



2021.08.30 (월 45일차) 09:40

* 호출자에게 예외상황을 알리는 방법 - 리턴값

com.eomcs.exception.ex1

.Exom 0130 - 0131

```
int compute (String op, int a, int b) {  
    =====  
    return ○;  
}
```



그러나 그 리턴값이

정상적인 계산결과일 수도 있다. ←

리턴값을 이용하여 예외상황을

알리는 방법의 한계다.

2021.08.30 (월 45일차)

* 예외 처리 방법의 종류이유.

대 4종? 에러상황을 스코프에게 "정확하게" 알려주기 위해서

↳ return 은 정확하지 X

ex2

① 리턴값으로 에러상황을 알리는 방식의 한계 극복!

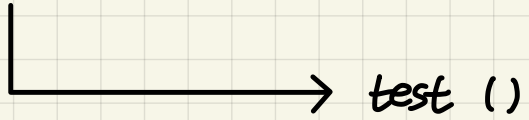
② 예외가 발생하더라도 JVM을 멈추지 않게 만드는 것!

2021.08.30 (월 45일차)

* 예외 던지고 받기

.ex3.Exam011 10:30

main ()



test ()



ml ()

=====

throw new RuntimeException ("_")

×

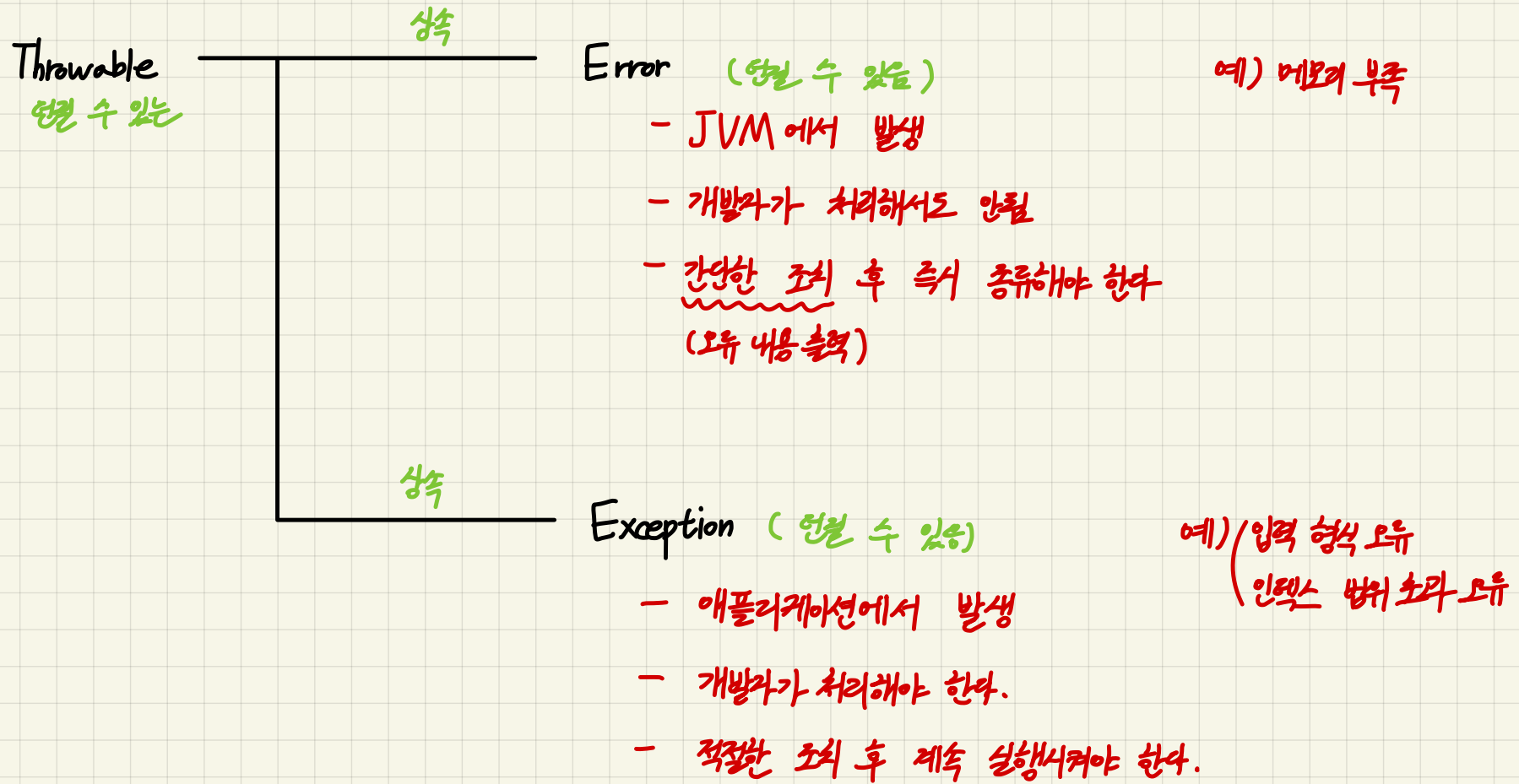
2021.08.30 (월 45일차)

```
try {  
    ===  
} catch
```

throw

2021.08.30 (월 45일차)

