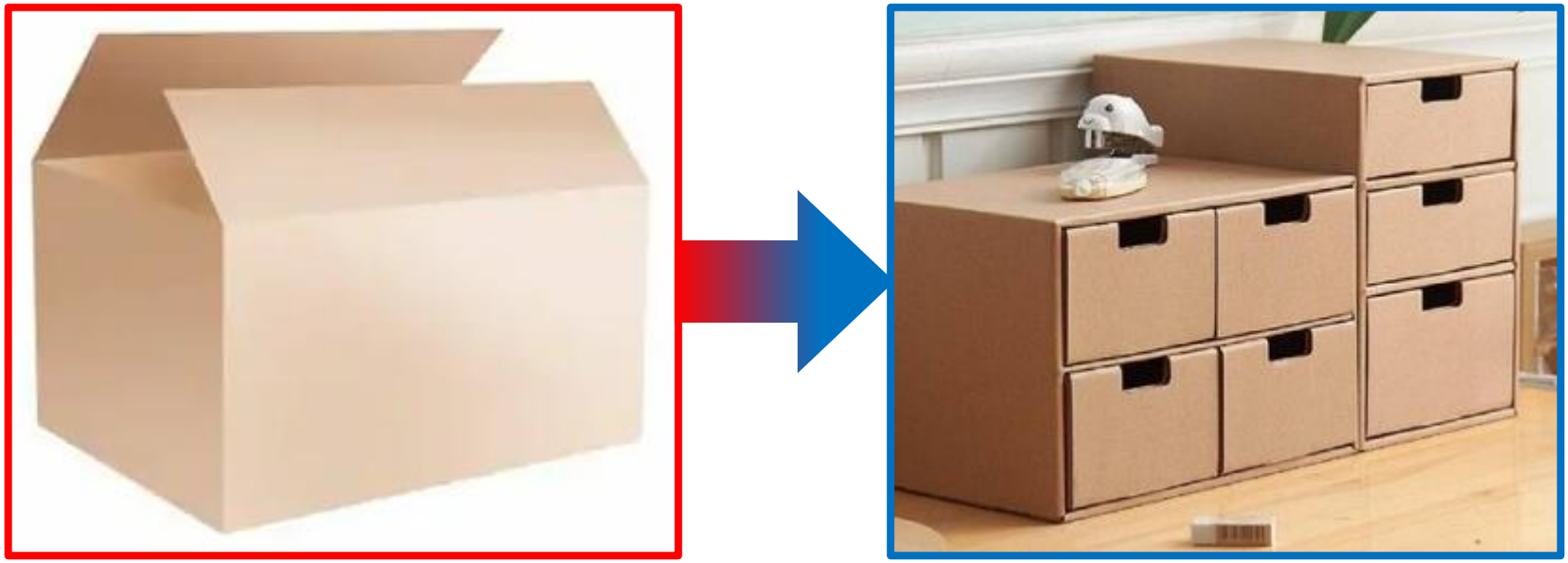

K G 아 이 티 뱅 크
JAVA
J A V A

참조 배열

참조 배열

❖ **기본을 갖춘 클래스가 정의되면 객체를 생성하게 됨**

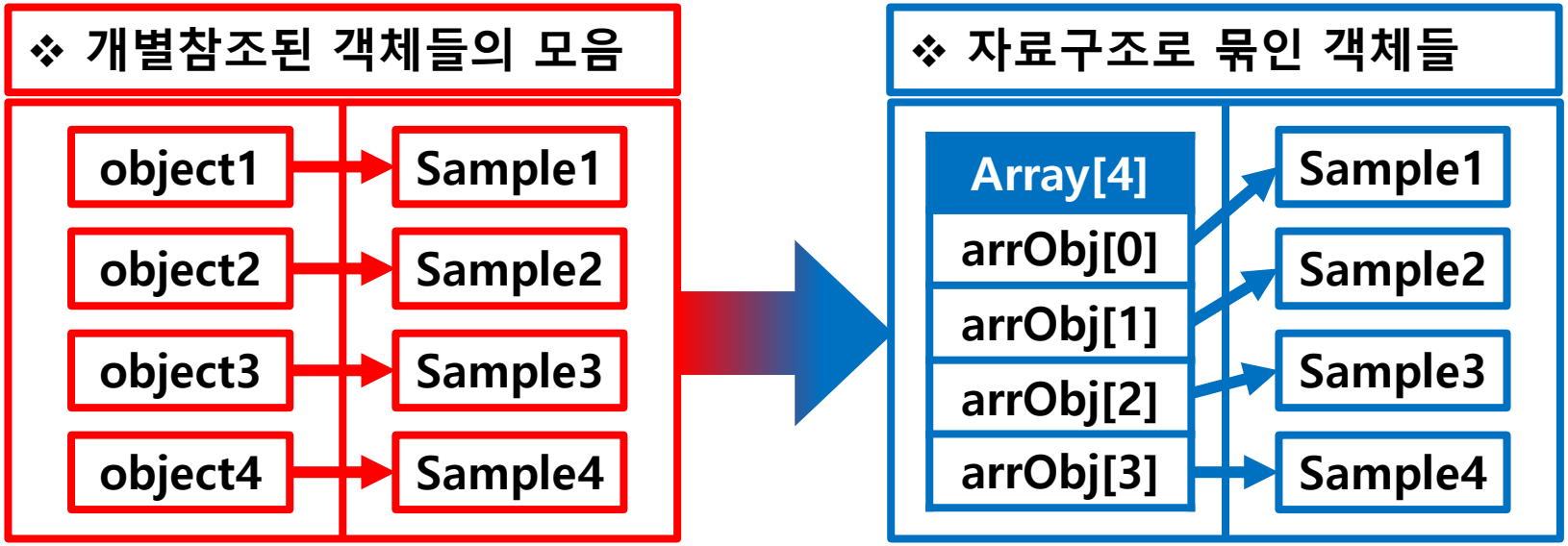
- ➢ 같은 클래스로 생성된 객체는 서로 같은 형제지간
- ➢ 형제 관계이기 때문에, 근본은 서로 다른 존재
- ➢ 단, **객체가 모여서 묘사되는 것들도 존재**하게 됨



참조 배열

❖ **참조 배열 : 참조 자료형으로 만들어진 배열 객체**

- 배열은 순번(인덱스)이 있고, 크기를 가짐
- 배열 외의 자료구조를 이용해서 만들 수 있음
- 단, 참조형이기 때문에 전부 비어 있는 상태임
- 참조형 자료구조를 만들어 구조를 이용하게 됨



참조 배열

❖ 참조 배열의 운용 : 생성은 개별로 처리해야 하니 주의

