

\* 클래스 문법

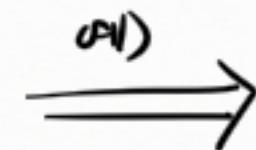
클래스

① 데이터 타입 정의 (User-defined Data Type)  
개별화

② 메서드 분류

\* ↗ 클래스 문법 - 데이터 타입 정의

class 데이터타입 {  
 변수선언  
 :  
}

예) 

class Contact {  
 String name;  
 String email;  
 String tel;  
 String company;  
}

← 메모리  
설정

||  
~~new~~ 명령을 실행하면  
클래스에 정의된 대로  
변수가 준비된다.

\* 클래스를 이용하여 새 데이터 타입의 변수 만들기

Contact c = new Contact();

Contact 클래스의 주소를 저장하는  
리퍼런스(reference)

리터리터링 = 클래스명

클래스 선언에 따라 메모리 할당 => Heap 영역

인스턴스(instance) = 개체(object)

c | 200

200 | name email tel company

## \* 레퍼런스 배열

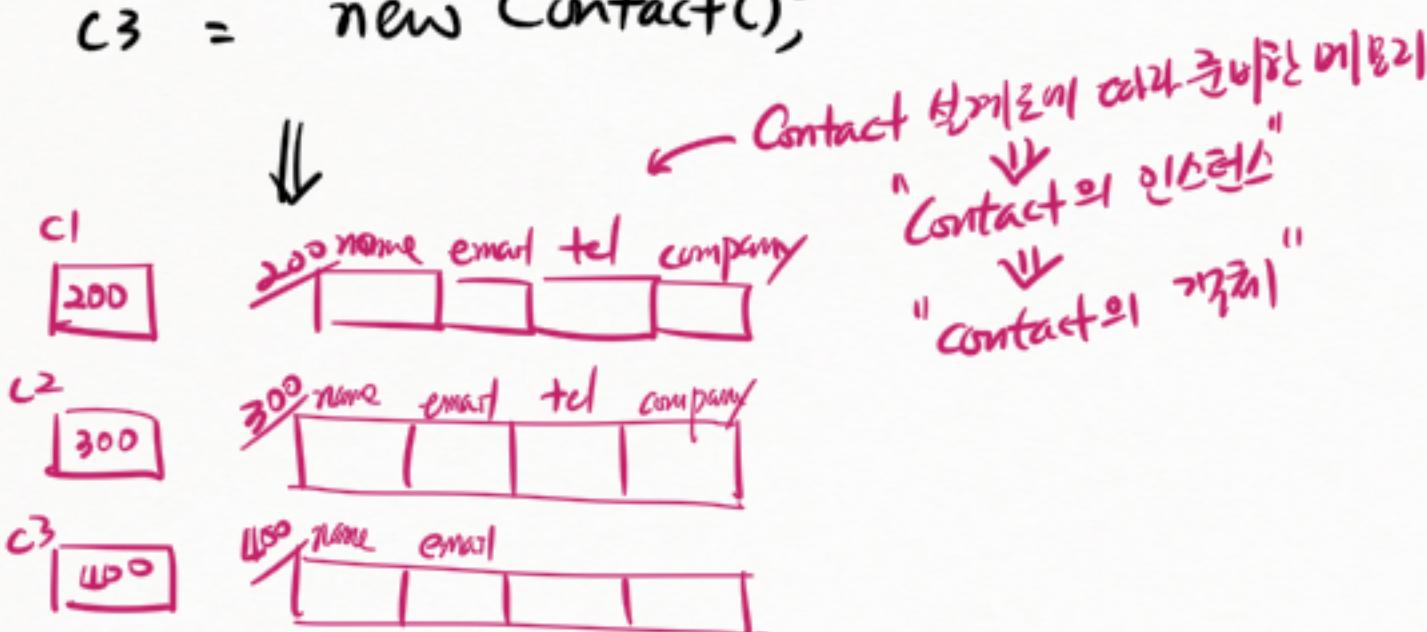
## ① 배포 사용 전

① 초기화 사용 선  
Contact c1, c2, c3;      ↪ 인스턴스의 주소를 저장하는 변수;  
    "리퍼런스"  
    "포인터(pointer)"

```
c1 = new Contact();
```

```
c2 = new Contact();
```

```
c3 = new Contact();
```

