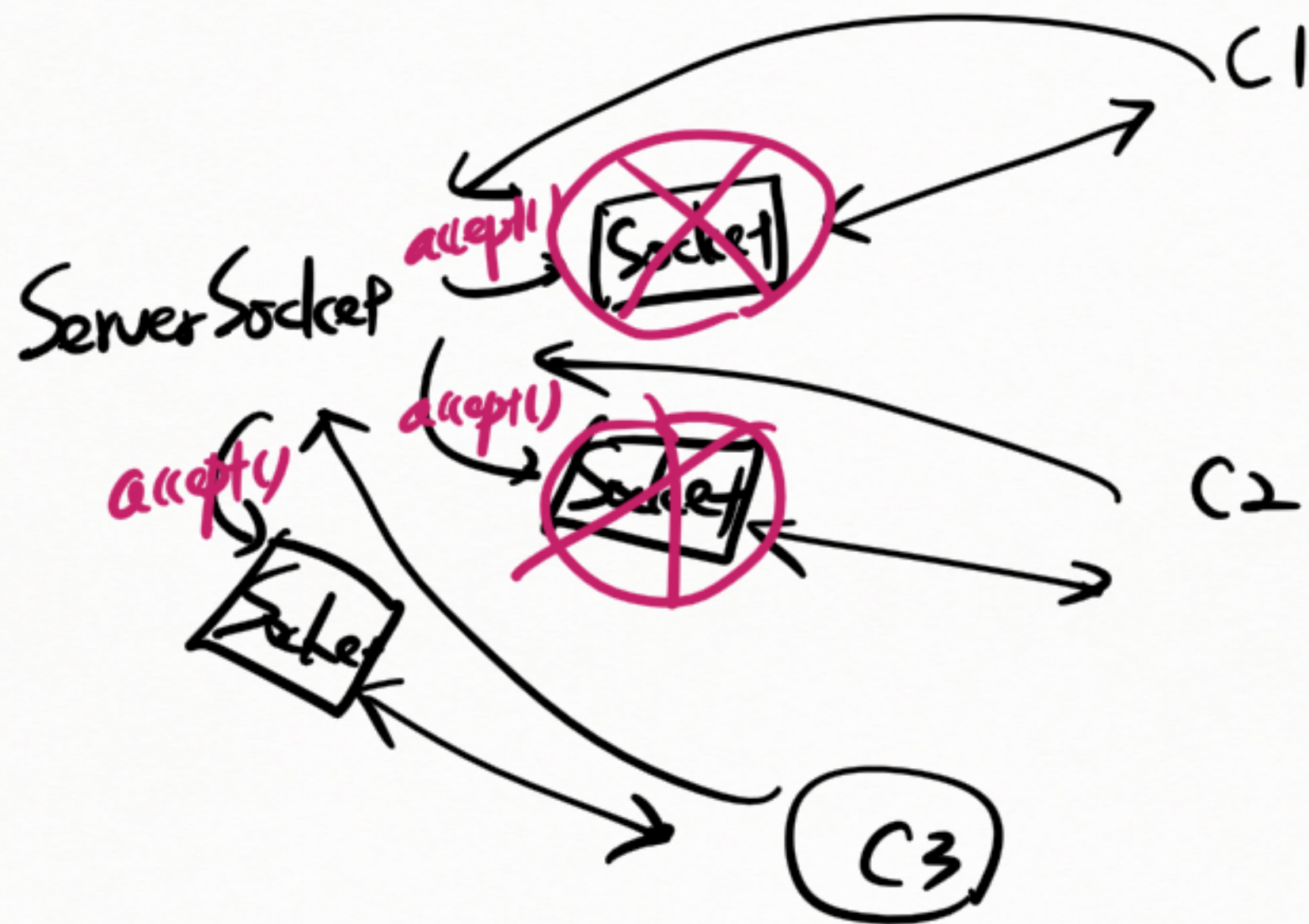


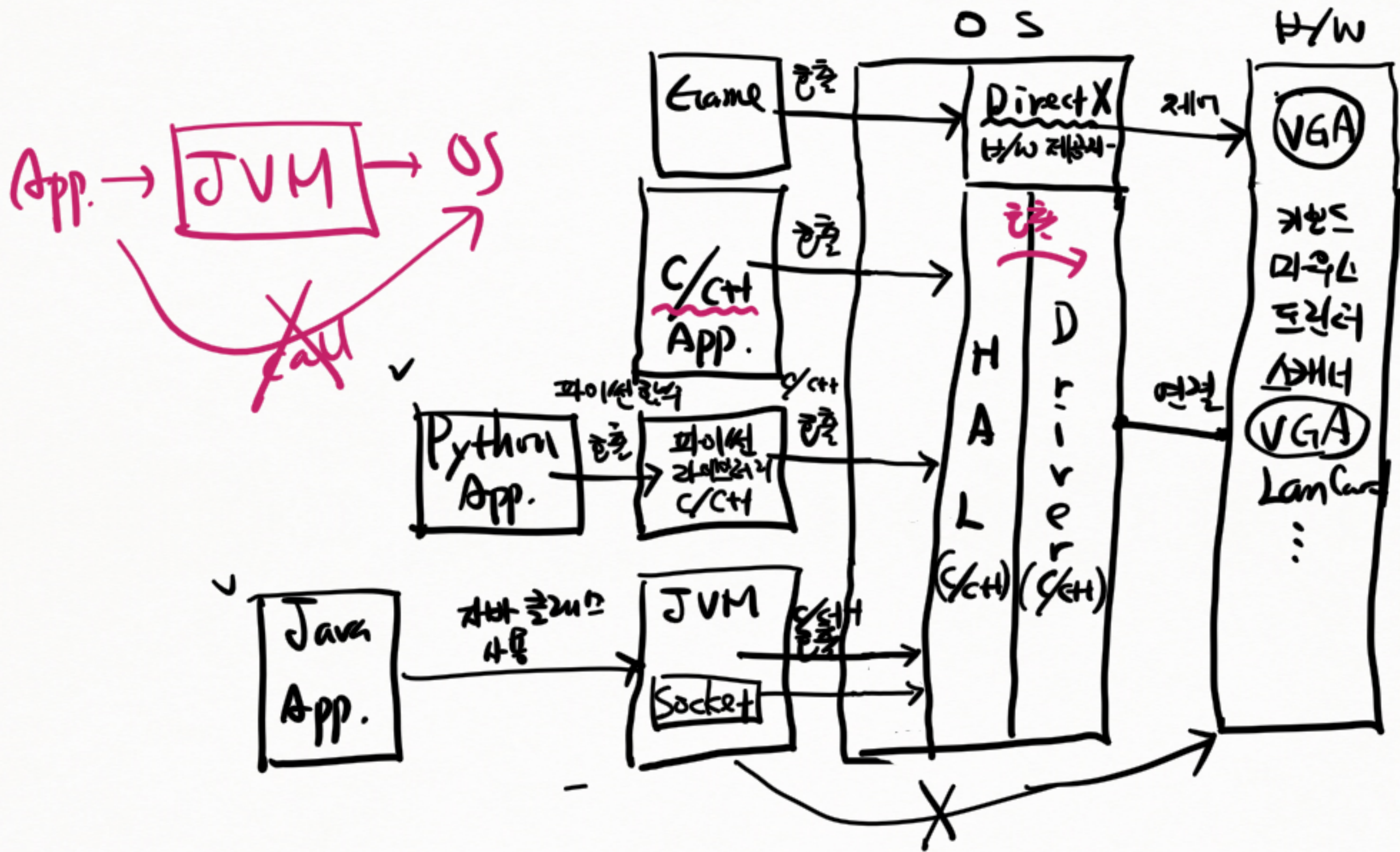
↓
accept() ↓ blocking ← 클라이언트가 데이터를 전송을 마칠 때까지 기다린다.