❖ 참조 배열의 운용

```
SampleClass[] classArr
     = new SampleClass[3];
classArr[0]
     = new SampleClass("A",1);
.......
classArr[0].set(...);
.......
for (int i =0; i < ..... ) {
     classArr[i].show(...);
     .......
}
.......
```

❖ 참조 배열의 생성

생성되는 객체들을 관리하는 자료구조를 생성하는 과정
객체들의 형태를 잡게 됨

❖ 객체를 생성하는 과정

배열은 어디까지나 구조이며 그 자체로는 형태에 불과함
객체를 내부에 생성해줘야 함

❖ 모양이 잡힌 구조를 이용

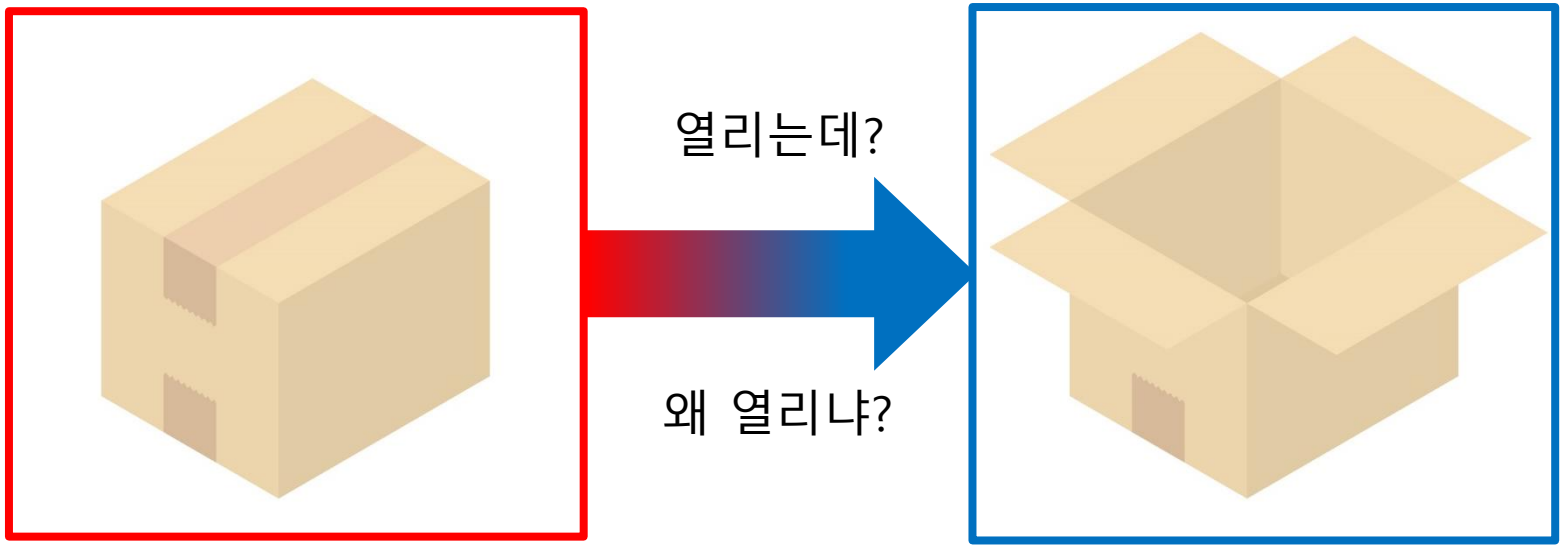
객체들은 서로 다른 객체이나 하나의 구조로 관리됨
각 객체 각각을 사용해야 함

