

class

* 팀스 운영의 패턴

- ① 멤버는 문구 : MemberHandler
Prompt
- ② 새 아이디 태입을 확장 : Score, Member

* 새 데이터 타입 정의

① class 정의

```
class Score {  
    String name;  
    int kor;  
    int eng;  
    int math;  
    int sum;  
    float aver;  
}
```

"인스턴스 변수" (field)

④ 인스턴스 초기화

obj. name = "홍길동"

② 인스턴스 할당

```
Score obj;
```

obj
200

obj = new Score();



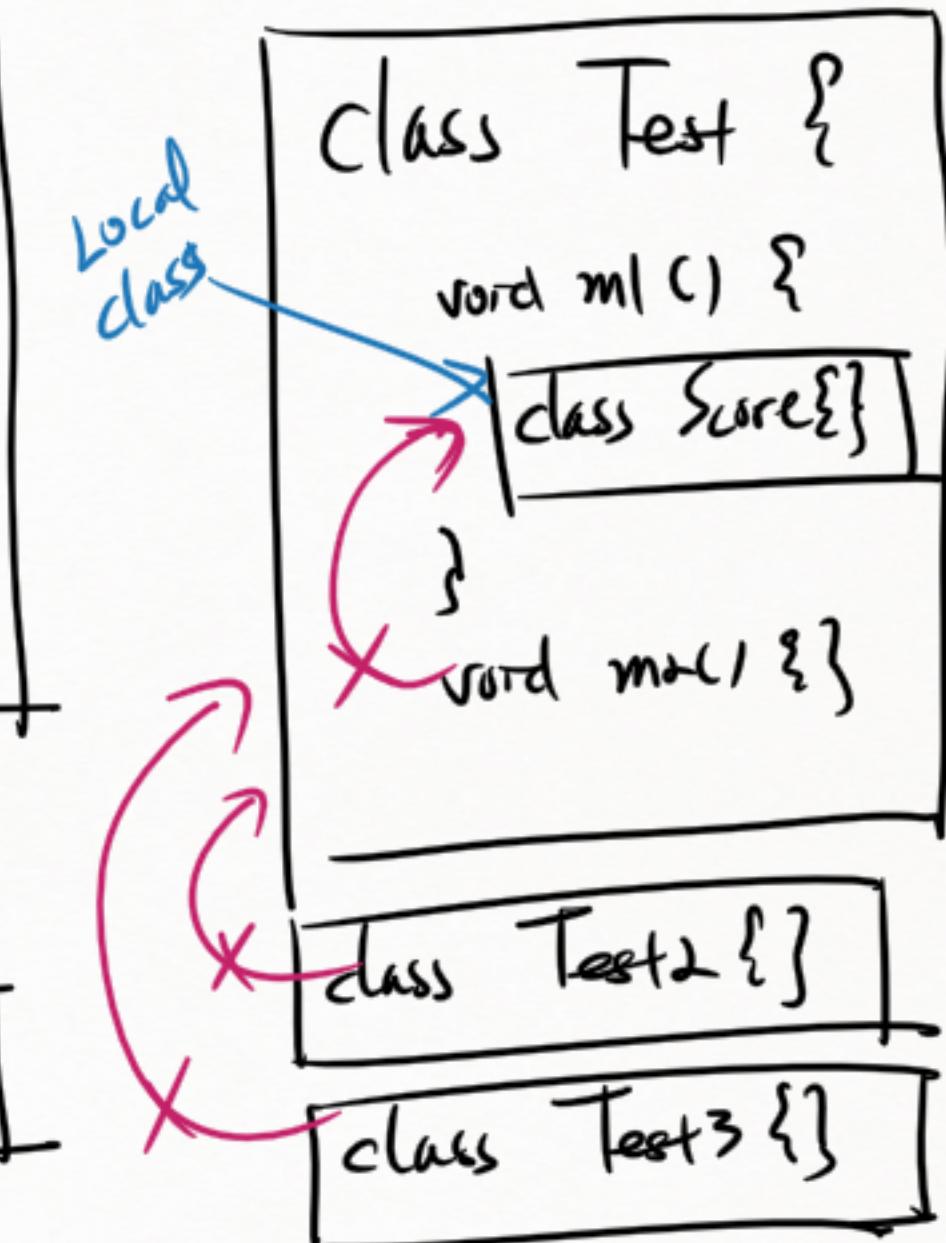
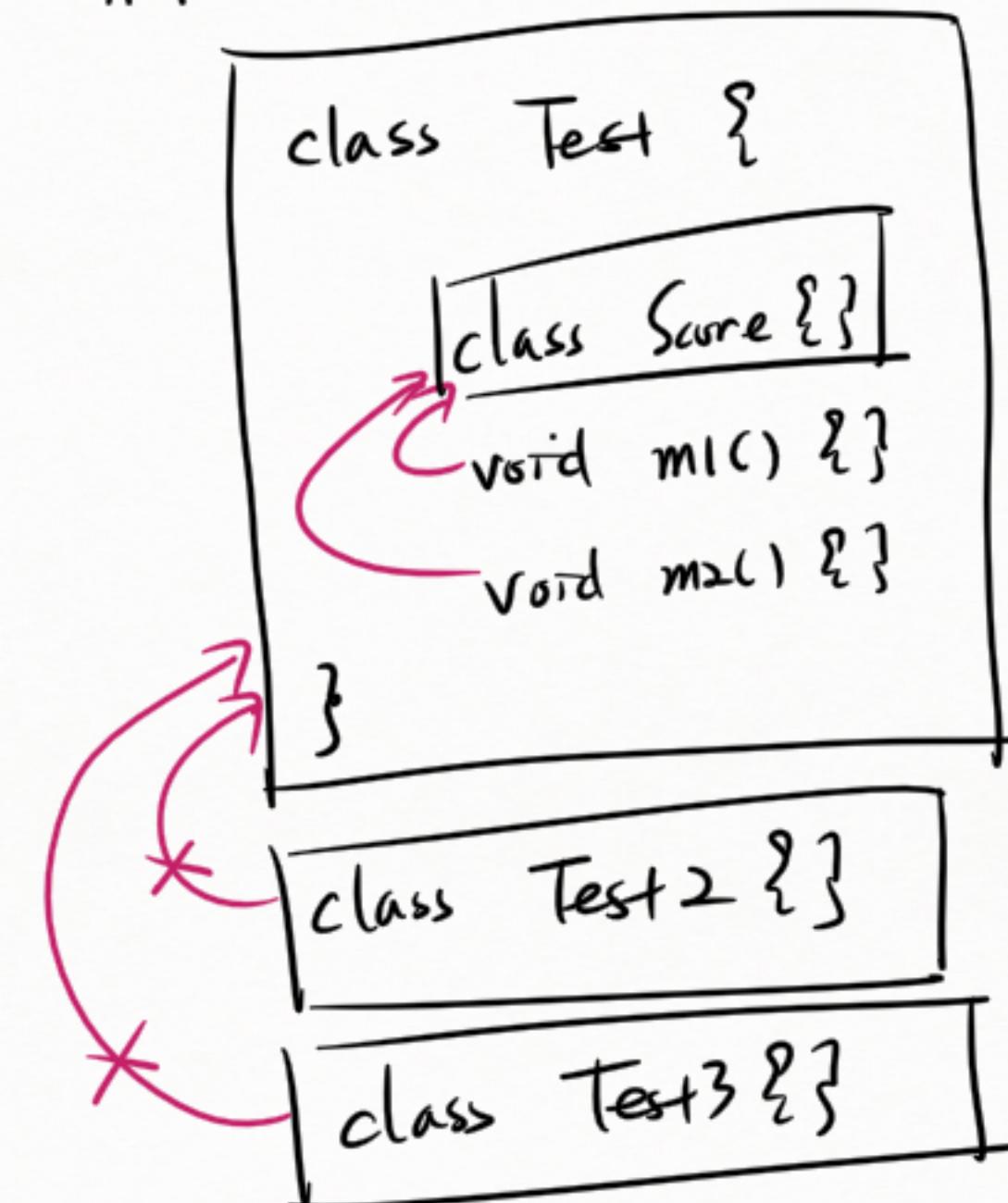
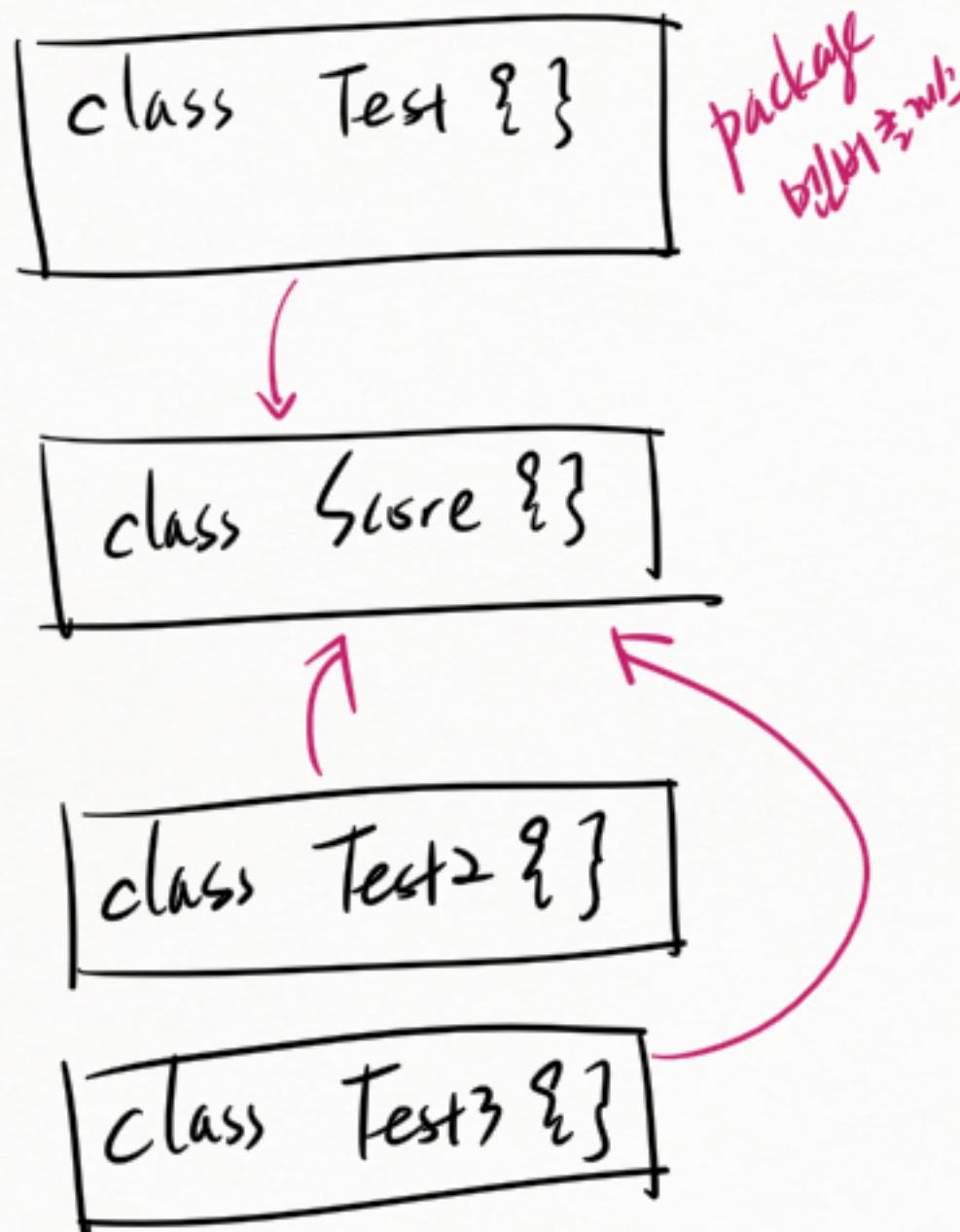
③ 인스턴스 사용

