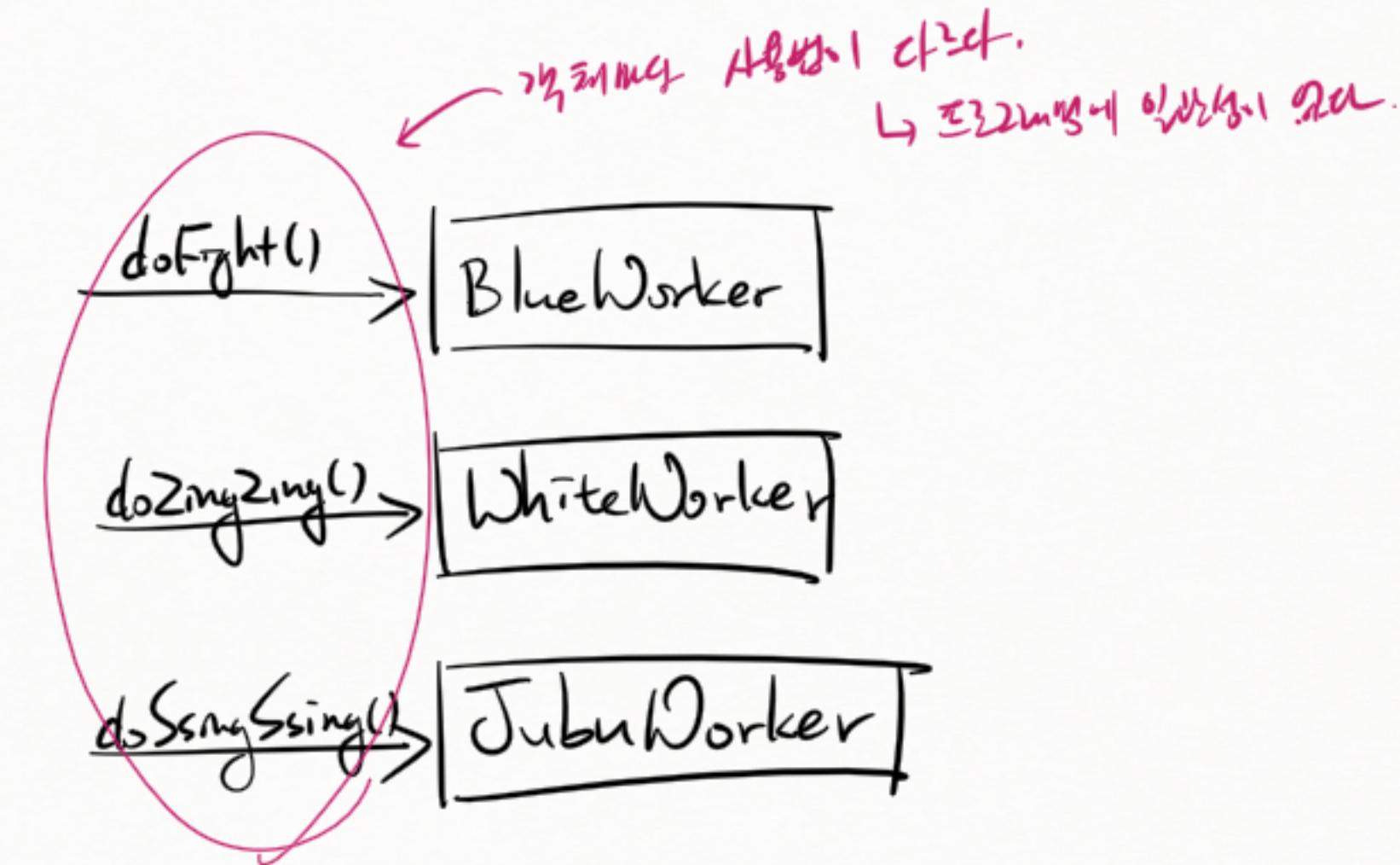
