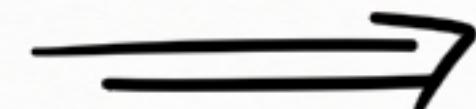
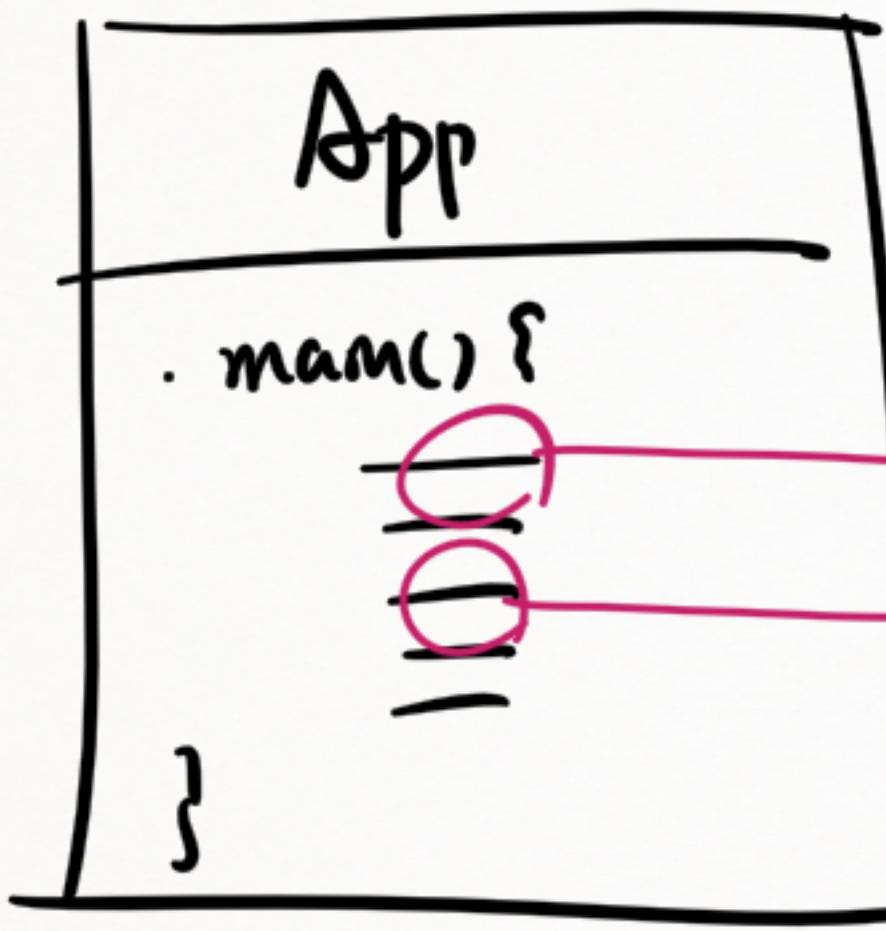
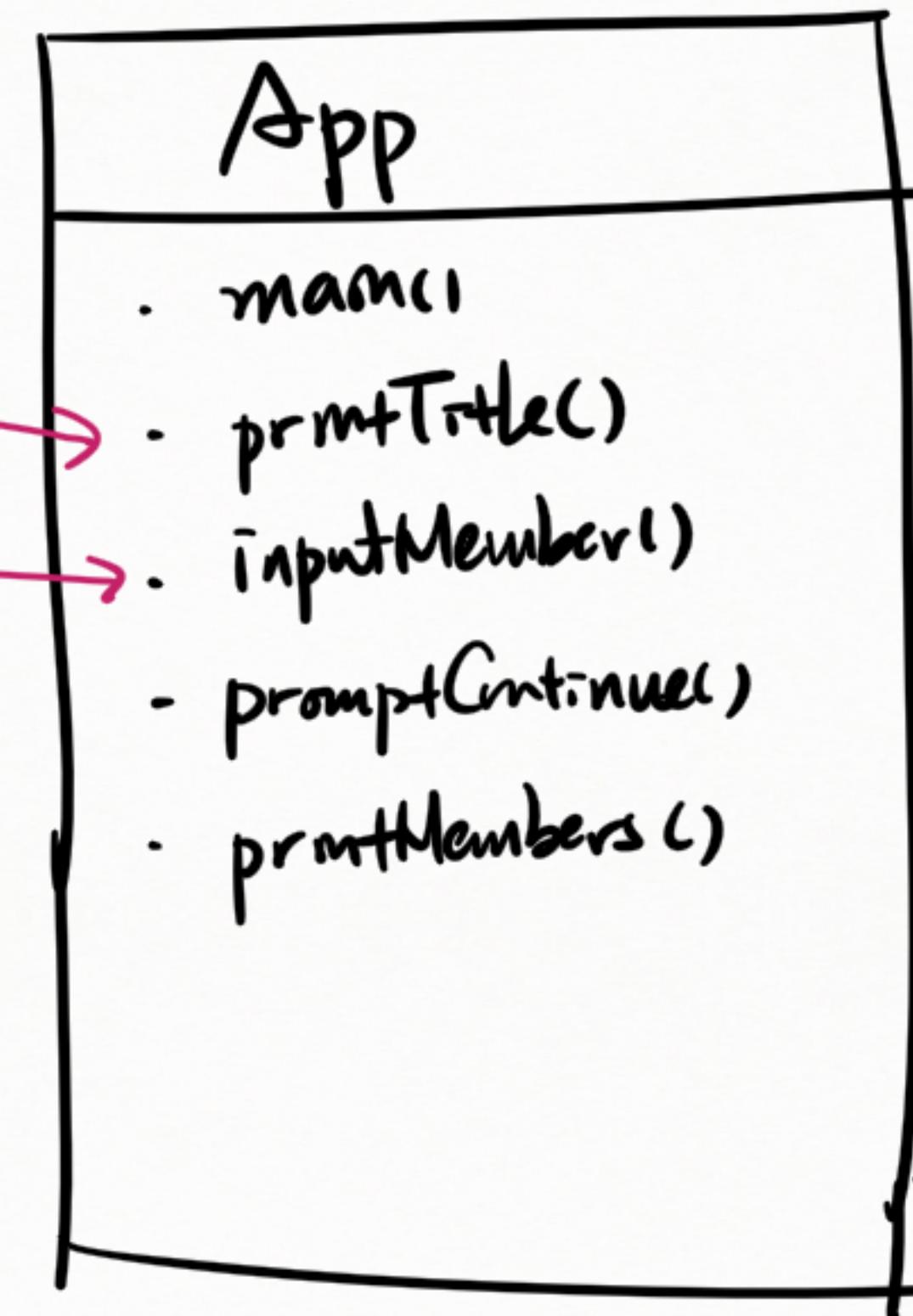


* 1. 디렉트 사용법

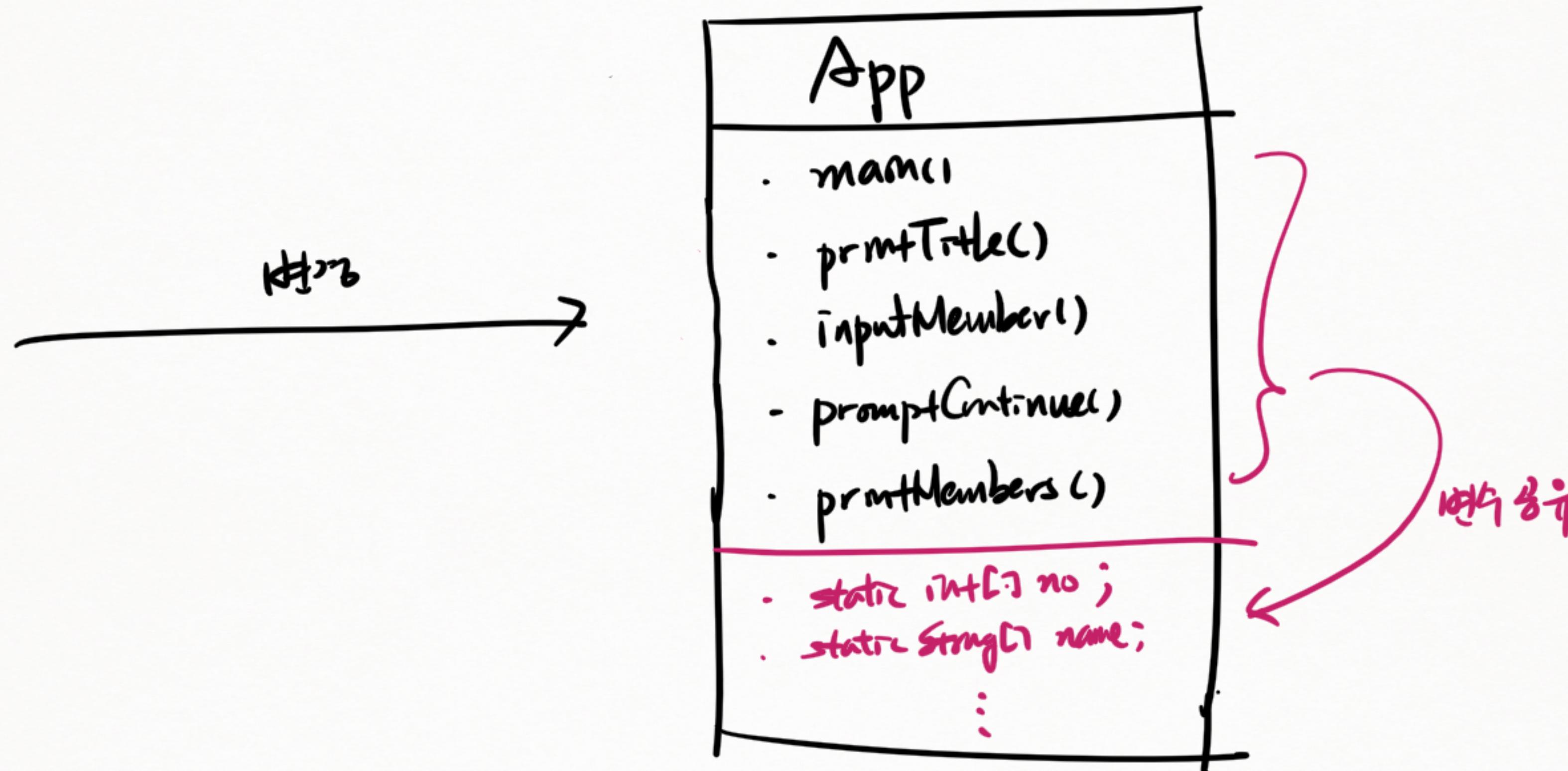
이전



변경

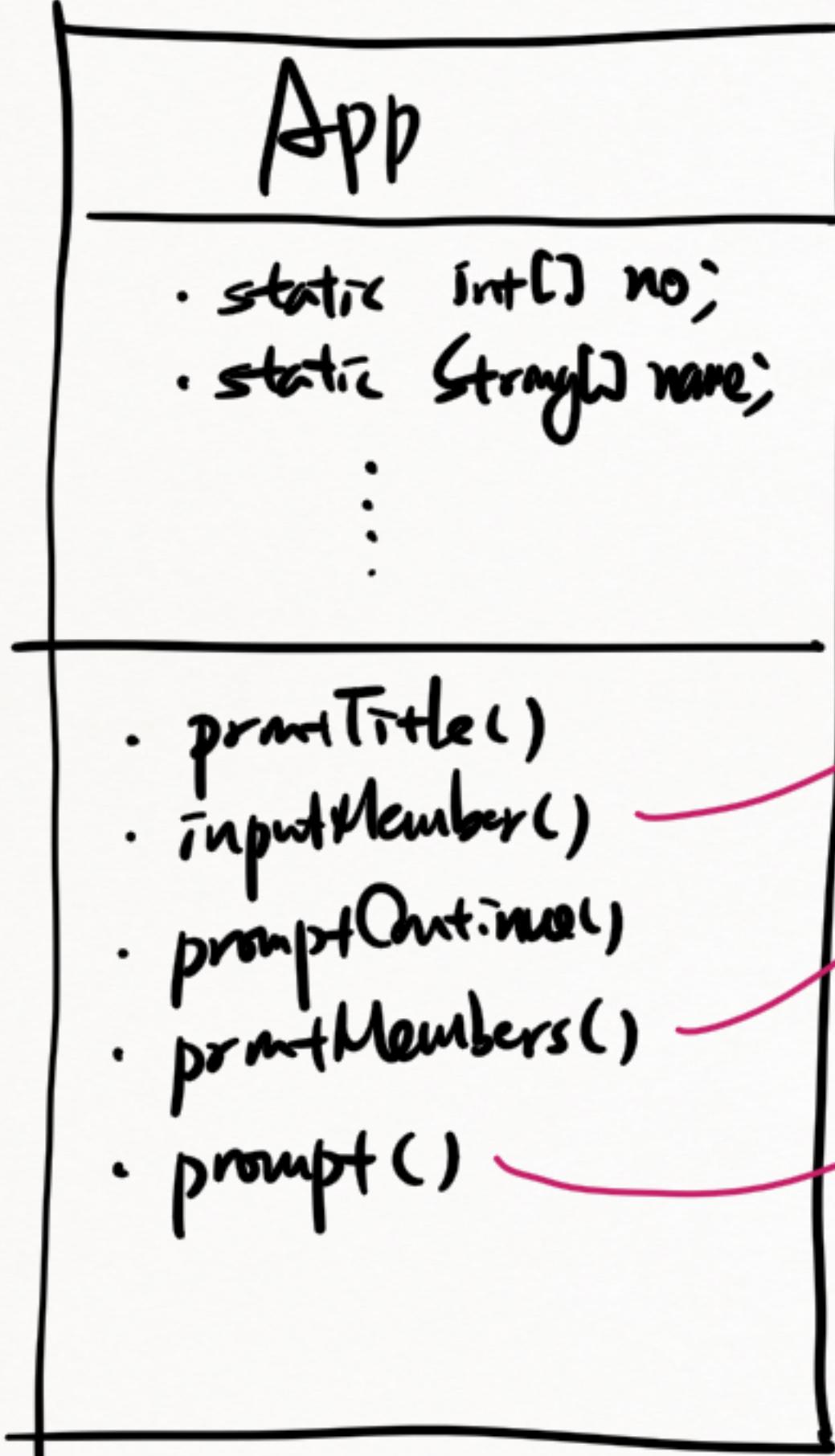


* Q. Lession with Array

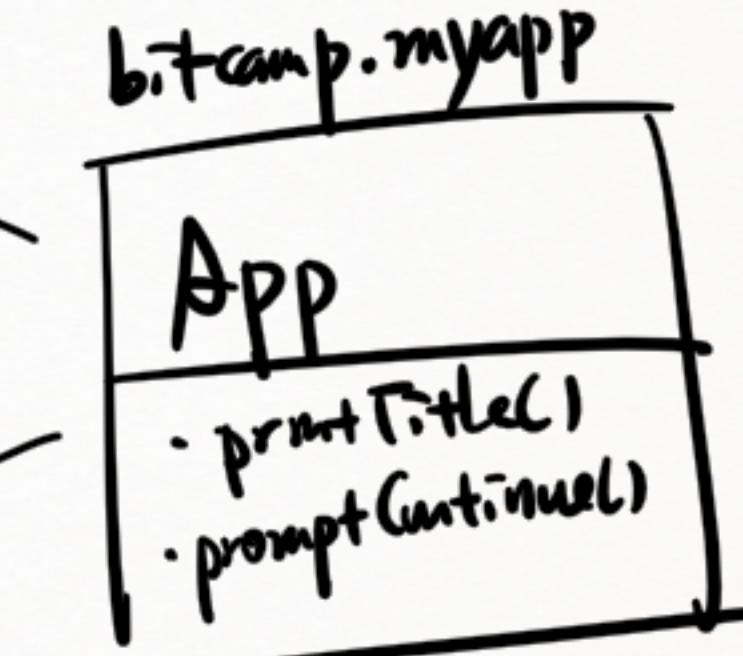
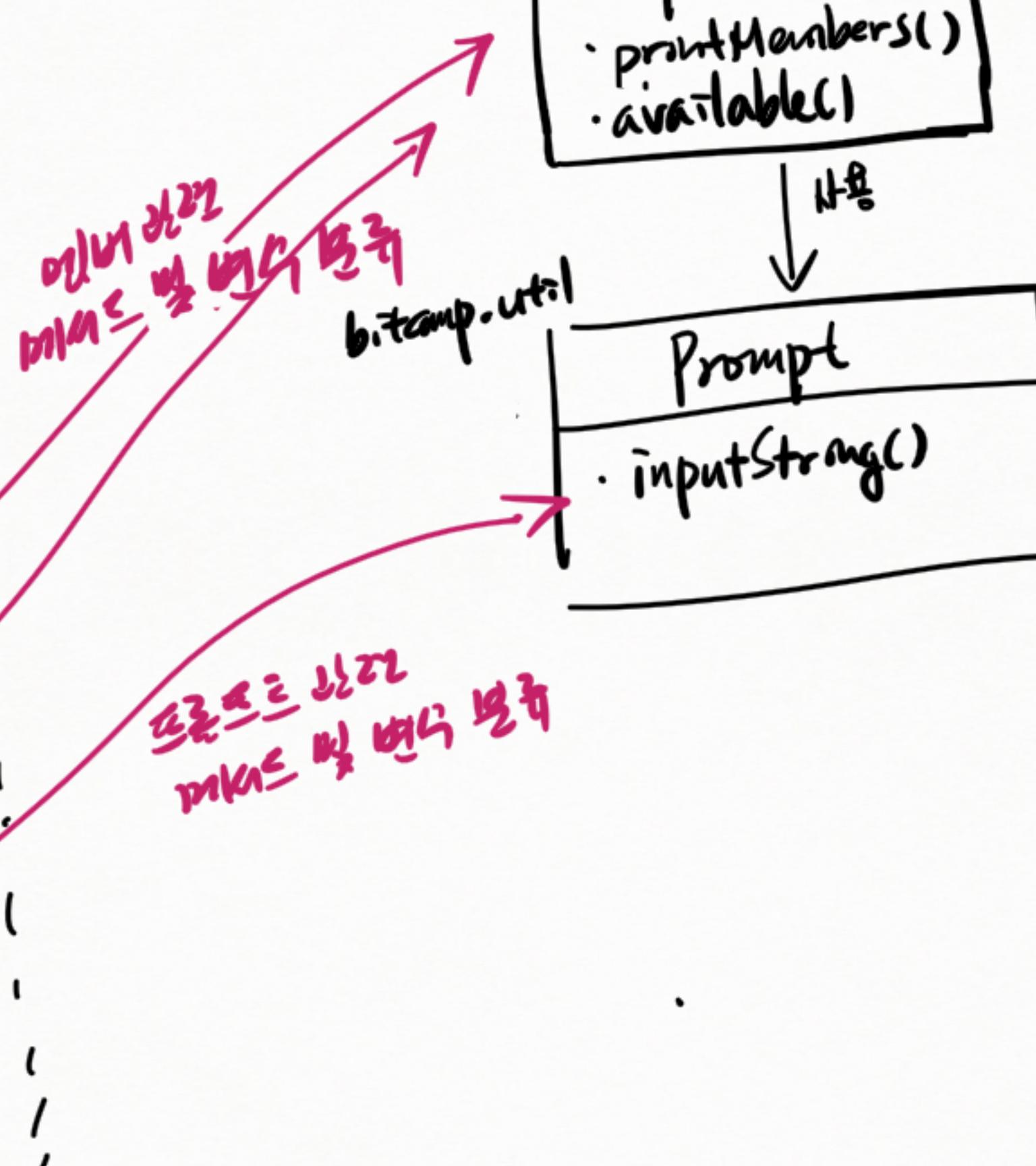
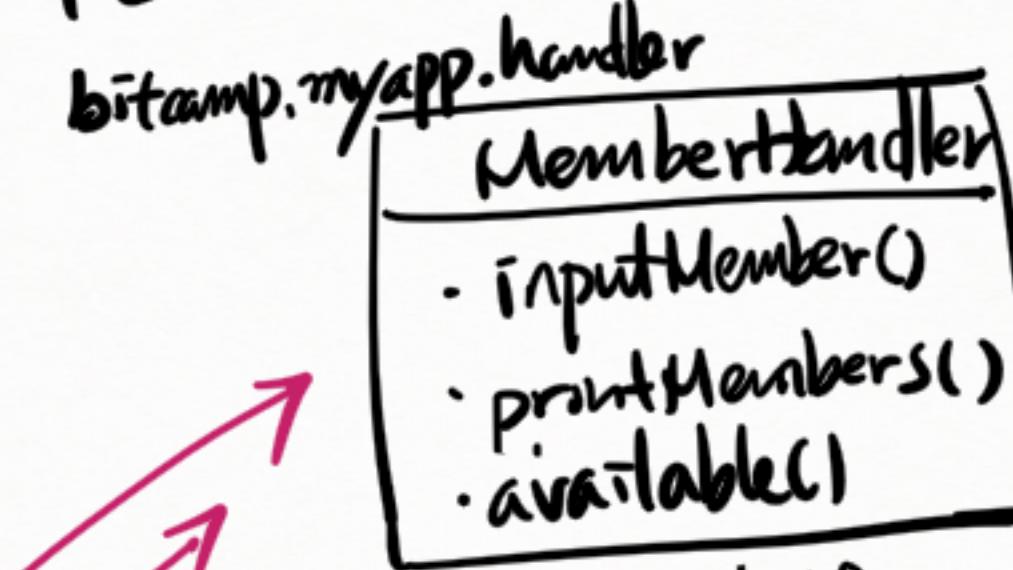