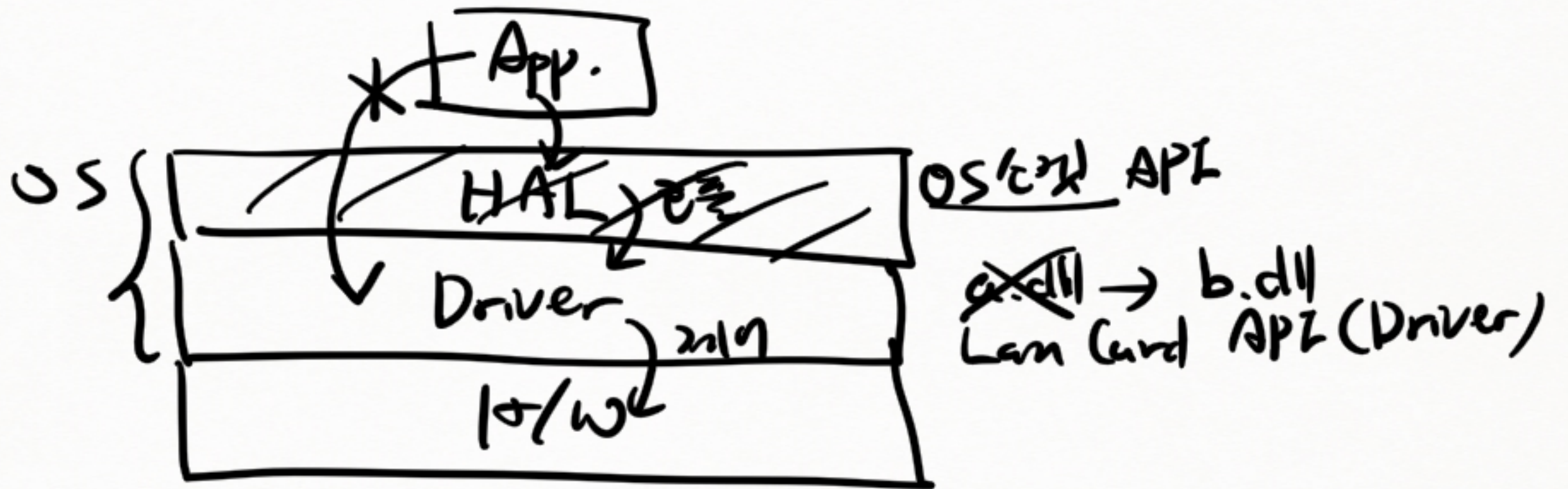


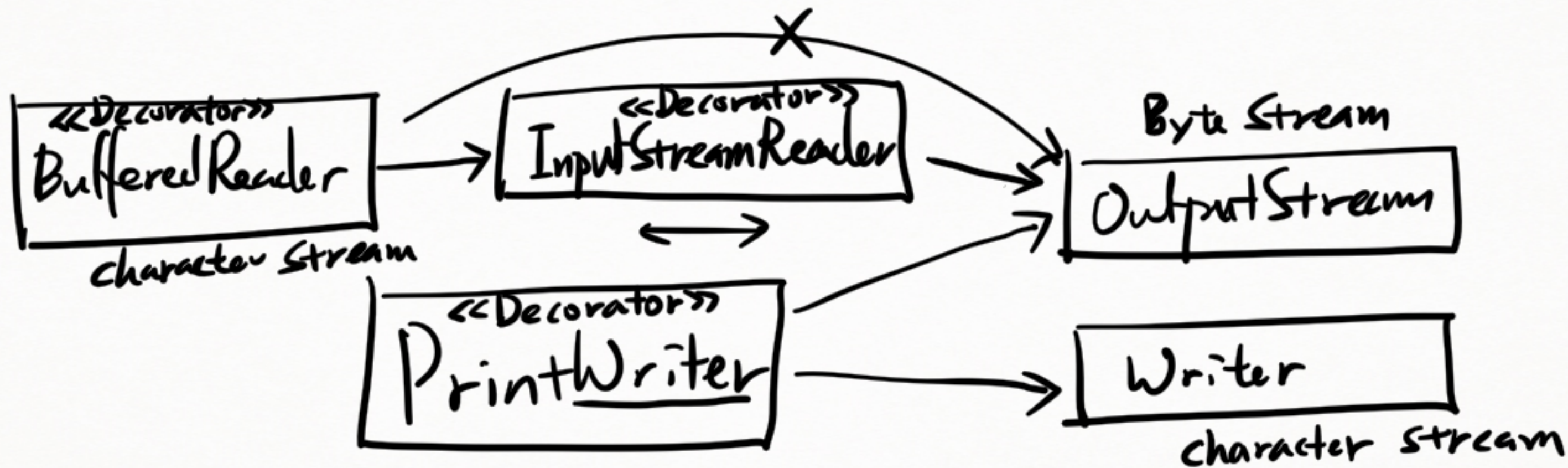










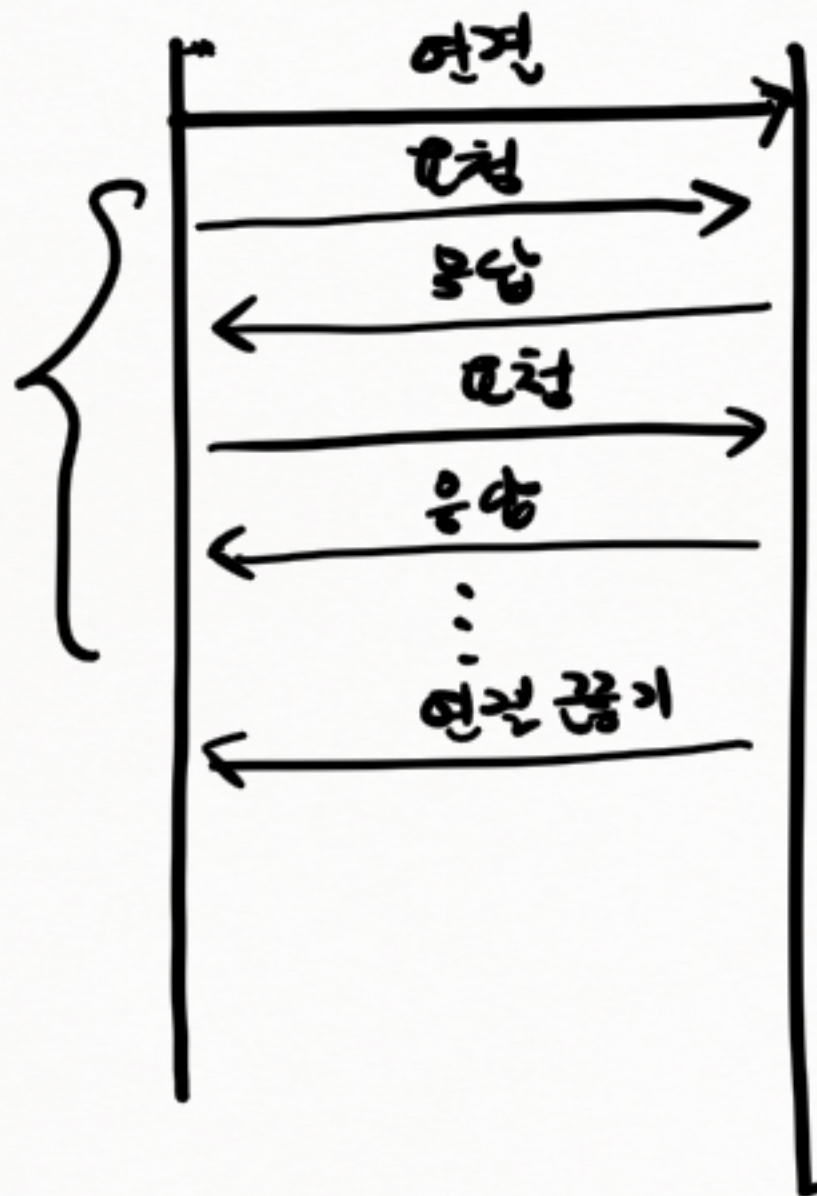


통신 방식

Stateful

Client

Server



- 고객센터 상담
- 한번만 연결
↓
전송속도 느리다.
- Client 작업 정보 유지
- 작은 packet 대응
예) FTP, Telnet 등

TCP
(Connectionful)

Socket

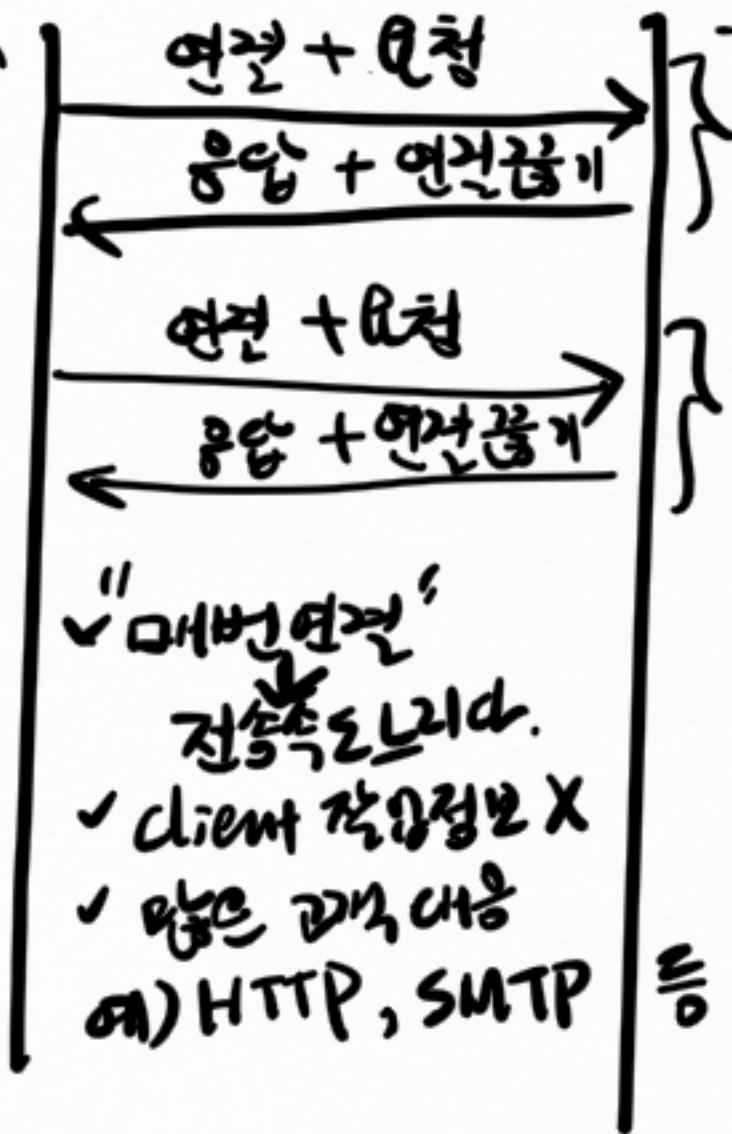
114 안내

114 안내

Stateless

Client

Server



- "대부분 연결"
↓
전송속도 빠르다.
- Client 작업 정보 X
- 많은 packet 대응
예) HTTP, SMTP 등

연결 방식

Connectionfull
Client Server

연결 후 전송 →

- ✓ 유튜브 } 채팅
- ✓ 아프리카.TV }
- ✓ 온라인게임
- ✓ 전화
- ✓

TCP = Socket

아프리카TV } 채팅
유튜브 }

메신저 - 채팅
쪽지

Connectionless
Client Server

전송 →

- ✓ 팩시
- ✓ 이메일
- ✓ 메신저
- ✓ 문자- (메시지)
- TV 라디오 } 방송

Datagram Socket = UDP

IP

✓ Connectimful
TCP
↳ Socket



```
graph LR; A["Connectimful  
TCP  
↳ Socket"] --> B["Stateful"]; A --> C["Stateless"]
```