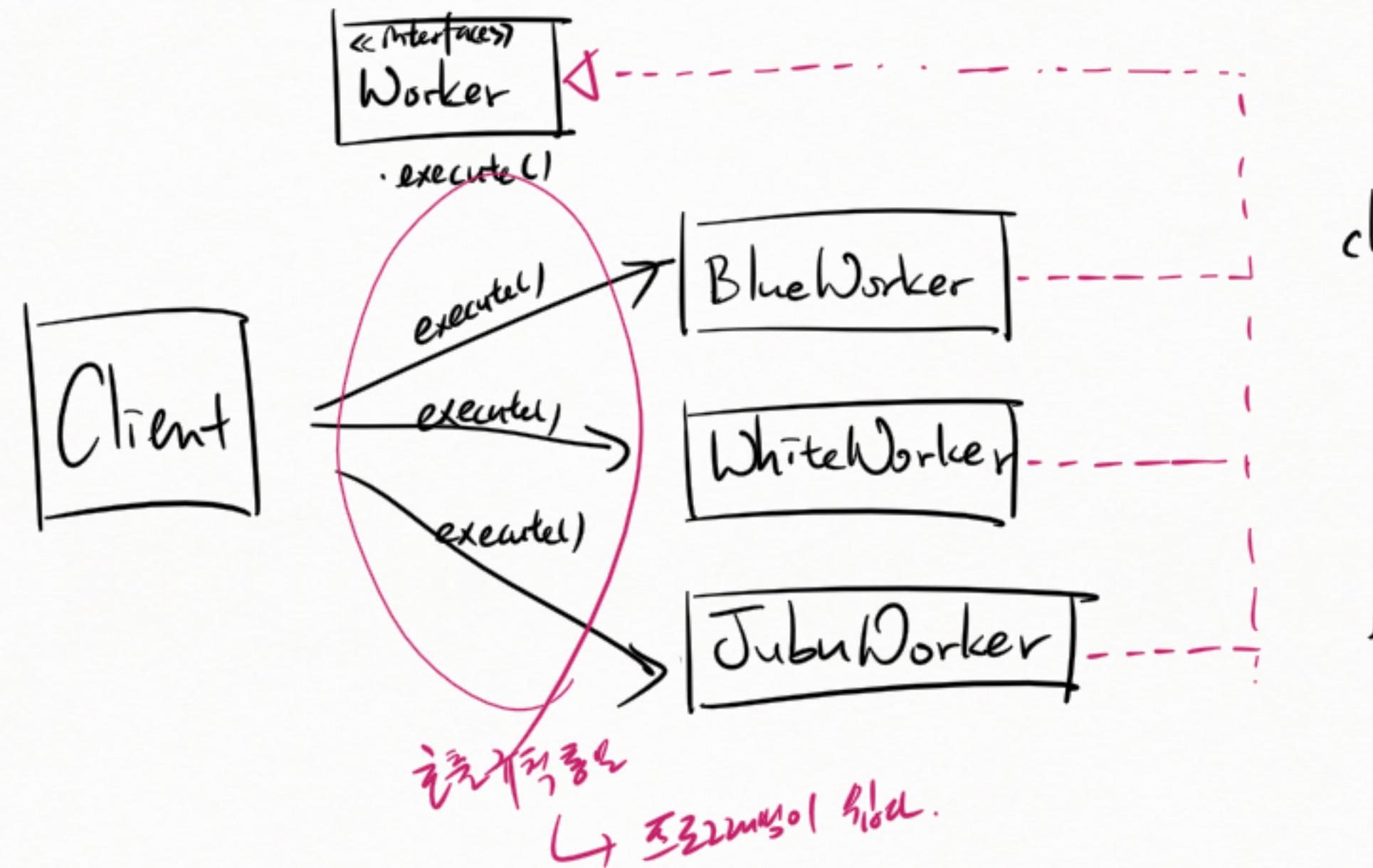