* 9. 클래스 및 패턴 학습

이전 구조
~
Architecture



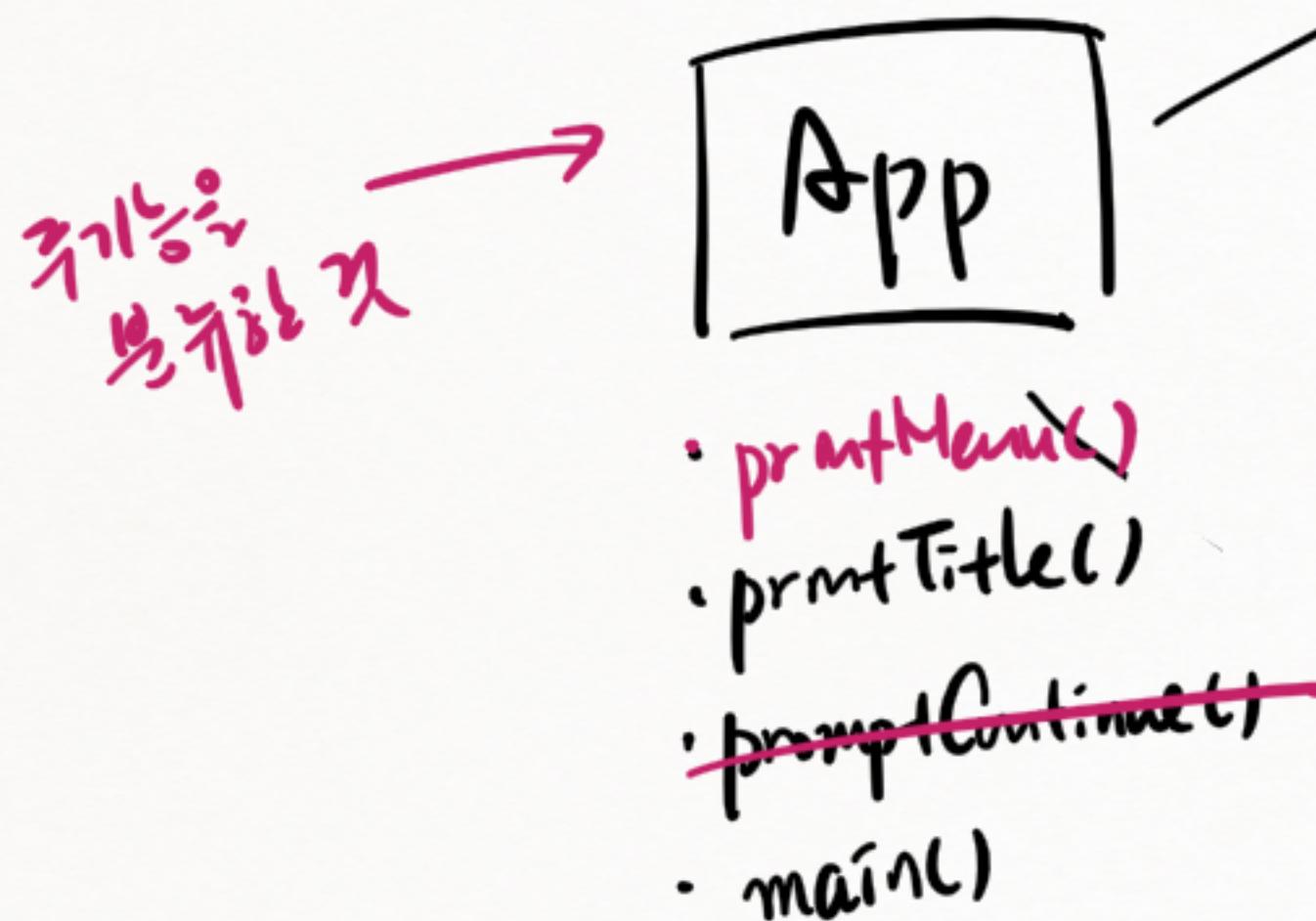
내 구조



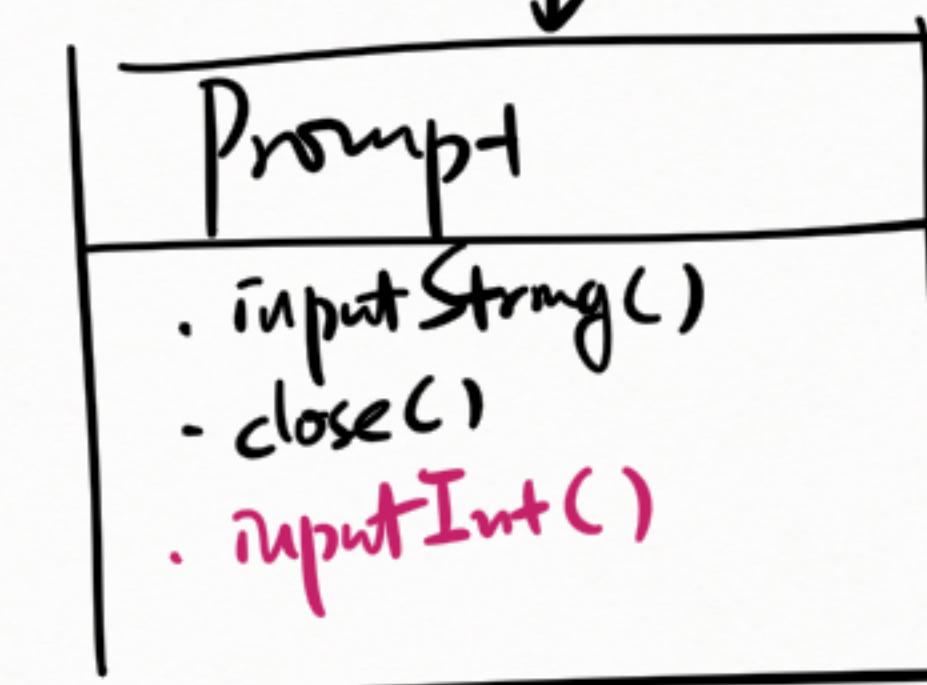
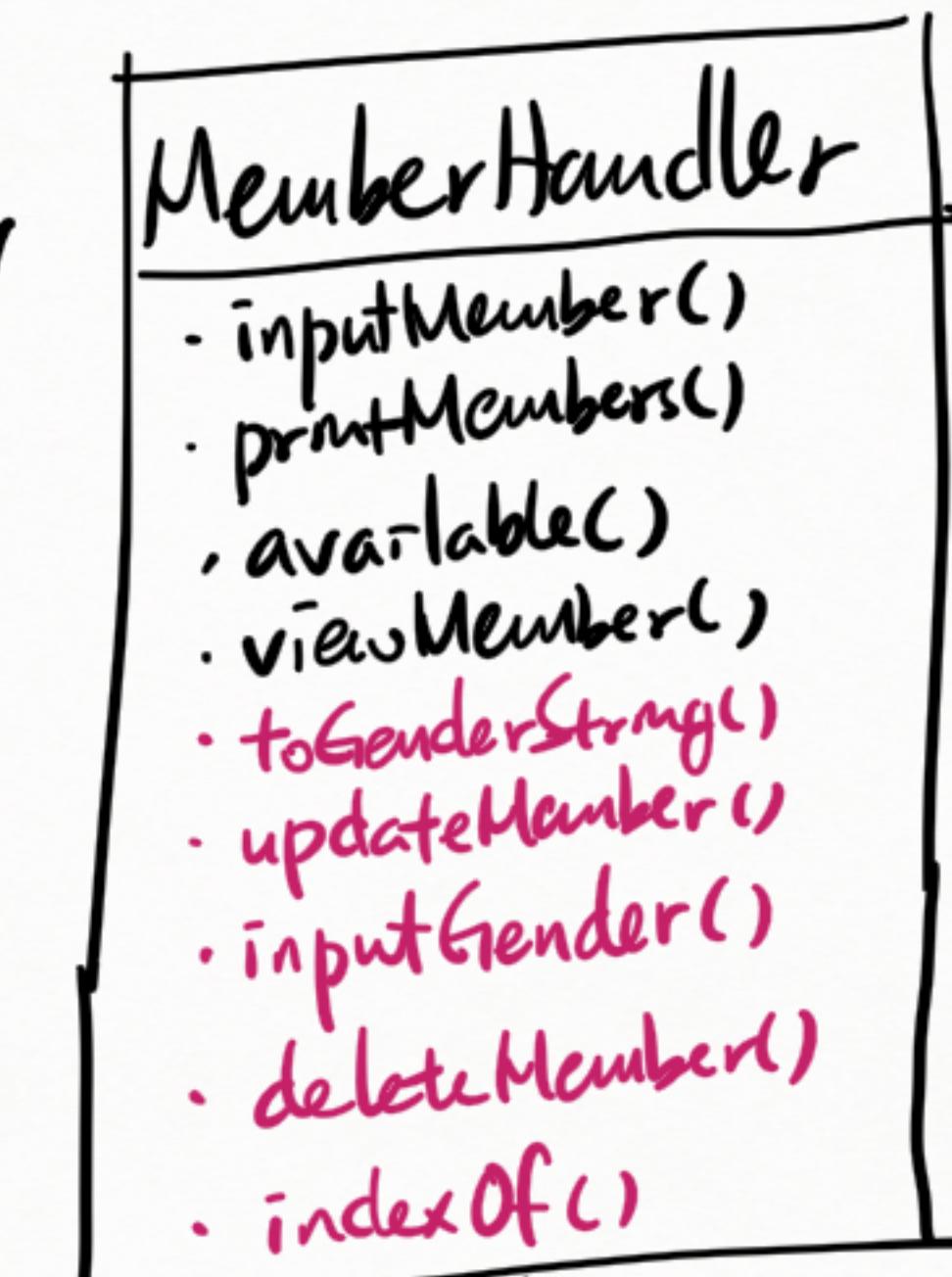
* 10. 멤버 및 CRUD 구현

* 클래스
↳ 애플리케이션 메서드를 분류할 것

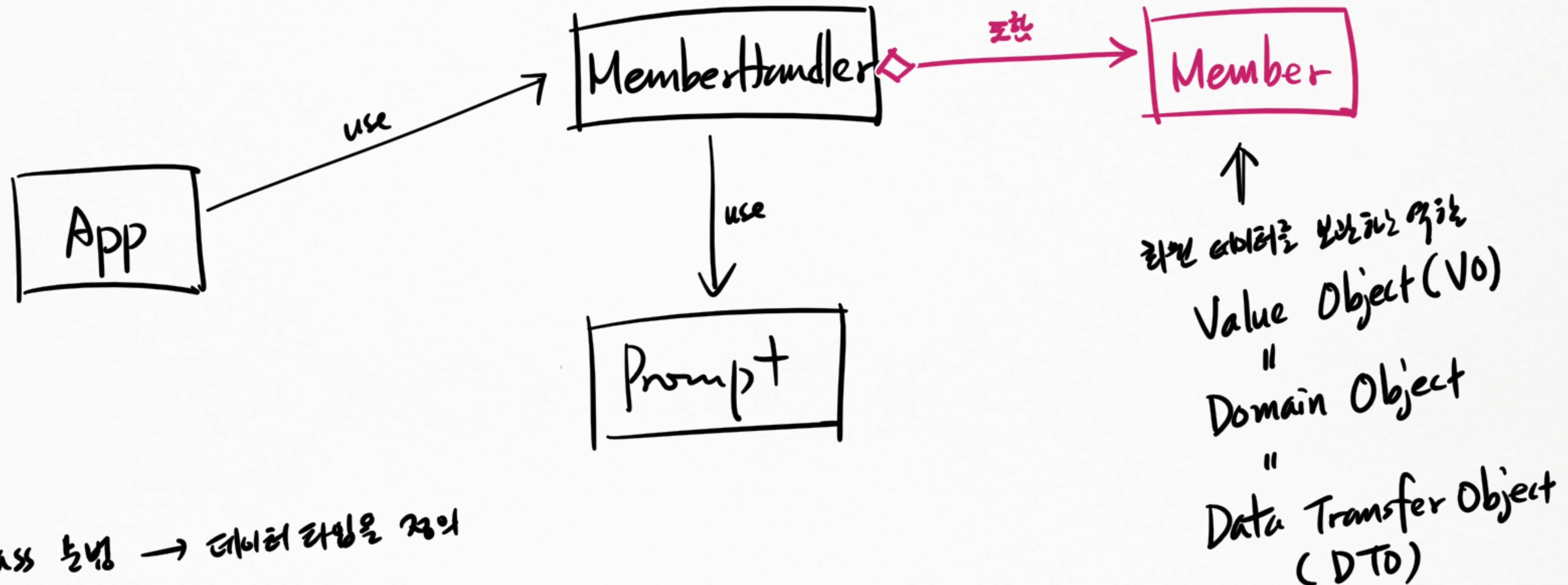
* 패키지
↳ 클래스를 분류할 것.