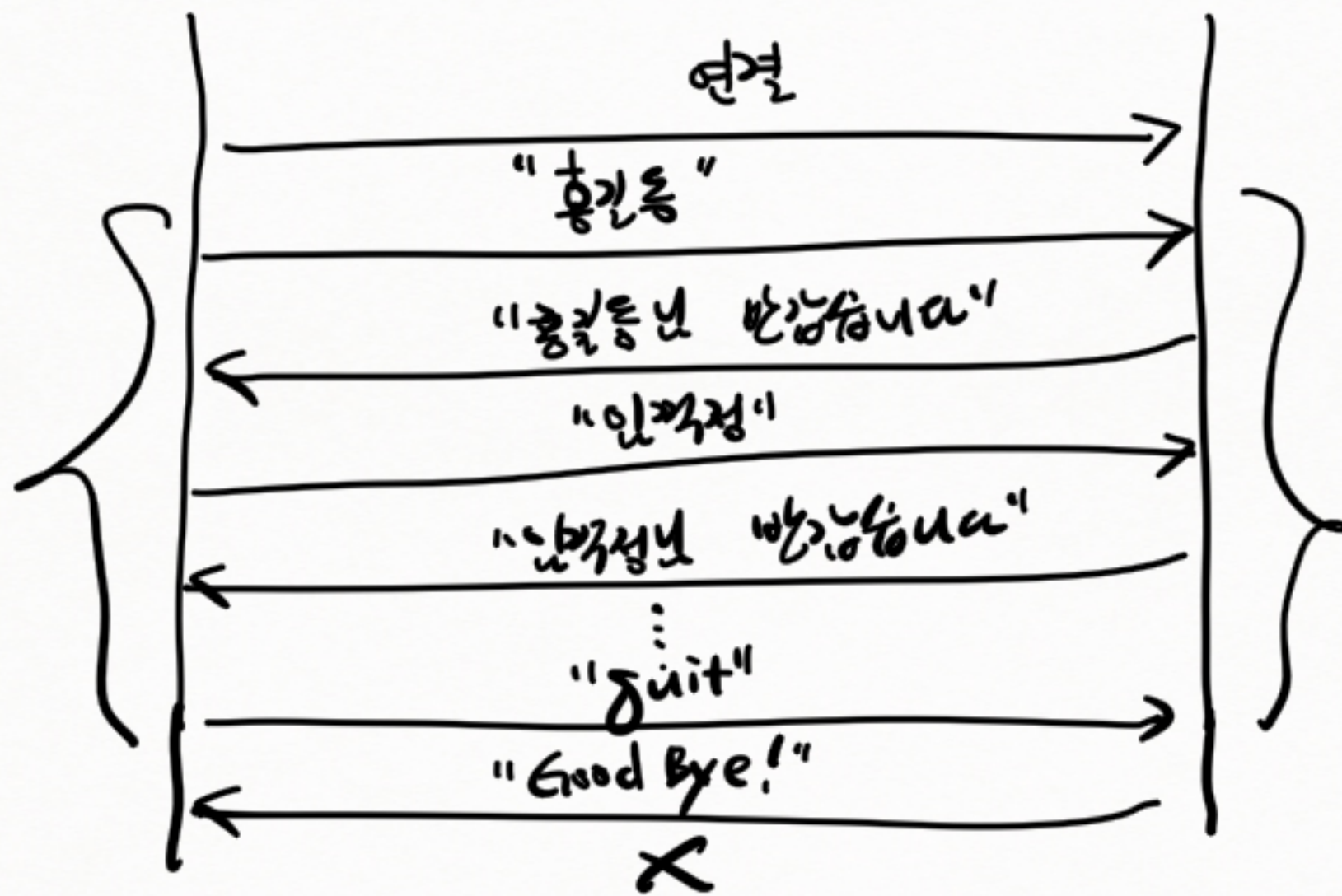
Stateful

Stateless

✓ Connectimless
UDP
↳ Datagram Socket

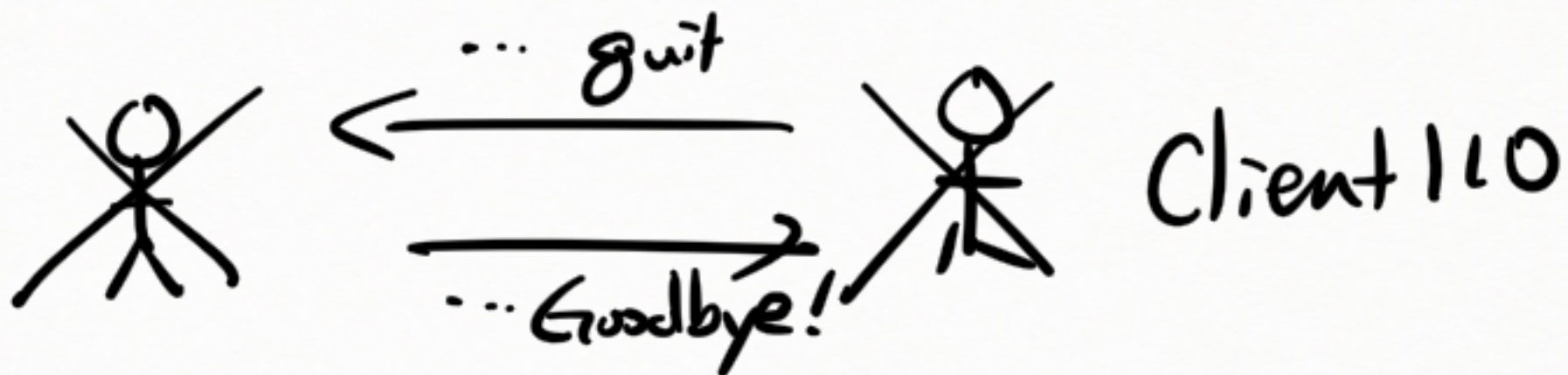
Client ☺ ☺ ☺

Server ☺ ☺ ☺

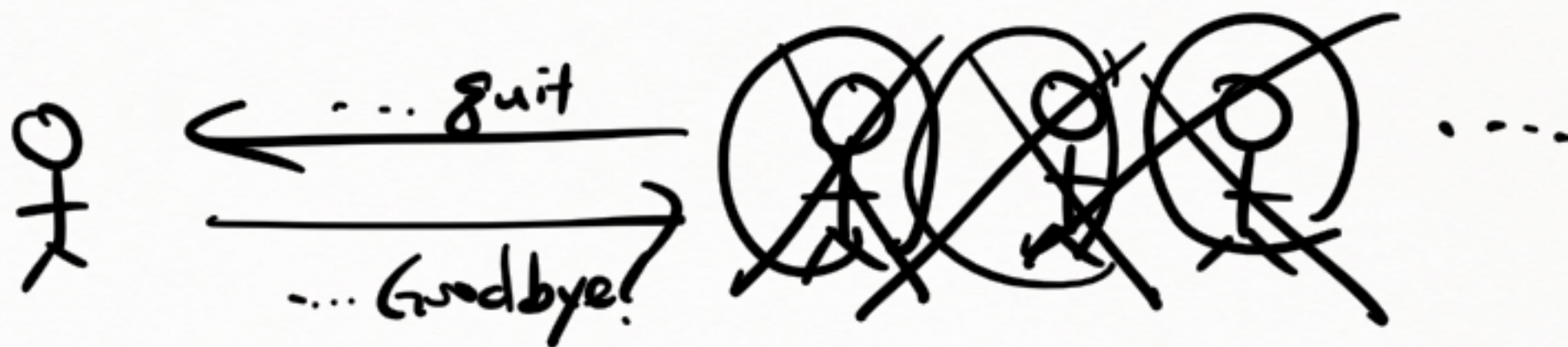


이런 연결 후
여러번 Data 송수신
↓
"Stateful"
통신

Server 110

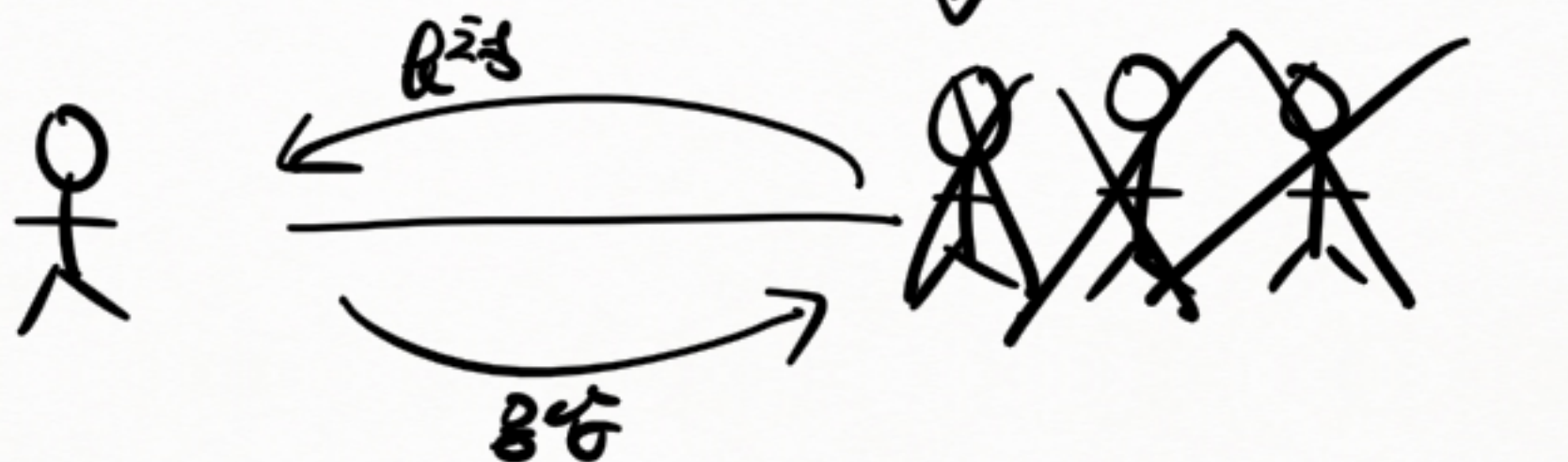


Server 120

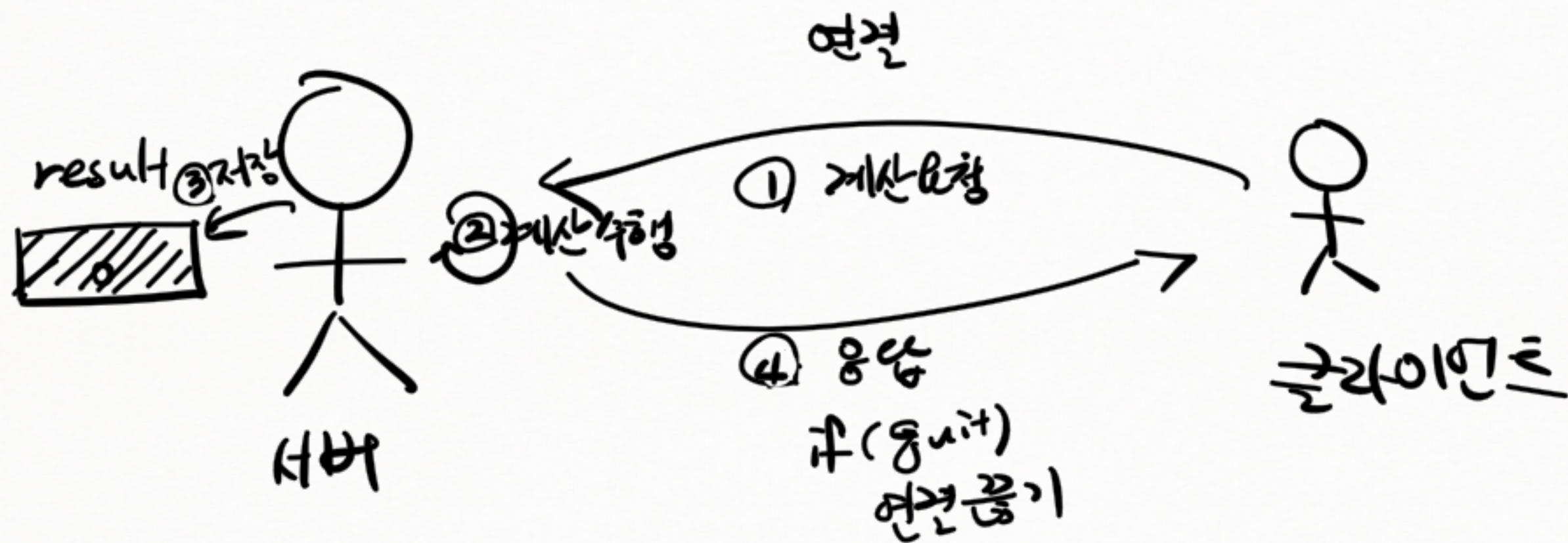