다른 객체간 상호작용

다른 객체간 상호작용

❖ **객체는 현실의 개념을 프로그래밍언어로 묘사한 것**

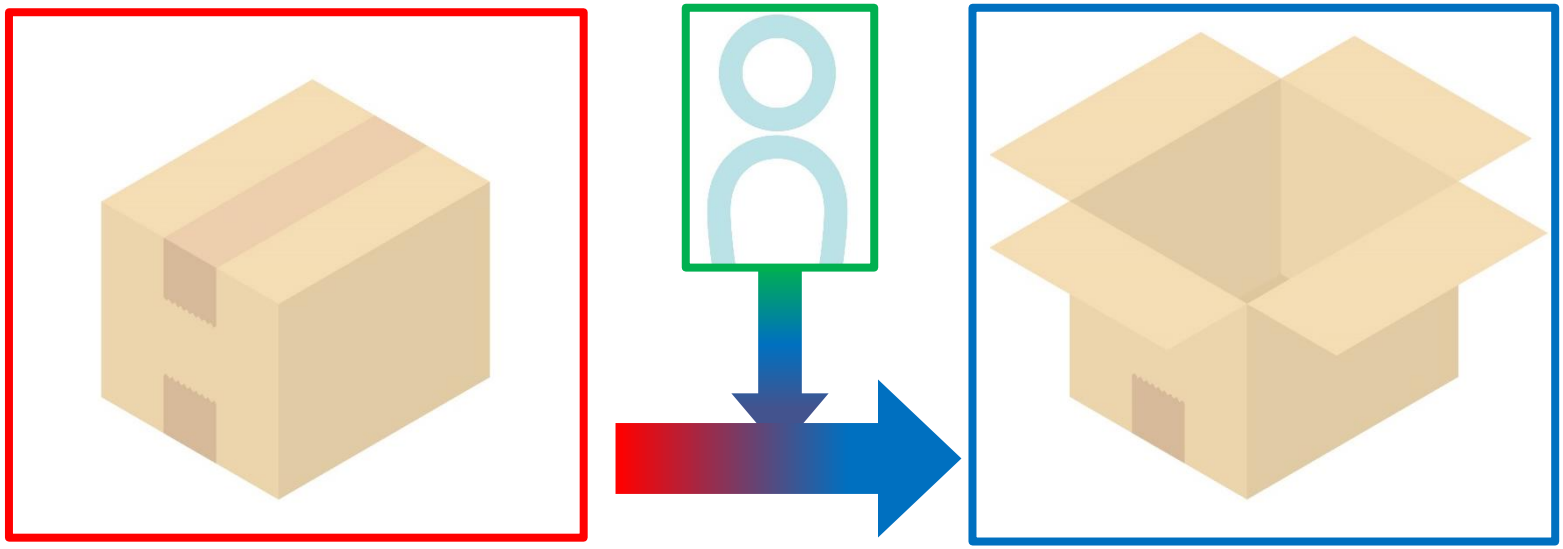
- ➢ 완벽한 묘사는 불가능하기에, "허용"이 존재함
- ➢ 하지만 묘사할 수 있는 것은 묘사되어야 함
- ➢ **상자는 기본적으로 절대 스스로 열리지 않음**
- ➢ **상자는 스스로 동작할 수 없고, 외부에서 동작시킴**



다른 객체간 상호작용

❖ **대상 객체를 이용하는 다른 객체가 준비되어 사용함**

- 다른 객체는 스스로 동작할 수 있도록 묘사된 객체
- 다른 객체는 대상 객체의 필드 및 메서드를 이용함
- **성립하도록 상자 클래스가 정의되어 있어야 함**
- **"상자" 객체와 "상자를 운용하는" 객체가 있게 됨**
- **상자는 사람이 열고, 닫고, 이름을 붙일 수 있음**