class 분류 → 메서드를 찾는 용도

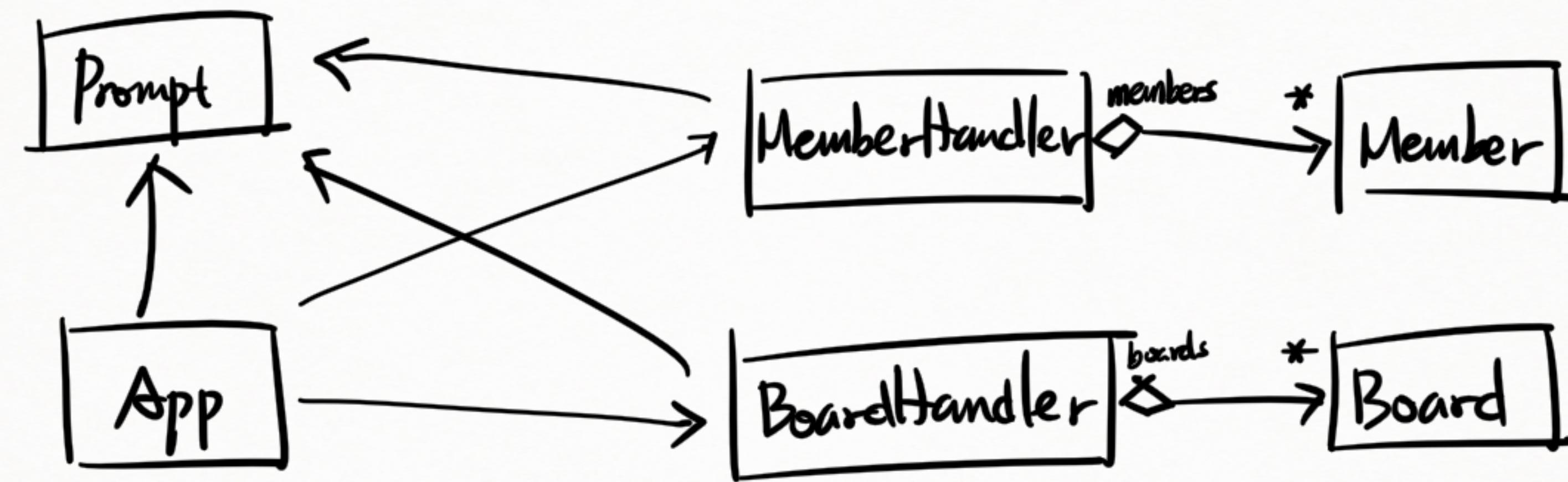


11. 사용자 정의 데이터 타입 만들기

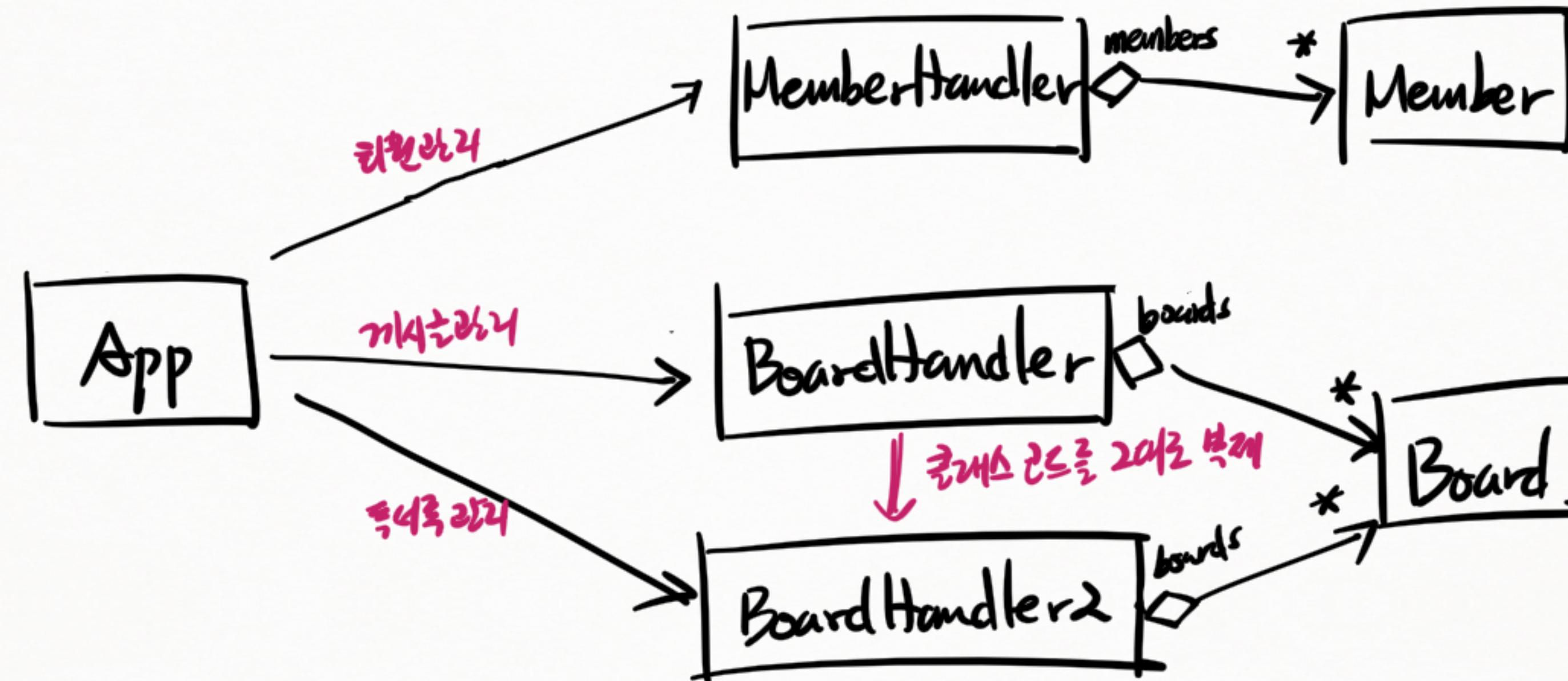


* class 늘기 → 데이터 타입을 확장

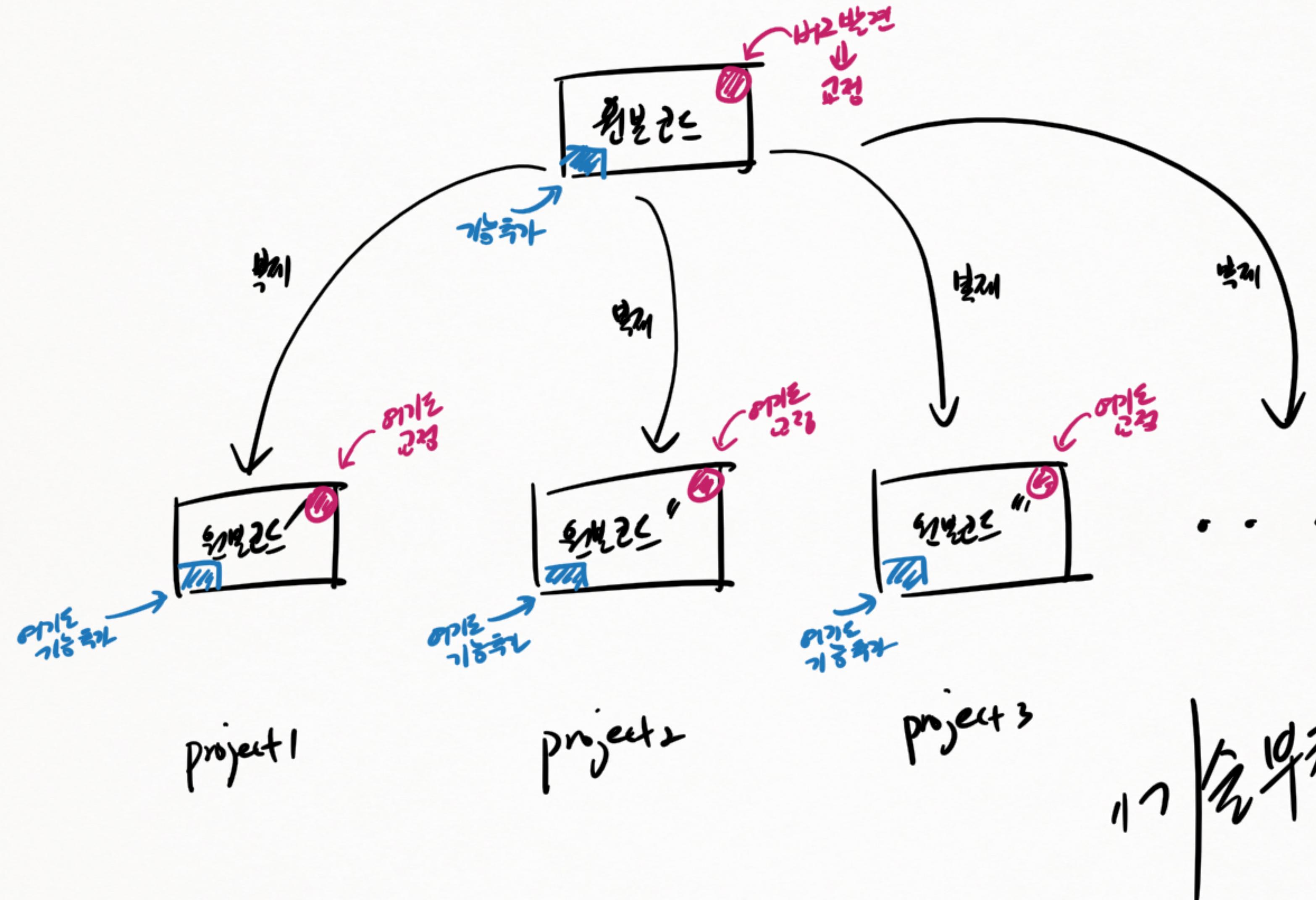
13. 깃허브 CRUD 추가



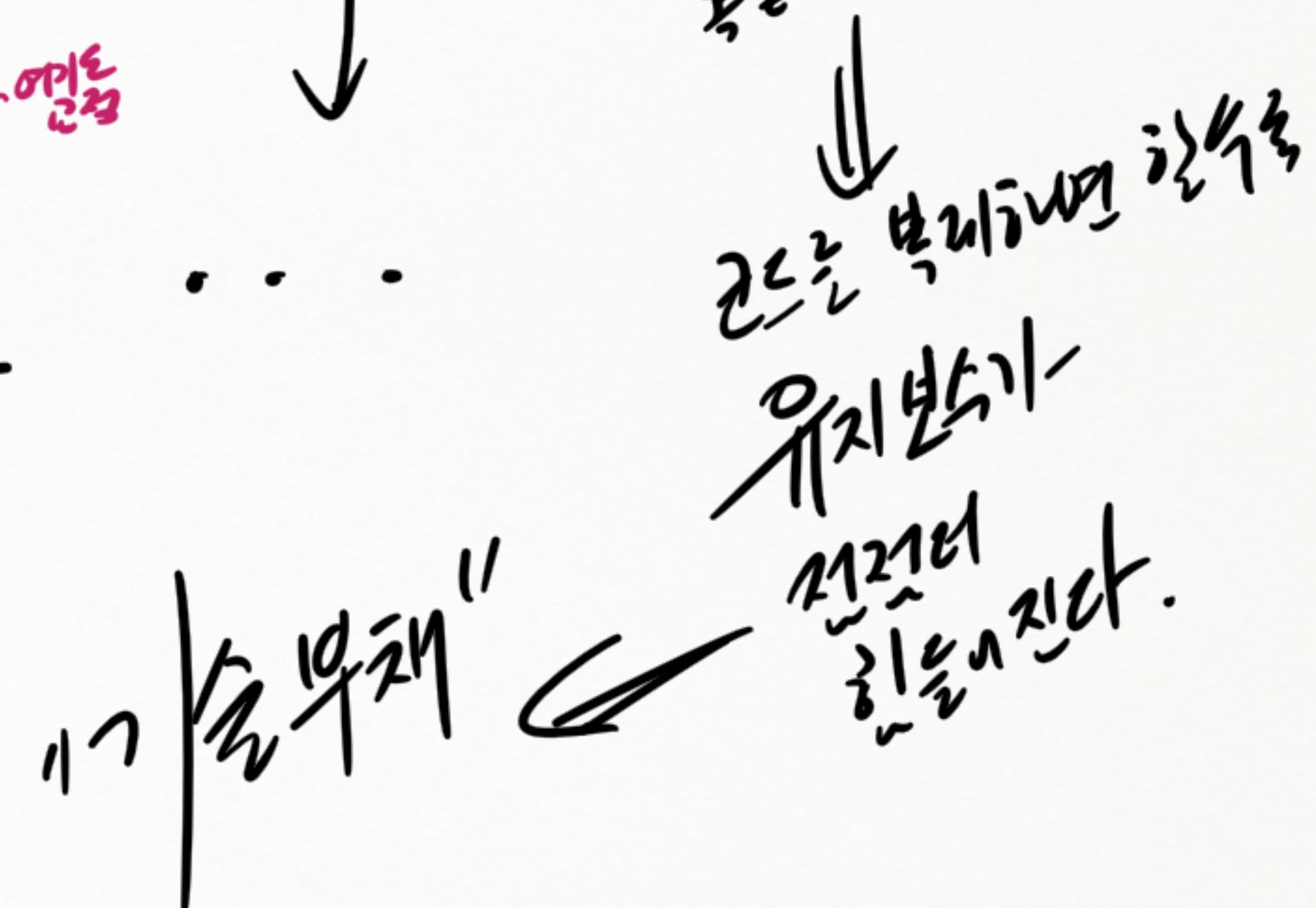
14. 토커를 CRUD 추가



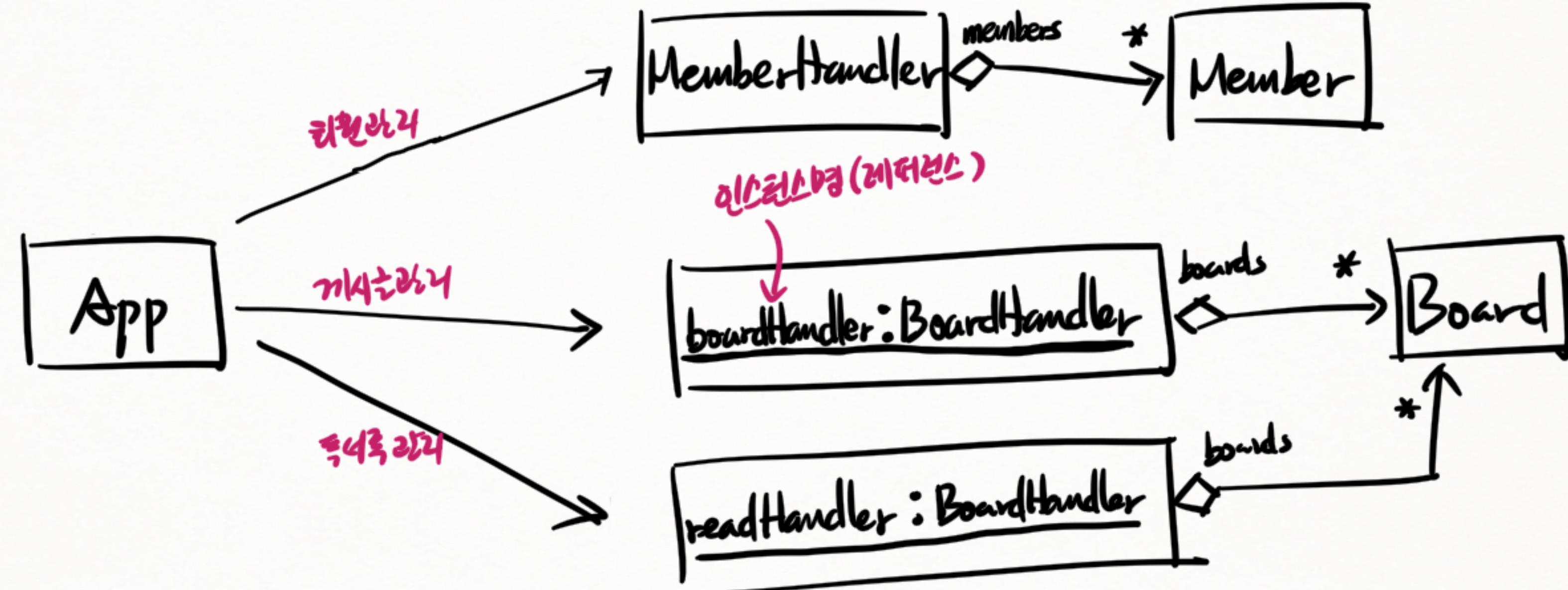
* 가로관 복제로써 새기능을 주입하는 예



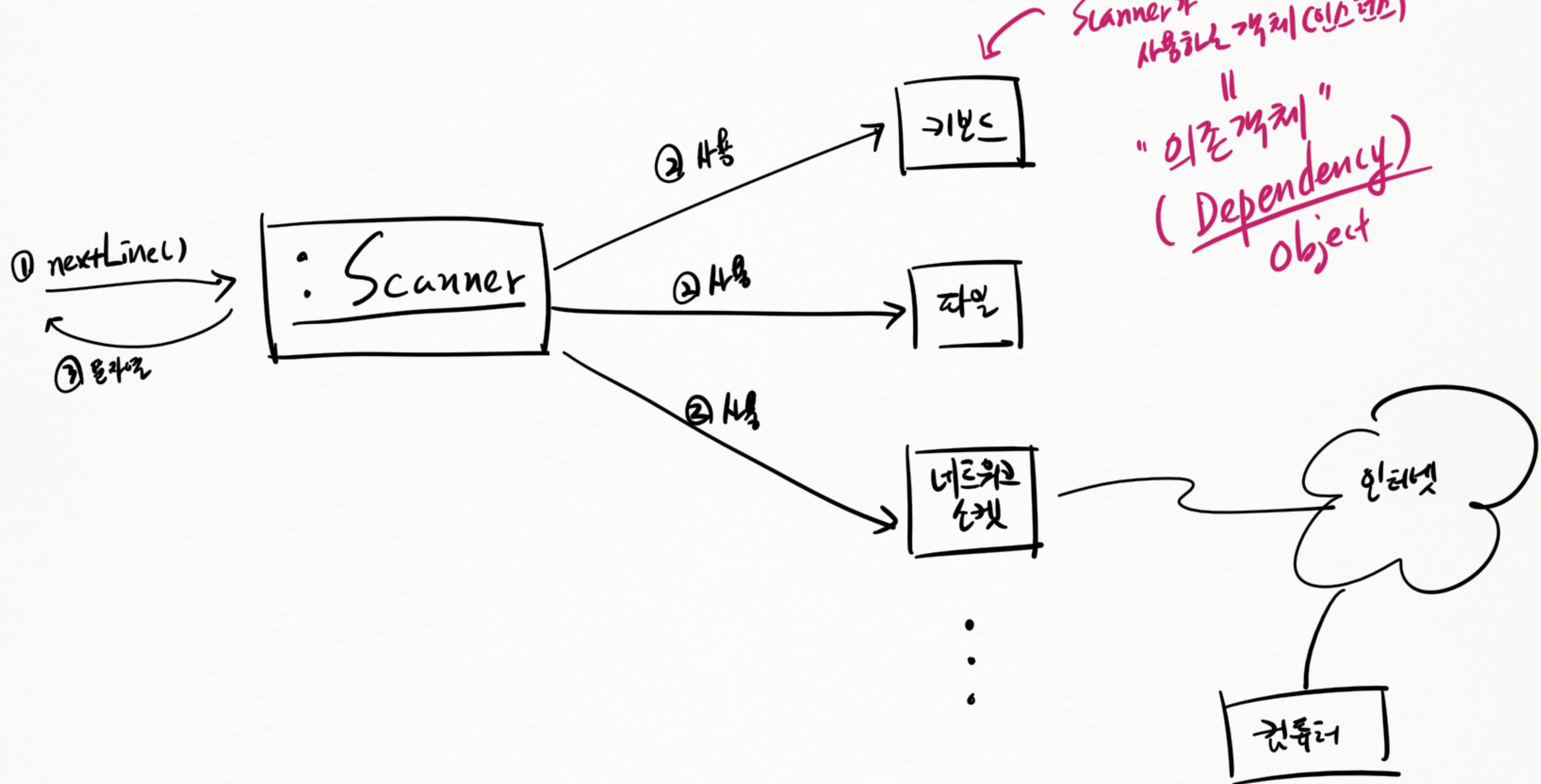
* API를 고정하여 기능을 추가하여 API를 복제한 코드의 디자인은 동일한 일을 수행하는 코드다