Server 210

"Stateless"



Stateful 계산기 서버



Stateless 계산기 서버

고객명단

clientID	결과
217	82
1P	201
mm8	15
2722	100

② 객체의 결과를 리턴한다



③ null x

서버

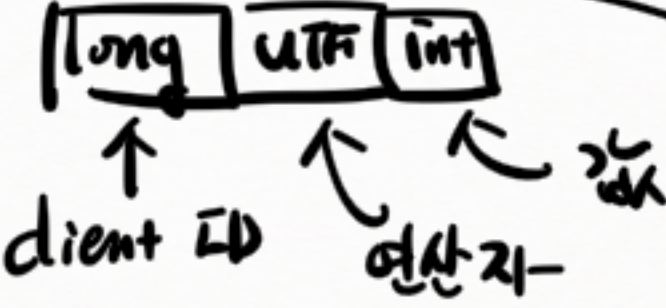
새 클라이언트 ID 생성
(예: 2722)

⑤ 계산 결과 저장

result

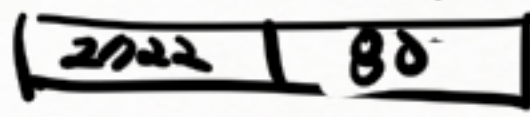
80

① 연결 + 읽기



readLong() readUTF() readInt()