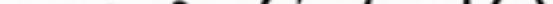


## ② 배포 사용 흐름

Contact[] arr = new Contact[3];  
레퍼런스를 3개 만드는 명령



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



A hand-drawn diagram of a Contact object. It is represented as a horizontal rectangle divided into four equal-sized boxes. Above the first box, the label 'name' is written. Above the second box, the label 'email' is written. Above the third box, the label 'tel' is written. Above the fourth box, the label 'company' is written. To the left of the first box, there is a crossed-out label 'who'.



```
arr[1] = new Contact();
```



~~arr[2] = new Contact();~~

## \* 레퍼런스와 인스턴스 변수

Contact c = new Contact()



① 인스턴스 변수가 없는 저장

c.name = "홍길동";

인스턴스 주소를  
알고 있는  
레퍼런스

↑  
인스턴스  
변수

c.email = "hong@";

c.tel = "1111";

c.company = "비트";

② 인스턴스 변경

c = new Contact()



c.name = "이꺽정";

c.email = "leem@";

c.tel = "2222";

c.company = "캐논";

기존 인스턴스의  
주소를 알고 있어  
레퍼런스가 한 개로 고정된  
''garbage'' 가 된다.

### Method Area

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

### JVM Stack

```
Score s;
```

s 200

↑  
Score의 레퍼런스

### Heap

```
new Score()
```



200 name kor eng math sum aver

↑  
Score의 인스턴스  
(기록체)

\* com.eomcs.oop.exam.Exam0114

Method Area

```
class Exam0114 {  
    public static void main(String[] args) {  
        System.out.println("Hello Java");  
    }  
}
```

```
class Score {  
    public static void main(String[] args) {  
        System.out.println("Hello Java");  
    }  
}
```

JVM Stack

```
main()  
args [ ] s [ ]
```

Heap

\* com.eomcs.oop.ex01.Exam0114

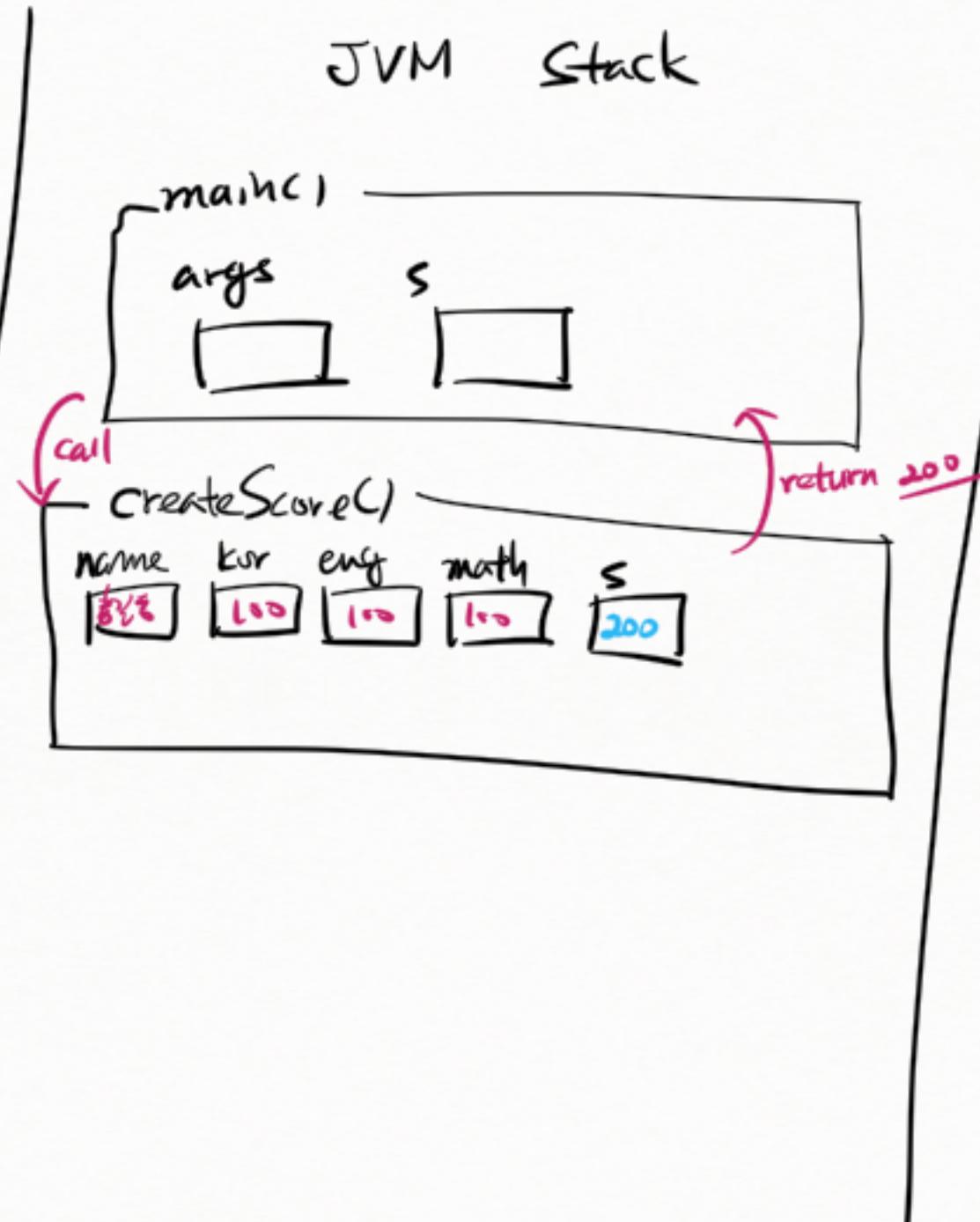
Method Area

```
class Exam0114 {
    public static void main(String[] args) {
        Score s = new Score();
        System.out.println("Sum: " + s.sum);
    }
}
```

```
class Score {
    private String name;
    private int kur;
    private int eng;
    private int math;
    private int sum;
    private double aver;

    public Score() {
        name = "BKT";
        kur = 100;
        eng = 100;
        math = 100;
        sum = 300;
        aver = 100;
    }
}
```