"Score의 인스턴스"
instance

"reference"

↑
Score의 인스턴스 주소를 저장하는 변수

* 클래스 정의의 유형



"Nested class"

* 인스턴스 생성, 메모리, call by reference

Score s = new Score();

s
200

200	name	kur	eng	math	sum	aver
	String	int	int	int	int	float
	한국어	100	90	80	270	90.0

s.name = "한국어";

s.kor = 100;

s.eng = 90;

s.math = 80;

s.sum = s.kor + s.eng + s.math;

s.aver = s.sum / 3f;

printScore(s);

인스턴스의 주소

printScore(Score s) {

}

System.out.printf();

}

* 디자인한 인스턴스 생성 후 출력

```

Score s = createScore("김민수", 100, 100, 100);    createScore (String name, int kor, int eng, int math) {
    s
    

|     |
|-----|
| 200 |
|-----|


    prntScore(s);
}

}
    Score s = new Score();
    s.name = name;
    s.kor = kor;           s.sum =
    s.eng = eng;           s.aver =
    s.math = math;
    return s;
}

s


|     |
|-----|
| 200 |
|-----|


    name kor eng math sum aver
    "김민수" 100 100 100 300 100.0

```

* 커스텀 클래스 사용 예

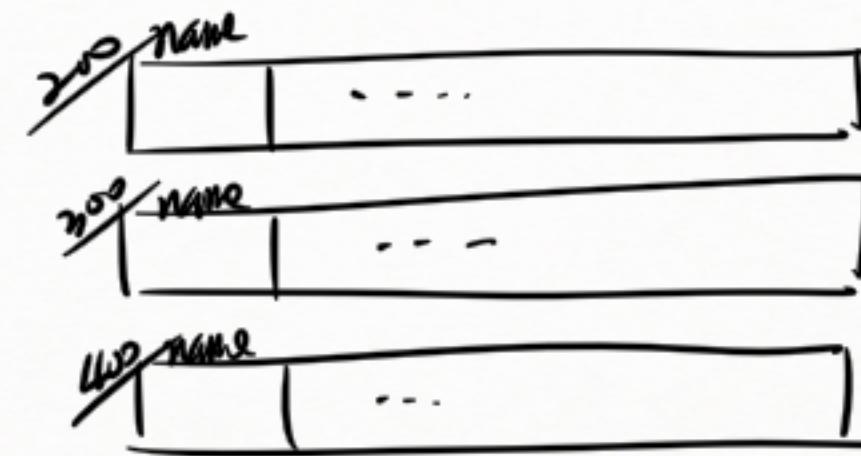
Score s1, s2, s3;

s1 | 200 s2 | 300 s3 | 400

s1 = new Score();

s2 = new Score();

s3 = new Score();



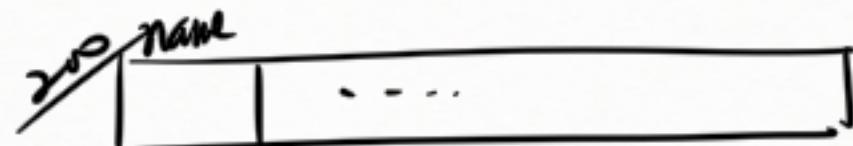
* 리퍼런스 변수 사용 후
 ↳ 리퍼런스의 대체값
Score[] scores = new Score[3];

scores

1700



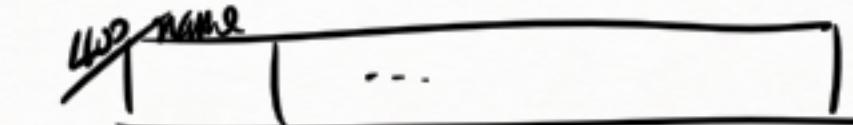
scores[0] = new Score();



scores[1] = new Score();



scores[2] = new Score();



* 인스턴스와 메서드:

Score s1 = new Score();



"Score의 인스턴스"
가짐

Score s2 = s1;

