Day11

Day9 5/26 리뷰&실습

Day10

- 리뷰 실습: Prime number class
- This p170
- 유효범위 p191

- 정보은닉 p162
  - 접근 제한자 (한정자, Modifier)
  - https://wikidocs.net/232
- 상속
  - https://opentutorials.org/course/1223/6060
  - https://wikidocs.net/280
- 객체지향 언어 특성
  - 추상화 https://codevang.tistory.com/78
  - 캡슐화 https://codevang.tistory.com/79
  - 상속 https://codevang.tistory.com/80
  - 다형성 https://codevang.tistory.com/81

Constructor, this, this() 정리

생성자(Constructor)

- 생성자의 정의

✓생성자의 이름은 클래스와 같아야 한다.
✓생성자의 리턴값이 없다. (하지만 void를 쓰지 않는다)

```
접근자 클래스이름 ( 파라미터 ) {

    // 인스턴스 생성시 수행할 코드
    // 주로 인스턴스 변수의 초기화 코드

}
```

```
public  class Goods {
    public Goods() {
        // 초기화 코드
    }

    public Goods( String name, int price ) {
        // 초기화 코드
    }

}
```

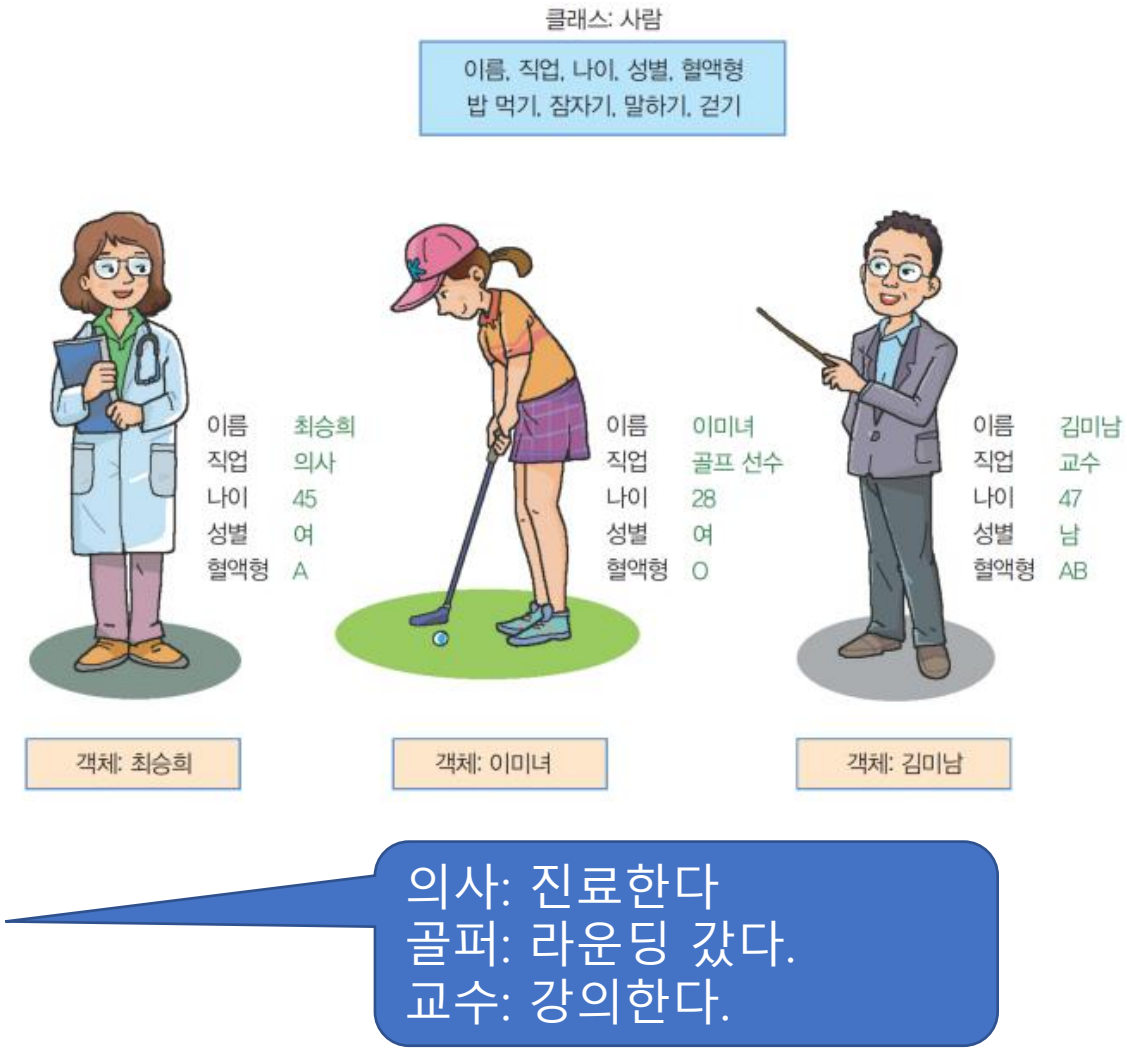
실습: 클래스 생성자, this, this() 이용

- 사람을 표현하는 Person 클래스
  - this, this() 생성자 사용.

Person

Person() // name=이름없음
Person(name, age)
Person(name, age, gender, blood, job)

name: String job: String age: int gender: String blood: String
play();



Constructor, this, this() 정리

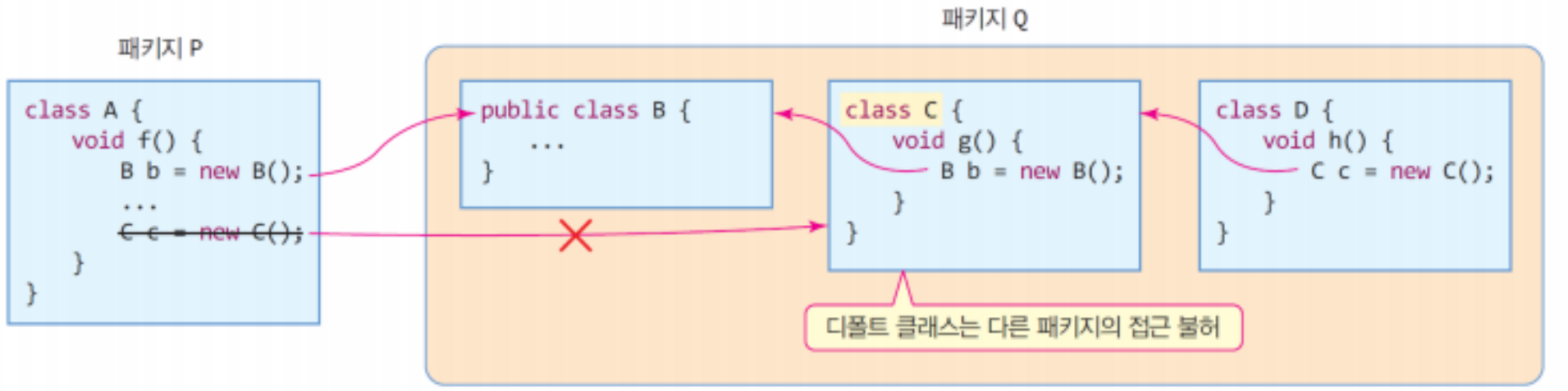
- 생성자의 특징
  - 생성자는 **메소드**
  - 생성자 이름은 클래스 이름과 반드시 동일
  - **생성자 오버로딩** == 메서드 오버로딩
  - **생성자는 리턴 타입을 지정할 수 없음**
  - 생성자의 목적은 객체 초기화
  - 생성자는 객체가 생성될 때 반드시 호출됨.
  - 생성자 없으면 컴파일러가 자동으로 기본 생성자 삽입
  - 생성자는 new를 통해 객체를 생성할 때, 객체당 한 번 호출
- this의 필요성
  - 객체의 멤버 변수와 메소드 변수의 이름이 같은 경우
  - 다른 메소드 호출 시 객체 자신의 레퍼런스를 전달할 때
  - 메소드가 객체 자신의 레퍼런스를 반환할 때
- this()
  - 클래스 내의 다른 생성자 호출
  - 생성자 내에서만 사용 가능
  - 반드시 생성자 코드의 제일 처음에 수행