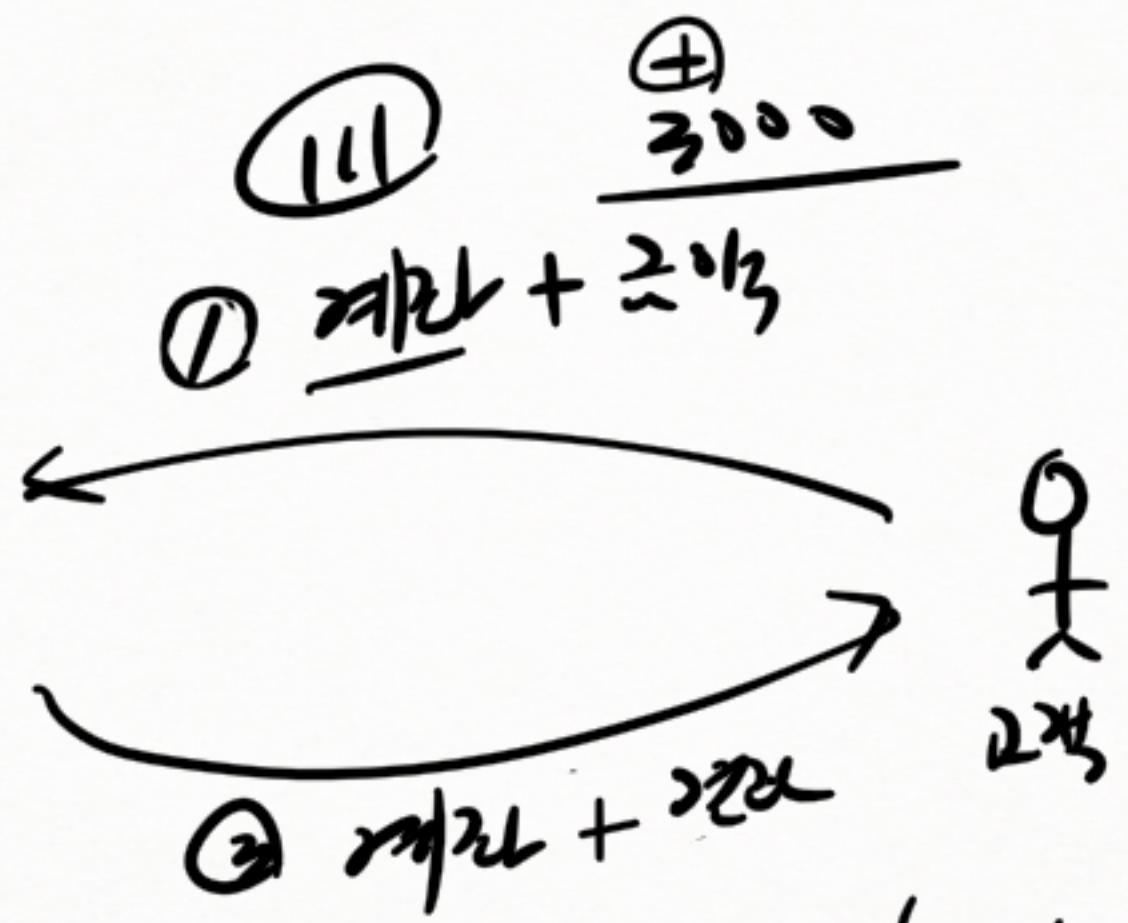
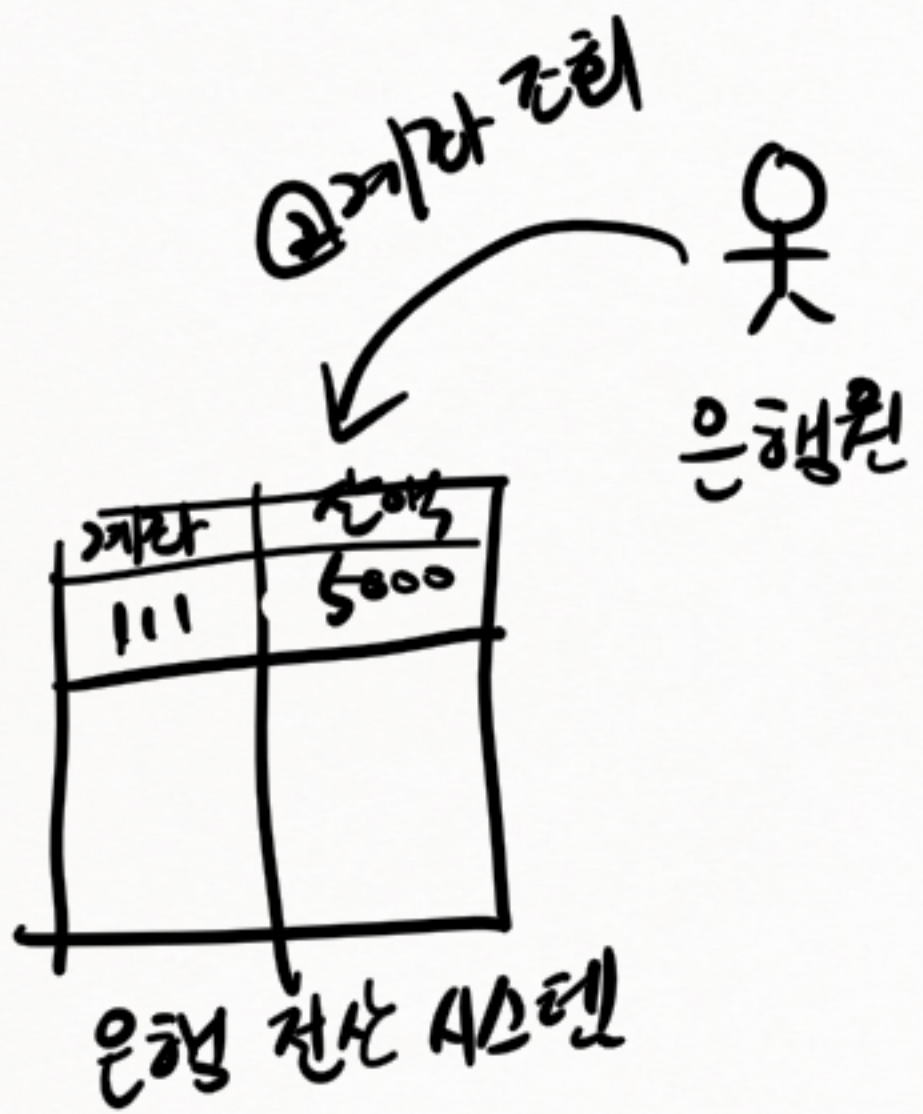
④ 클라이언트 ID + 계산 결과



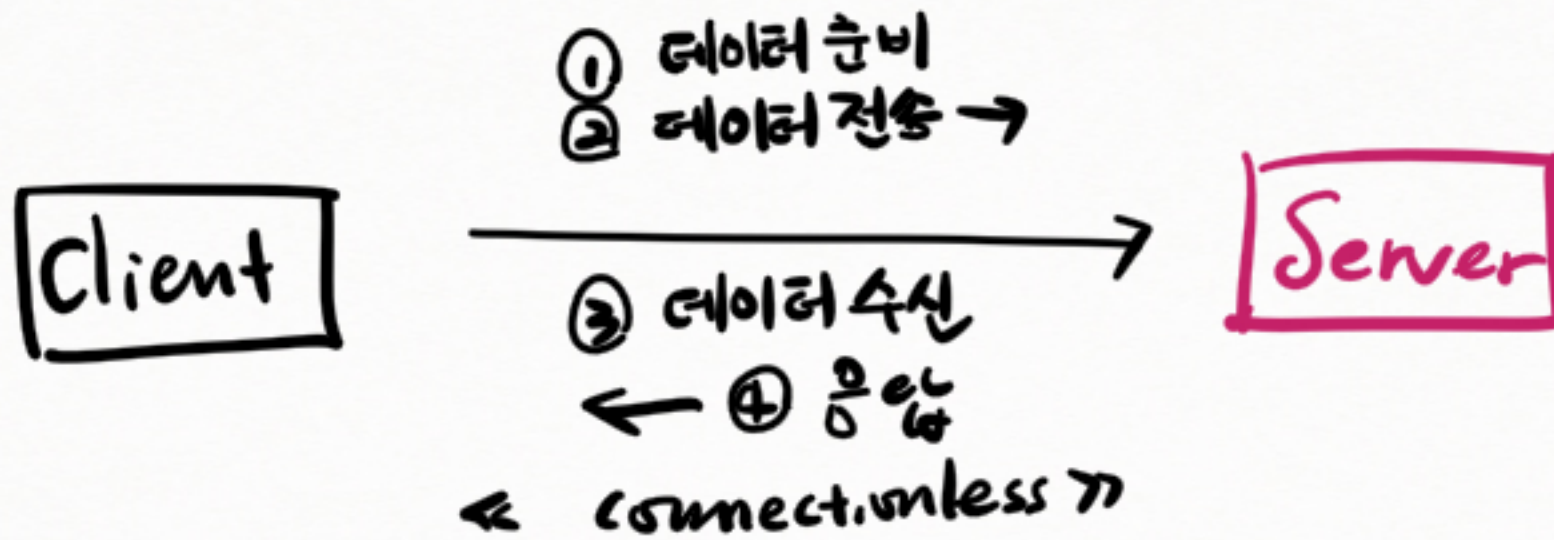
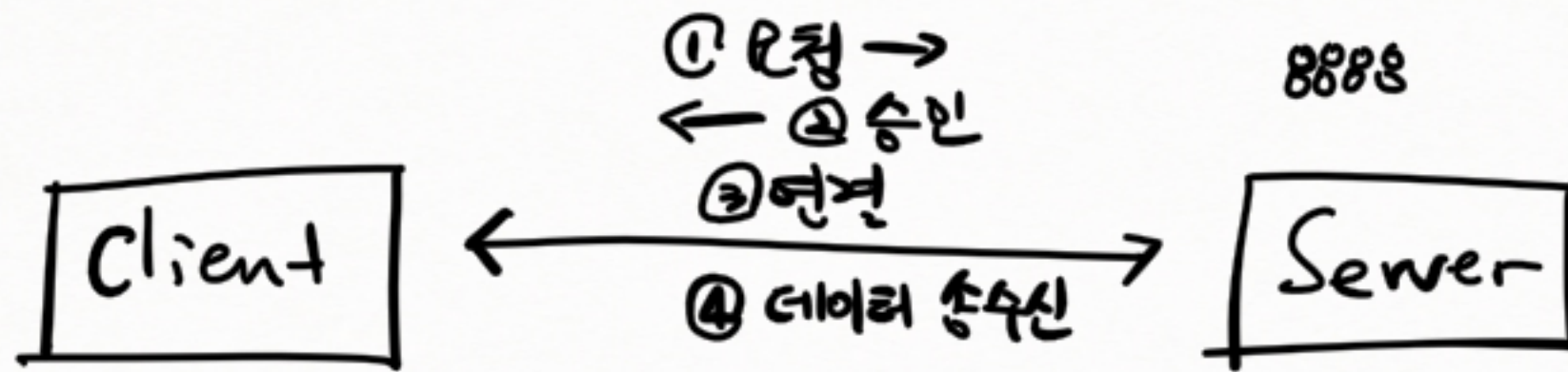
클라이언트