인터페이스 (Interface)

* 인터페이스 사용 전

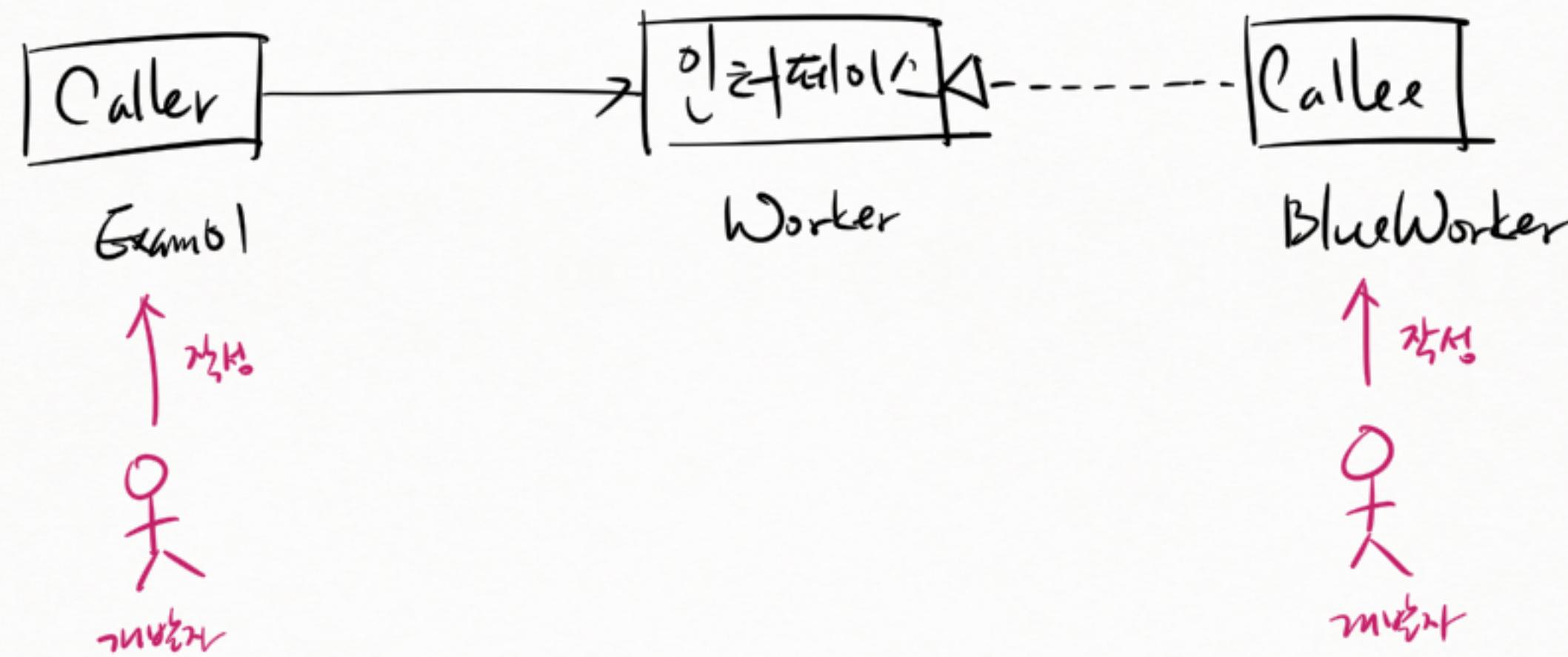


* 인터페이스 사용 후

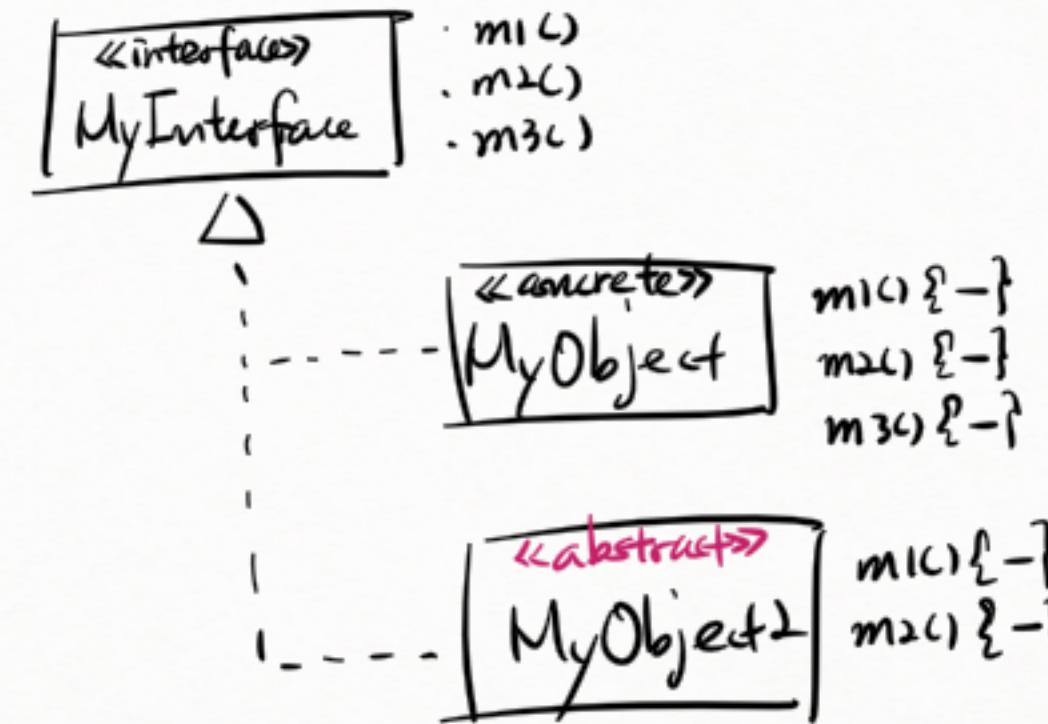


```
class BlueWorker  
implements Worker {  
    ...  
}
```

* 인터페이스 와 caller / callee



* 인터페이스의 구현



MyInterface ref;

ref = new MyObject();