접근제한자

- 클래스 접근지정
  - 다른 클래스에서 사용하도록 허용할 지 지정
  - public 클래스
    - 다른 모든 클래스에게 접근 허용
  - 디폴트 클래스(접근지정자 생략)
    - package-private라고도 함
    - 같은 패키지의 클래스에만 접근 허용

```
public class World { // public 클래스
.....
}
```

```
class Local { // 디폴트 클래스
.....
}
```



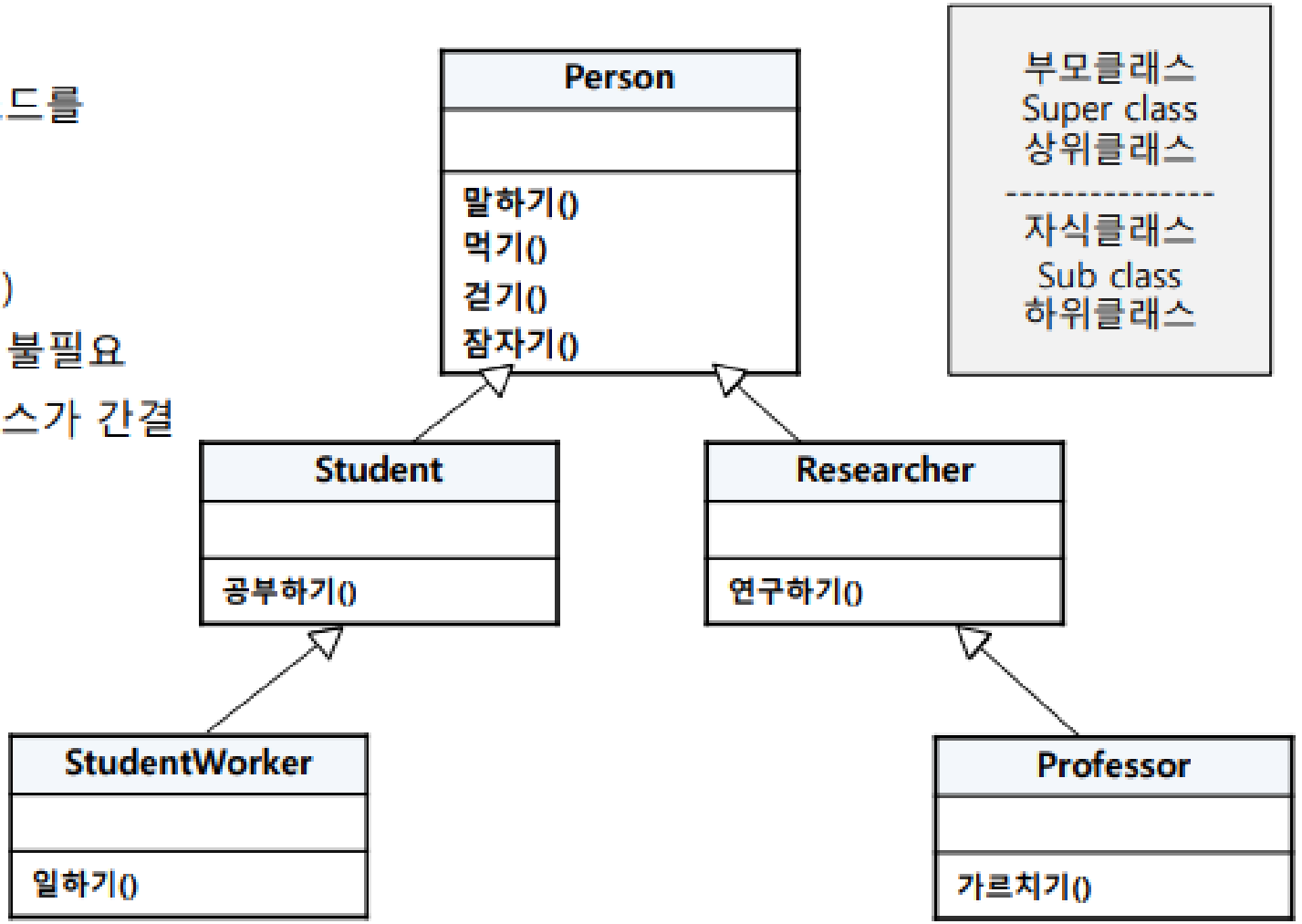
public 클래스와 디폴트 클래스의 접근 사례

상속, super

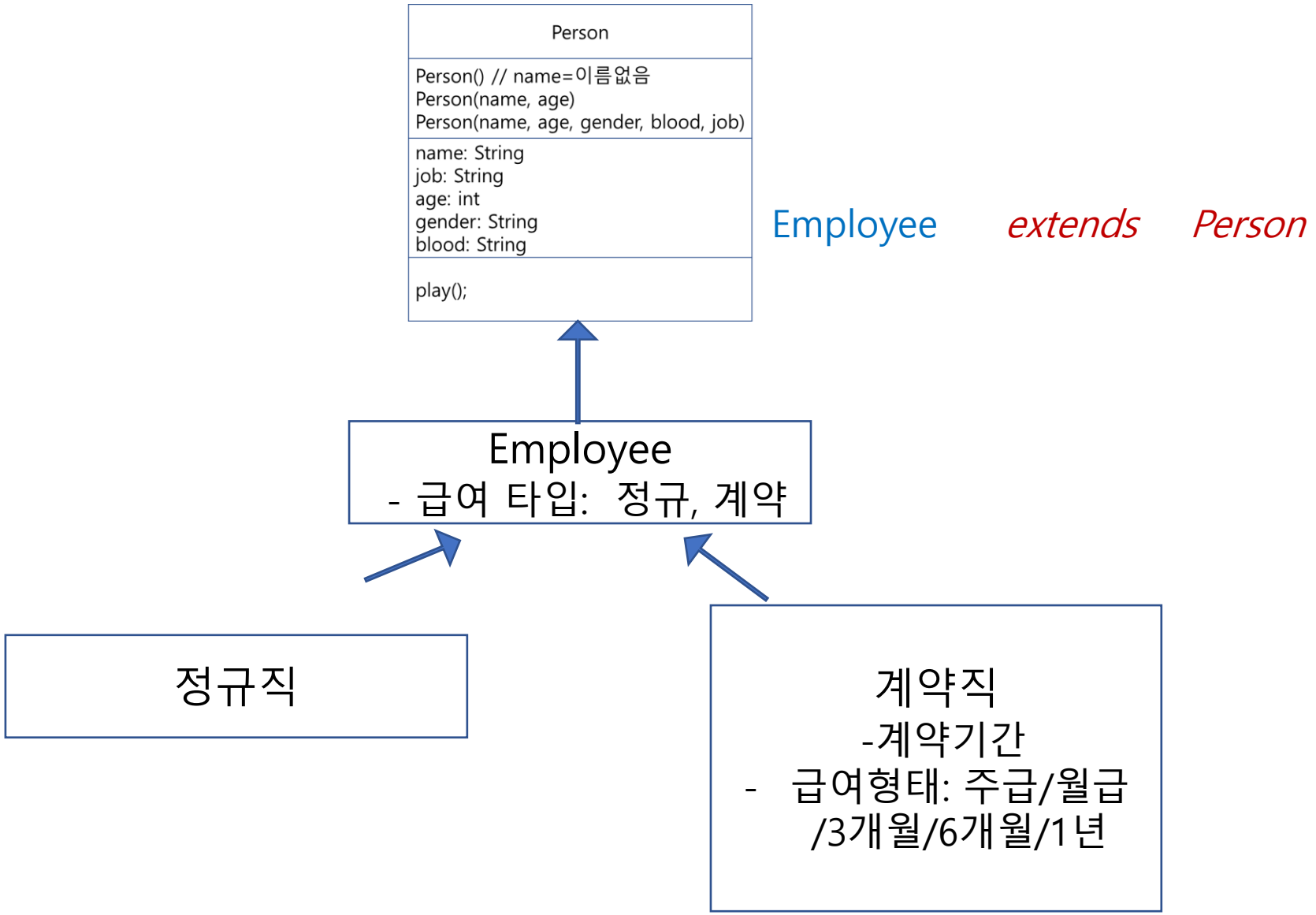
■ **상속**

부모 클래스에 정의된 필드와 메소드를
자식 클래스가 물려 받는것

- 왜 상속을 하는가?(상속의 필요성)