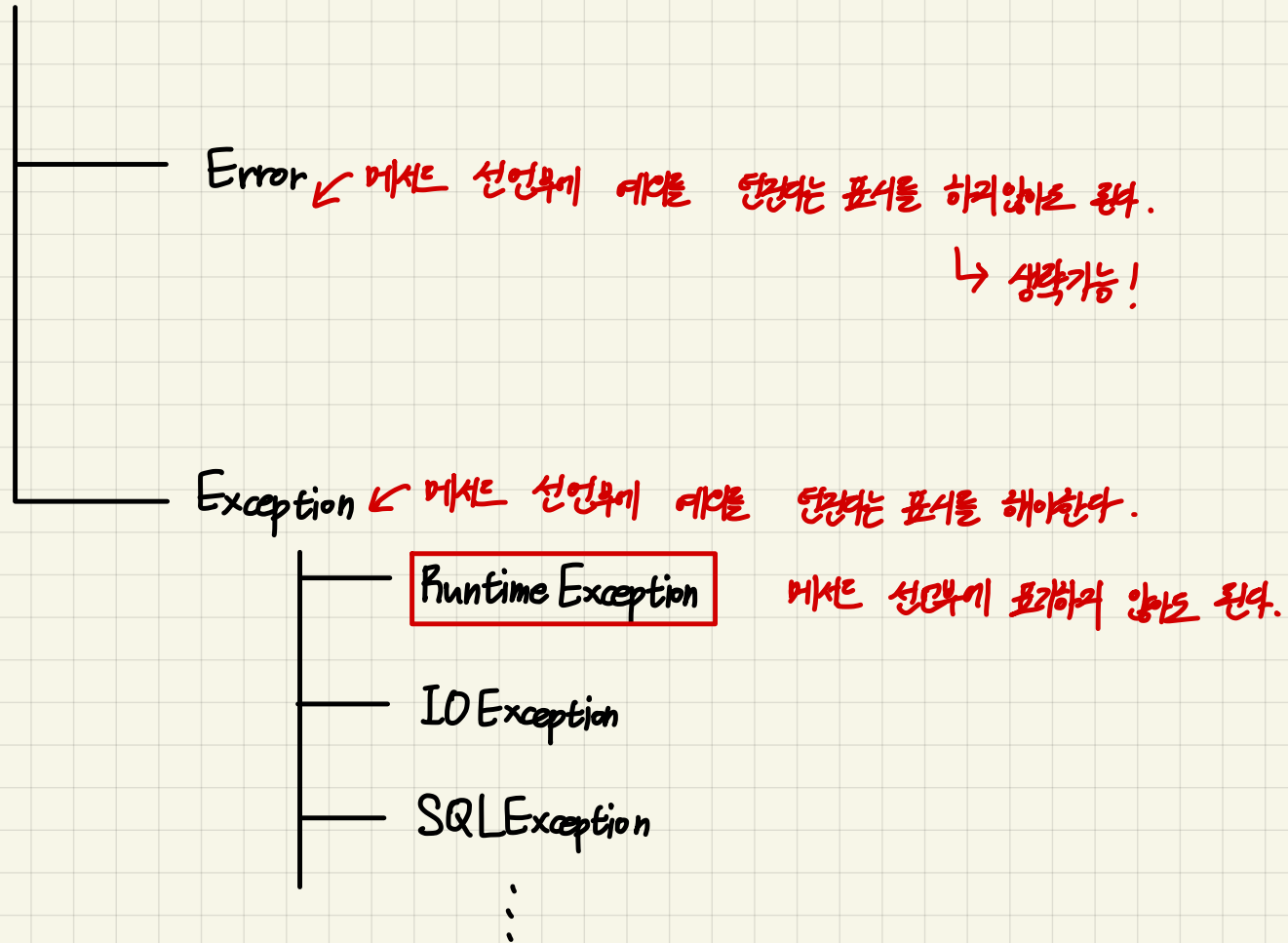
* 예외 종류



//

throws String (X)
└─ 연결 수 없음

Throwable



2021.08.30 (월 45일차) 11:30

Throwable

└─ Exception

├─ RuntimeException → catch 해주려고 안 받아도 됨.