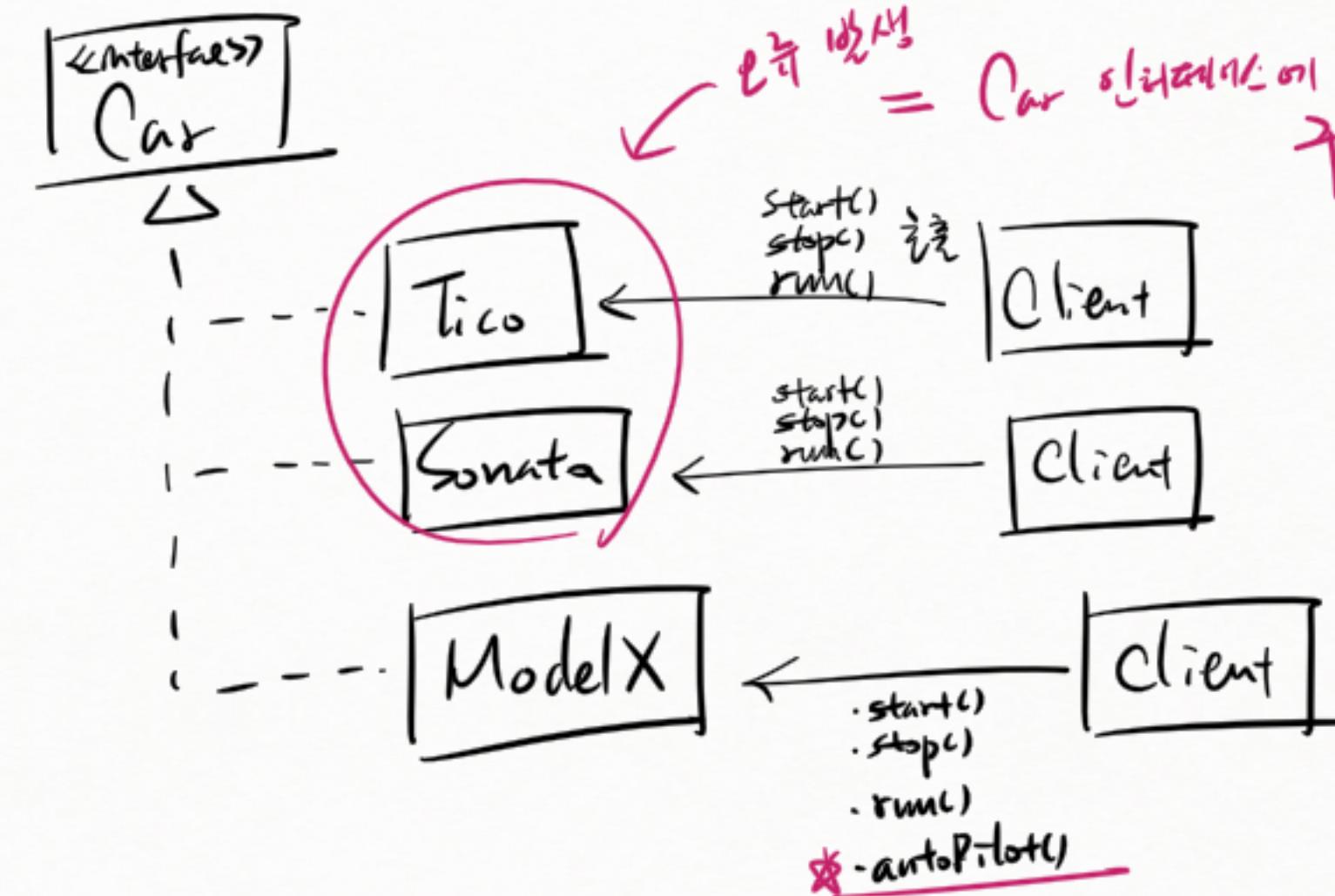


인터페이스를 구현한 클래스의
인스턴스화

* default 메서드 사용

```
interface Car {  