  - 클래스 사이의 맴버 중복선언 불필요
  - 필드, 메소드 재사용으로 클래스가 간결
  - 클래스간 계층적 분류 및 관리

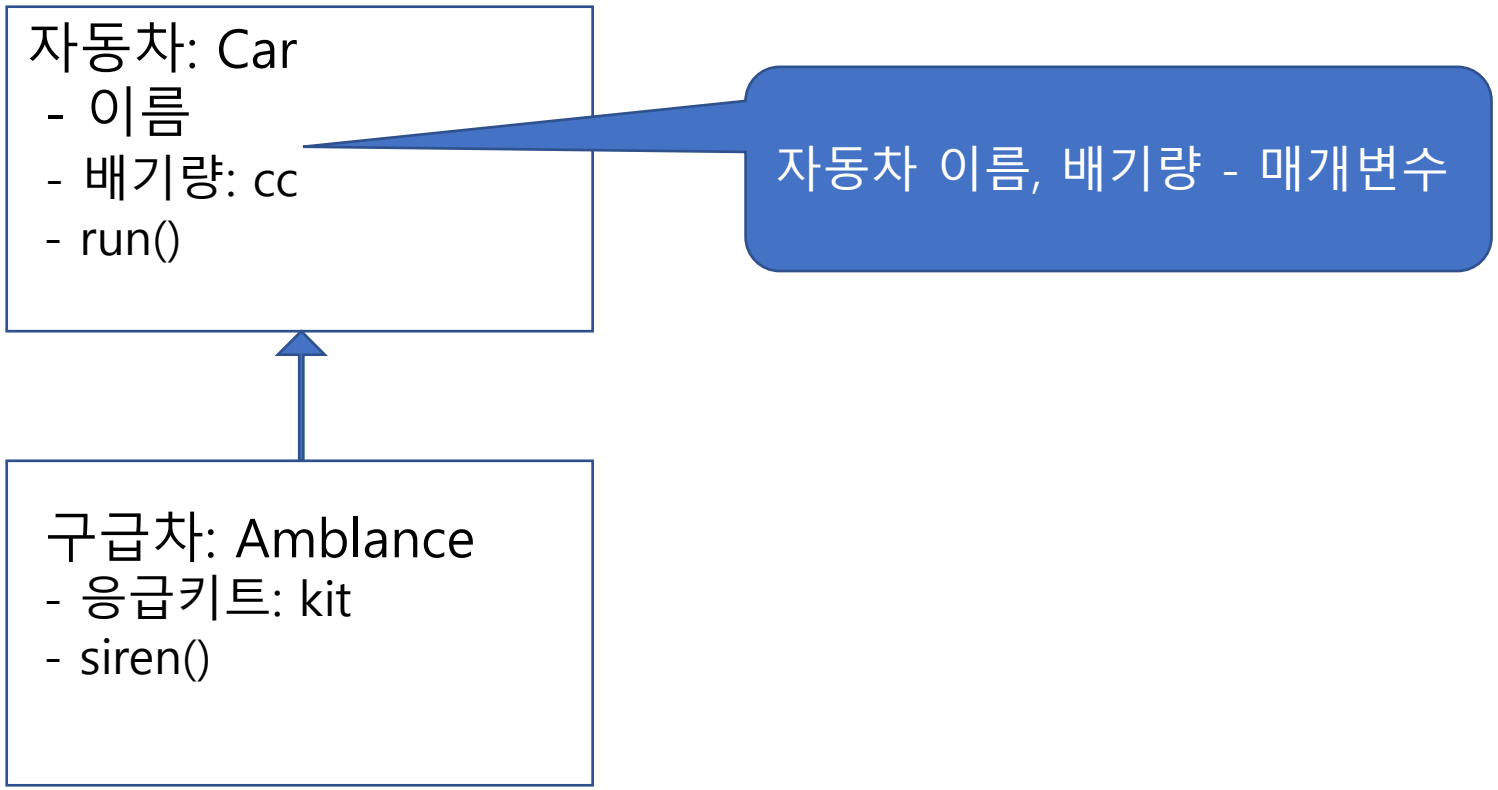


상속 extends


Person
Person() // name=이름없음
Person(name, age)
Person(name, age, gender, blood, job)
name: String
job: String
age: int
gender: String
blood: String
play();
Employee extends Person
Employee
- 급여 타입: 정규, 계약
정규직
계약직
-계약기간
- 급여형태: 주급/월급
/3개월/6개월/1년

상속 실습: 자동차

- 자동차는 run() 으로 달린다
- 구급차는 사이렌을 울리고
  - 응급키트가 배치되어 있다.