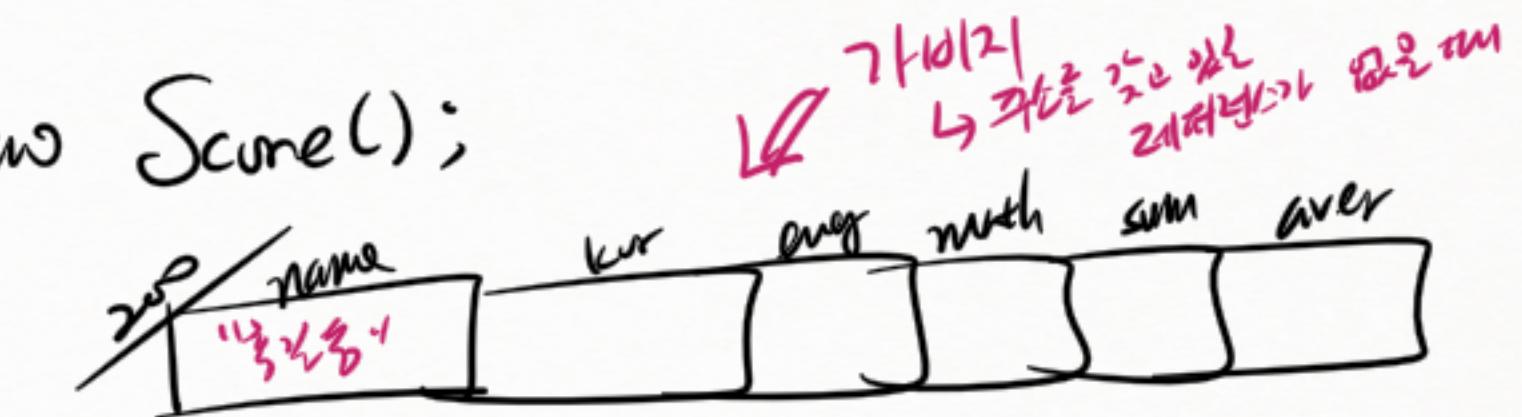


* 111011 (Garbage)

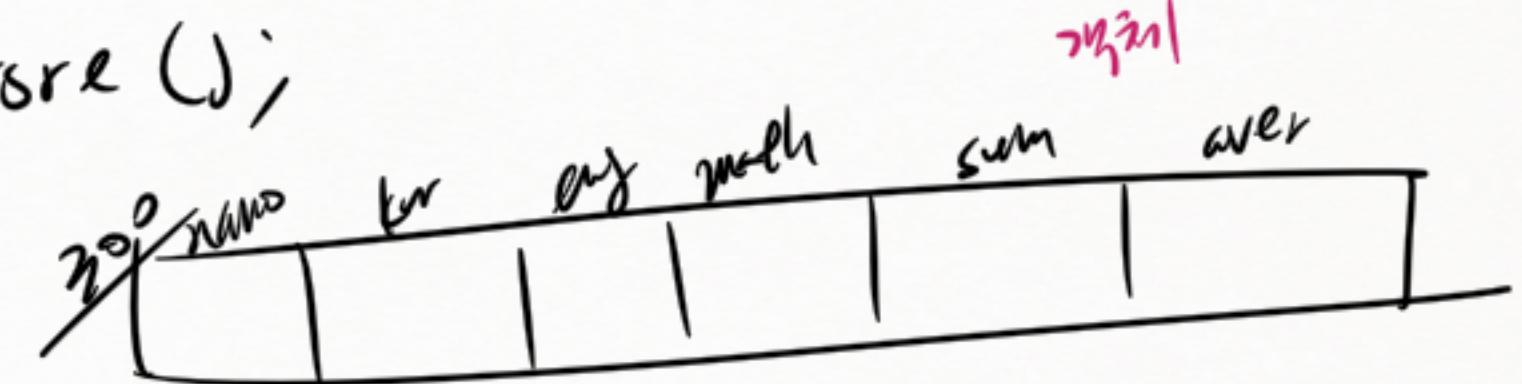
Score s1 = new Score();



s1 = new Score();



"Score의 인스턴스"