├─ SQLException

└─ IOException

2021.08.30 (월 45일차) 11:50

exam 0460

* catch 문 parameter 에서 (RuntimeException ! SQLException ! IOException) 가능

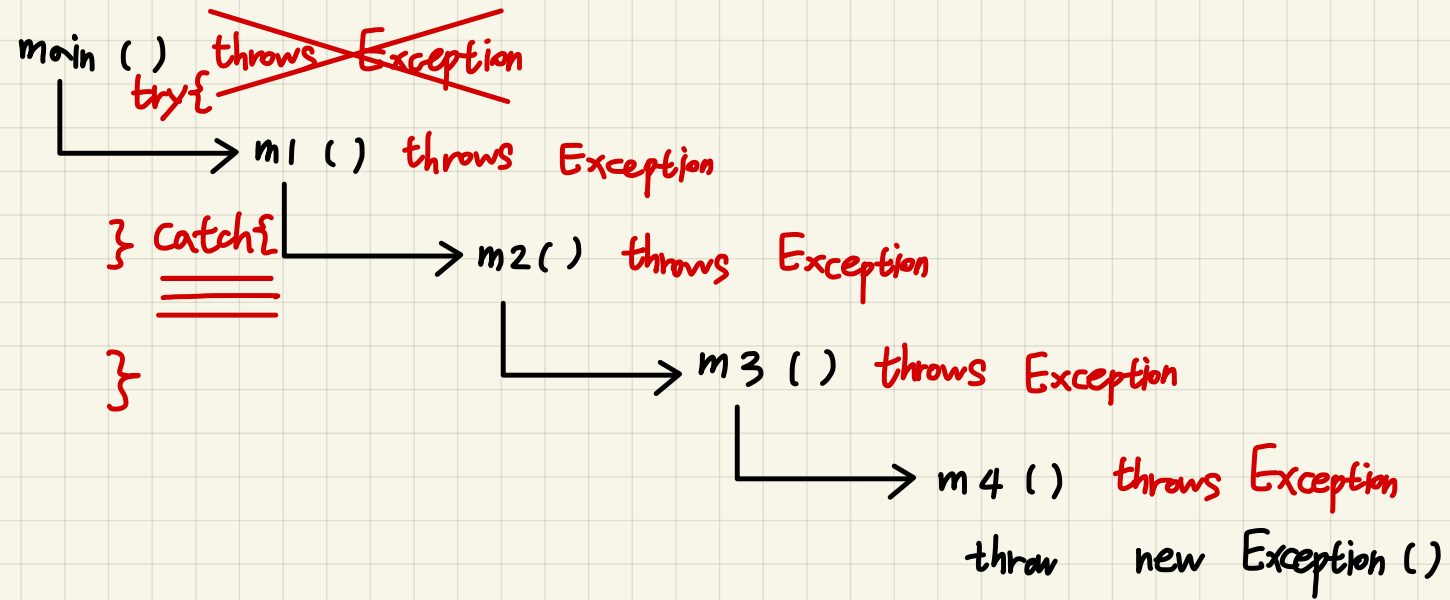
13:20

exam 0610

2021.08.30 (월 45일차) 13:55

ex 4 . Exam 0130

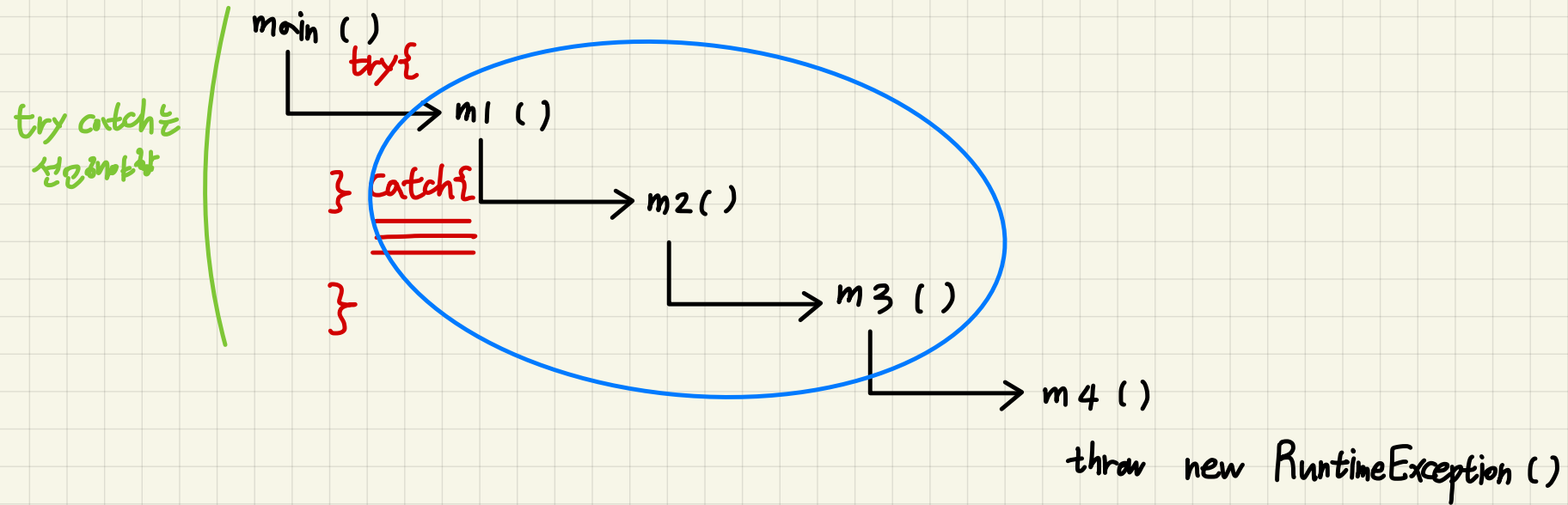
* Exception 객체 만들기



2021.08.30 (월 45일차)

ex 4 . Exam 0130

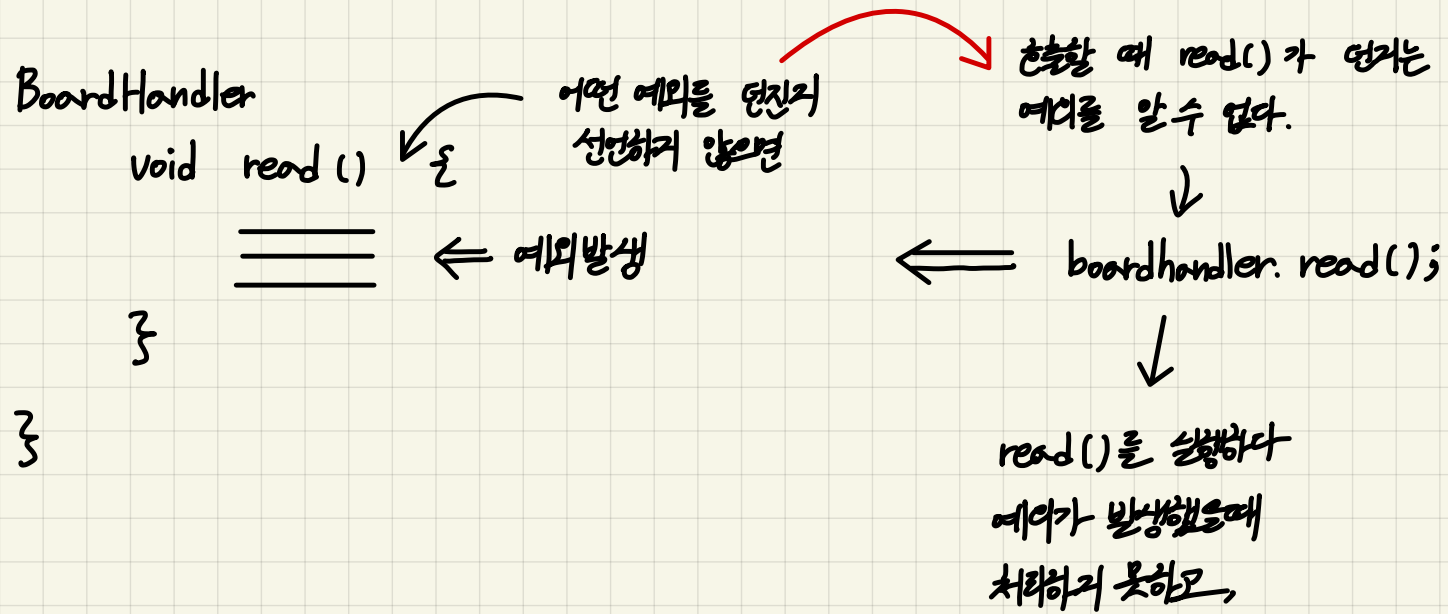
* Exception 예외 던지기



2021.08.30 (월 45일차)

* 예외 클래스 상속받기

14:30



2021.08.30 (월 45일차)

* 예외 클래스 상속받기

BoardHandler

void read () throws RuntimeException {

≡

← 예외발생

}

}

① 이렇게 예외를 선언하면

try {

boardHandler

① 아왕 예외를 당기는 김에
의미있는 이름의 예외를 당기면

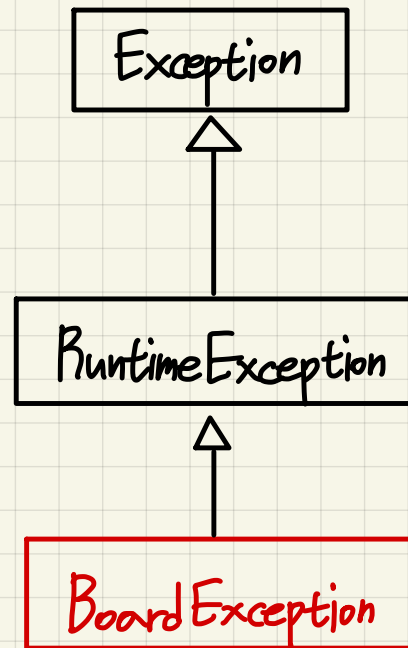
3

2

3

2021.08.30 (월 45일차)

* 예외 클래스 상속 받기



서브클래스는 이름을 통해
예외에 대한 의미를 전달하는 것이 목적이다.