객체 캐스팅

```
public class CarMain {

    public static void main(String[] args) {
        Car car = new Car("자동차", 1000);
        car.run();

        CarAmbulance am = new CarAmbulance();
        am.run();
        am.siren();

        Car car2 = new CarAmbulance();
        car2.siren();

    }
```

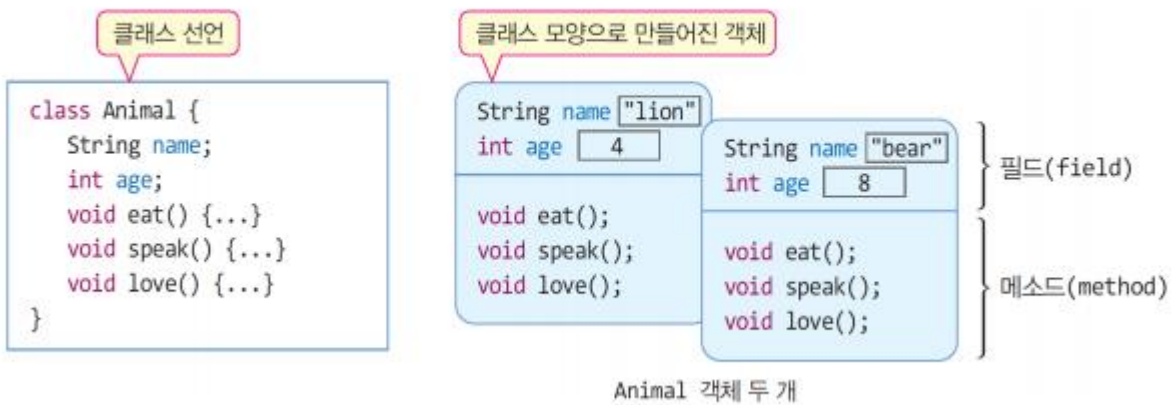
Day11

Day11

- 메서드 오버라이딩
  - 가상메서드
  - 어노테이션
- 다형성
- 업/다운캐스팅
- instanceOf
- 배열과 복사 P211
- 추상클래스 abstract

자바의 캡슐화

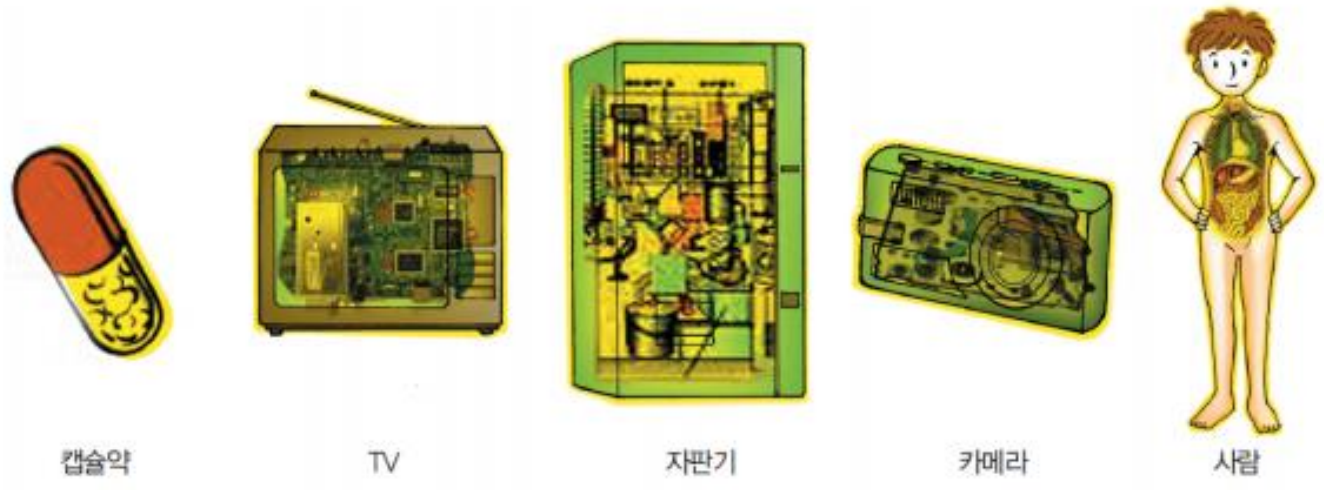
- 클래스(class): 객체 모양을 선언한 틀(캡슐화)
  - 메소드(멤버 함수)와 필드(멤버 변수)는 모두 클래스 내에 구현
- 객체
  - 클래스의 모양대로 생성된 실체(instance)
  - 객체 내 데이터에 대한 보호, 외부 접근 제한
    - 객체 외부에서는 비공개 멤버(필드, 메소드)에 직접 접근할 수 없음
    - 객체 외부에서는 공개된 메소드를 통해 비공개 멤버 접근



Animal 객체 두 개

객체 지향 특성 : 캡슐화

- 캡슐화 : 객체를 캡슐로 싸서 내부를 볼 수 없게 하는 것
  - 객체의 본질적인 특징
    - 외부의 접근으로부터 객체 보호



메소드 오버로딩

- 메소드 오버로딩(Overloading)
  - 이름이 같은 메소드 작성
    - 매개변수의 개수나 타입이 서로 다르고
    - 이름이 동일한 메소드들
  - 리턴 타입은 오버로딩과 관련 없음

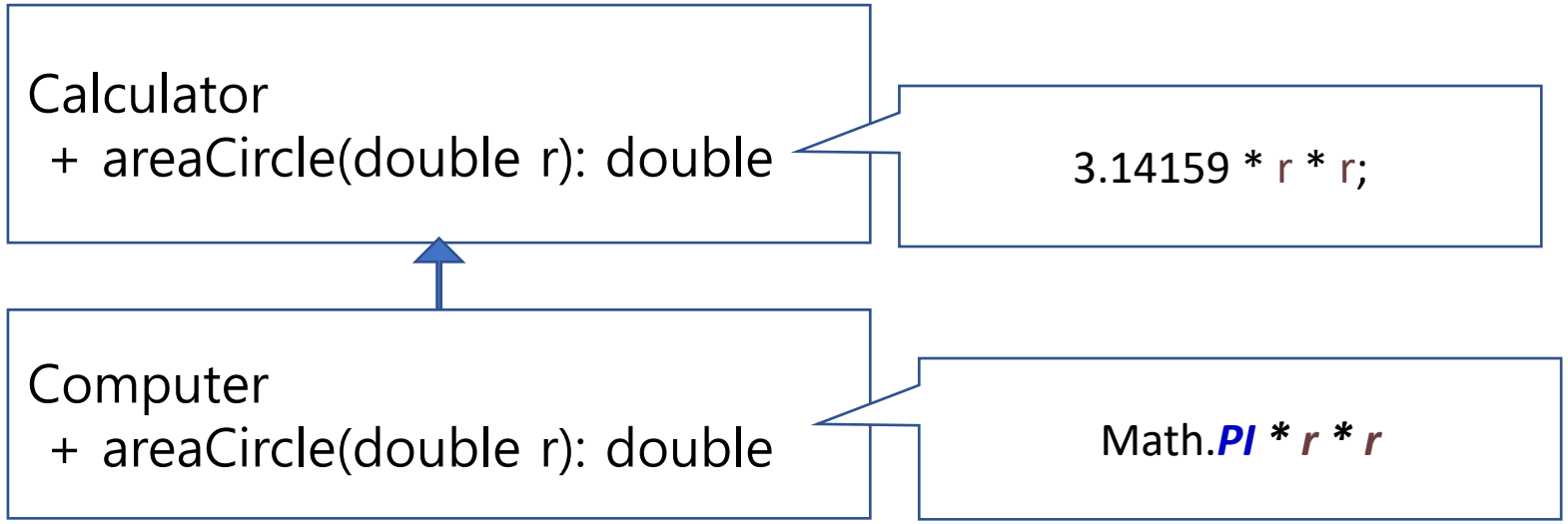
```
// 메소드 오버로딩이 성공한 사례

class MethodOverloading {
   public int getSum(int i, int j) {
      return i + j;
   }
   public int getSum(int i, int j, int k) {
      return i + j + k;
   }
}
```

```
// 메소드 오버로딩이 실패한 사례

class MethodOverloadingFail {
   public int getSum(int i, int j) {
      return i + j;
   }
   public double getSum(int i, int j) {
      return (double)(i + j);
   }
}
```