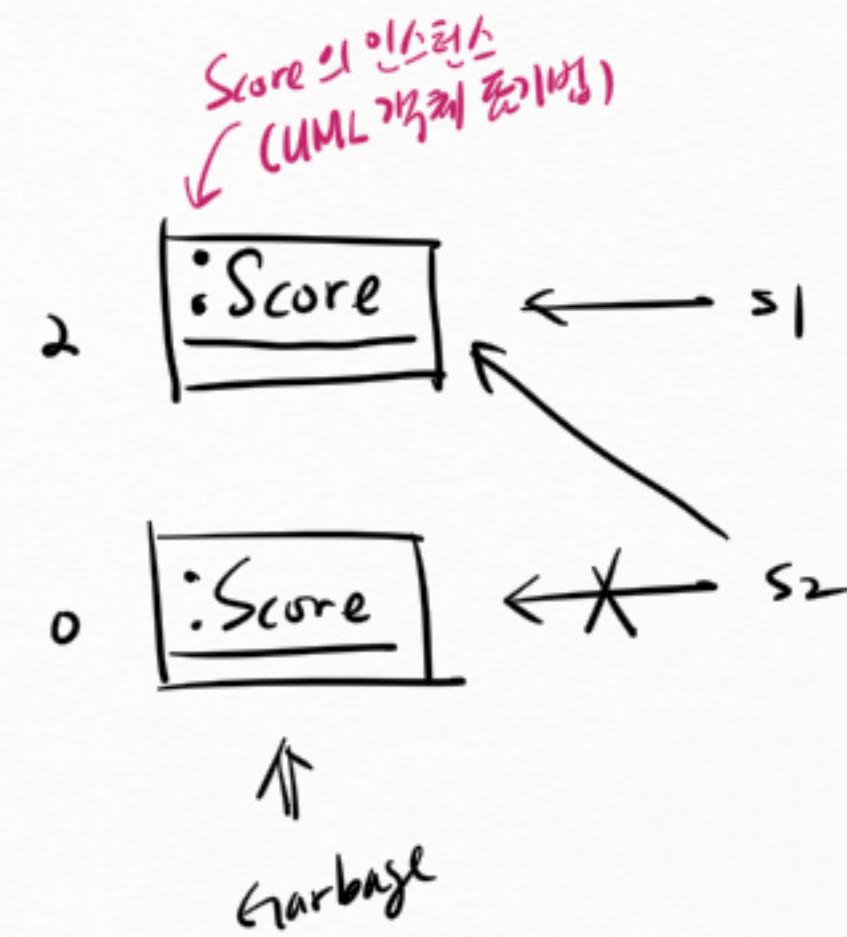


* 인스턴스와 리퍼런스 차운트

```
Score s1 = new Score();
```

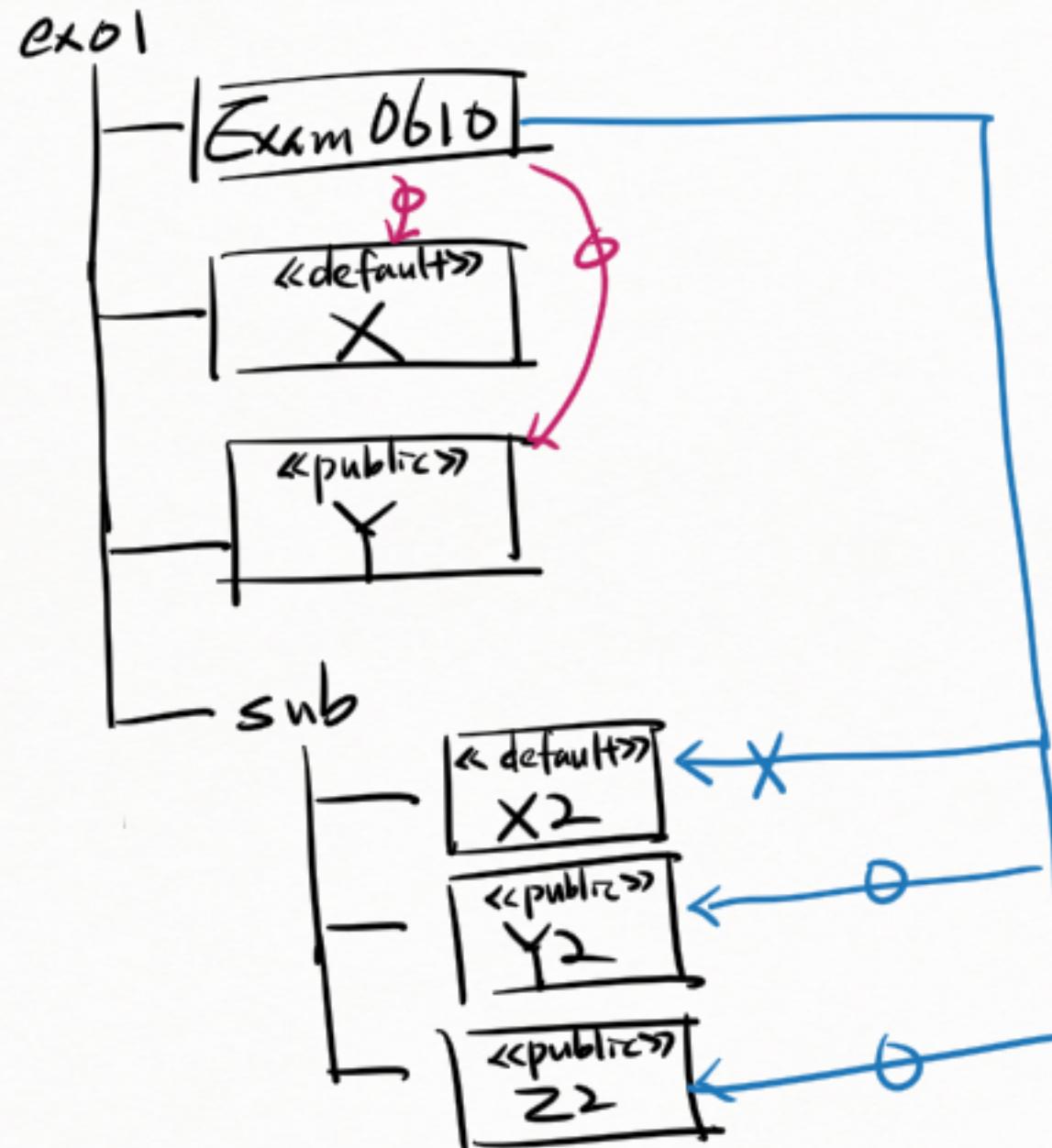
```
Score s2 = new Score();
```

```
s2 = s1;
```