15. 인스턴스 있는 디자인 패턴



Scanner 와 의존 객체



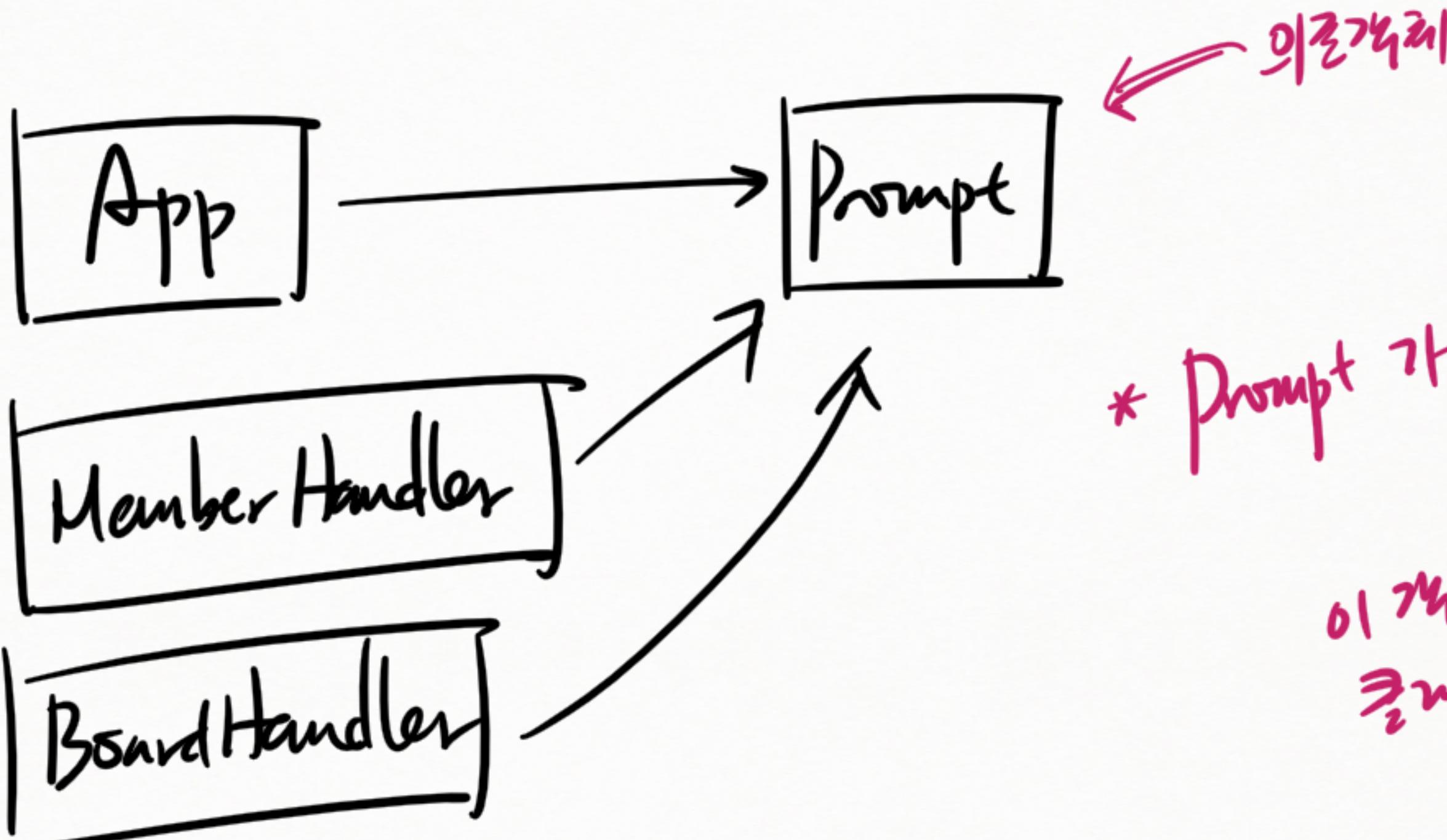
App, MemberHandler, BoardHandler et prompt

생성자 주입!



부모에
속해
하는가?

"생성자 주입"
Dependency Injection
(DI)