JVM Stack



Heap

name	kur	eng	math	sum	aver
BKT	100	100	100	300	100

\* com.eomcs.oop.exo1.Exam0114

Method Area

```
class Exam0114 {  
    public static void main(String[] args) {  
        System.out.println("Hello Java");  
    }  
}
```

```
class Score {  
    public static void main(String[] args) {  
        System.out.println("Hello Java");  
    }  
}
```

JVM Stack



Heap

name	kor	eng	math	sum	aver
200	32.5	100	100	100	300

\* com.eomcs.oop.exam. Exam0114

Method Area

```
class Exam0114 {  
    public static void main(String[] args) {  
        Score s = new Score();  
        s.printScore();  
    }  
}
```

```
class Score {  
    public void printScore() {  
        System.out.println("Hello Score");  
    }  
}
```

JVM Stack



Heap

name	kor	eng	math	sum	aver
200	72.7	100	100	100	300

name	kor	eng	math	sum	aver
200	72.7	100	100	100	300

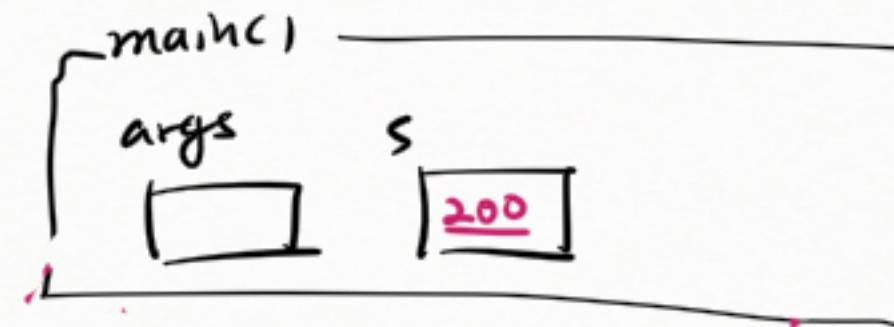
\* com.eomcs.oop.exo1. Exam0114

Method Area

```
class Exam0114 {  
    public static void main(String[] args) {  
        System.out.println("Hello Java");  
    }  
}
```

```
class Score {  
    public static void main(String[] args) {  
        System.out.println("Hello Java");  
    }  
}
```

JVM Stack



Heap

name	kor	eng	math	sum	aver
200	72	100	100	300	100

\* com.eomcs.001.ex01 . Exam0210

Score  $s_1, s_2, s_3$ :

$s_1$  | 200

$s_2$  | 300

$s_3$  | 1100

200	name	kur	...