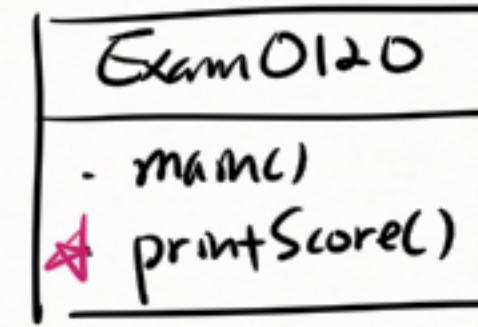
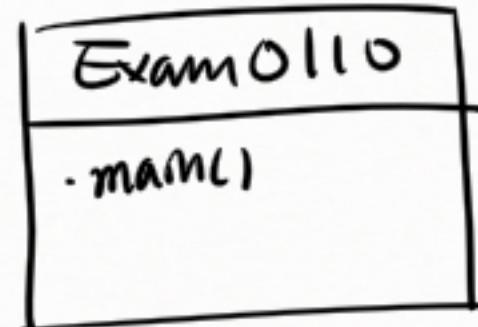
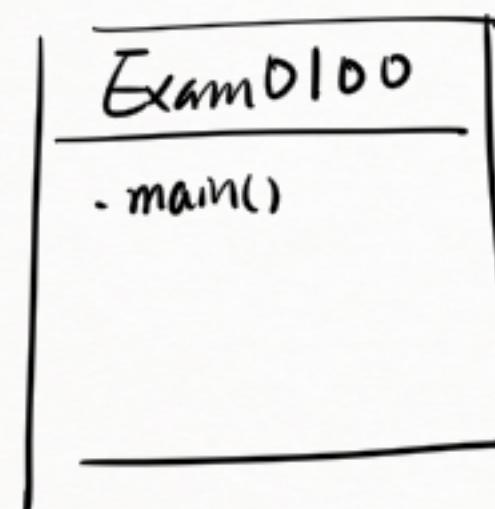
* public $\frac{3}{2}$ mLcf

default $\frac{3}{2}$ mL
(package private class)