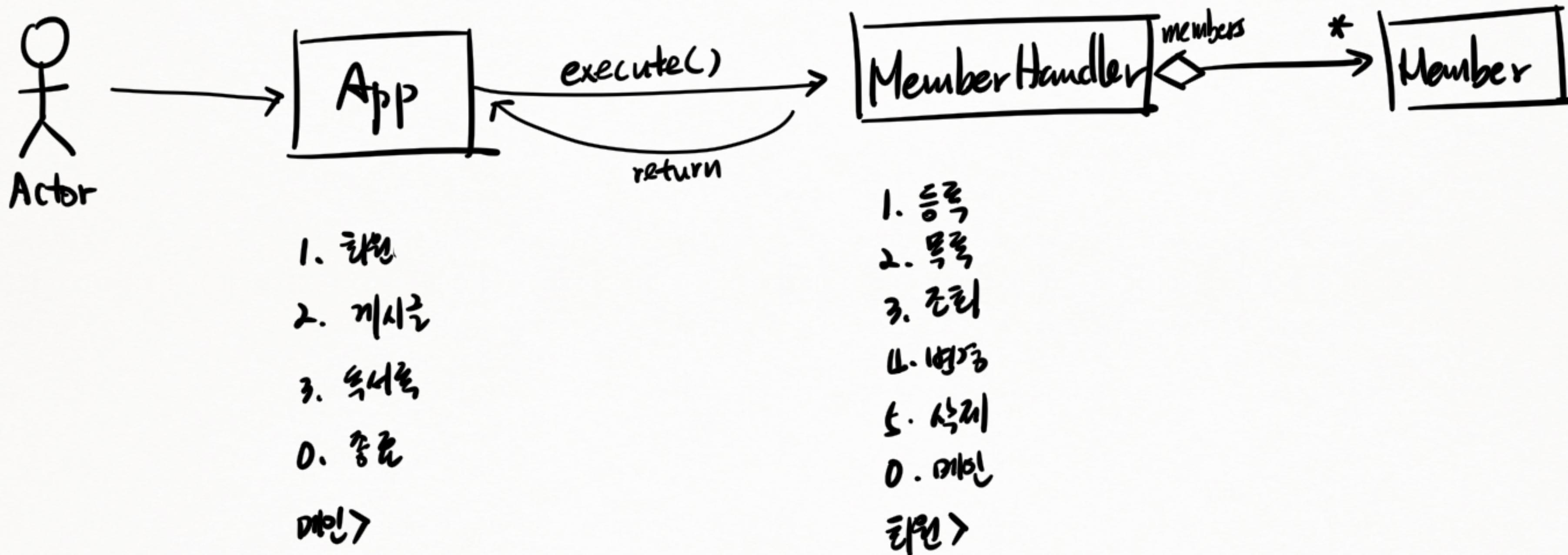


* Prompt 가 인스턴스 별로 만들기
의존주입

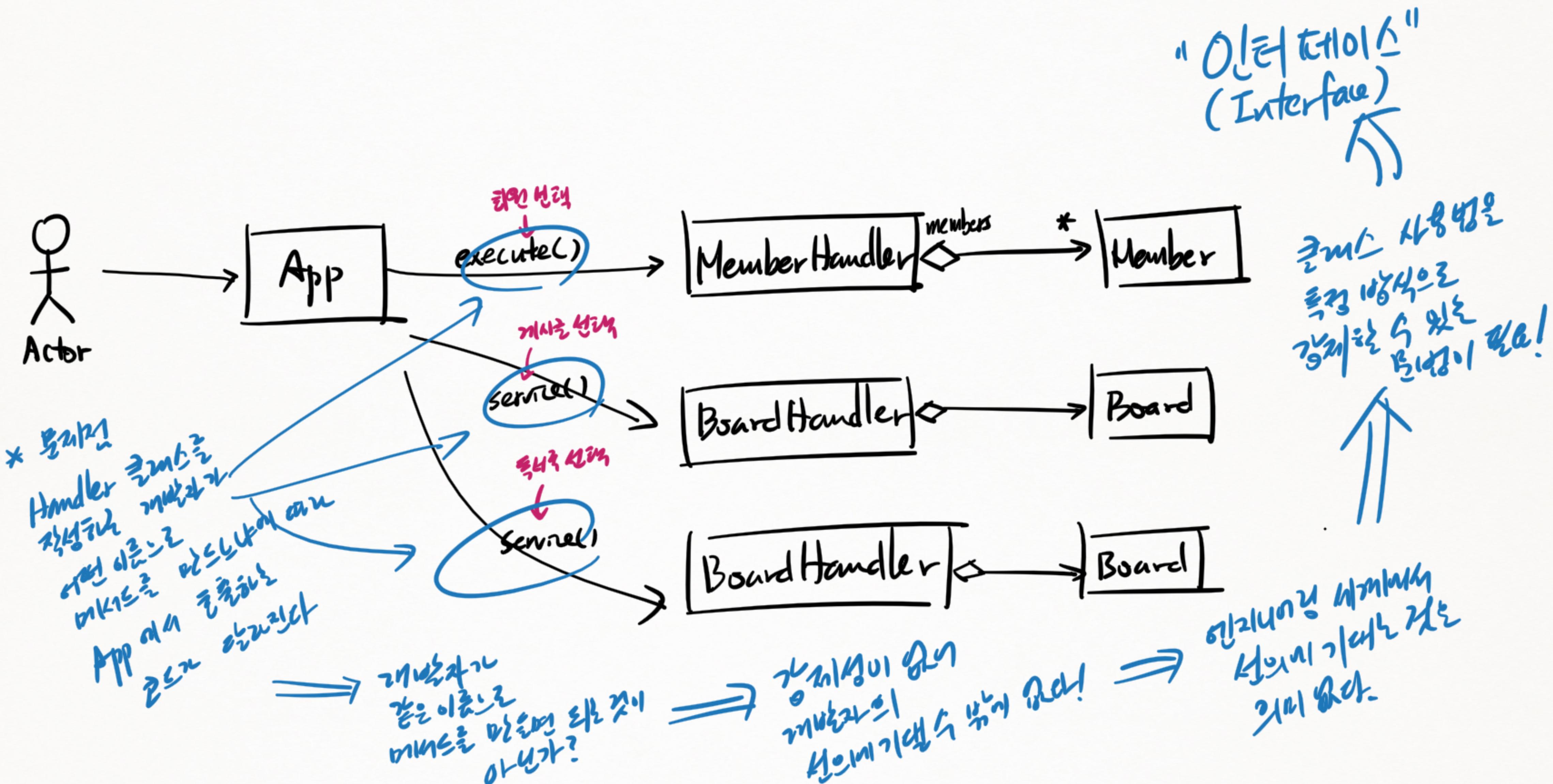
이 개체를 Algorithm
으로 변경되었을 때
↓

생성자를 쓸
의존주입(Prompt)
같이!
Injection

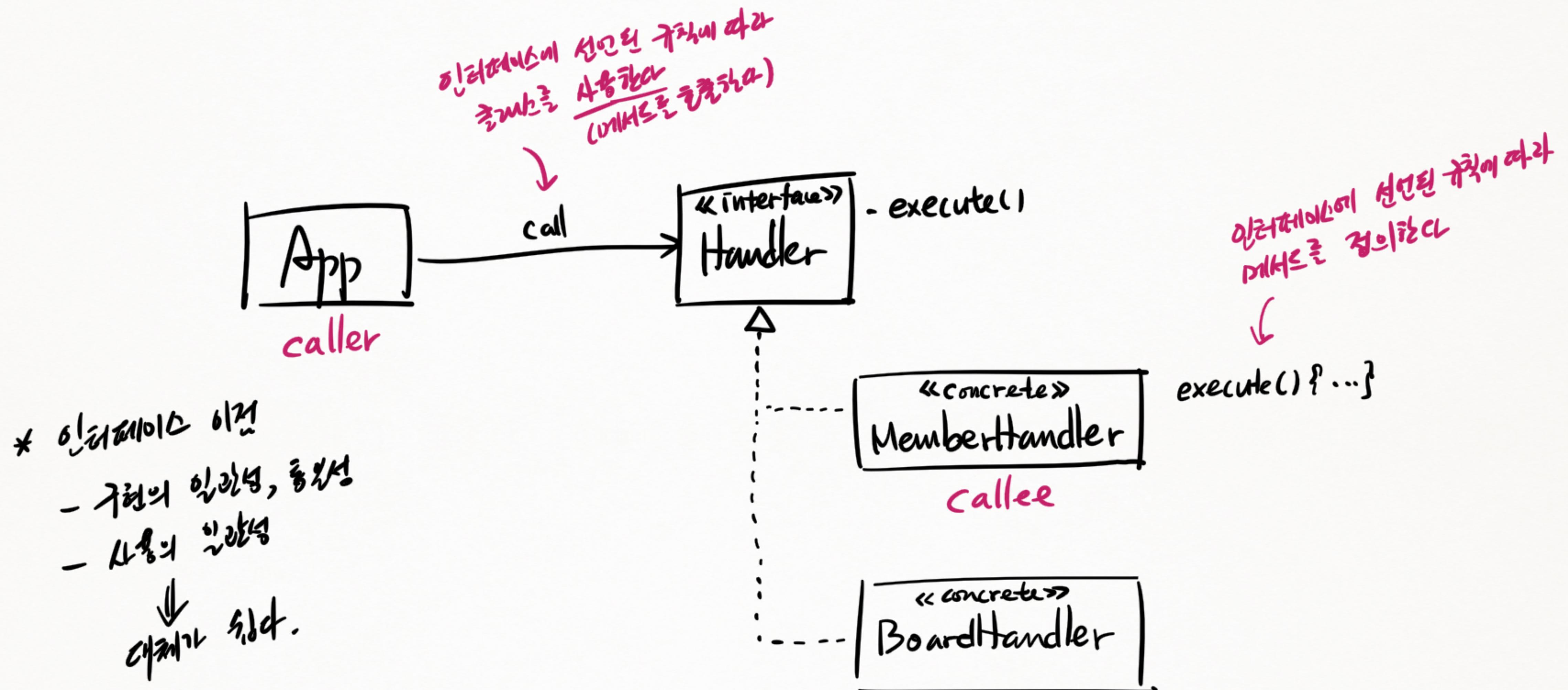
16. Handler에게 메뉴 기능을 위임



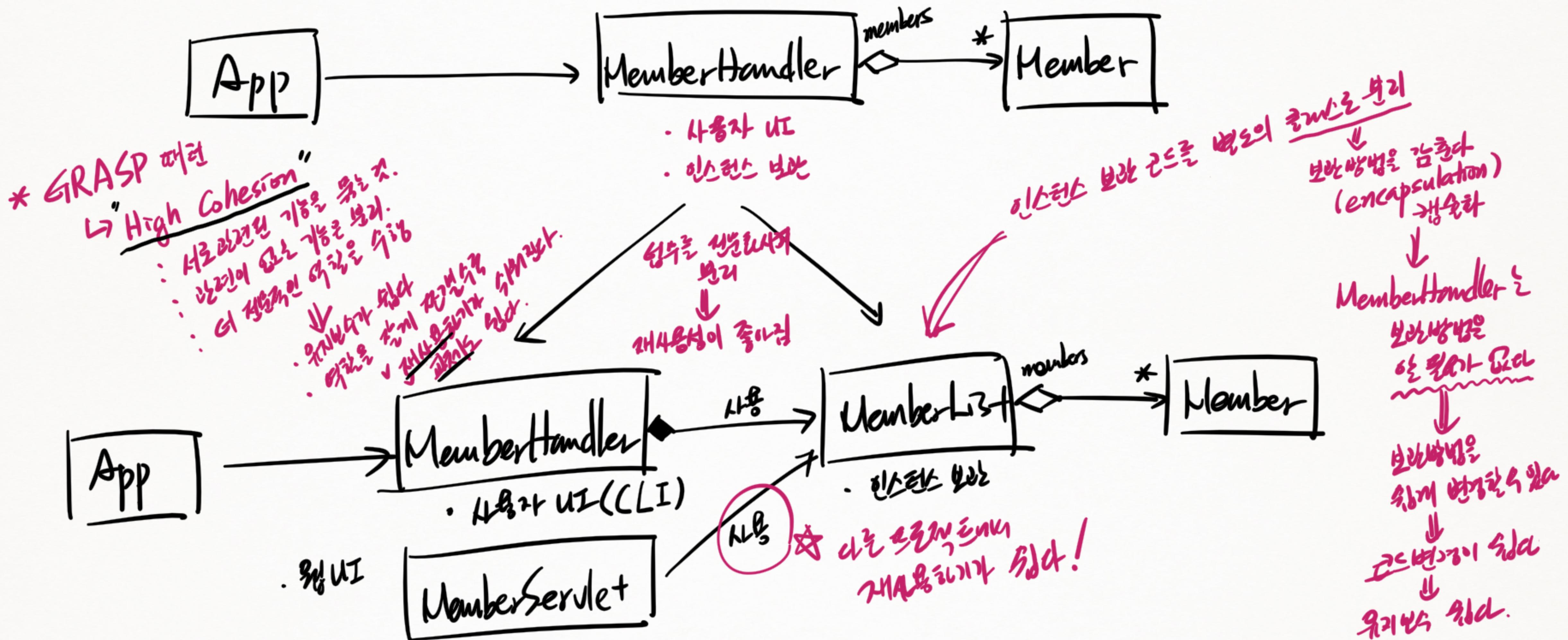
16. Handler에게 메뉴 기능을 위임



III. Handler의 사용 구조를 인터페이스로 정의하기



18. 인스턴스 목록을 다수로 코드를 빼는 축제로 보기



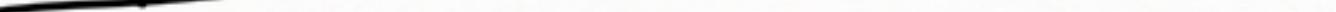
* 배운 늘하기

$$\frac{5}{2} = \cancel{2} \cancel{\times}$$

Garbage \Rightarrow 0 1 2 3 4
 G | a | r | b | a | r | a | g
 | | | | | | | |

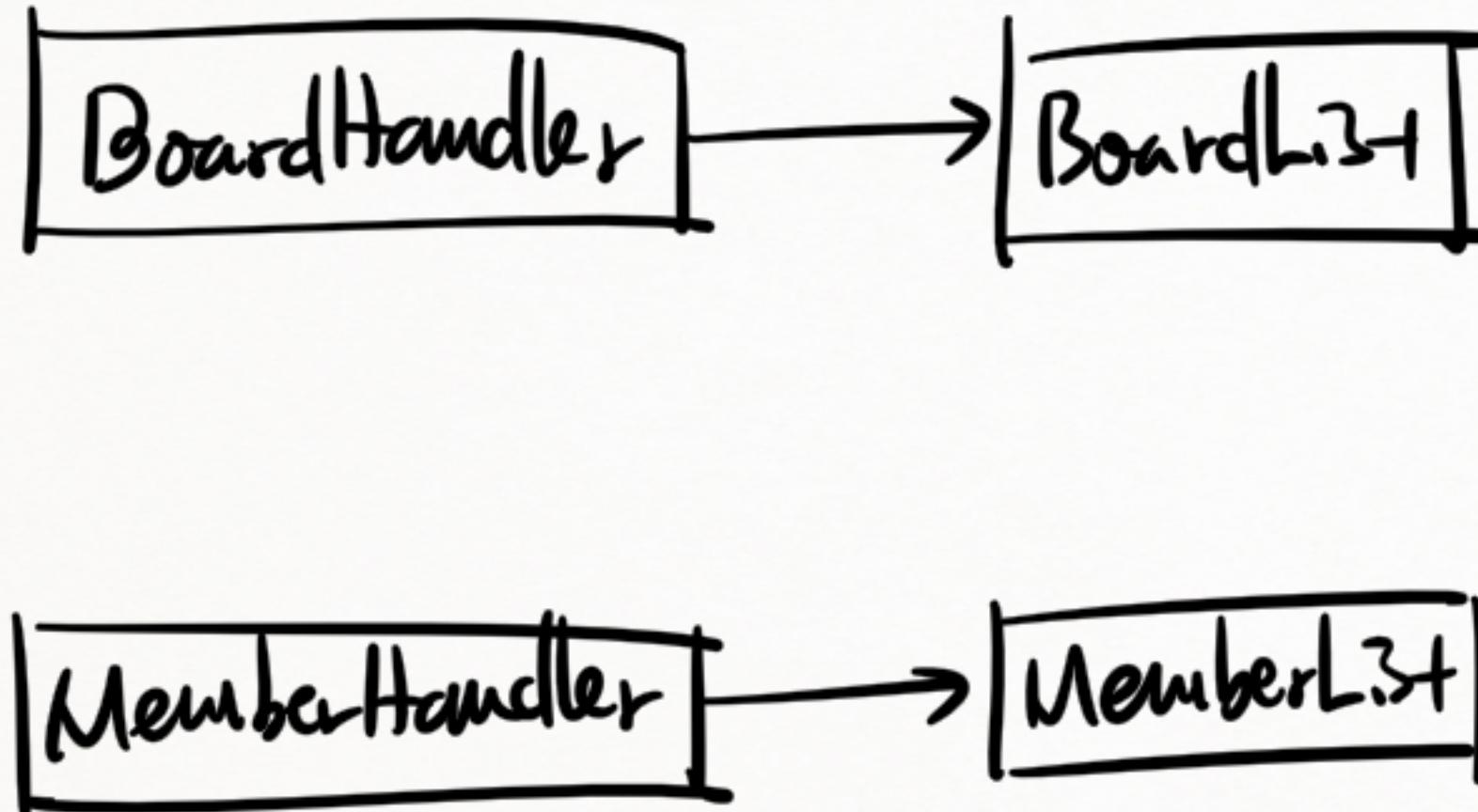
$$\underline{t+2} = 4$$

$$7 + \frac{1}{3} = 10$$

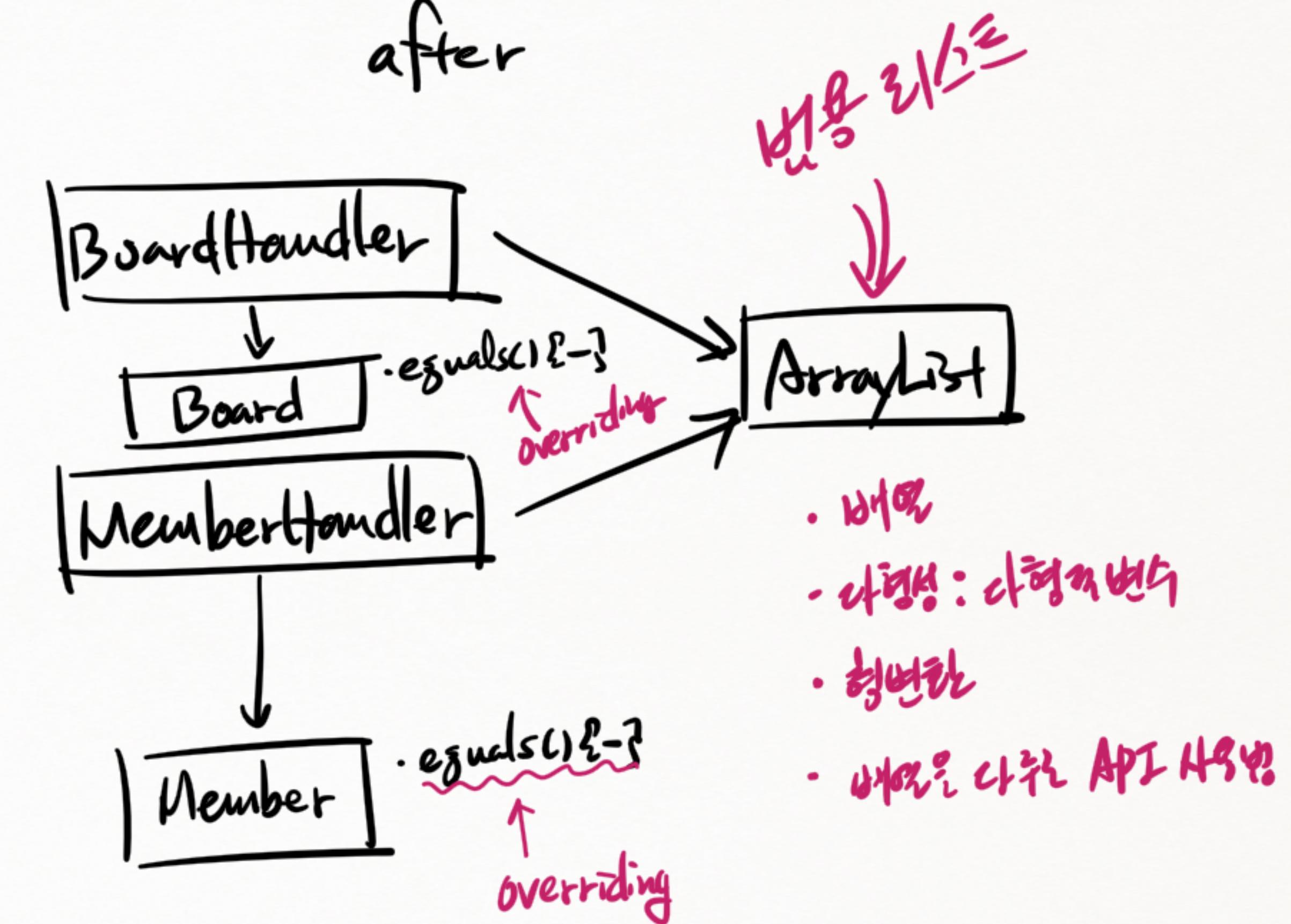
boards → 

19. 범용리스트 만들기

before



after

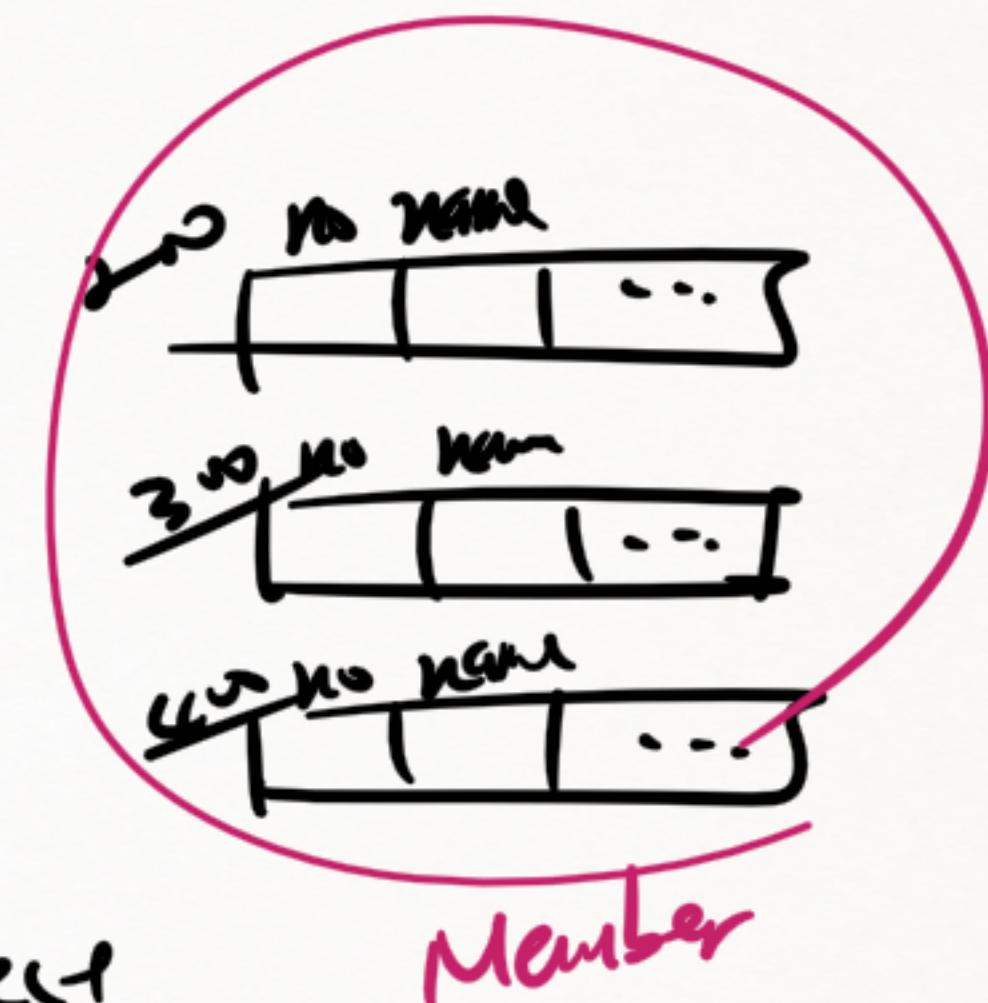
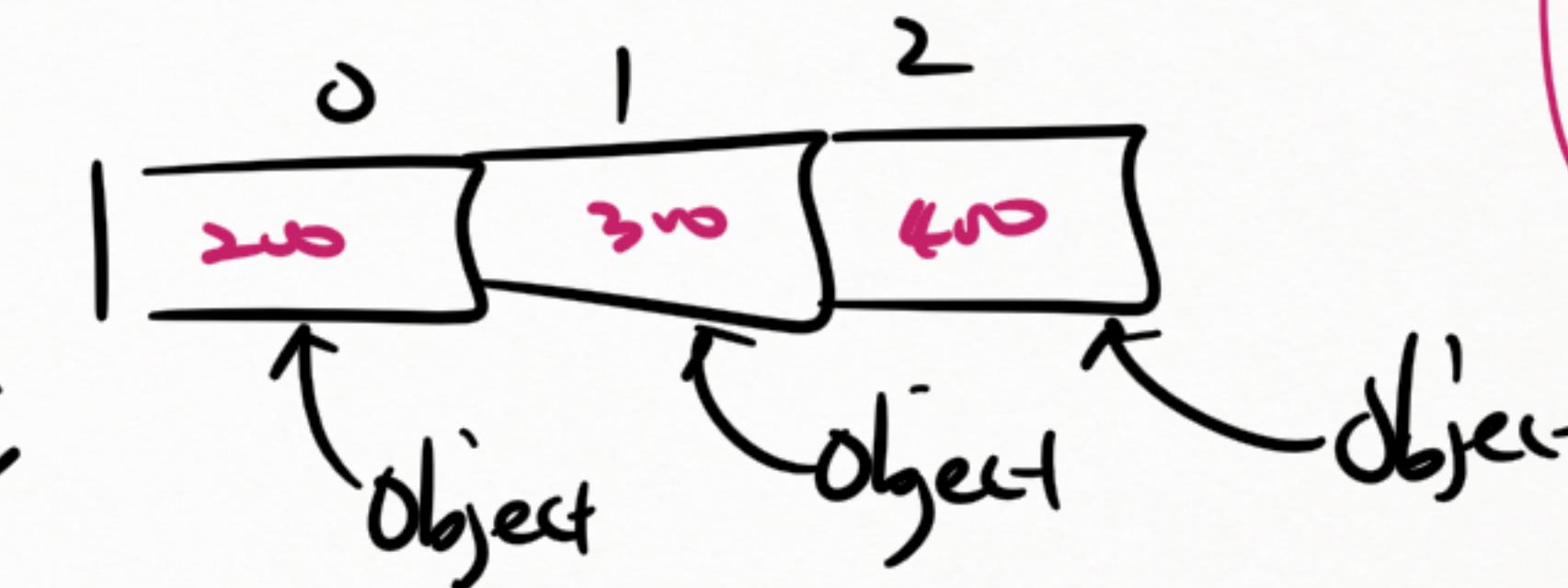


`Object[] arr = new Object[3];`

```
arr[0] = new Member();
```

obj[1] = new Member()

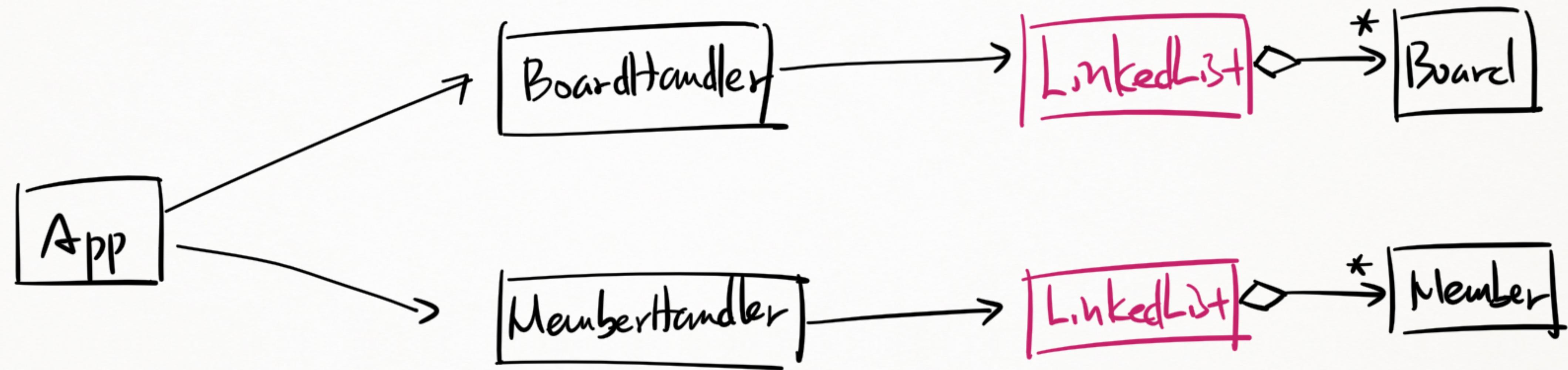
obj1[2] = new Member();



~~Member[] arr2 = (Member[]) arr1;~~

[1] arr1 ;
↑
arr1 이 가리키는 것 ~ Member 라이브러리의 멤버이며.
Object 라이브러리 멤버로는 Object 라이브러리에 정의된 멤버이다.