두 개의 getSum() 메소드는 매개변수의 개수, 타입이 모두 같기 때문에 메소드 오버로딩 실패

실습: 메서드 오러라이딩

- Calcurator 를 상속받는 Computer

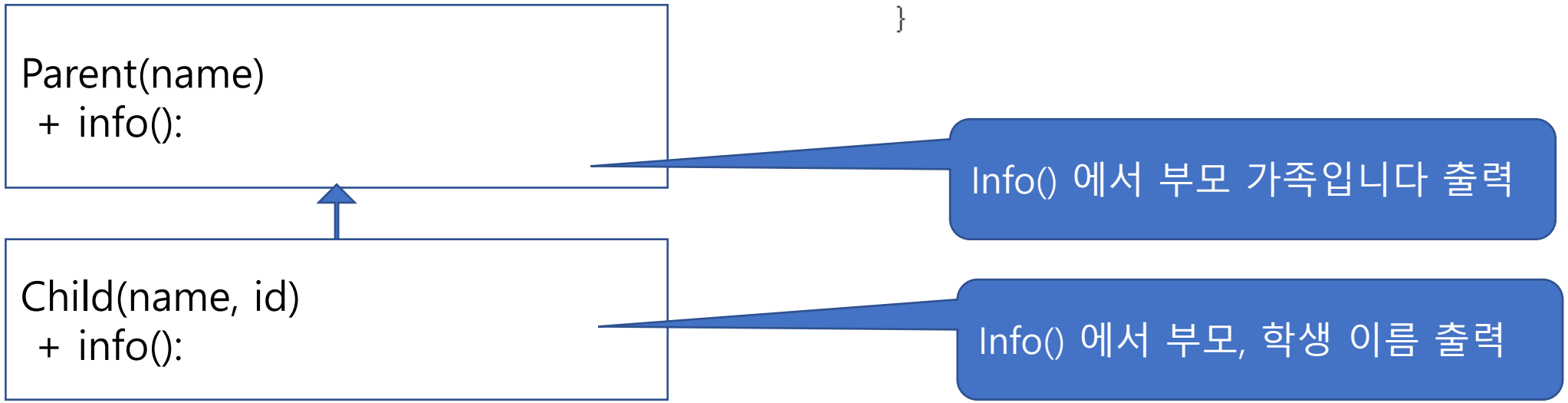



```
Main:
int r = 10;
Calculator calculator = new Calculator();
System.out.println("원면적 : " + calculator.areaCircle(r));
System.out.println();
Computer computer = new Computer();
System.out.println("원면적 : " + computer.areaCircle(r));
```

실습: 상속-super

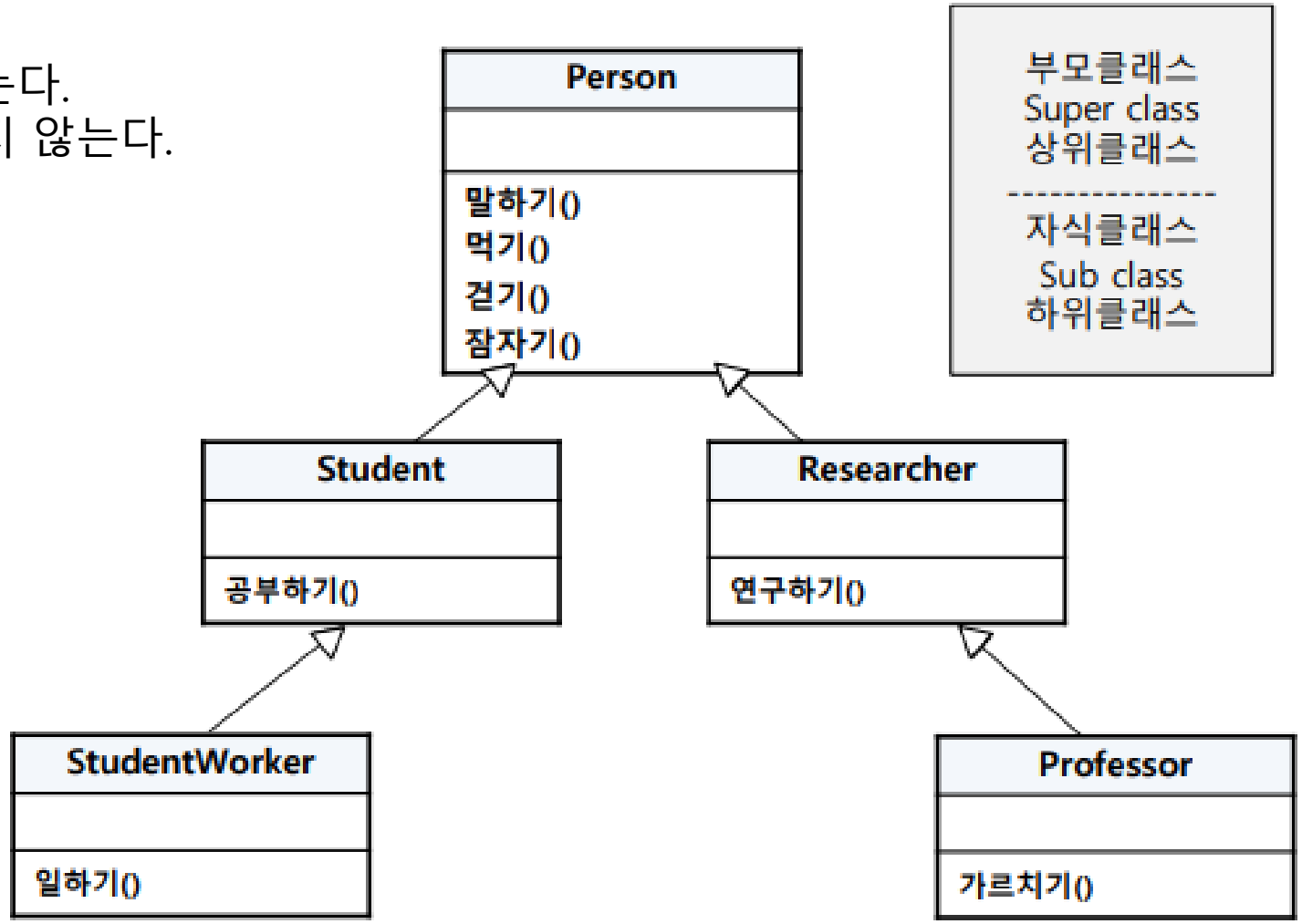
- Parent 클래스를 상속받는
- Child 클래스는 이름, 학번으로 생성한다.
  - Child class 와 info overriding

```java
public class Parent {
    public String name;

    public Parent(String name) {
        this.name = name;
    }

    public void info() {
        System.out.println(name + " 가족입니다");
    }
}
```