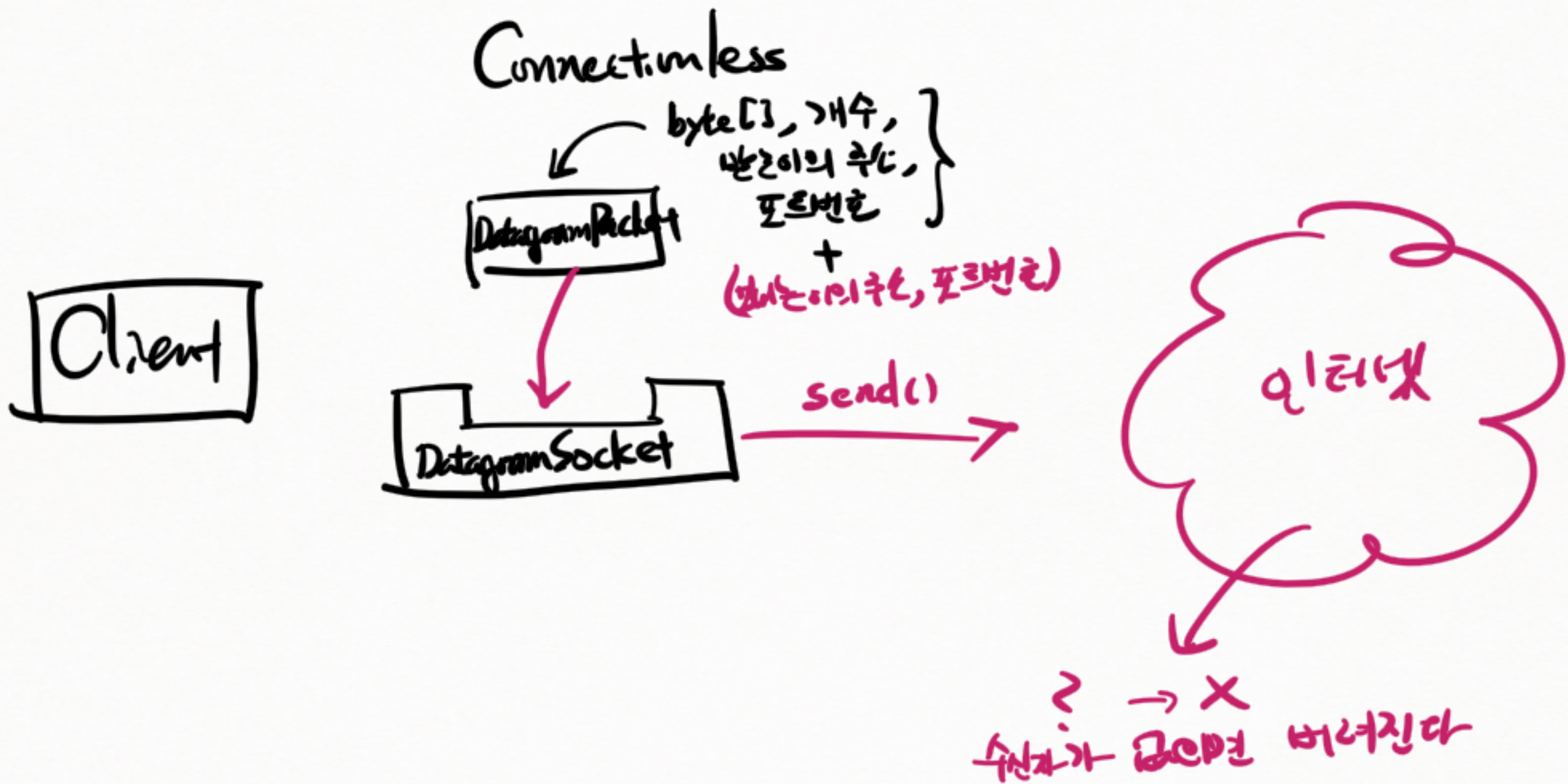


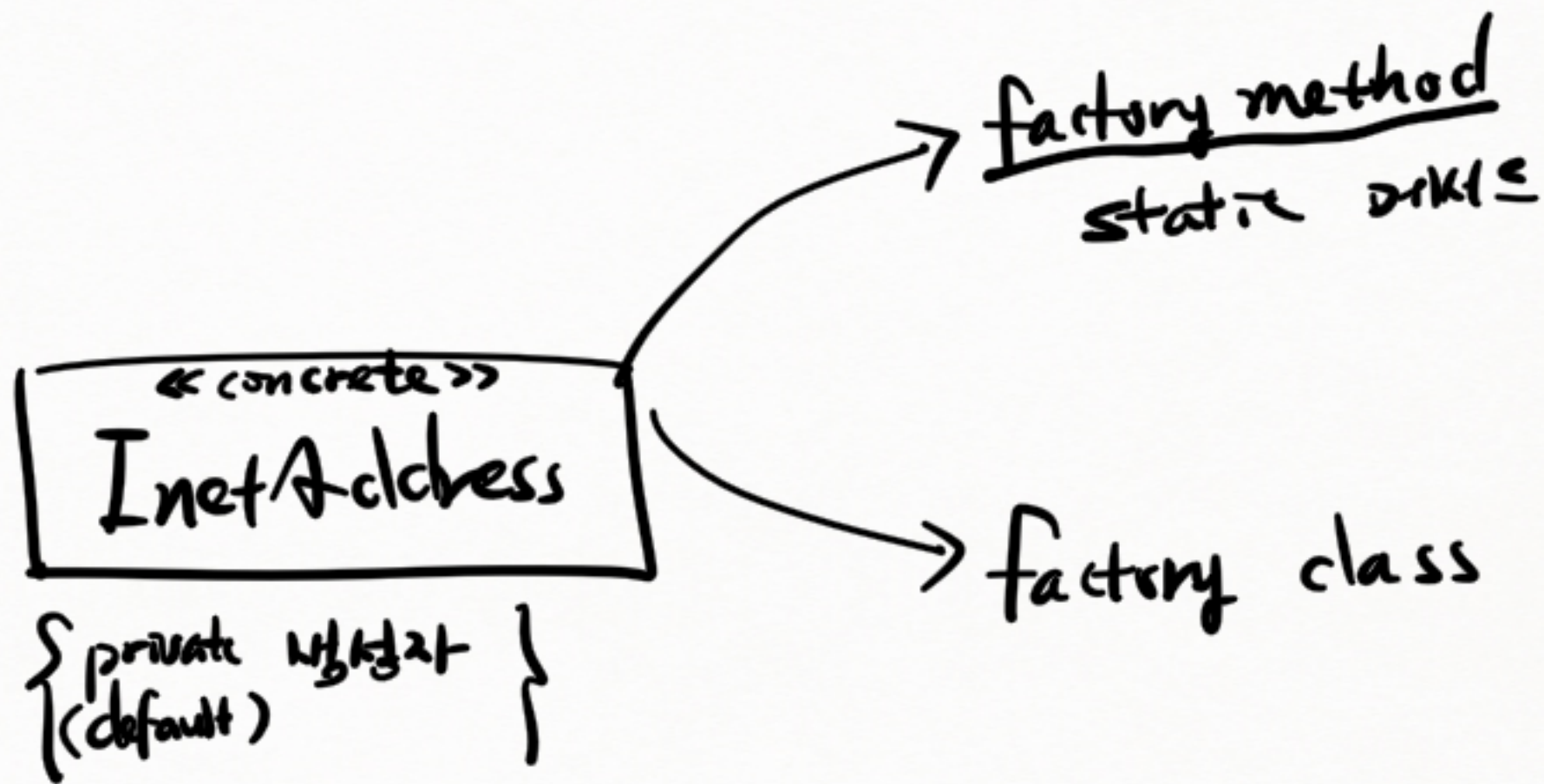
readLong()
readInt()

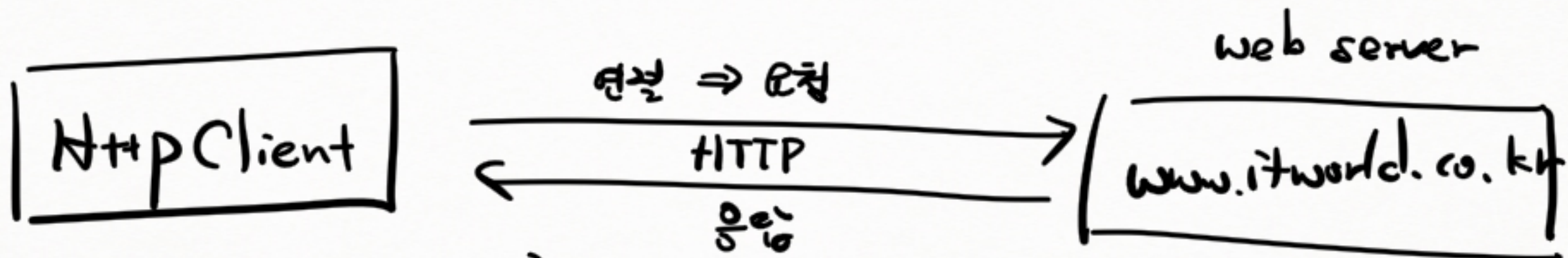


Stateless 방식!