    - start();  
    - stop();  
    - run();  
}
```

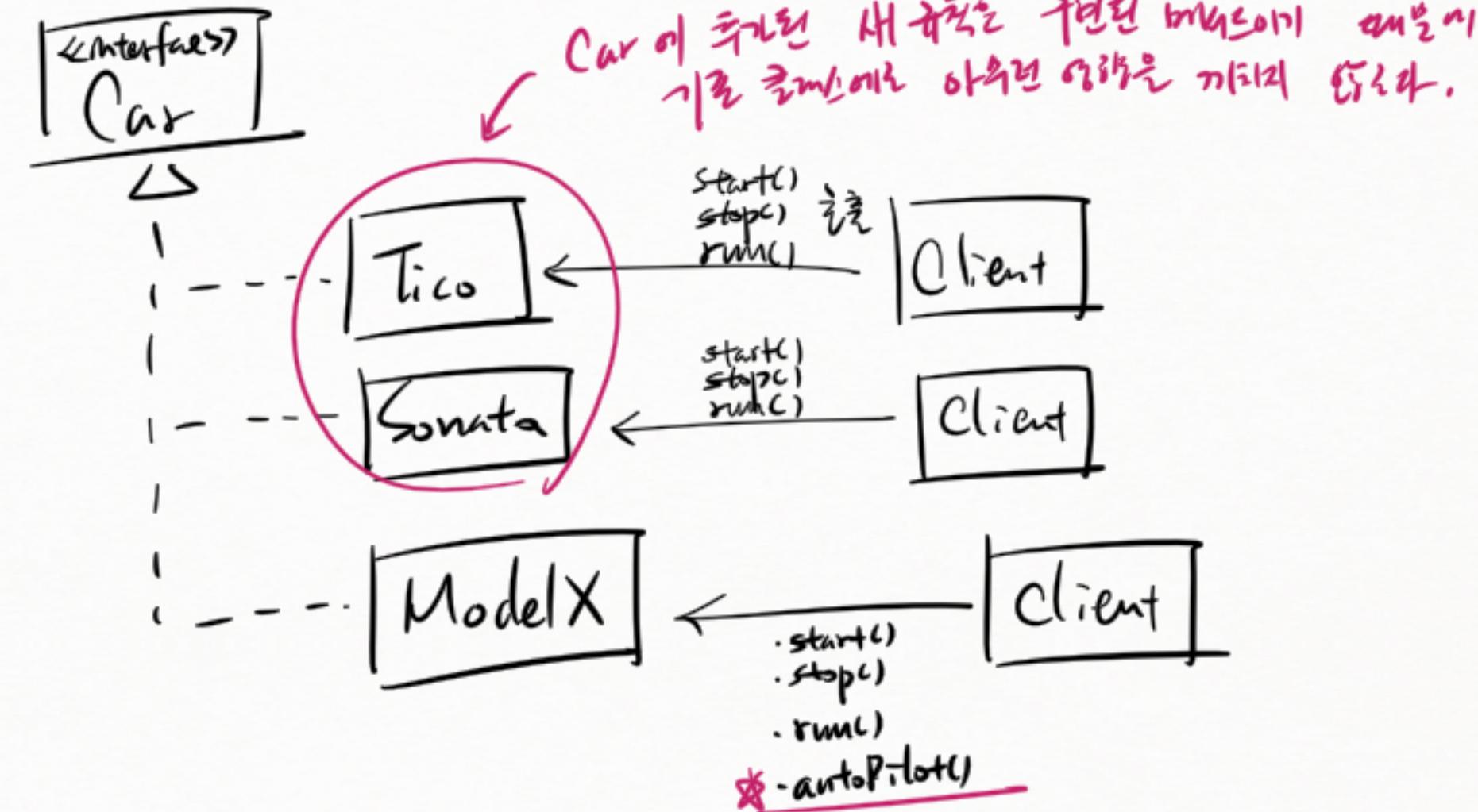
↑
구현 추가
+ autopilot();