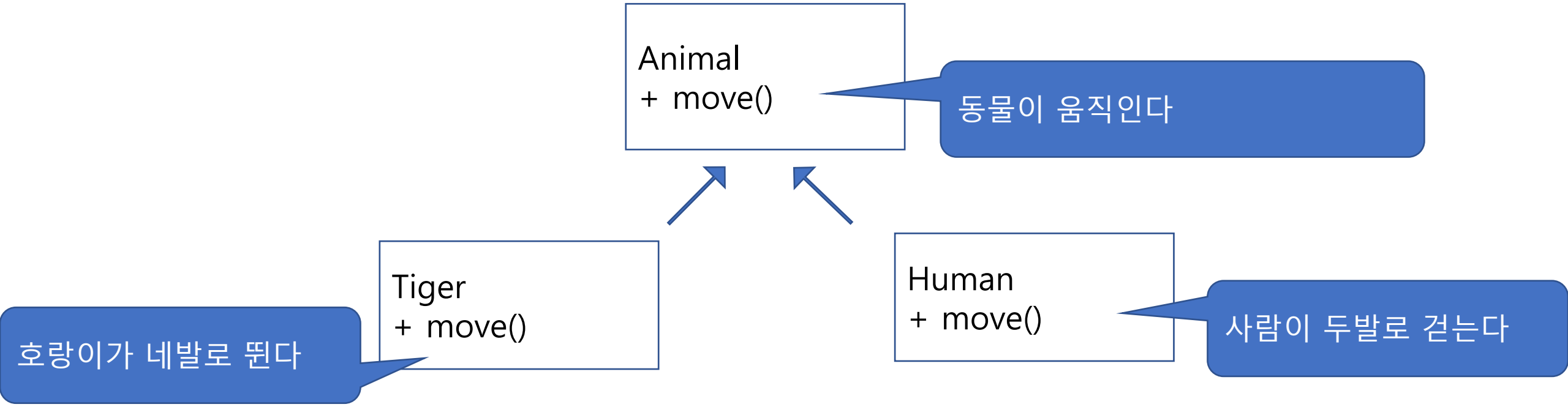


다형성

자바상속의 특징
– 자바에서는 다중 상속을 지원하지 않는다.
– 자바에서는 상속의 횟수에 제한을 두지 않는다.
– 자바에서 계측구조의 최상위에 있는
클래스는 java.lang.Object 이다.



실습: 다형성


Animal
+ move()
동물이 움직인다
Tiger
+ move()
호랑이가 네발로 뛴다
Human
+ move()
사람이 두발로 걷는다

상속,다형성,관계 참고:

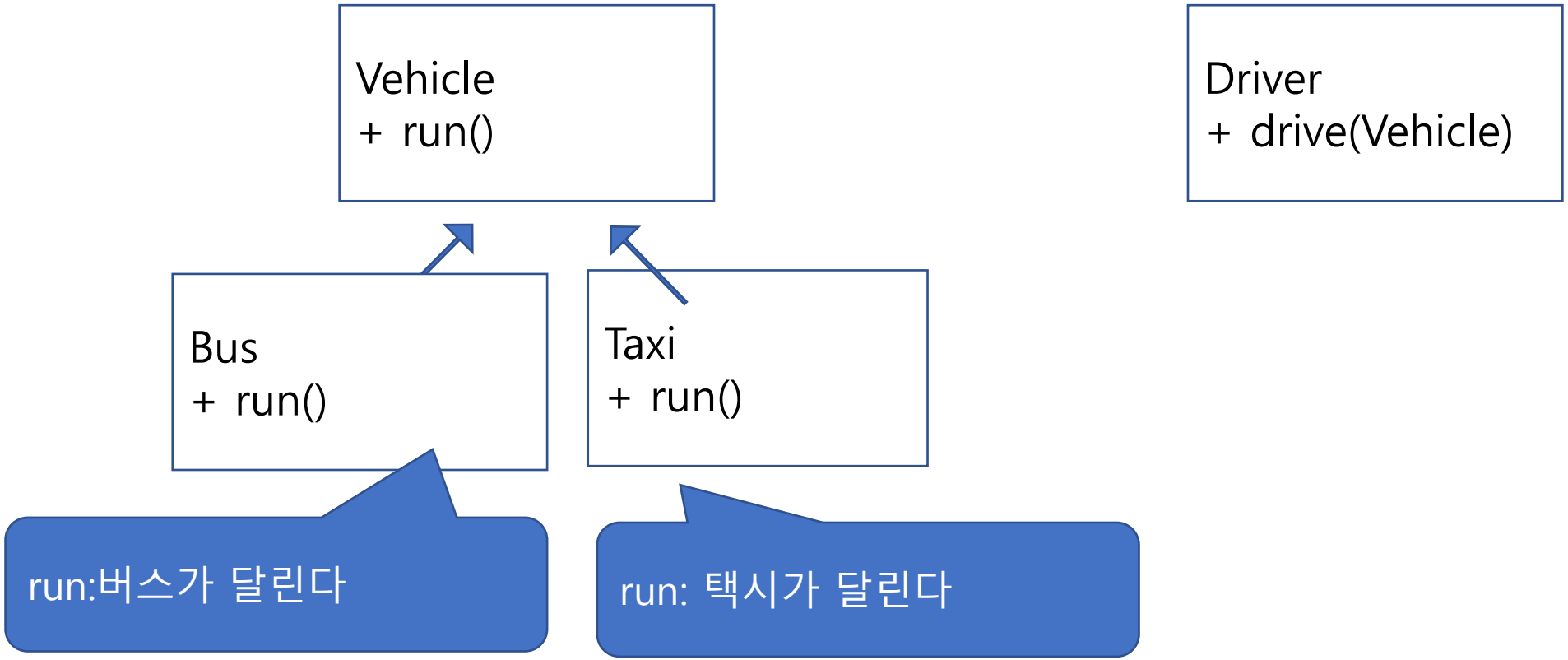
- 다형성
  - 오버라이드 https://opentutorials.org/course/1223/6090
  - 다형성 https://wikidocs.net/269