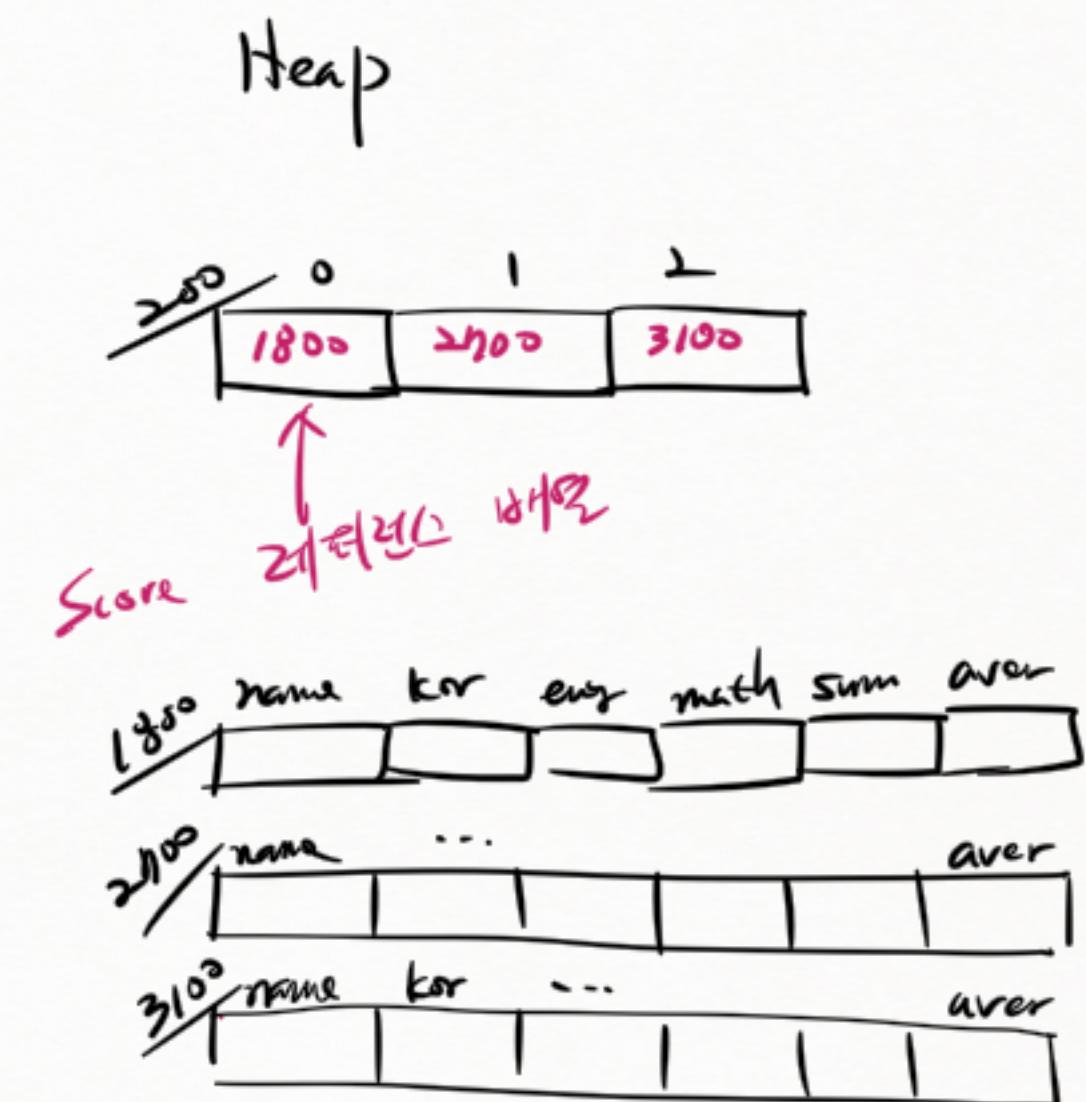