Hyper-Text Transfer Protocol => HTML 송수신 통신규칙

고수준의 문서
↓
문서 (text) + 다른 문서의 연결정보 + 문서 구조 ... => 문서포맷

Hyper-Text
Markup
Language

⇒ RFC-2616 명세서

HTTP

① 요청

method

protocol

version

GET /index.html HTTP/1.1 ← request-line

{ Host: www.itworld.co.kr
User-Agent: ... } header (부가정보)

{ } ← 요청 끝

HttpClient

HttpServer

명령: 해석해서 관성에 처리

② rendering

{ HTTP/1.1 200 OK ← status line

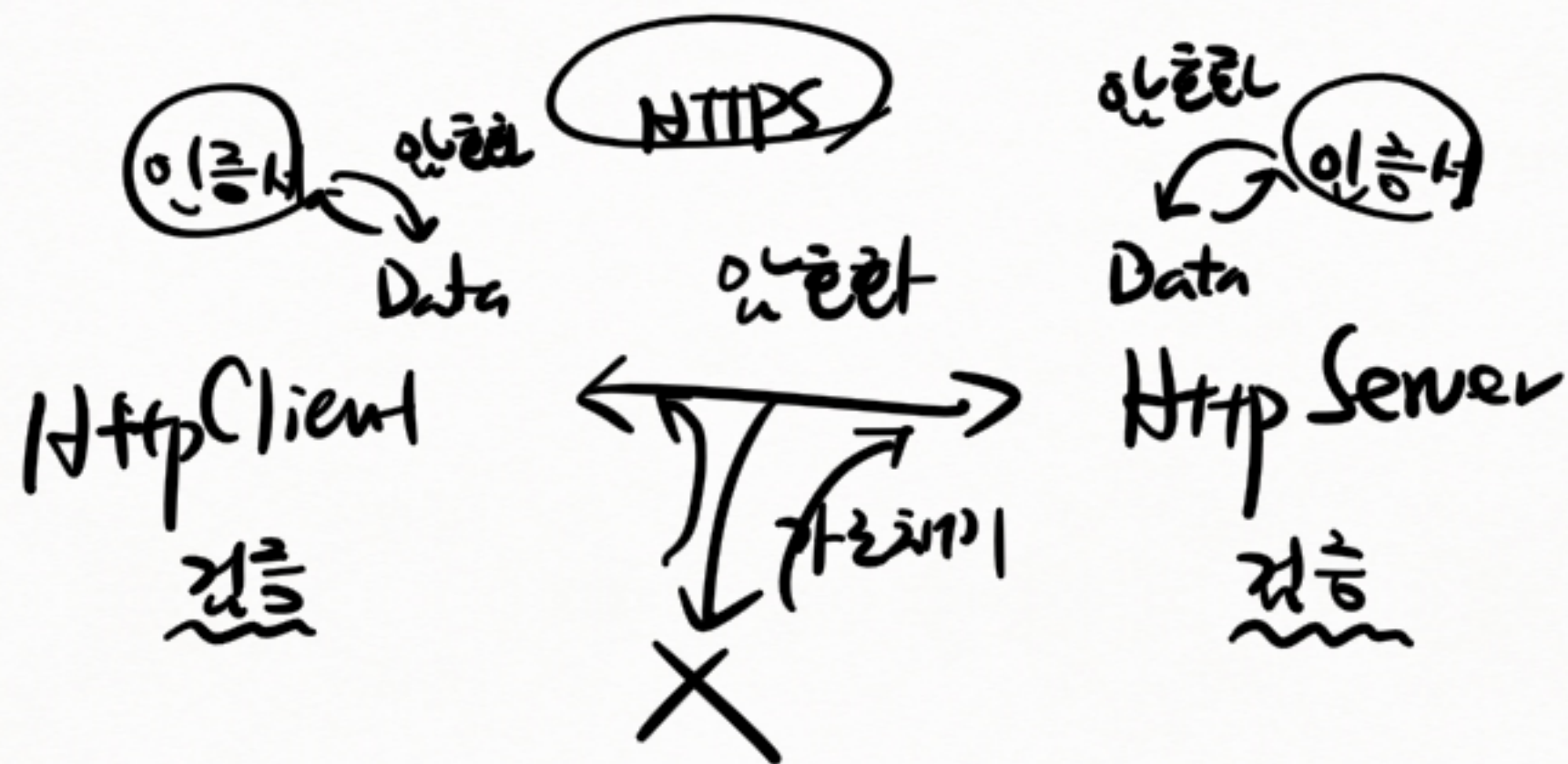
Content-Type: text/html; charset=UTF-8

Content-Length: ...

} header (부가정보)

③ 응답

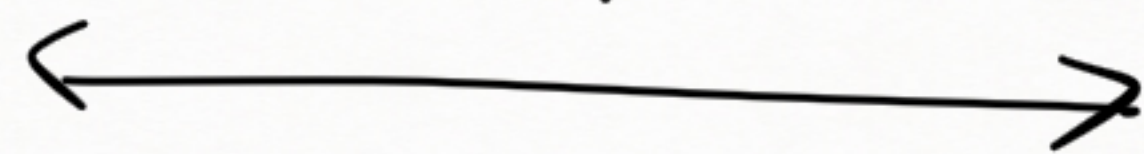
{ } ← 부가정보 부분은 규약 (message-body) { 렌더링 데이터



프로토콜

a) FTP

프로토콜 client



프로토콜 Server

a) FTP Client

HTTP client

||
Web Browser

FTP Server

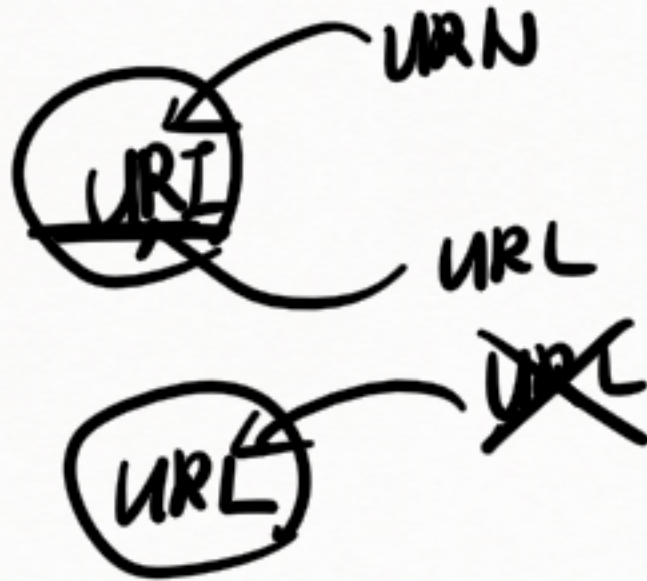
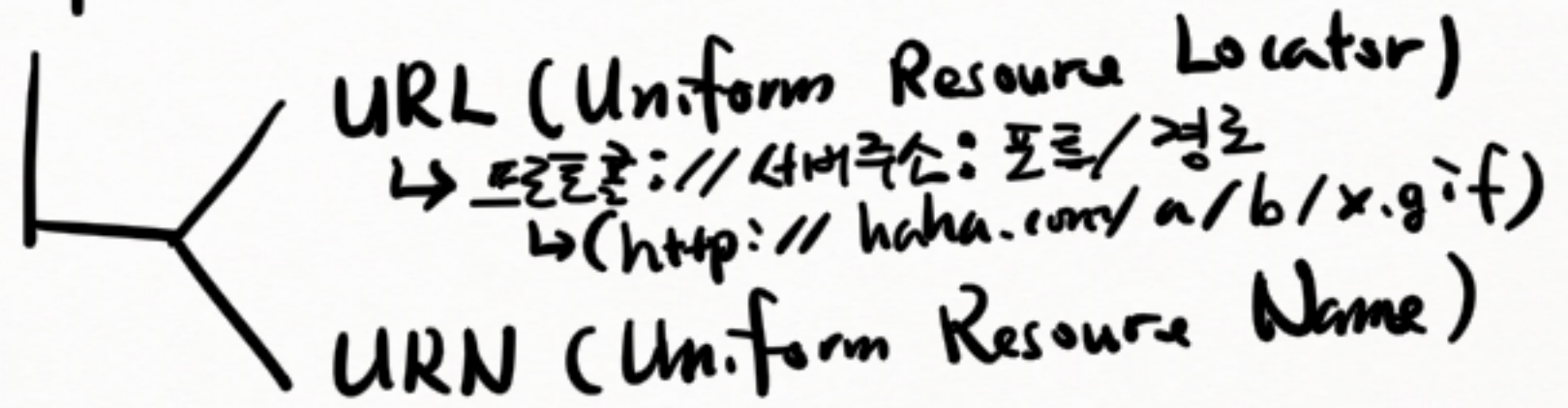
HTTP Server

||
Web Server

인터넷 상의 자원 위치를 표시하는 방법



Uniform Resource Identifier (URI)



Base 64

binary $\xrightarrow{\text{변환}}$ text

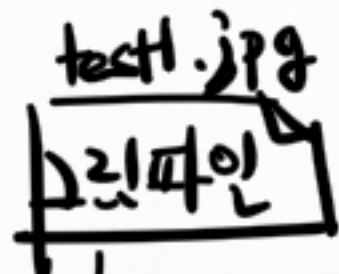
Why?

binary data를 텍스트 데이터에 포함시키고 싶은 때.

test1.html

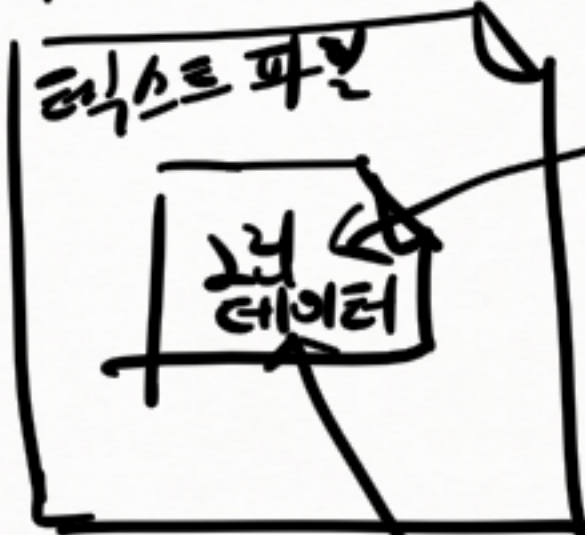


링크

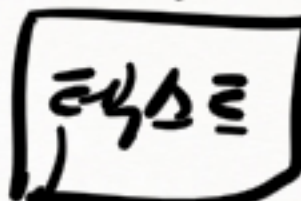


임의 붙자!