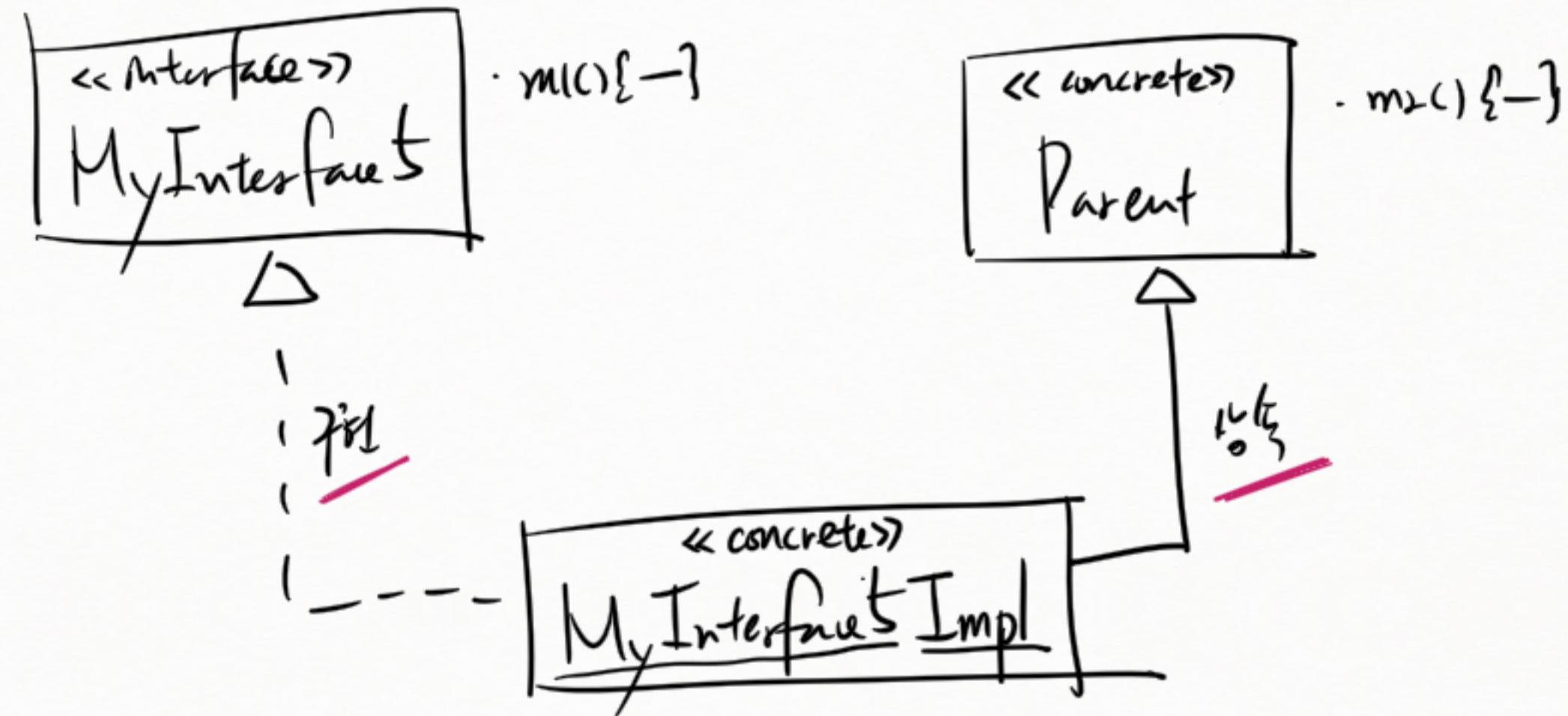
→ = Car 인터페이스에 추가된 새 기능은 구현(정의)되어 있겠지
증명이다.