그래서

ex03

ex05

0640 -

com.comcs.exception.ex91 ~ ex 92 자습

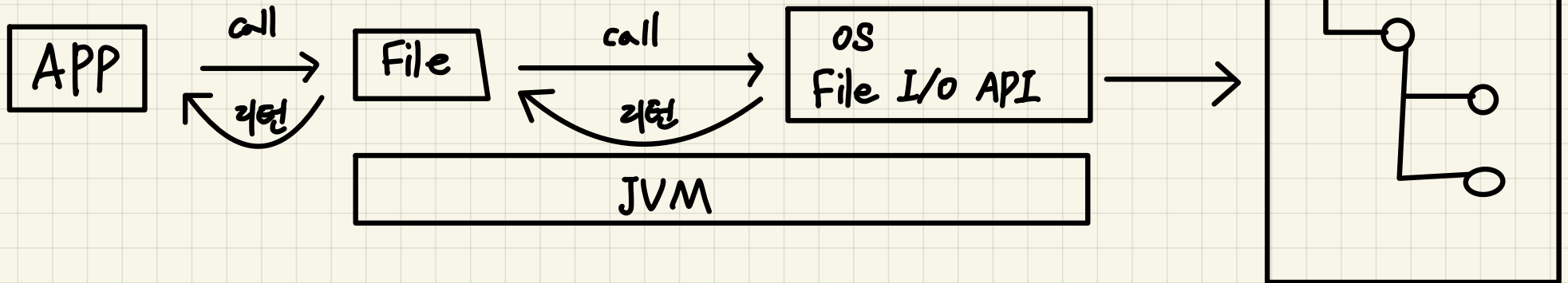
2021. 09. 08 (52화) 9:30

* File 클래스

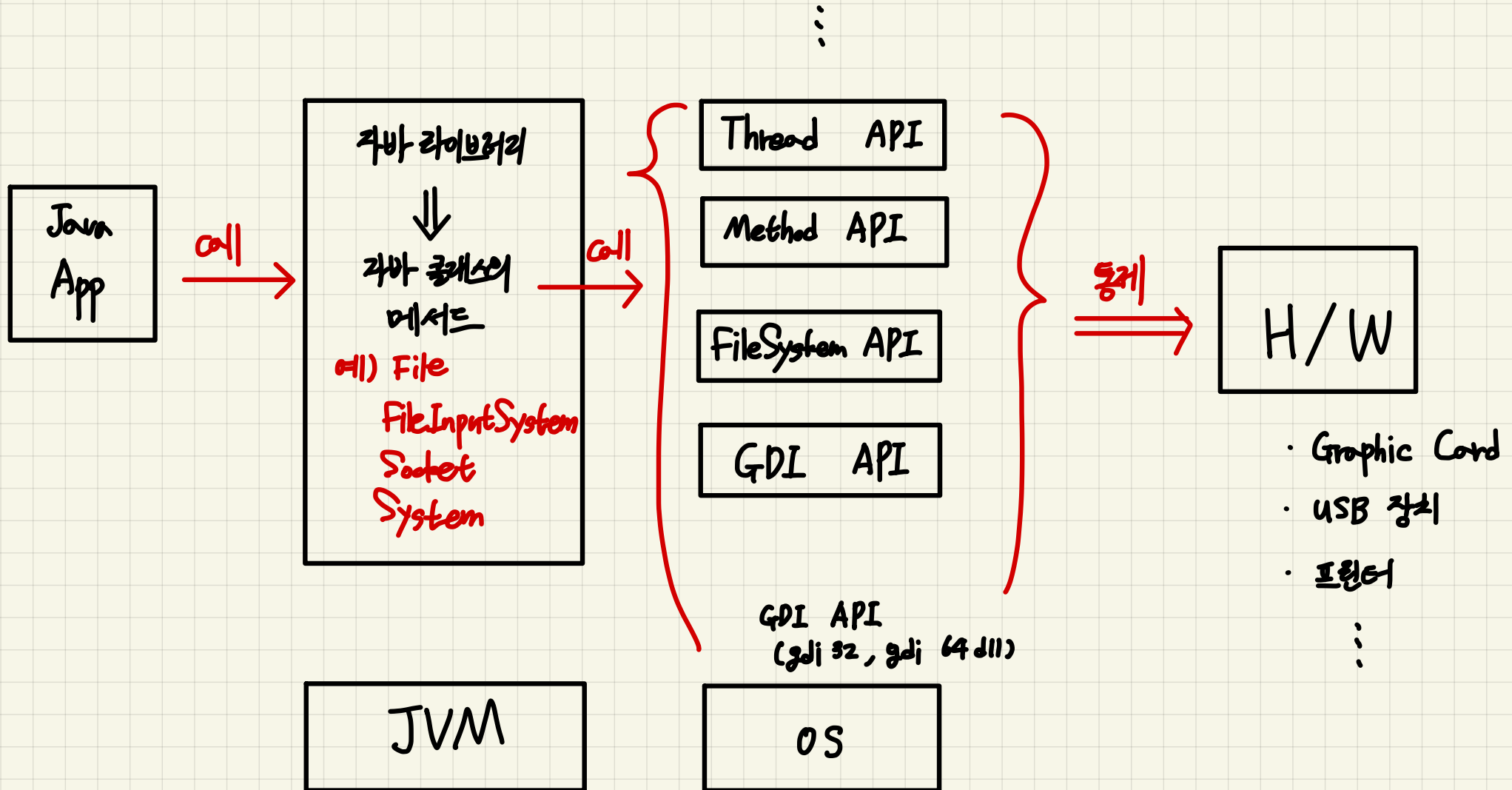
String → 문자열

Data → 바이트

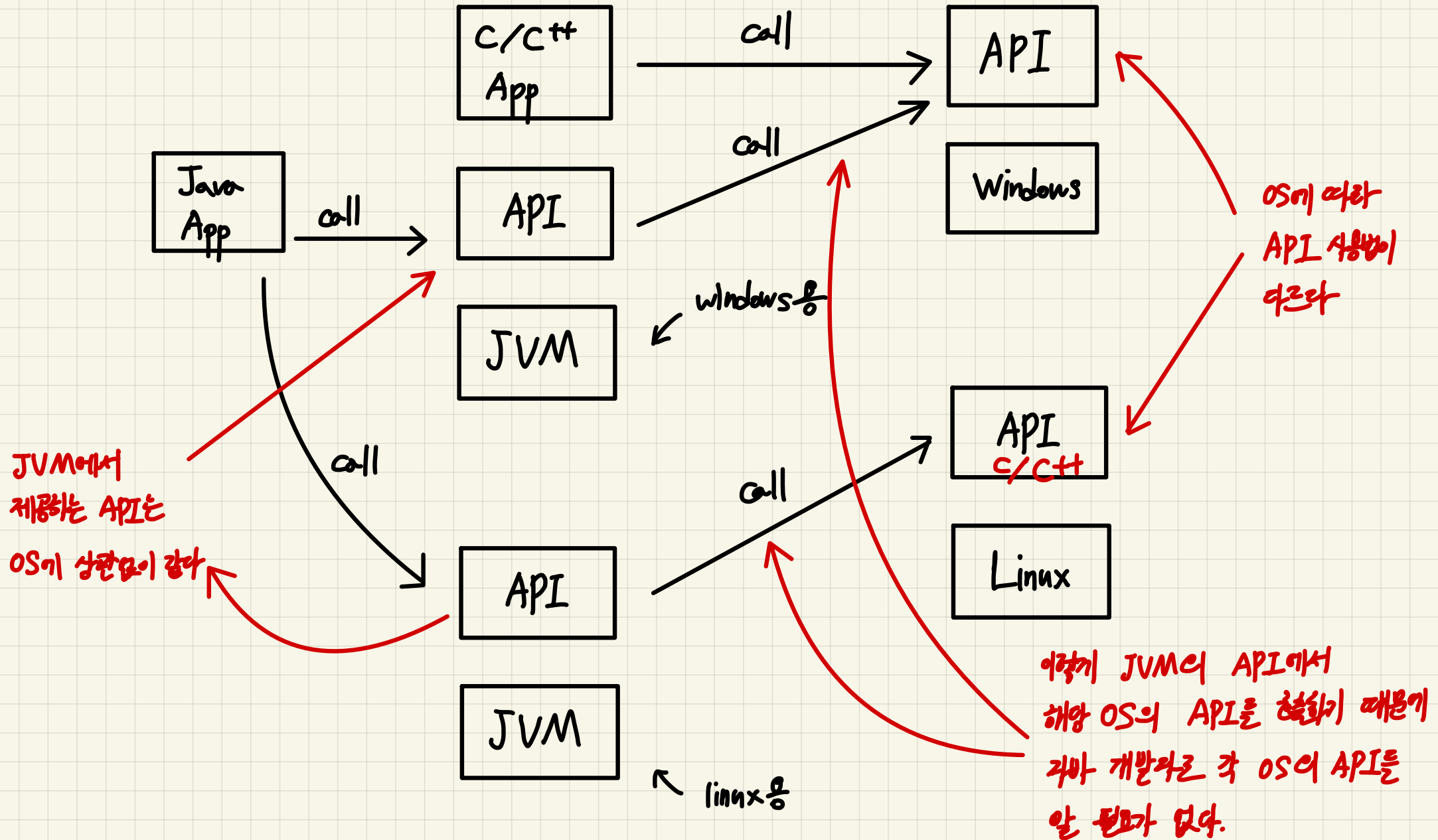
File → 파일의 객체



2021. 09.08 (52화) 9:35
loison

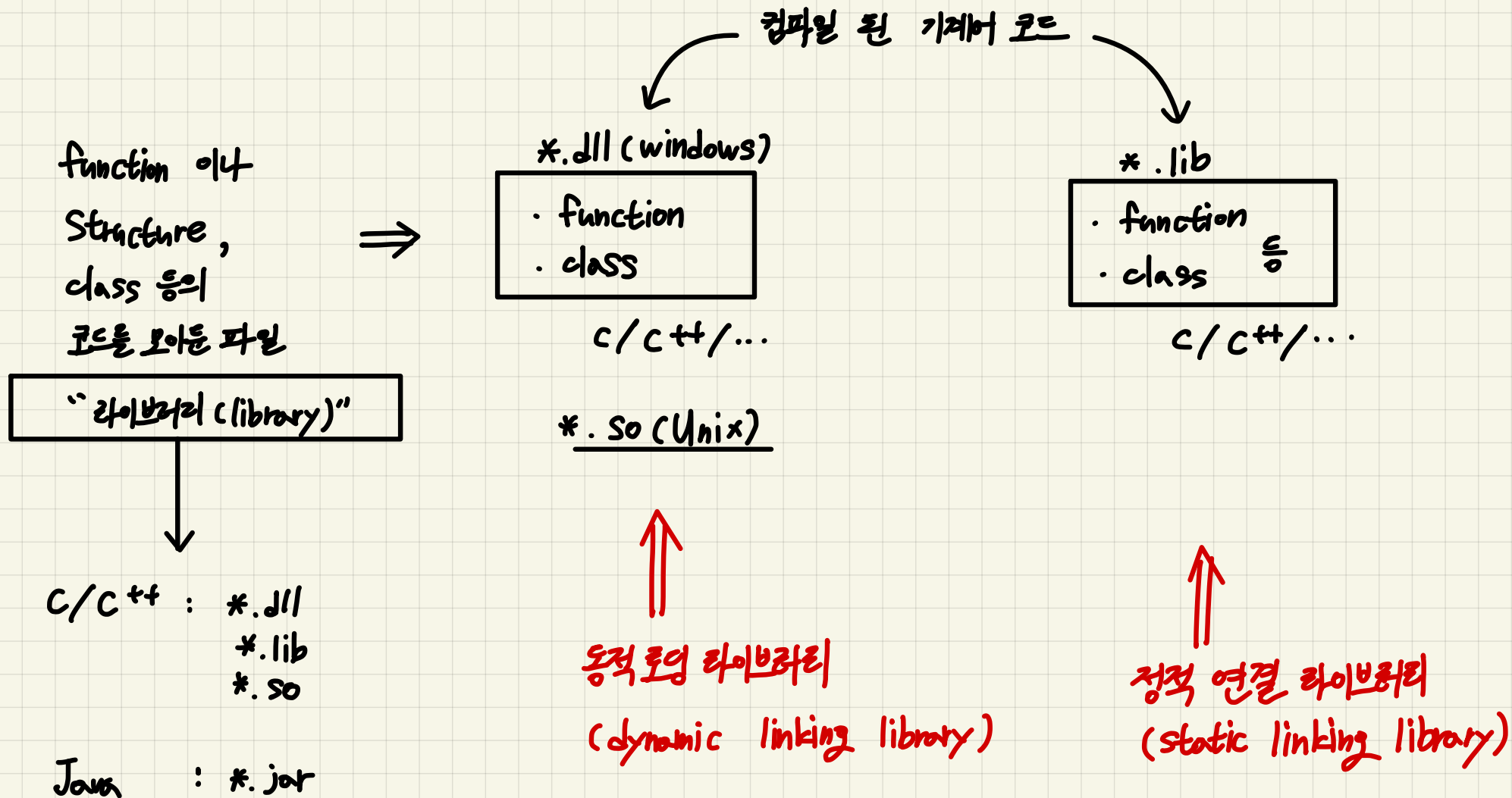


2021. 09. 08 (52화) 10:45

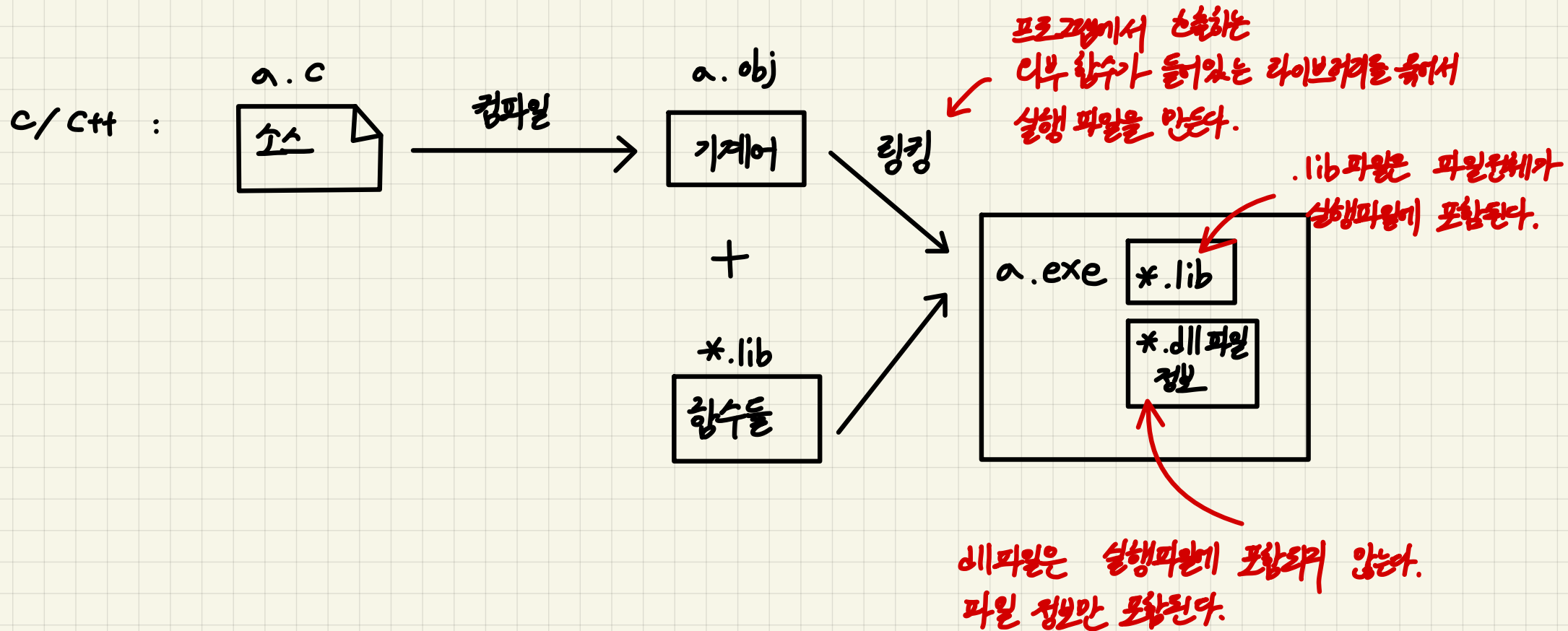


2021. 09.08 (52화) 9:40

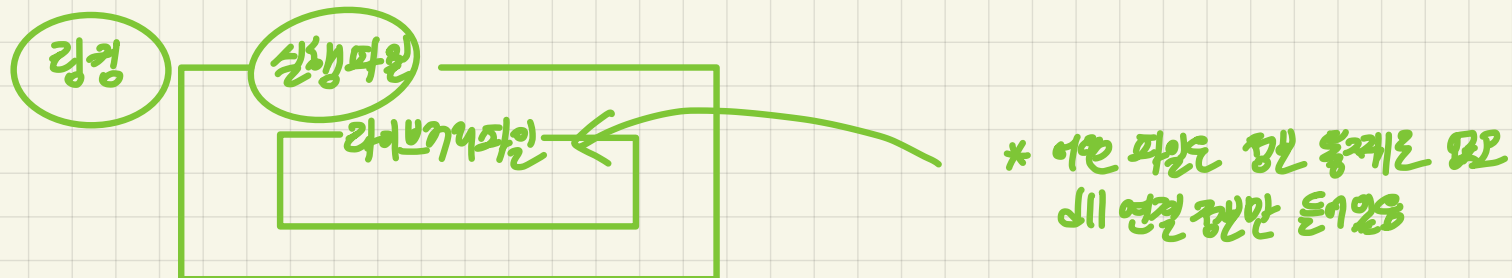