test2.html

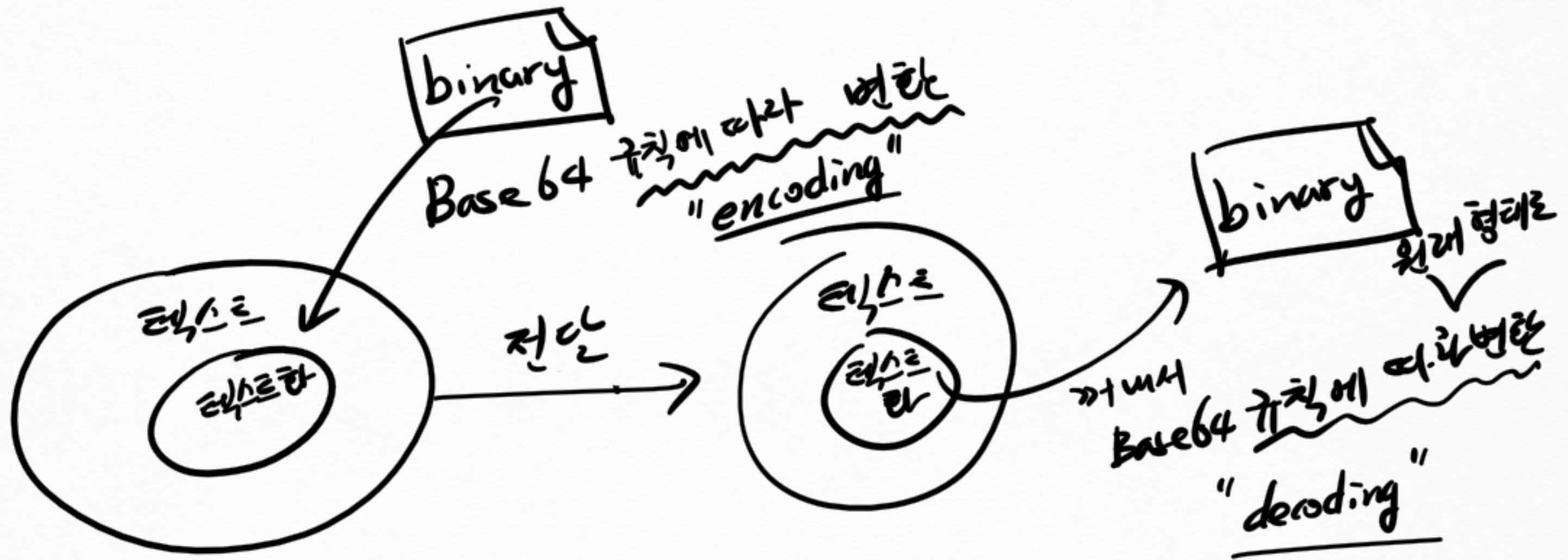


Base64 $\rightarrow$ 텍스트



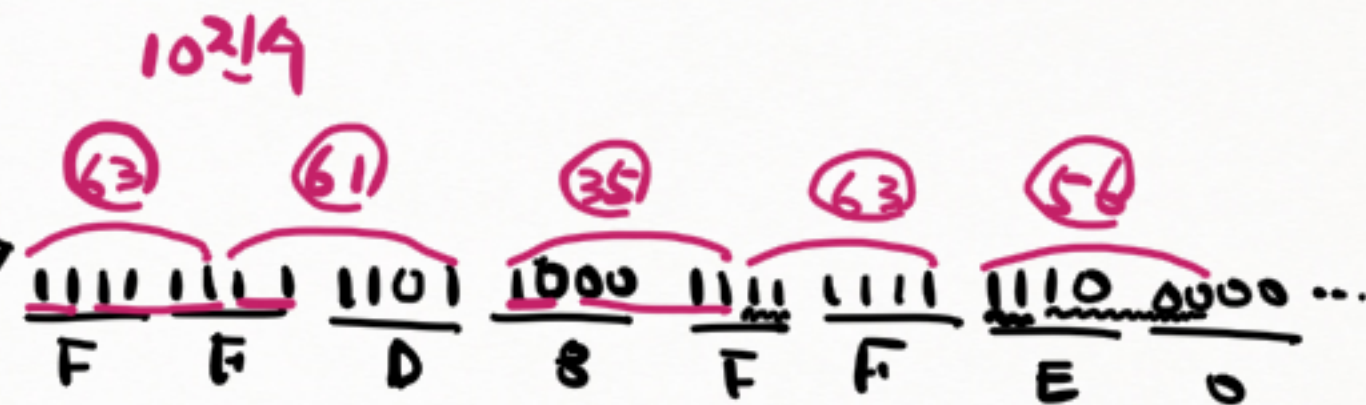
OK!

- 영자 대소
- 숫자
- 특수문자



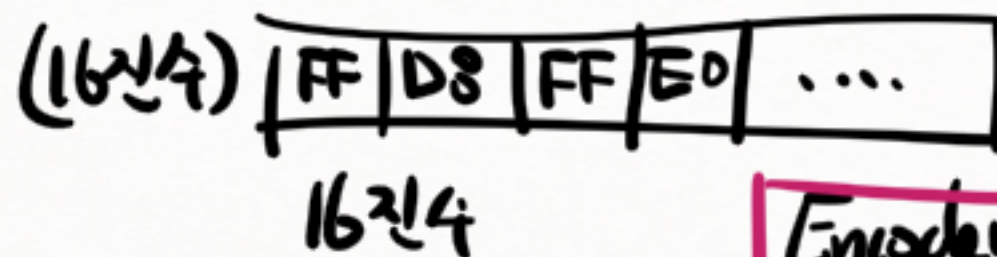


Base64



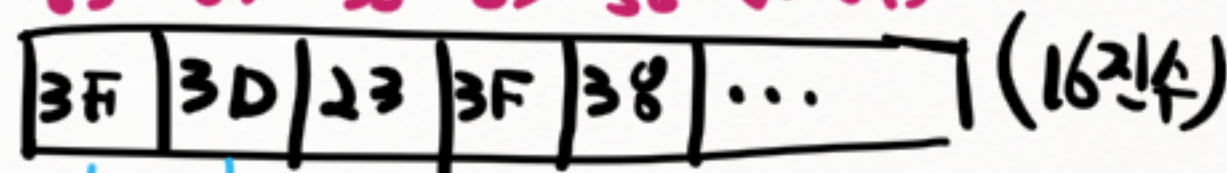
8 bit 단위

① 6 bit 단위로 잘라서 1 byte 버퍼에 넣는다.



Encoder.encode()

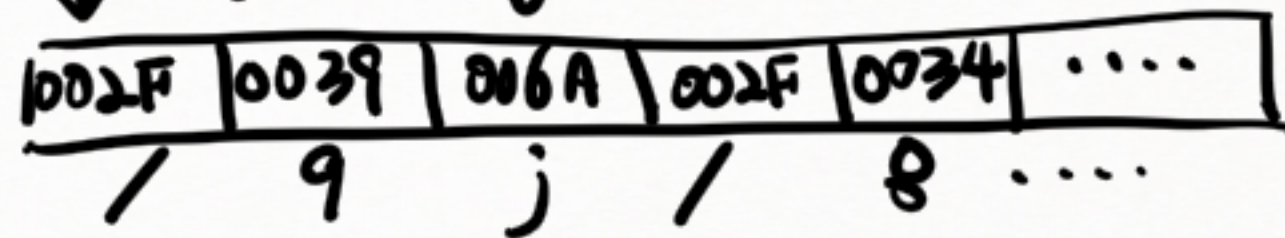
return



② Base64 인코딩 규칙에 따라 변환



(UTF-16BE)
new String();



PC

Server

Desktop App.

계산기

- ① 로컬에 설치
- ② 버전업 → 재설치
- ③ 대량의 PC 관리가 어렵다

Application의 서버에서 실행
"Web" + "Application Server" (AS)
(WAS)

Network App.

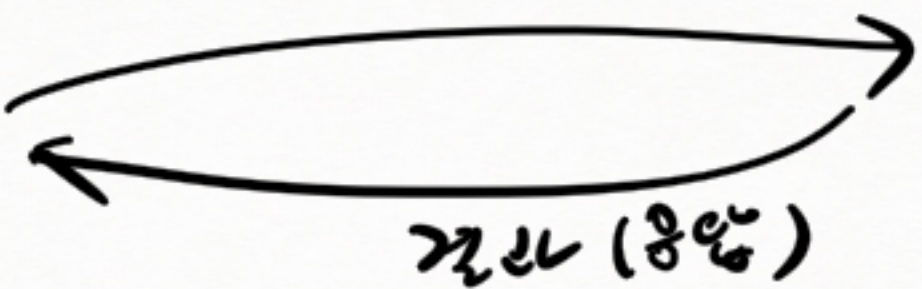
계산기 Client

- ① 로컬에 설치
- ② 버전업 → 재설치
- ③ 대량의 PC 관리가 어렵다

계산기 Server

+ 기능

Web Browser



기능 추가