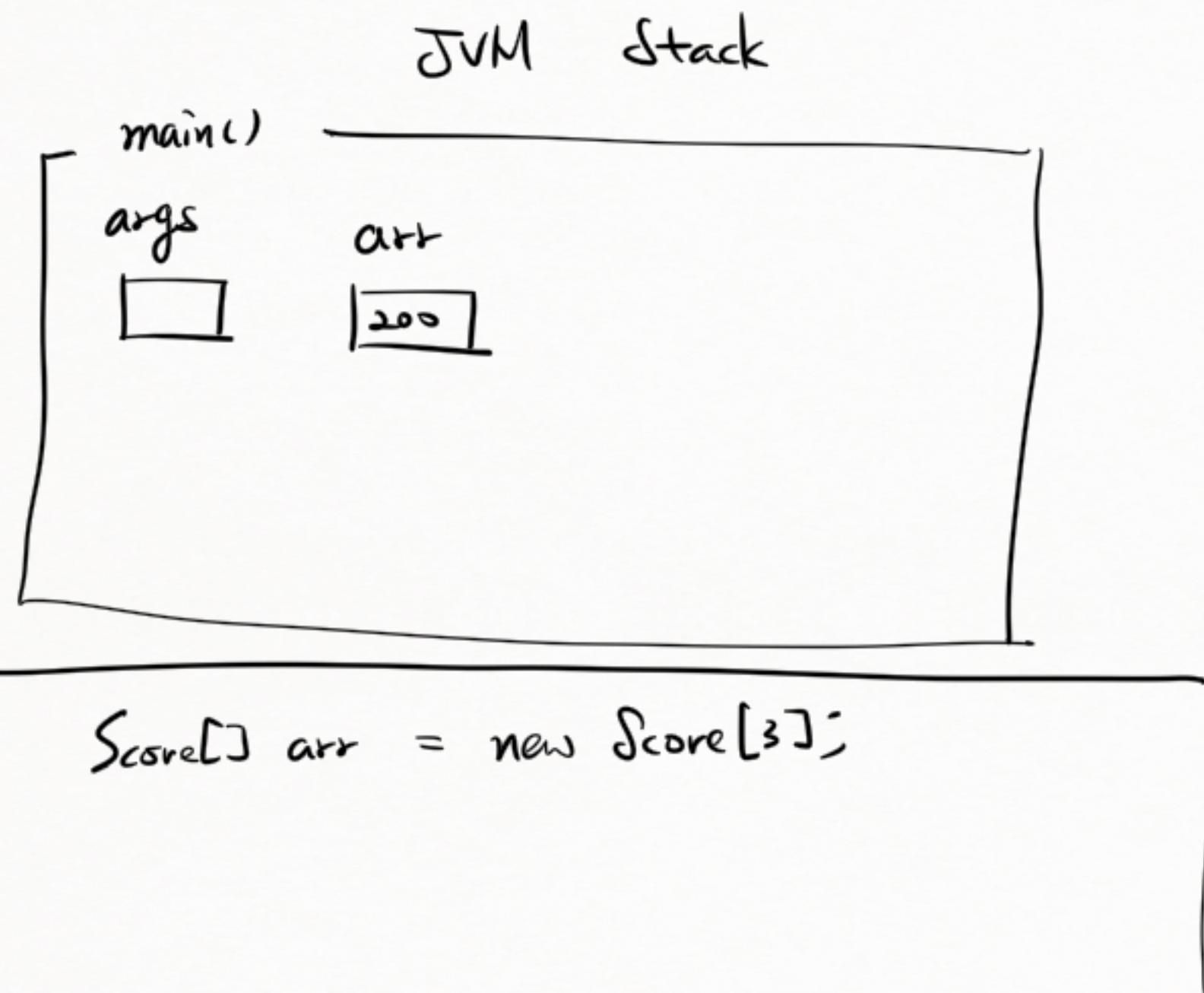
  