* 개발 기반 객 : dll 와 lib



2021. 09.08 (52화) 9:50

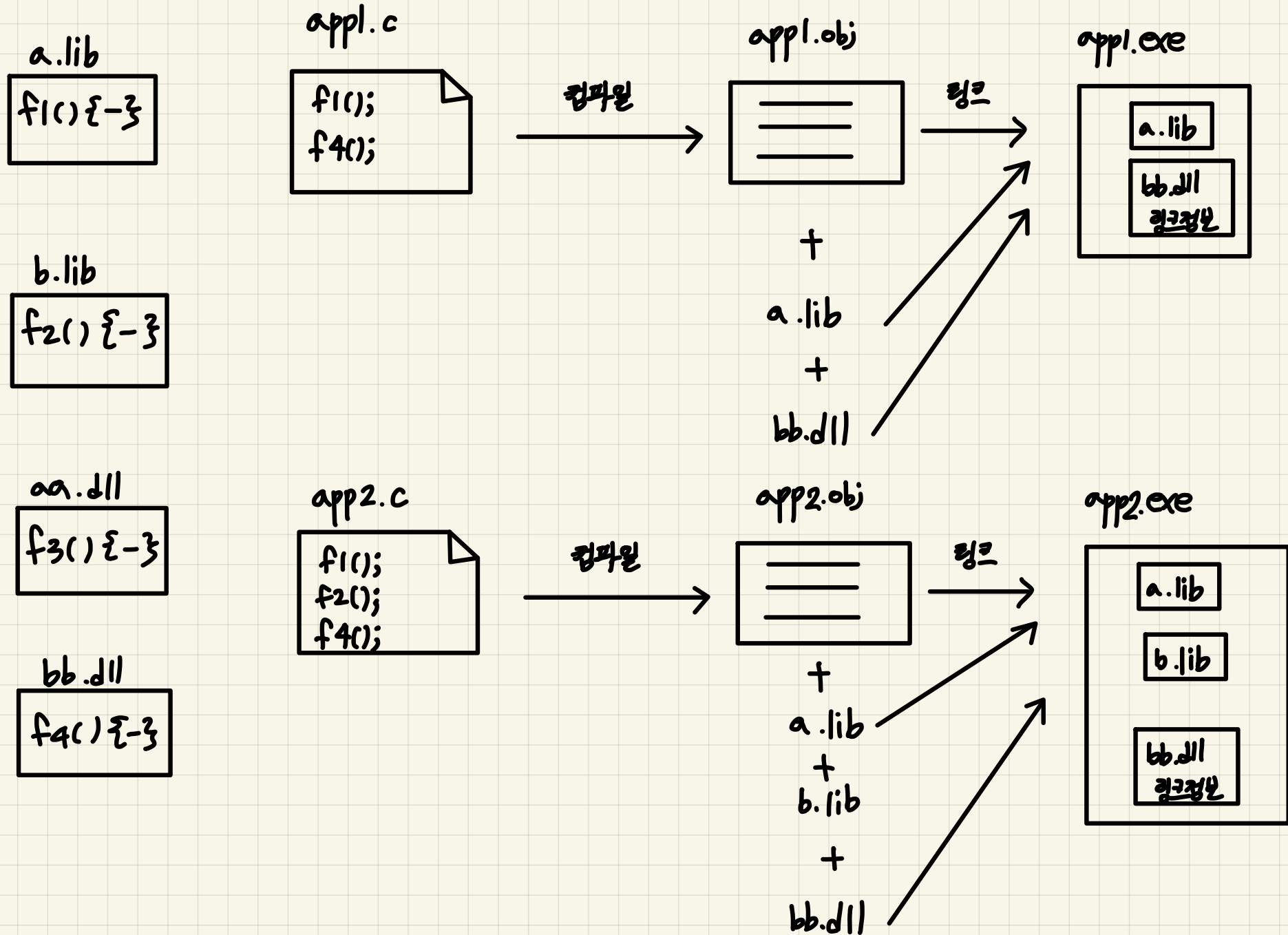


모든건 운영체제에 function을 호출하는 것



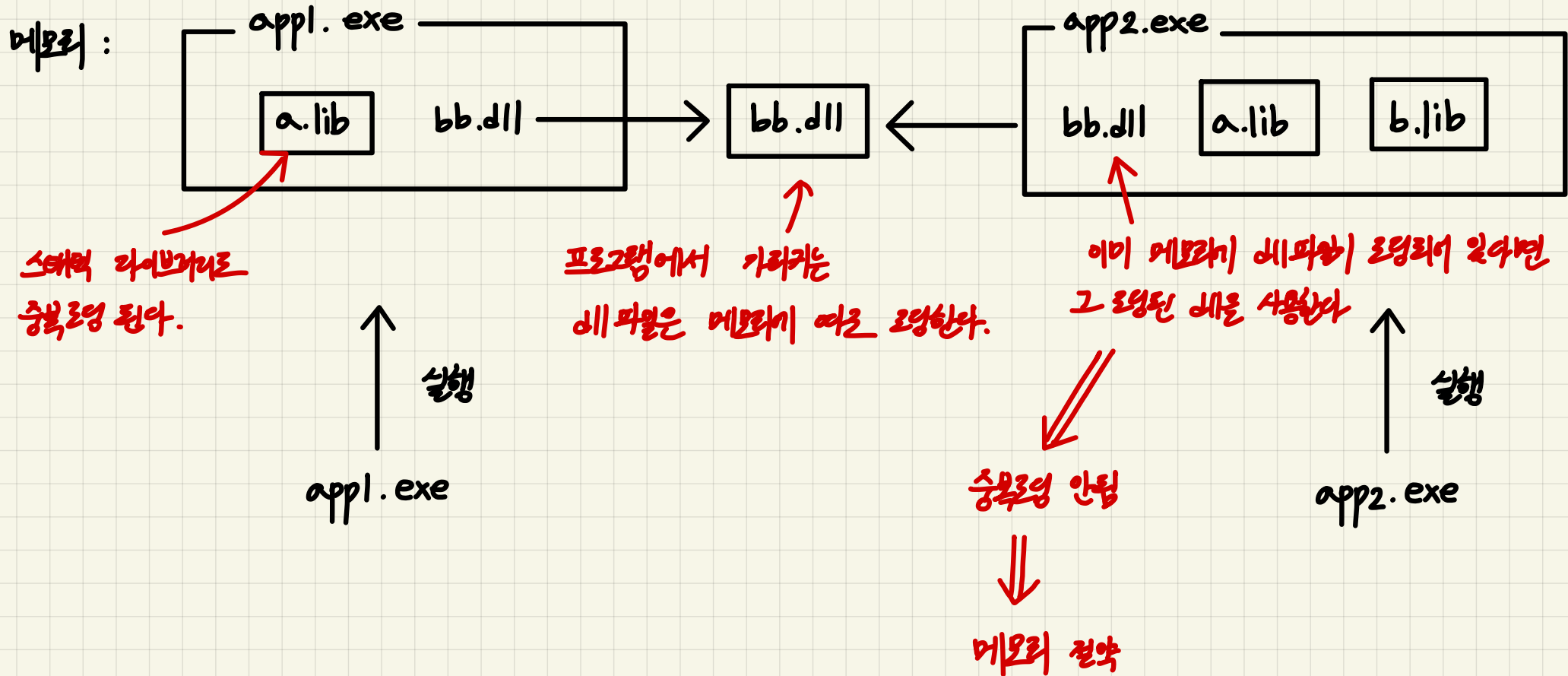
2021. 09.08 (52화) 7:55

* 프로그램 실행과 라이브러리



2021. 09. 08 (52주차) 10:05

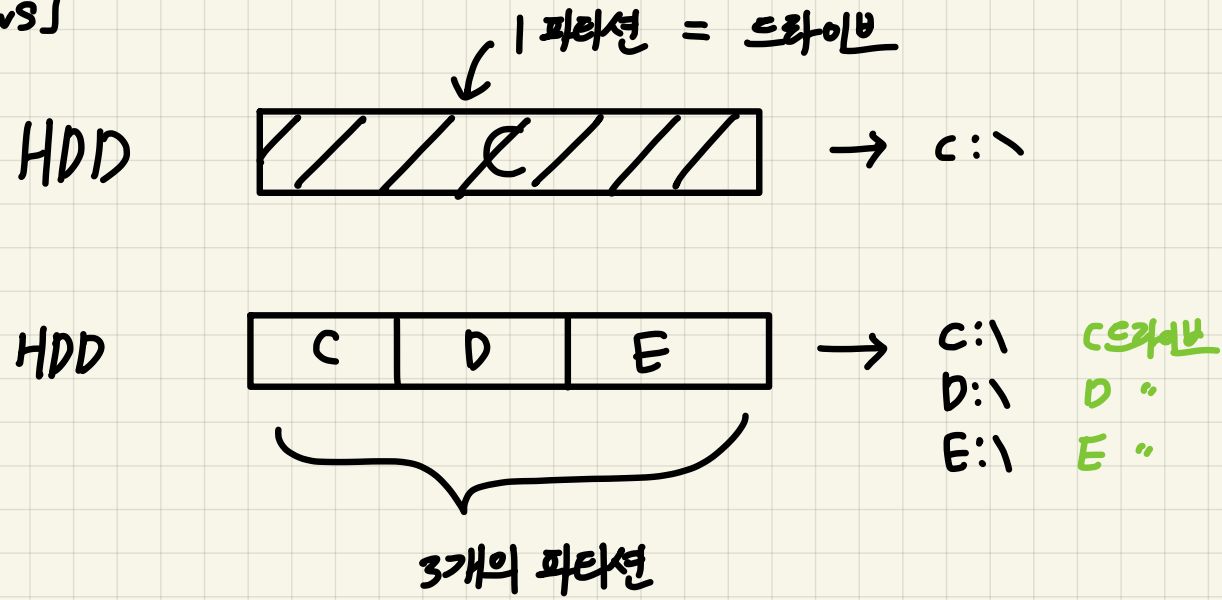
* 프로그램 실행과 라이브러리 로딩



2021. 09. 08 (52주차) **3125**

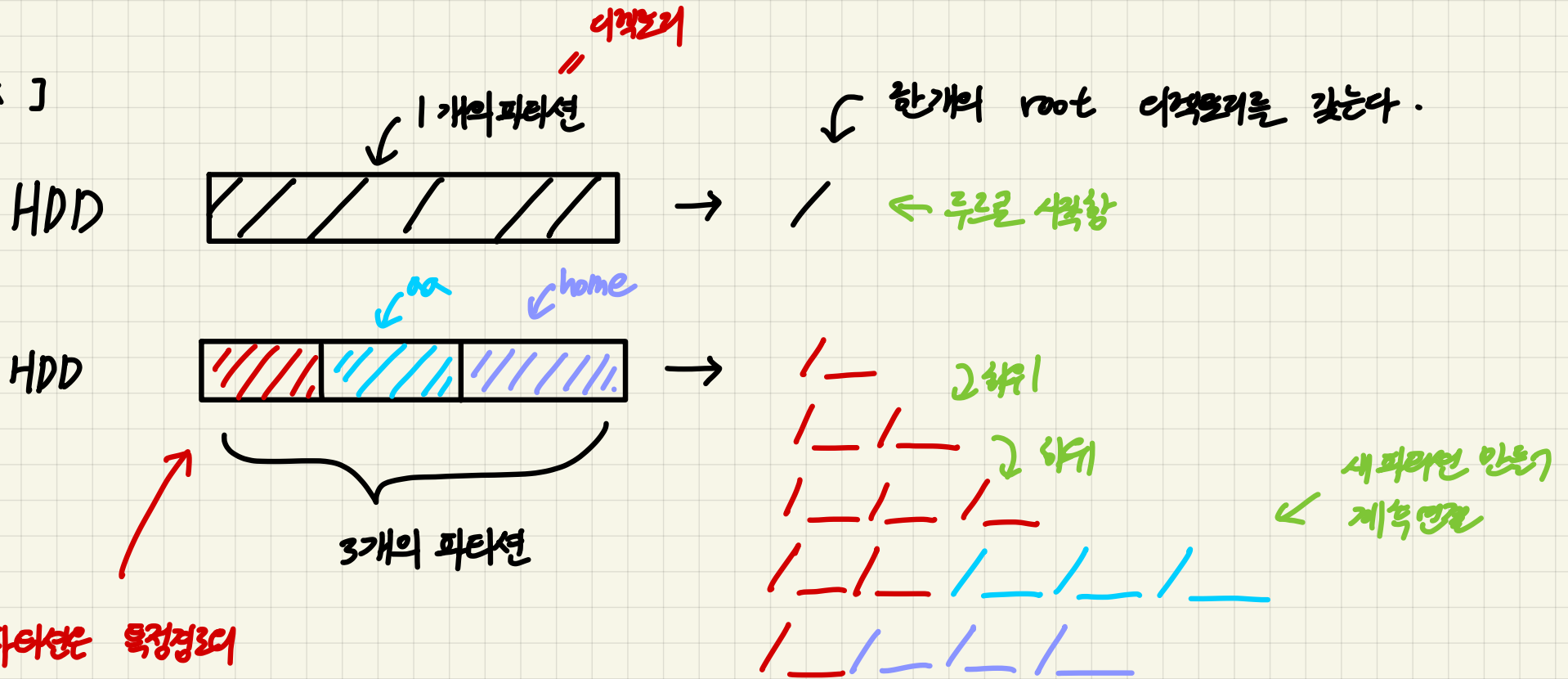
* HDD와 드라이브

[windows]



* HDD와 드라이브

[Unix]



각 파티션은 특정정도의 디렉토리를 연결된다.

예시