다른 객체간 상호작용

다른 객체간 상호작용 : 주체와 대상이 있으니 주의

❖ **Sample2가 주체이며, Sample1이 대상인 경우**

```
public class Sample1 {
    private int num;
    public void method() { ... }
}
public class Sample2 { ......
    public void method1(Sample1[] arr) { ... }
    public void method2(Sample1 target) { ... }
}
```

❖ **클래스의 순서 :** 대상 클래스는 주체 클래스보다 **위에** 정의함

❖ **대상이 자료구조일 수 있으나 내부의 객체들이 중요함**

자료구조는 **객체를 효율적으로 관리하기 위한 껍데기**에 불과함

❖ **단, 객체 클래스의 접근제어 지시자에 주의할 것**

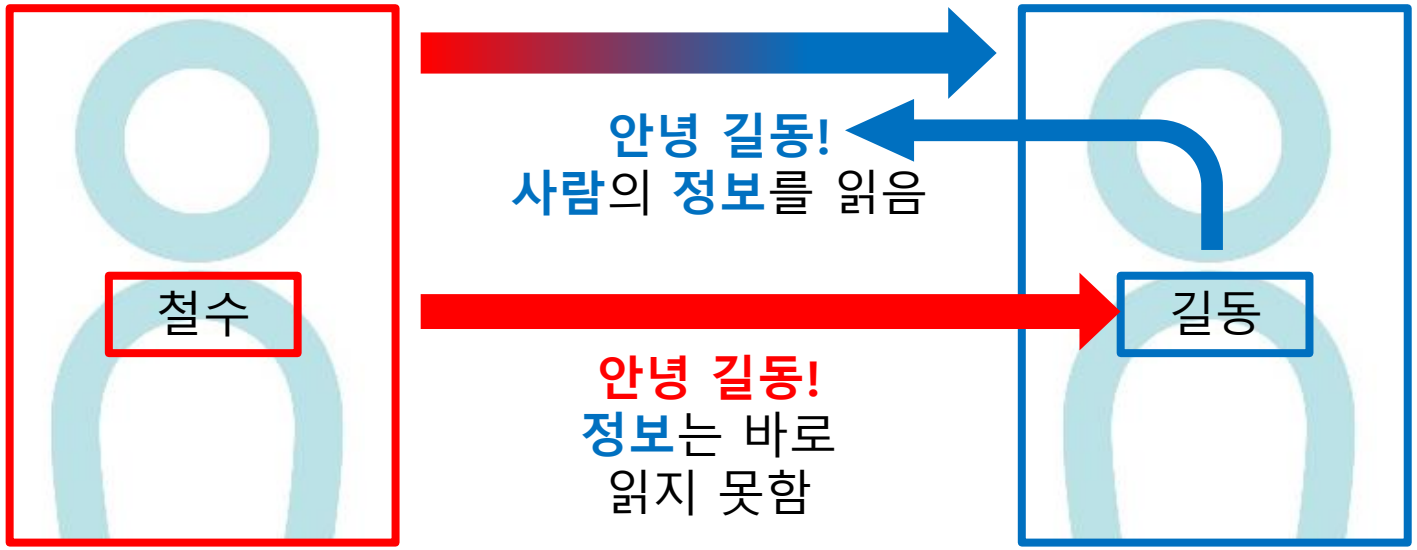
자료구조는 대부분 **public**이지만 정의한 클래스는 **그렇지 않음**

같은 객체간 상호작용

같은 객체간 상호작용

❖ **객체의 상호작용은 같은 종류끼리도 발생할 수 있음**

- **사람**과 **사람**끼리는 서로 인사할 수 있음
- **사람**이 **사람의 이름**에 대하여 인사하지 않음
- 인사할 대상이 사람이라면 이름을 알 수 있음
- 사람끼리는 그 사람의 특징을 알 수 있음



같은 객체간 상호작용

같은 객체간 상호작용 : 주체와 대상을 구별해야 함

```
public class Tank {
    private int hp; private int atk;
    public void attack1(Tank that) {
        that.hp -= this.atk;
    }
    public void attack2(Tank that) {
        this.hp -= that.atk;
    }
}
```

❖ **올바른 상호작용 : this는 that에게 상호작용을 능동적으로 함**

이 때 매개변수명은 구별하기 쉽도록 설정하는 것이 좋음

❖ **잘못된 상호작용 : this는 that에게 상호작용을 당하지 않음**

행동을 할 수 있는 클래스는 행동을 해야 하며, 받아오지 않음