300	name	kur	

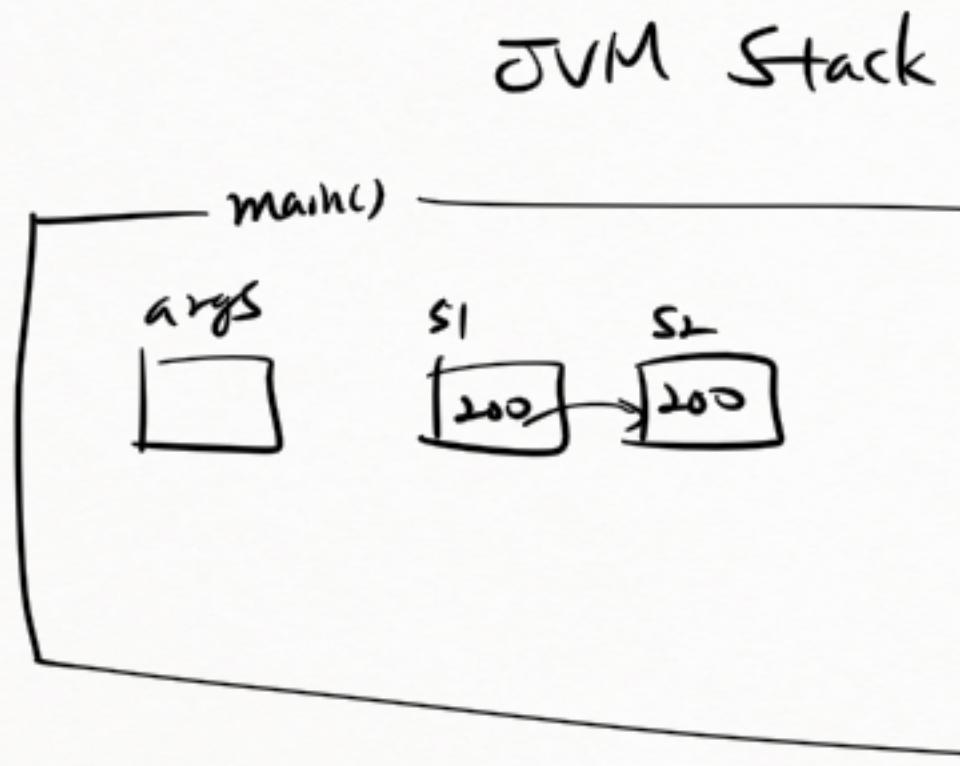
  

1100	name	kur	

\* com.eomcs.oop.ex01.Exam0220

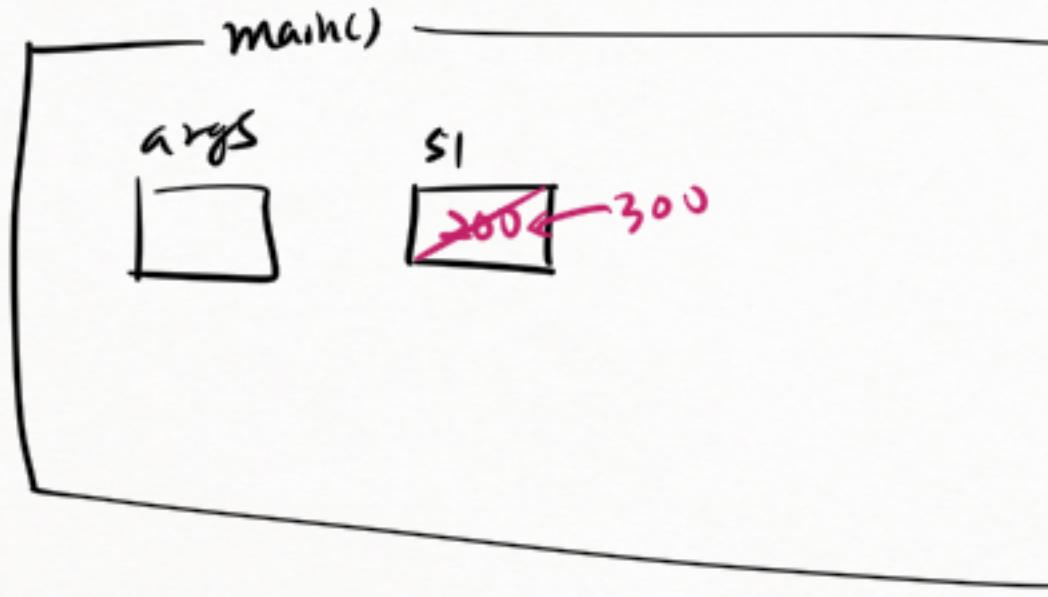


\* com.eomcs.007. ex01. Exam0310



\* com.eomcs.06p. ex01. Exam0320

JVM Stack



Score s1;

s1 = new Score();

s1 = new Score();

Heap

name	kor	eng	math	sum	aver
null	0	0	0	0	0.0

↑  
인스턴스를 생성하면 각 필드 기본값이 설정된다.