* 20. LinkedList 자료구조 구현하기



ArrayList vs LinkedList

	ArrayList	LinkedList
추가/삭제	<u>일정 범위내</u> 빠르 (IMPL)	O 부터 시작
크기 증가	Yes → 가로세로로 가로나가면서	Yes → 가로이자 높이.
검색	인덱스로 조회 (LinkedList는 불가능)	링크를 따라가야 한다 (ArrayList는 가능)
교환, 삽입, 삭제	삽입 → 배열값 옮기기 삭제 → 배열값 옮기기 <i>(LinkedList는 불가능)</i>	해당 값을 찾고자 노드 추가/삭제 <i>(List는 가능)</i>

LinkedList - add()

tail 3100

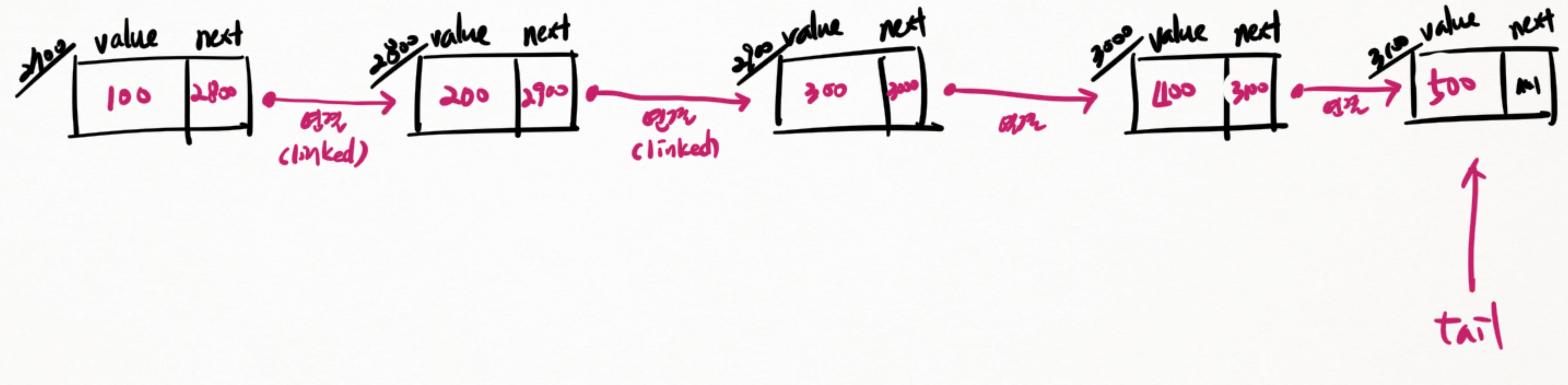
add(100);

add(200);

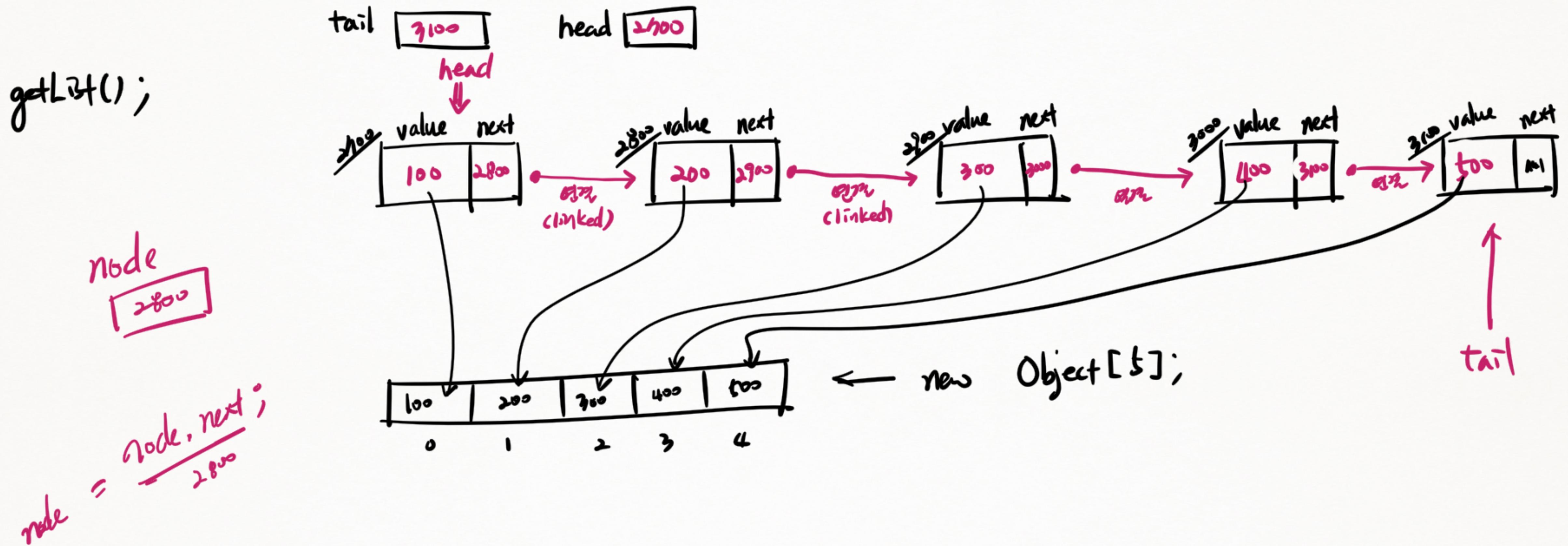
add(300);

add(400);

add(500);



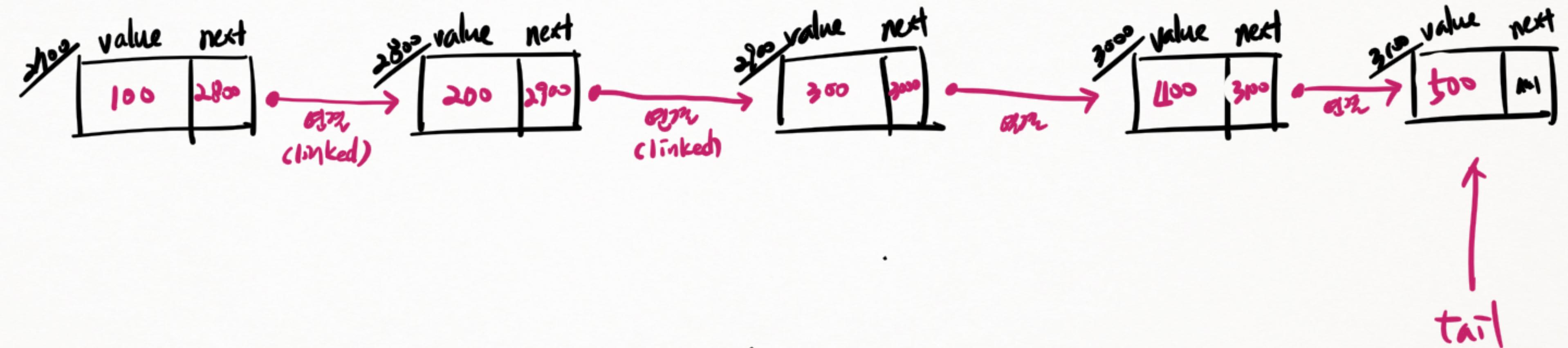
LinkedList - getList()



LinkedList - retrieve()

tail 3100

retrieve(100);



cursor

null

~~500.equals(100)~~

cursor

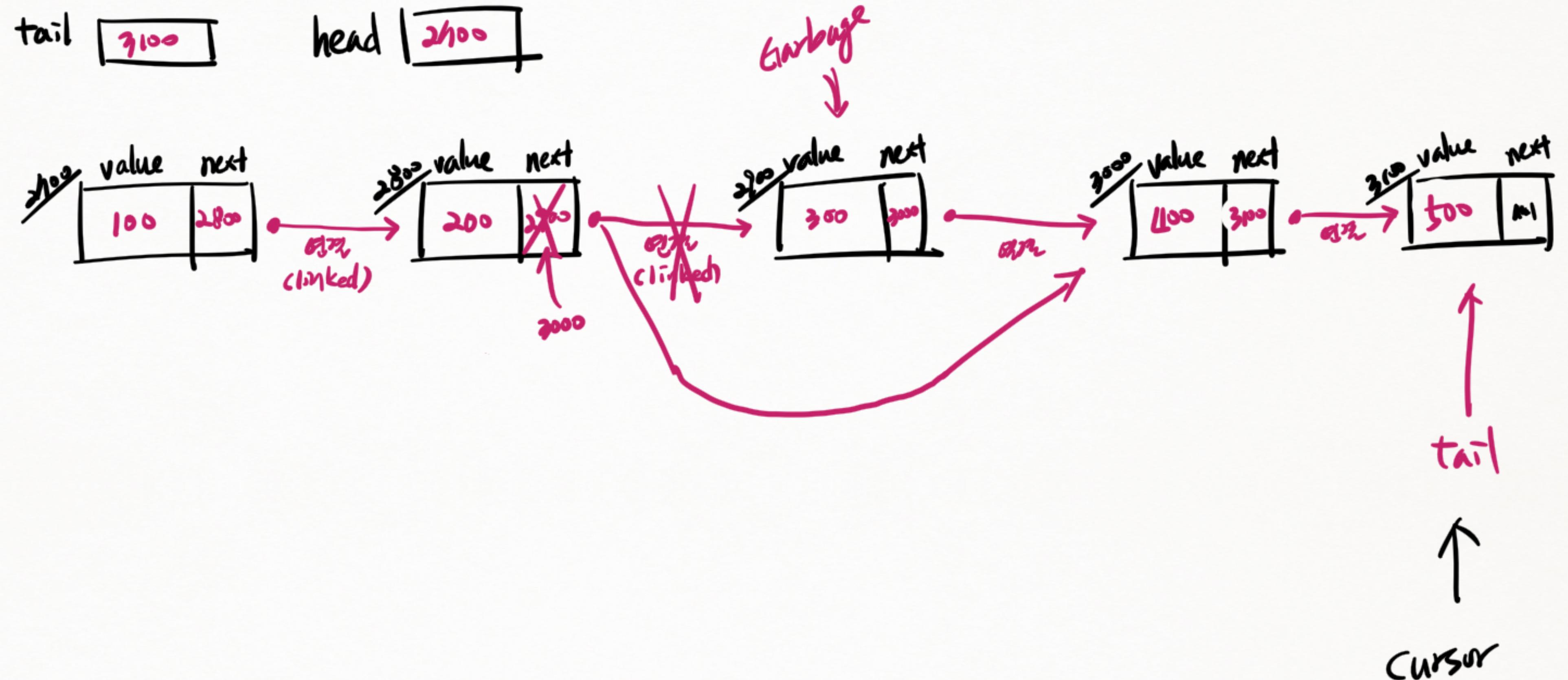


LinkedList - remove() : 중간 항목 삭제

`remove(300)`

tail 3100

head 2100



LinkedList - remove(): 정간, 흉목, 뒤지 \Rightarrow 가비지가 인스턴스를 가리킬 때 문제점.

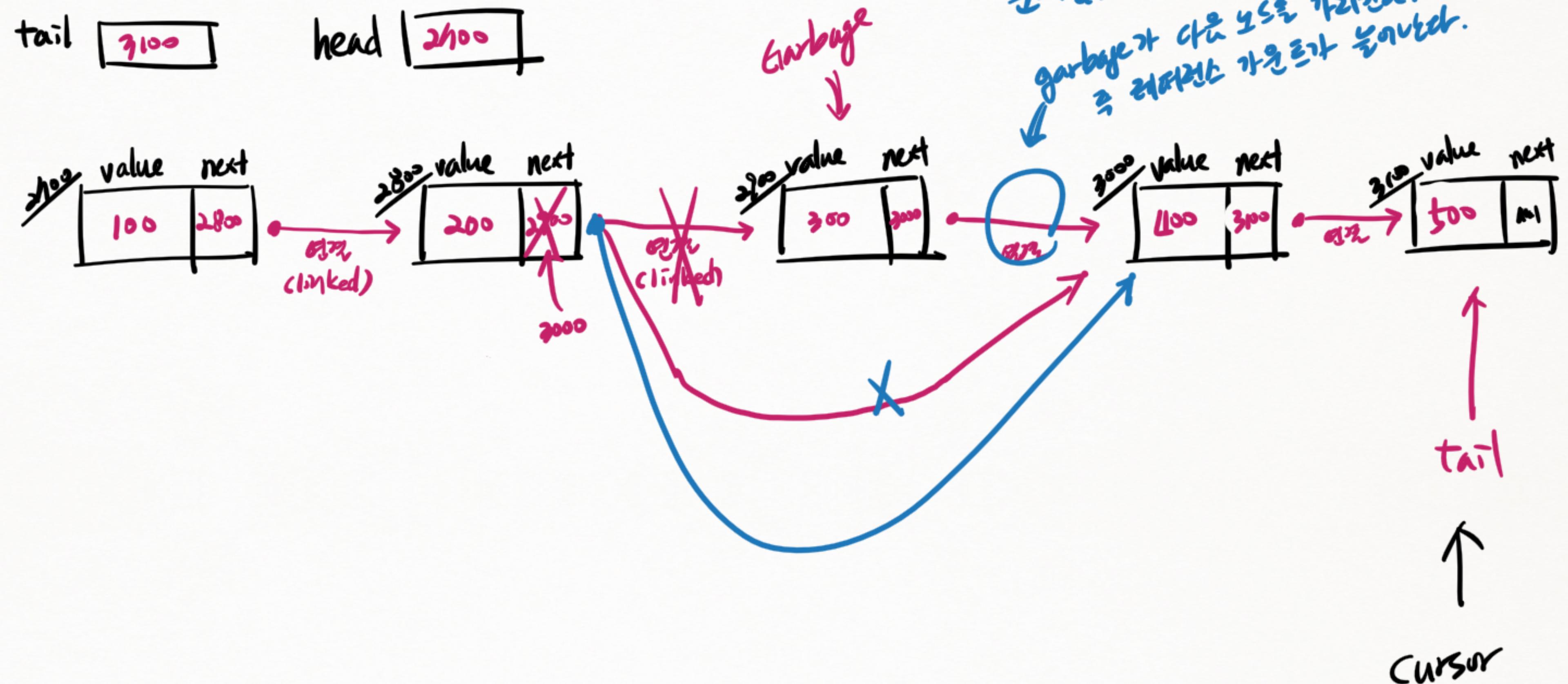
remove(300)

tail
3100

head
2100

prev
2800

cursor
2900



LinkedList - remove(): 정간, 흐름, 노드 ⇒ 가비지가 인스턴스를 가리킬 때 문제점.

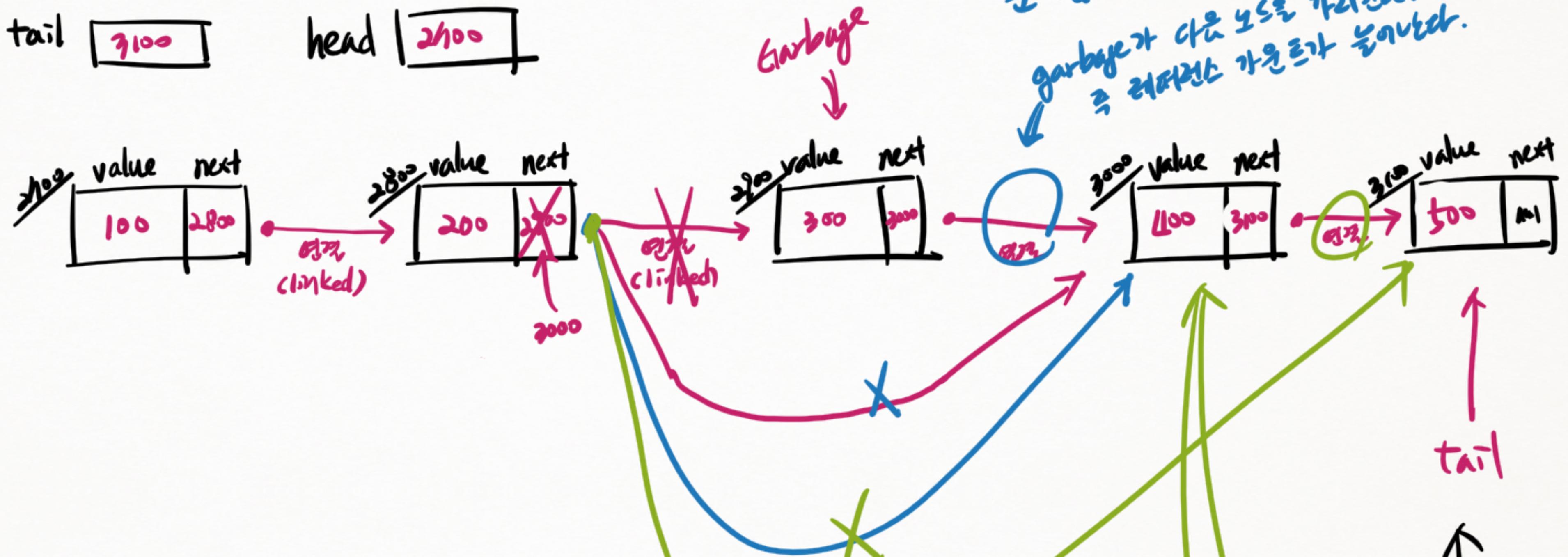
remove(300)

tail
3100

head
2100

prev
2800

cursor
2900

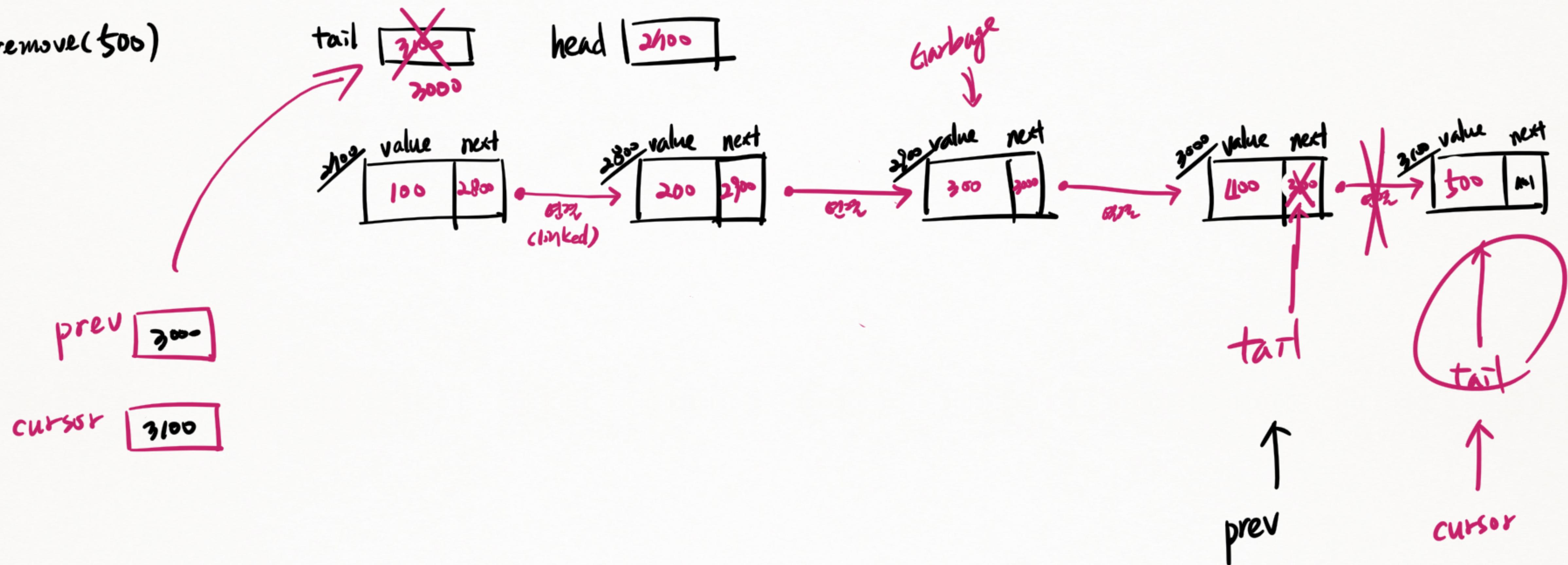


이 노드는
가비지가 되었나?
아니? 이전에 속해있던 노드에서
이 노드를 가리키고 있다.