- 참고
  - 자바를다루는기술_VOL1_5장-객체지향1 pdf
  - 자바를다루는기술_VOL1_5장-객체지향2 pdf

연습문제 P278, 279

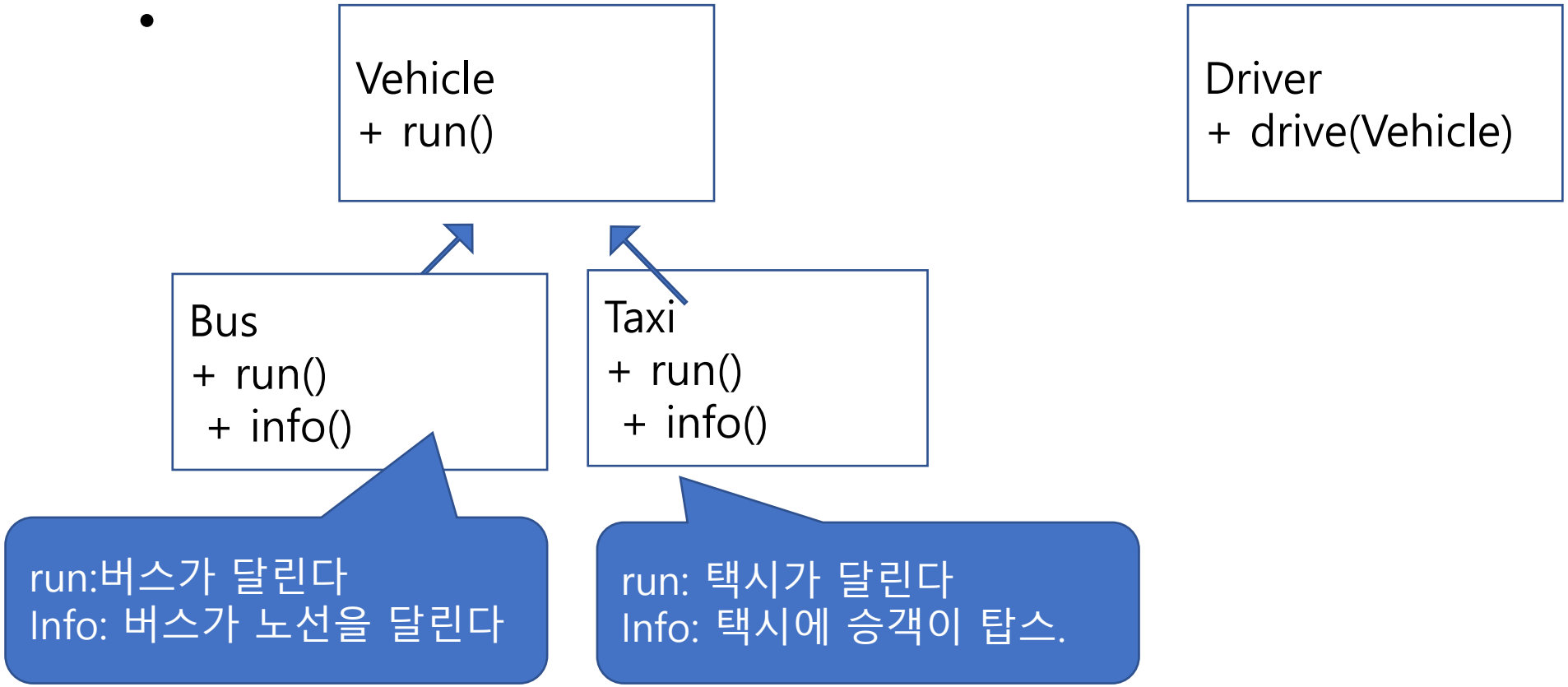
실습:

- Main class 에서 Bus, Tax를 Driver가 drive 하게 실현
  -



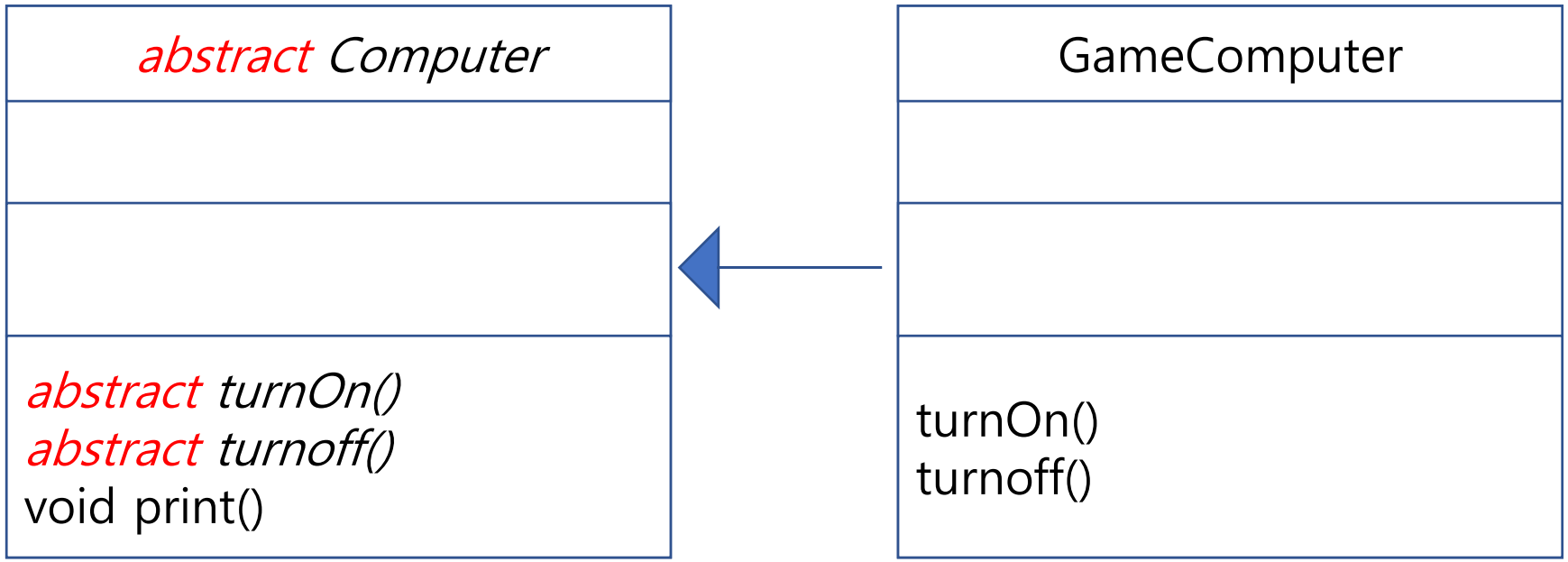
실습:

- Main class 에서 ArrayList 를 사용해서 5대의 Bus, Taxi에서 info() 를 사용해 보자
  -



추상 클래스/메서드

- P280



인터페이스

- P314

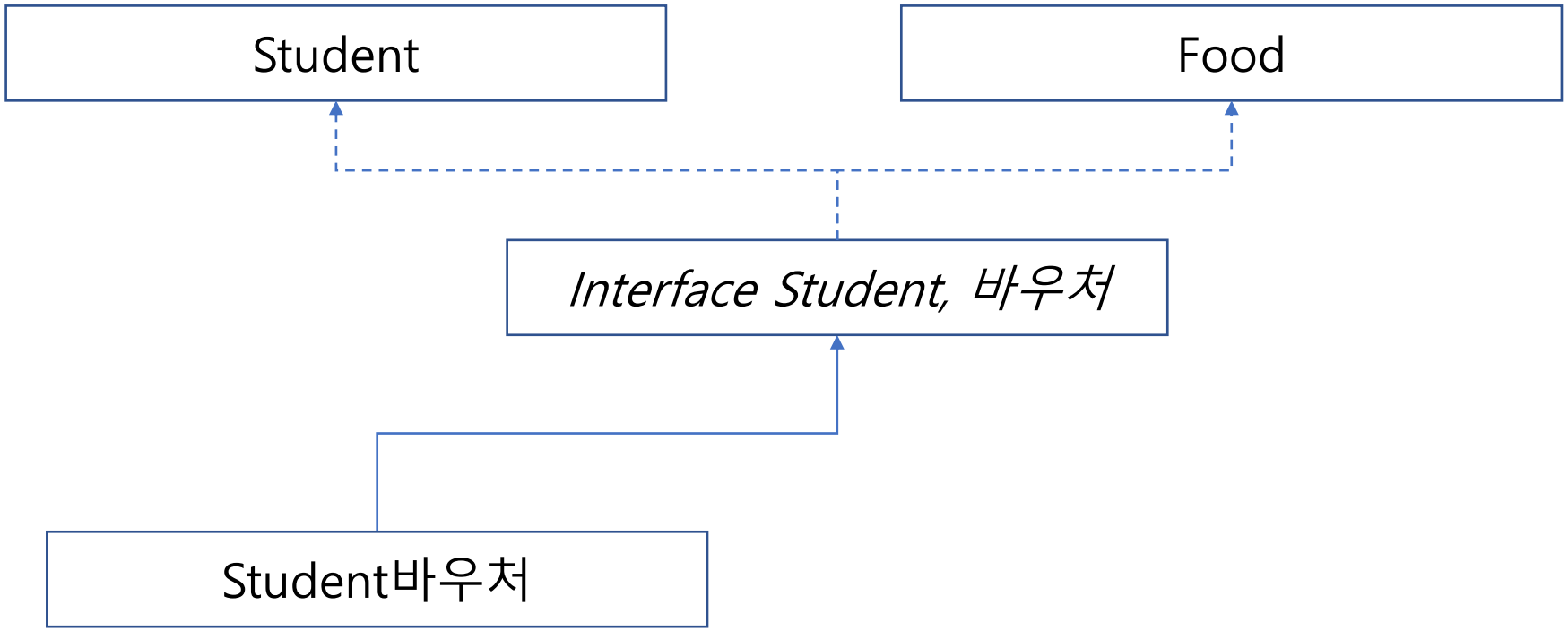

9GAG
NEW DEFINITION OF FUN
A GUIDE TO THE WORLD'S ELECTRICAL OUTLETS
seen on 9GAG.COM


SAMSUNG
SAMSUNG
ZOOM LENS
GALAXY


Anycall

인터페이스

- P314



5/28일 Day12

- Ch11 P351 기본 클래스
  - 컬렉션
  - 내부 클래스
- Ch7 P199 배열/객체복사
- 실습?

Garbage Collection

Garbage Collection

- 객체 소멸을 다루는 JVM 메커니즘

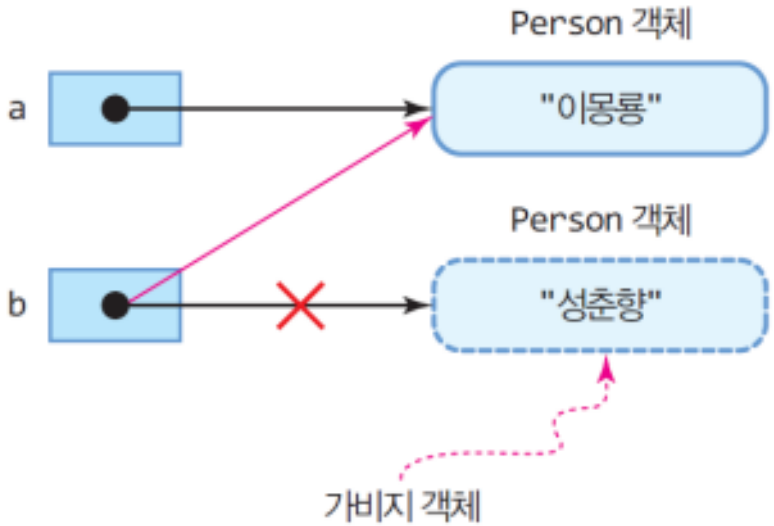
객체의 소멸과 가비지 컬렉션

- 객체 소멸
  - new에 의해 할당된 객체 메모리를 자바 가상 기계의 가용 메모리로 되돌려 주는 행위
- 자바 응용프로그램에서 임의로 객체 소멸할 수 없음
  - 객체 소멸은 자바 가상 기계의 고유한 역할
  - 자바 개발자에게는 매우 다행스러운 기능
    - C/C++에서는 할당받은 객체를 개발자가 되돌려 주어야 함
      - C/C++ 프로그램 작성을 어렵게 만드는 요인
- 가비지
  - 가리키는 레퍼런스가 하나도 없는 객체
    - 누구도 사용할 수 없게 된 메모리
- 가비지 켈렉션
  - 자바 가상 기계의 가비지 컬렉터가 자동으로 가비지 수집 반환

Garbage Collection

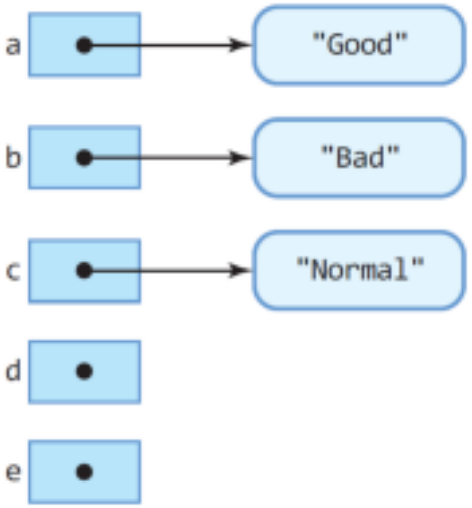
- 소멸 대상

```
Person a, b;
a = new Person("이몽룡");
b = new Person("성춘향");
b = a; // b가 가리키던 객체는 가비지가 됨
```

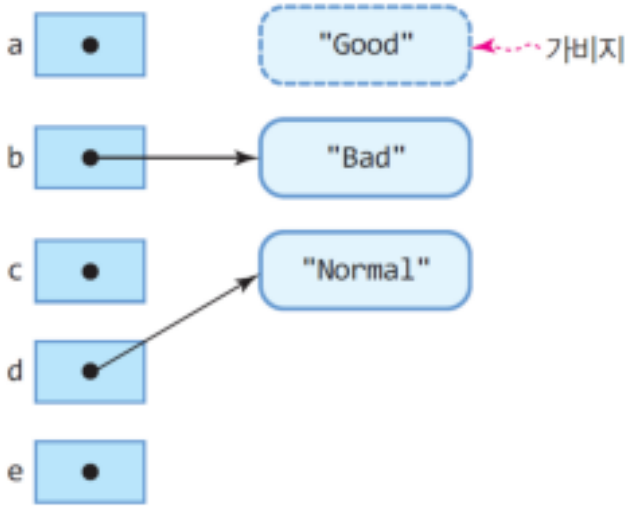


실습: Garbage Collection

```
public class GarbageEx {
    public static void main(String[] args) {
        String a = new String("Good");
        String b = new String("Bad");
        String c = new String("Normal");
        String d, e;
        a = null;
        d = c;
        c = null;
    }
}
```



(a) 초기 객체 생성 시(라인 6까지)



(b) 코드 전체 실행 후

Garbage Collection 정리

- 가비지 컬렉션
  - 자바에서 가비지를 자동 회수하는 과정
    - 가용 메모리로 반환
  - 가비지 컬렉션 스레드에 의해 수행
- 개발자에 의한 강제 가비지 컬렉션
  - System 또는 Runtime 객체의 gc() 메소드 호출

```
System.gc(); // 가비지 컬렉션 작동 요청
```

    - 이 코드는 자바 가상 기계에 강력한 가비지 컬렉션 요청
      - 그러나 자바 가상 기계가 가비지 컬렉션 시점을 전적으로 판단