Example file system structure:

- `/user`
- `/bin/sbin`
- `/usr/local/bin`
- `/data/dl/aa/-/-`
- `/data/home/ / -`

Annotations include:

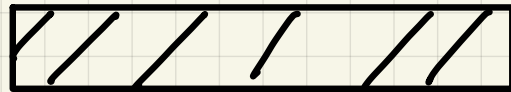
- * aa 폴더에 저장 (Save to aa folder)
- home 폴더에 저장 (Save to home folder)
- 그밖의 모든것 위에 계속+ (Continue everything else on top)

* HDD와 드라이브

모든 파일에 저장 (4개의 dir.file)
특정 dir → 별도의 파티션 가진 것

[Unix]

HDD



1개의 파티션
디렉토리

한개의 root 디렉토리를 갖는다.

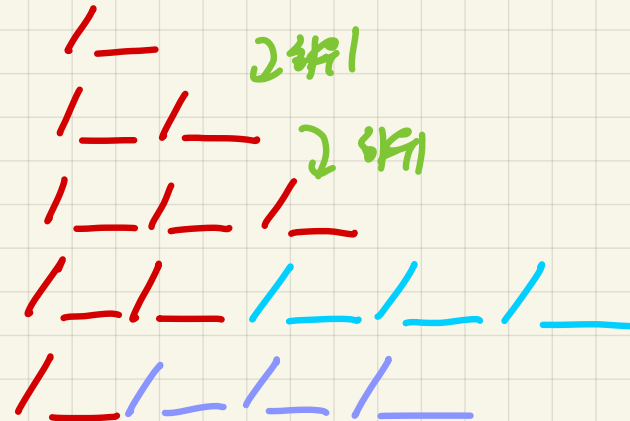
/ ← 루트로 시작함