CURSOR

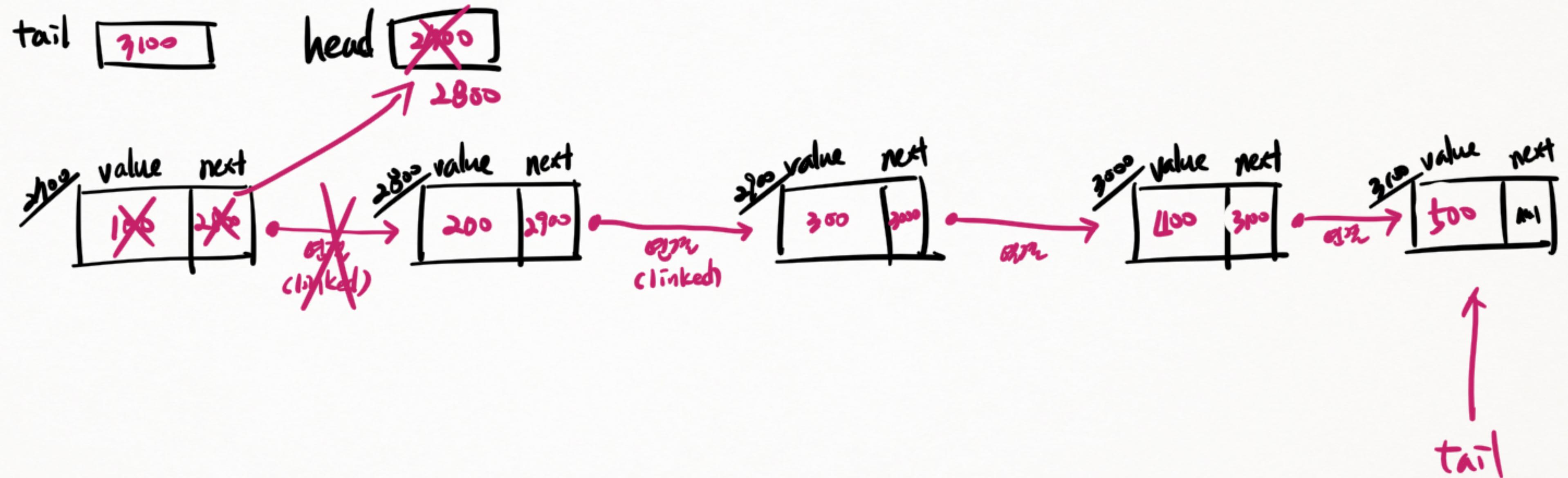
LinkedList - remove() : 끝 항목 삭제

remove(500)



LinkedList - remove() : 시작 노드

remove(100)

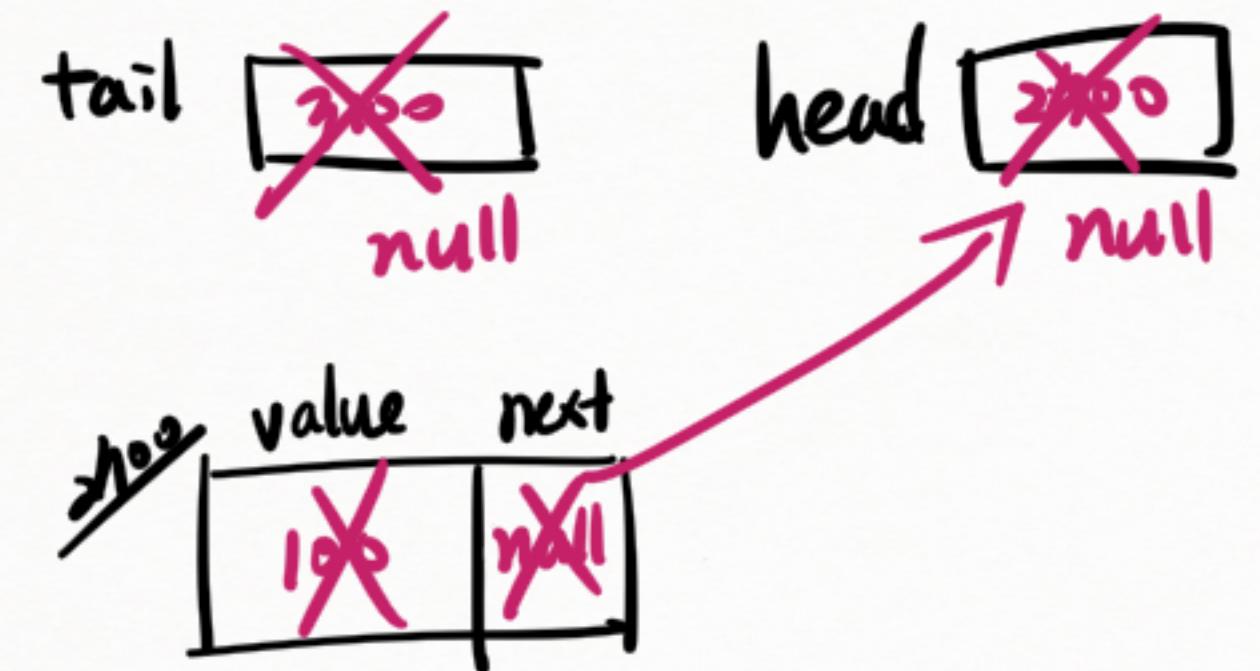


prev
[null]

CURSOR
[200]

LinkedList - remove() : 시작노드 + 끝노드

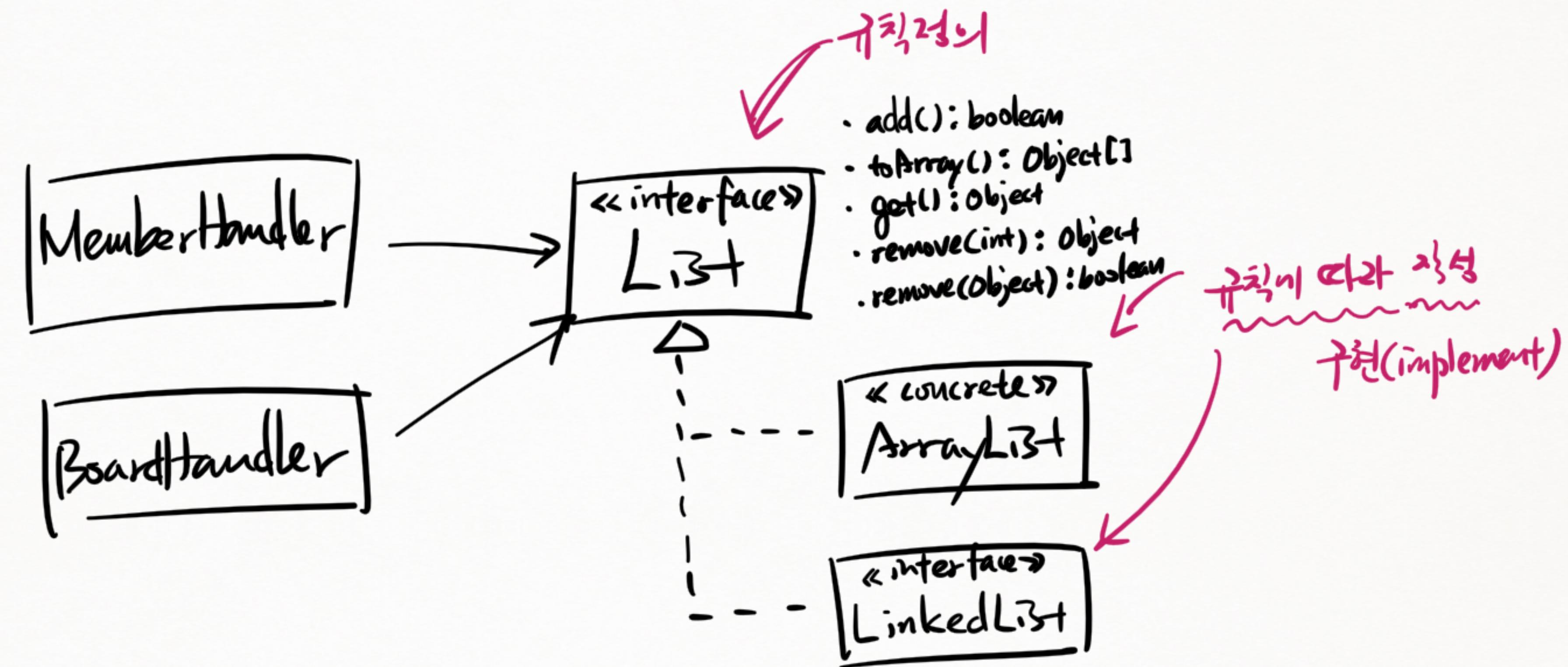
remove(100)



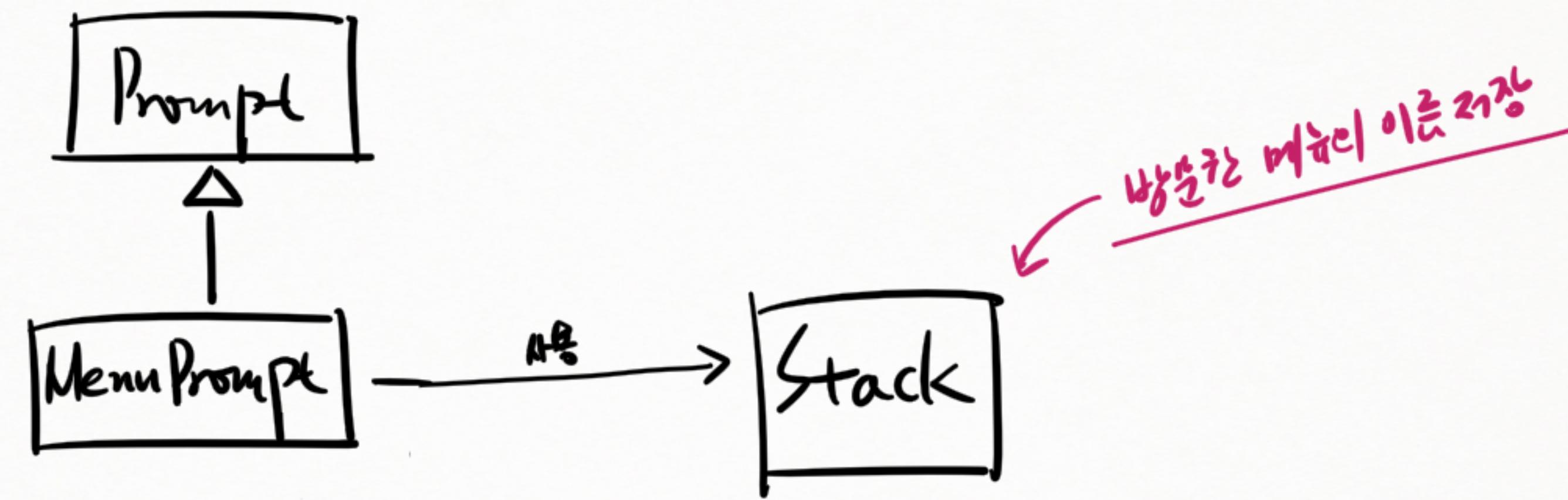
prev
[null]

CURSOR
[200]

21. 인터페이스로 목적관리 가능한 사용법을 기반으로 정의.