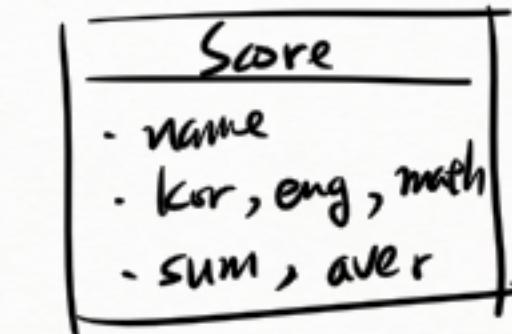


* com. cs. oop. ex02. Exam01xx

① 놓개 변수 사용 → ② class, 블법: 새 데이터 타입 정의 → ③ method 블법: 중복코드 제거



- 메모리와 인스턴스
- ↓
 new
 ↓
 Heap 영역
 ↓
 garbage
 ↓
 garbage collector

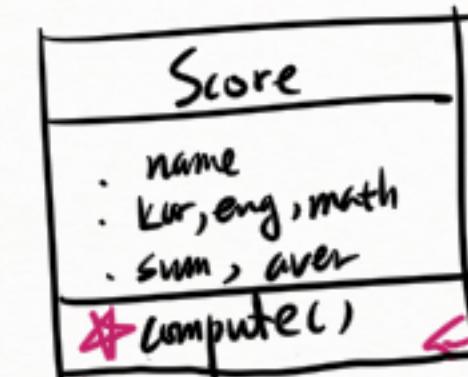
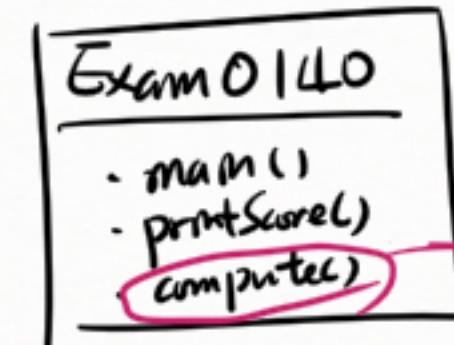
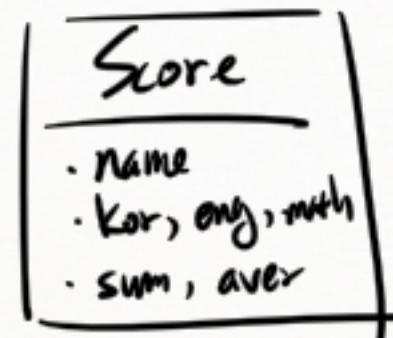
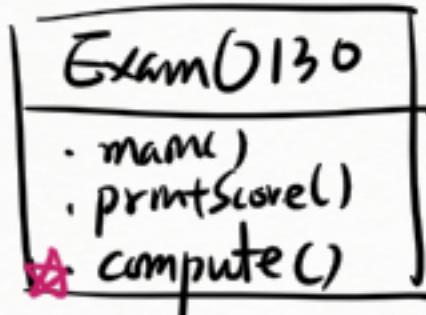


④ 리팩토링: 1기능 → 1 메서드

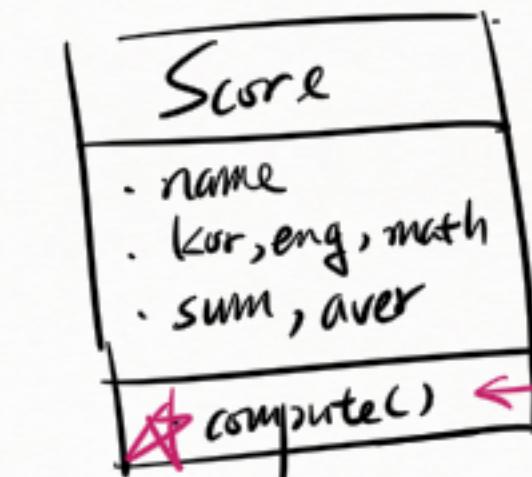
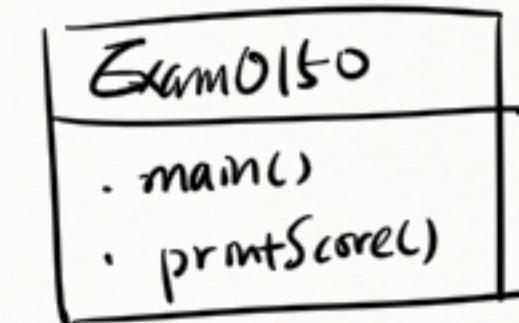
→ ⑤ 리팩토링: 멤버드 이동

→ ⑥ 인스턴스에 더 쉽게 접근하는 법: 인스턴스 메서드

non-static



이동



non-static 으로
변경

GRASP 의
Information
Expert

이전 방식
Score. compute(인스턴스주소)

변경 후
인스턴스주소. compute()

* 스태틱 메서드와 인스턴스 메서드

static 메서드 \Rightarrow Score. compute(s1);
||
클래스 메서드

메서드가 소속된 클래스

non-static 메서드 \Rightarrow s1. compute();
||
인스턴스 메서드

메서드가 소속된 클래스의 인스턴스 주는
~~~ 인스턴스를 보다 쉽게 다루는  
메서드 문법

\* 인스턴스 메서드와 this

인스턴스  
메서드를 호출할 때  
this는  
매개변수로  
传여  
된다.  
이쪽에서 넘겨온 인스턴스 주소를 받는  
built-in 로컬 변수

non-static 멤버에만 존재한다.

\* this 가는  
this 멤버변수에 저장된다

↓  
인스턴스 주소를 넘기지만  
이로 인수를 전달할  
필자가 있다.

```
void compute() {  
    this.sum = this.kor + this.eng + this.math;  
    :  
}
```