HDD



각 파티션은 독립적인
디렉토리를 연결된다.

3개의 파티션



새 파티션 만들기
← 계속 연결



/user
/bin/sbin
/usr/local/bin
/data/d1/aa/-/-
/data/home/ /-
/data2/ok

* aa 폴더에 저장
home 폴더에 저장
그밖의 모든 것 위에 계속+

2021.09.08 (52일차) 3:47

* File Input Stream

`read(): int`

1byte를 읽어
int 값으로 리턴한다

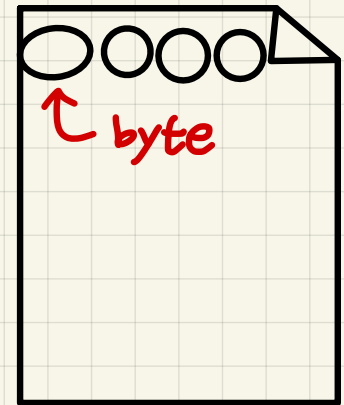
`read(byte[]): int`

배열이 다 찰때까지
파일에서 바이트를 읽어
배열에 저장한다.

byte를 읽어
배열에 저장한 개수

FileInputStream

temp/test1.data



`read(byte[], int, int): int`

바이트를 저장할
배열의 시작 위치
(인덱스)

읽을 개수

2021.09.08 (52일차) 4:45 * String 객체에서 byte[] 얻기

```
String s = new String("AB가각");
```

char[]

0041	0042	AC00	AC01
------	------	------	------

자바는 문자를 다룰 때 UTF-16 문자표기 따라 문자코드를 다룬다.

s.getBytes()



\$ java -Dfile.encoding = utf-8 --
JVM 환경 변수 설정

41 42 EA B0 80 EA B0 81 ← eclipse에서 실행할 때 : file.encoding = utf-8

41 42 B0A1 B0A2 ← windows 콘솔창에서 실행

41 42 EA B0 80 EA B0 81 ← Unix/Linux 콘솔창에서 실행

} file.encoding을 설정하지 않았을 때 OS 기본 인코딩으로 변환한다.