22. Stack, Queue 자료구조 구현



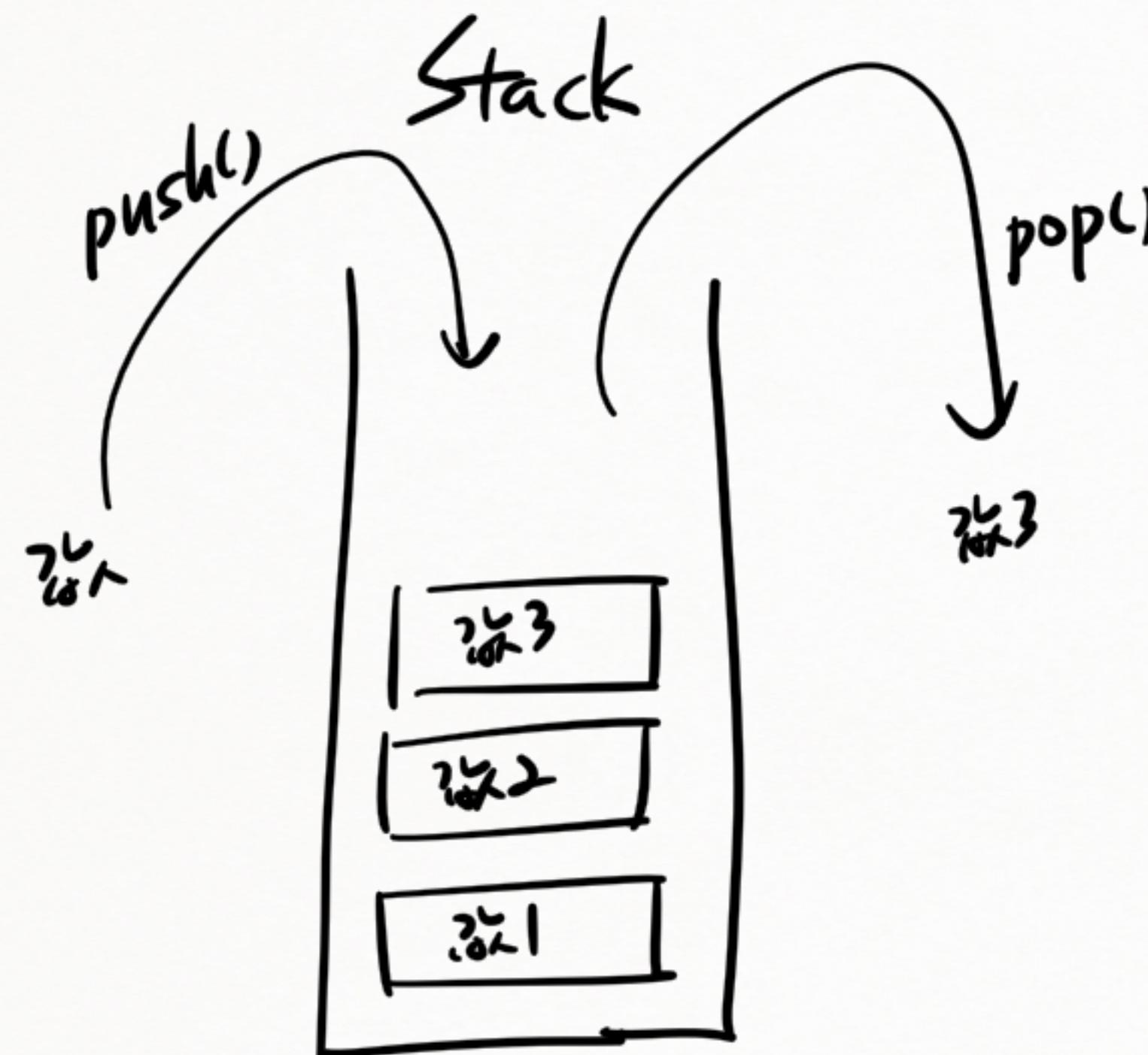
1. 카운트
2. —
3. —
메인 > 1 ↳

1. 등록
2. 목록
⋮
카운트 >

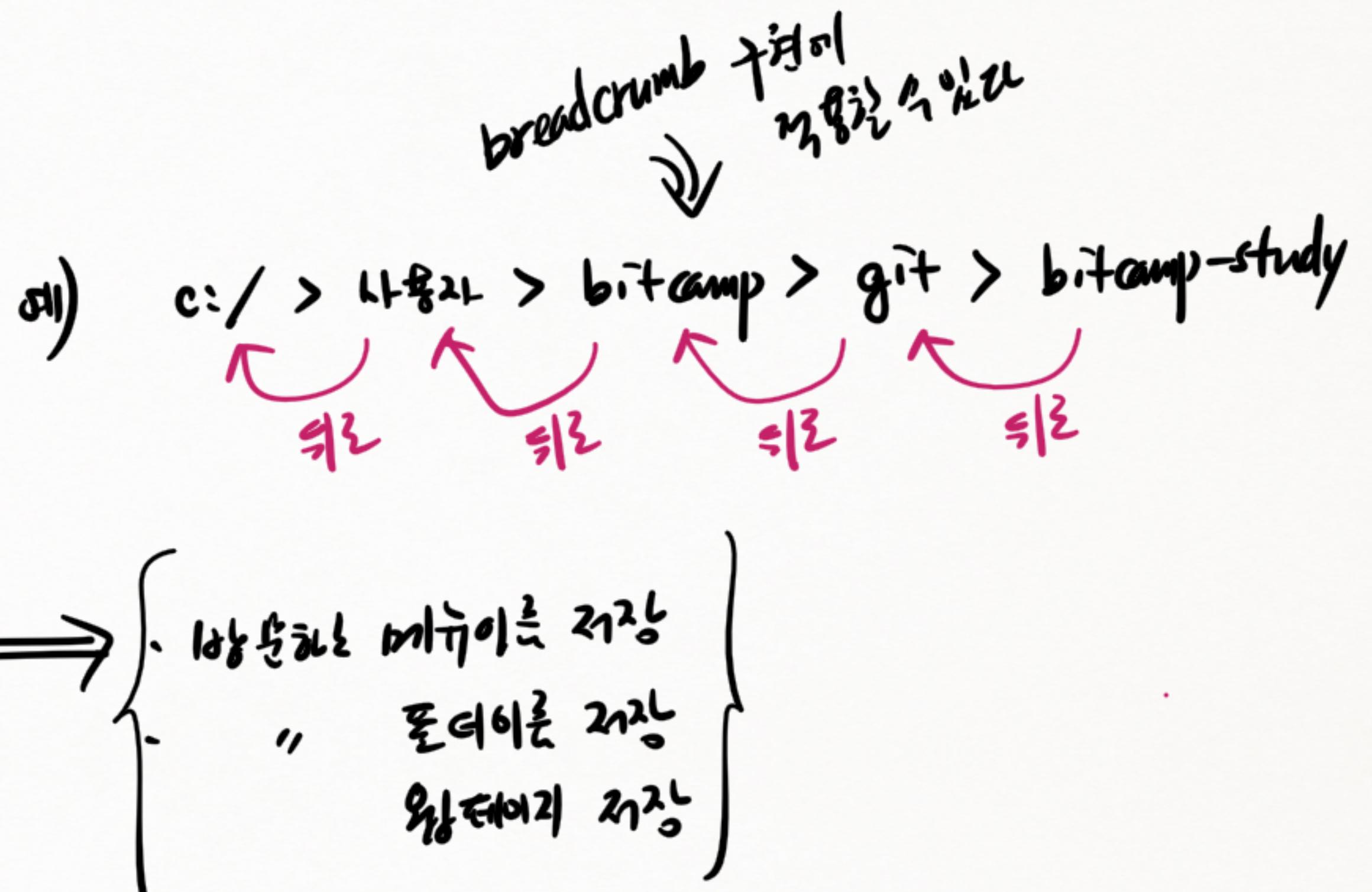
드로잉 메뉴
Breadcrumb 메뉴

메인 / 카운트 >

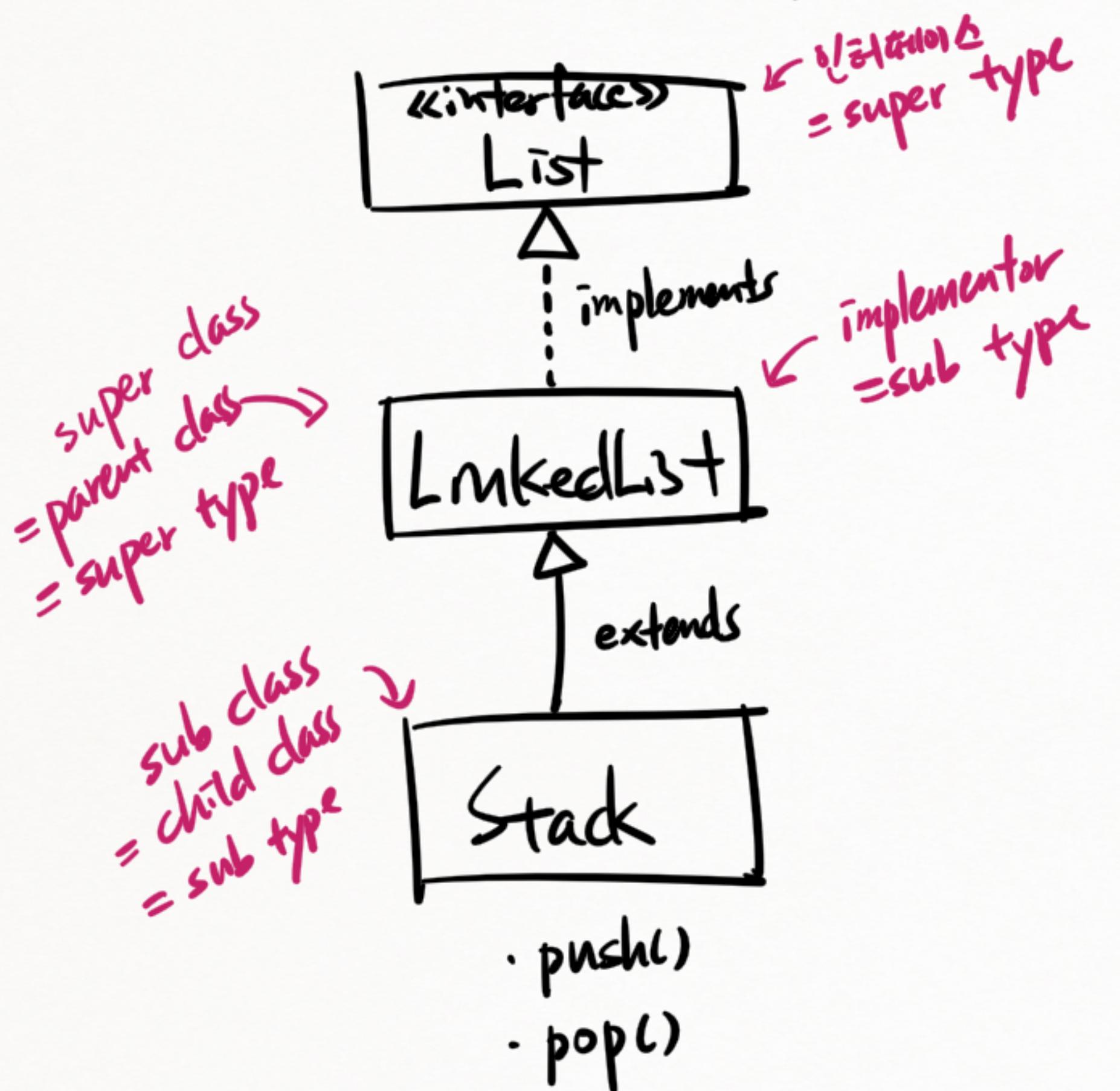
* Stack 구조화



Last In First Out
(LIFO)



* Stack 티입



List obj;

obj = new ArrayList();

obj = new LinkedList();

obj = new Stack();

List 인터페이스를
가지고 있는 구현.
LinkedList를 상속하는
방법으로 구현.
super 클래스가 구현
하는 메소드를 상속.
Stack은 LinkedList를
상속하는 구조로
구현된다.

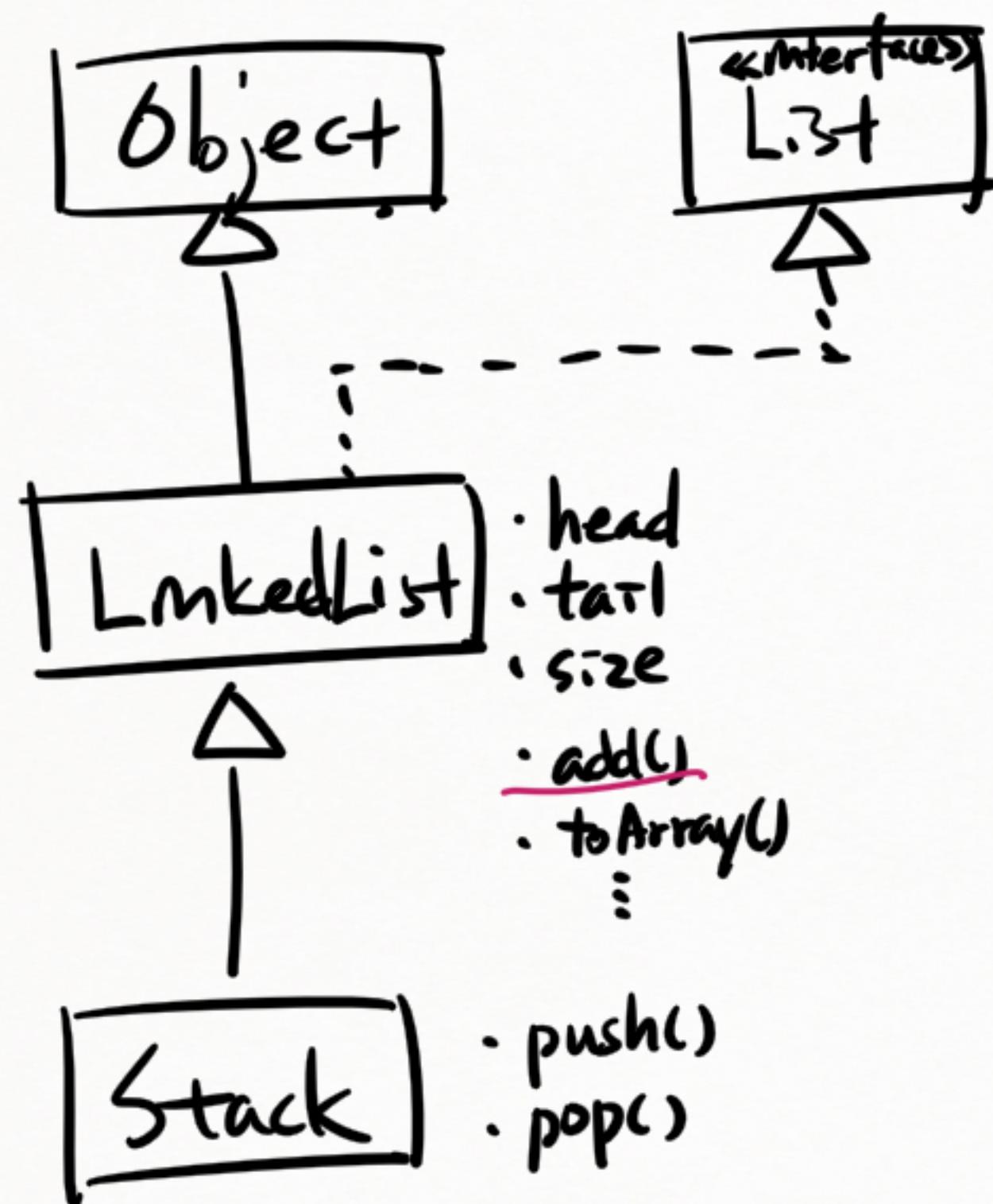
리스트 인터페이스
List
TDL

super 클래스가 구현
하는 메소드는
Stack은 상속하는
방법으로 구현.

Stack은 LinkedList를
상속하는 구조로
구현된다.

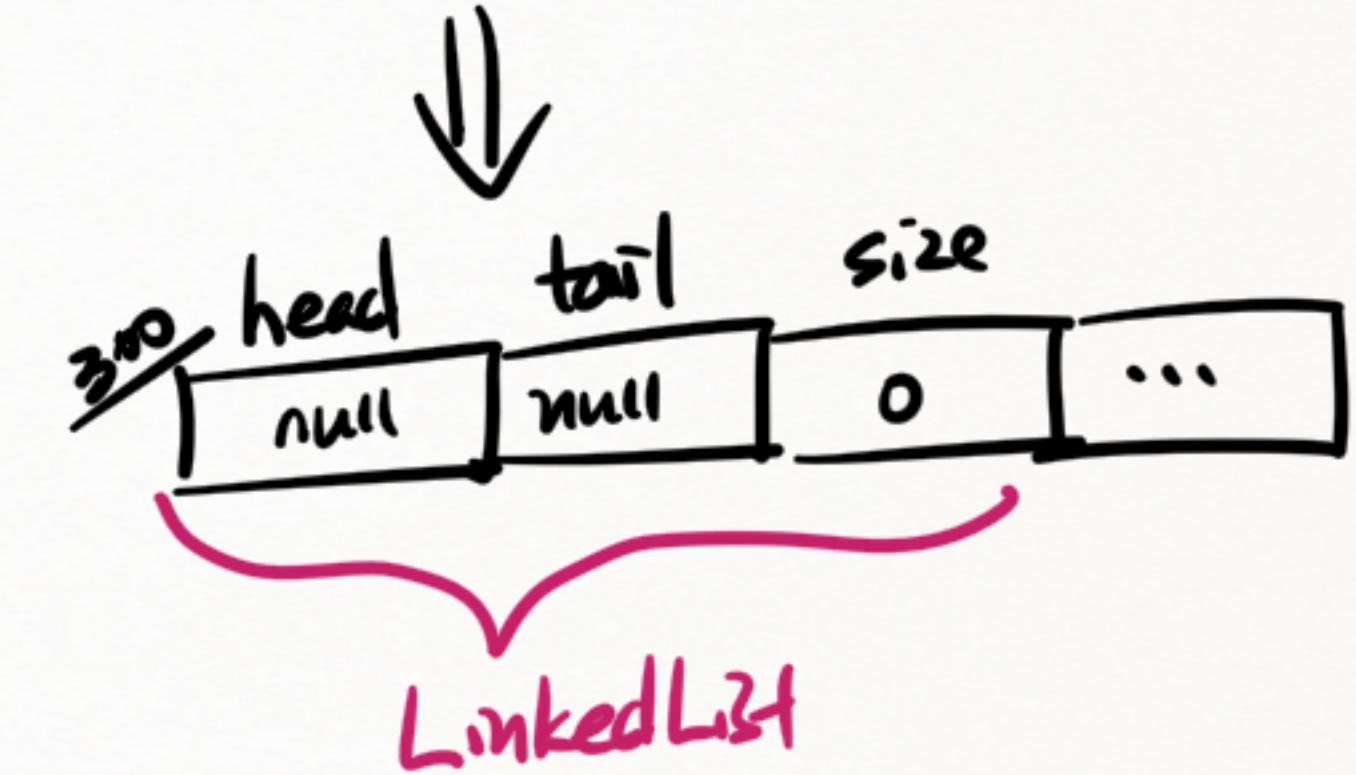
Stack은 LinkedList를
상속하는 구조로
구현된다.

* Stack 인스턴스



Stack s = new Stack();

s
300

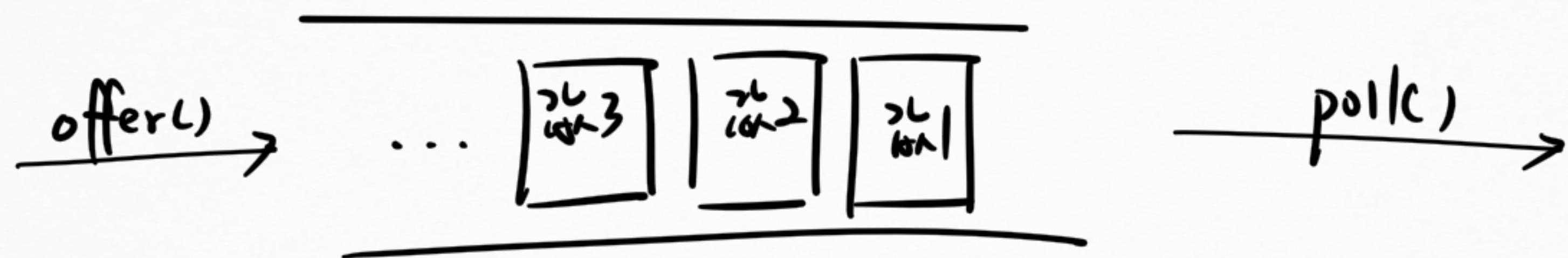


s.push(100);

300 ↳ this.add(100);

300 ↳ this.head
this.tail
this.size

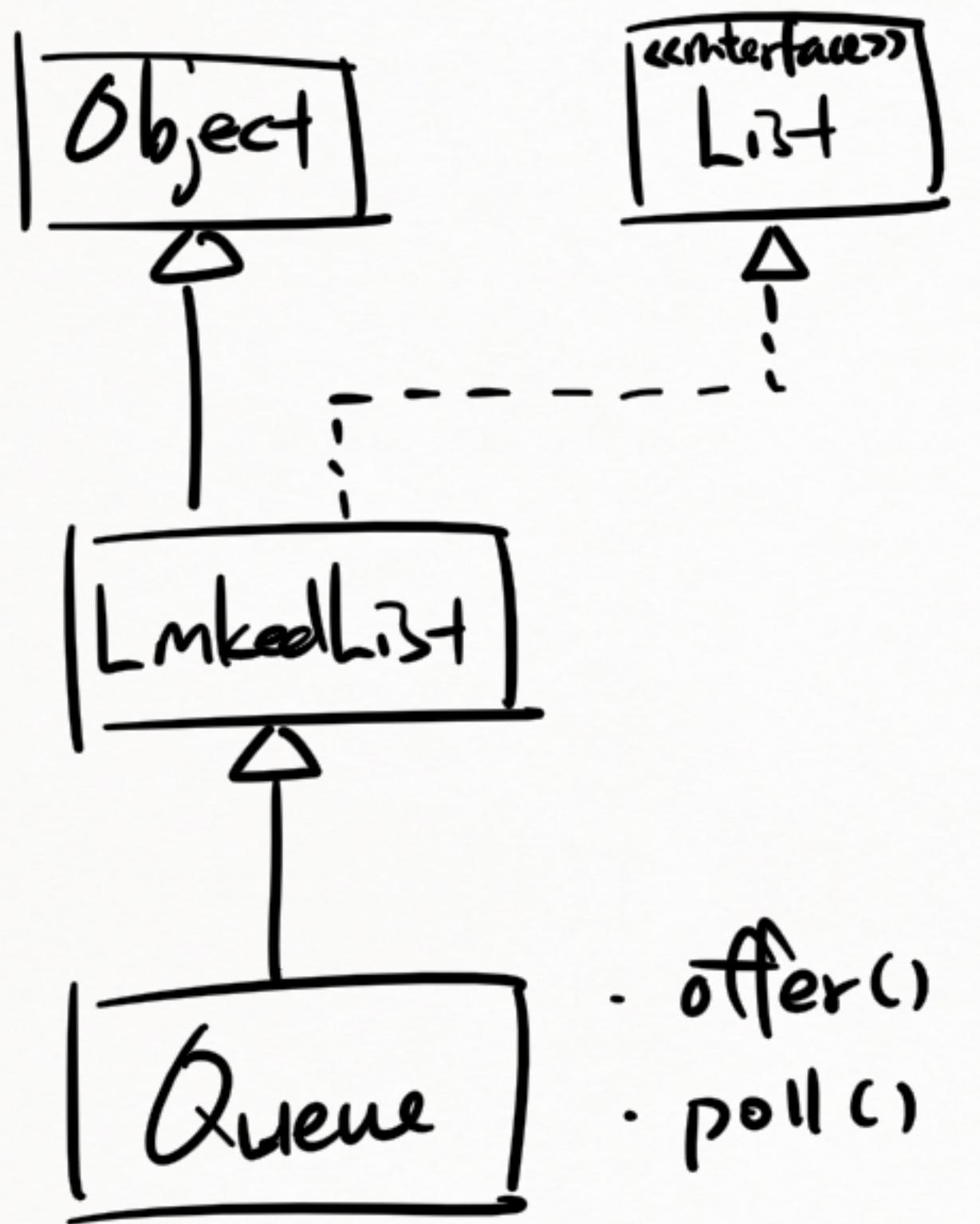
* Queue 풀이



First In First Out
(FIFO)

- ⇒
- 명령어 출력, history 기능에 쓰임
 - (예약)
(주문)

* Queue 구현과 예제



- `offer()`
- `poll()`