- 객체변수 = null
- 정수변수 = 0
- 부동소수점 변수 = 0.0
- 냄비변수 = false

name	kor	eng	math	sum	aver
300	0	0	0	0	0.0

이 인스턴스의 주소를 갖고 있는  
리터럴스가 단 한개도 없어  
내용은 쓰지 않다  
(적어도 한 개 써야)  
"garbage" 라  
부른다.

# \* com.eomcs.opp.ex01.Exam0330

## JVM Stack



Score s1 = new Score();

Score s2 = new Score();

s2 = s1;

인스턴스 주소는  
200으로 초기화된다.



## Heap

name	kor	eng	math	sum	aver
null	0	0	0	0	0.0

인스턴스를 생성하면 각 필드 기본값이 설정된다.

- 리퍼런스 변수 = null

- 정수 변수 = 0

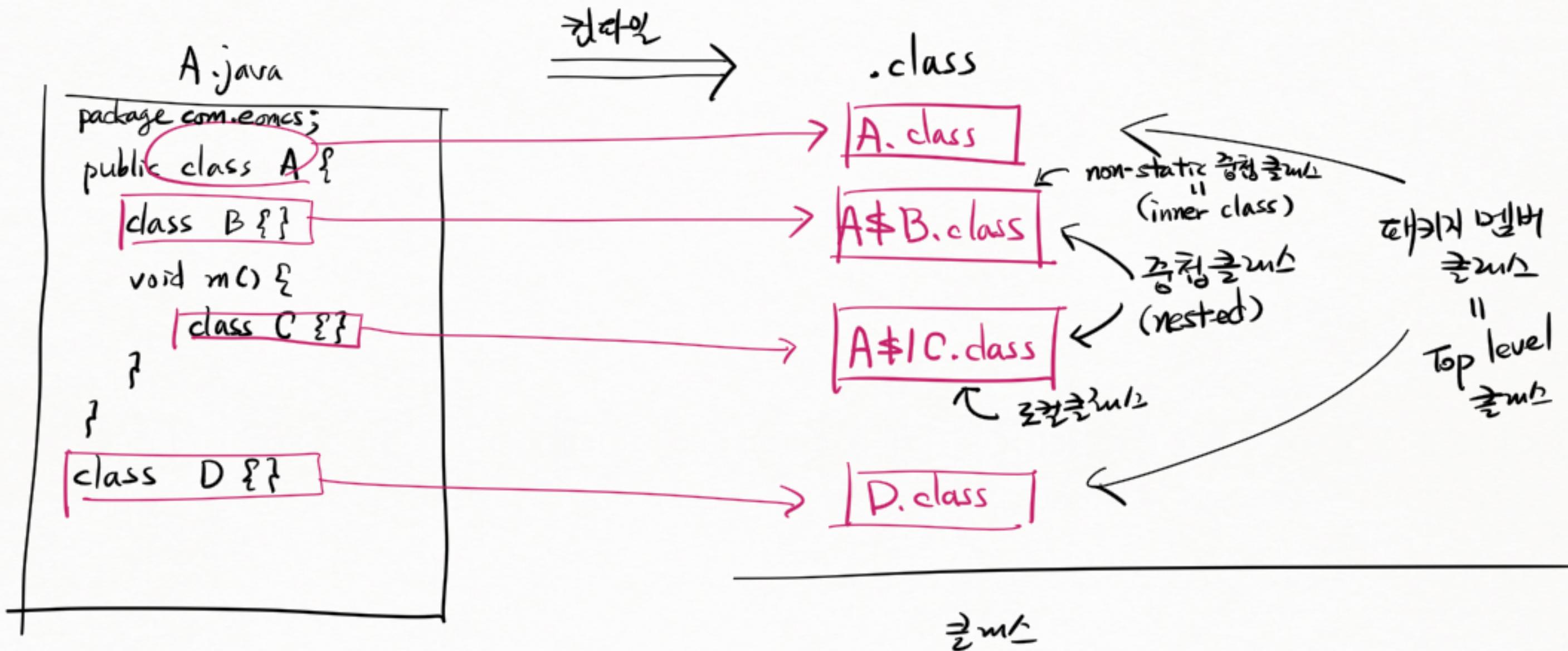
- 부동소수점 변수 = 0.0

- 끝나 변수 = false

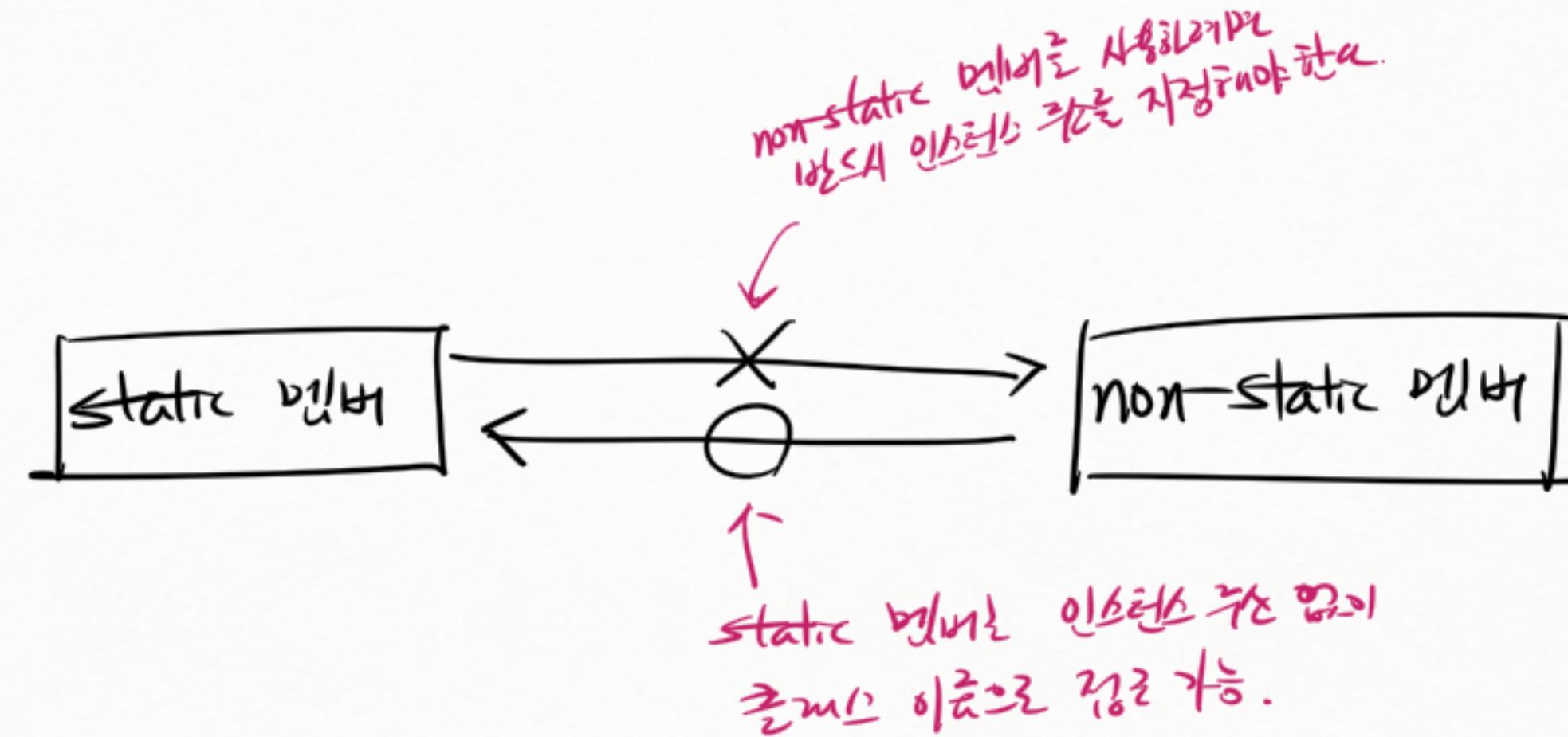
name	kor	eng	math	sum	aver
null	0	0	0	0	0.0

리퍼런스 카운트 개수가 0 이면  
"가ARBAGE (Garbage)" 가 된다.

\* 클래스 복수와 .class 파일



\* static 멤버와 non-static 멤버 간의 접근 규칙



\* 대기지 멤버 구조 : (default) vs public