* default interface 헬퍼

```
interface Car {  
    - start();  
    - stop();  
    - run();  
}  
  
default autopilot() {}
```

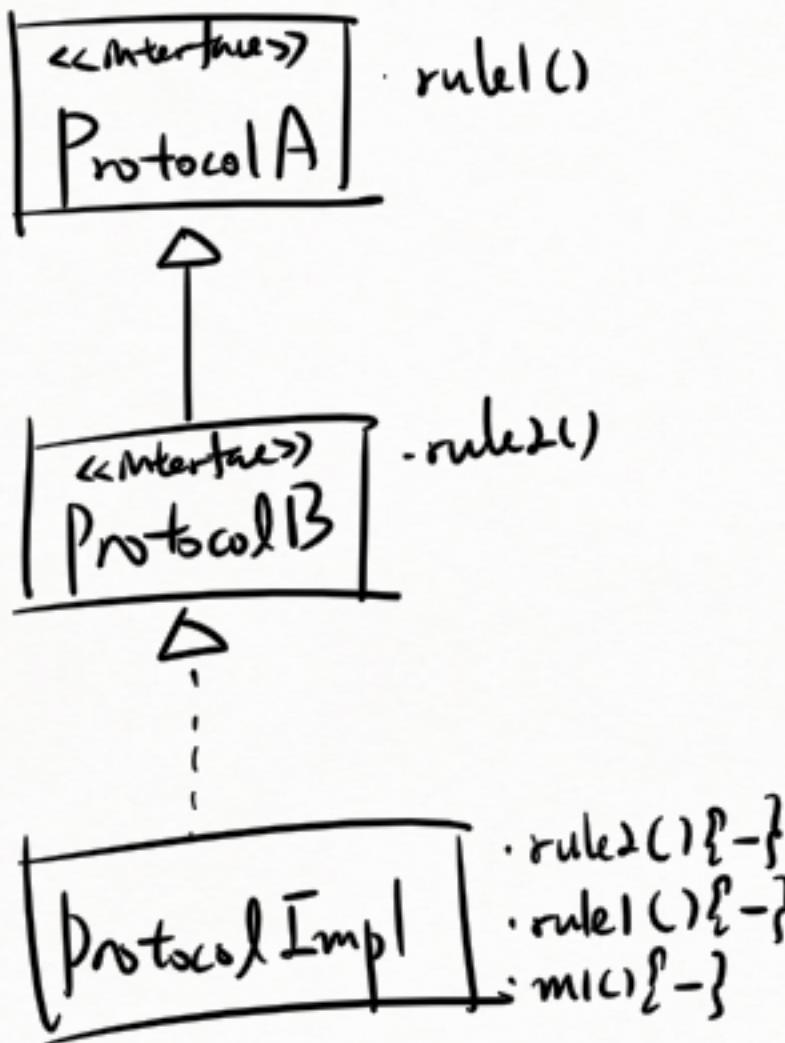


* static b1/b2



* 인터페이스 상속 구현

↳ 인터페이스를 통한 메서드 호출 방식



ProtocolImpl obj = new ProtocolImpl();

obj. m1();
obj. rule1();
obj. rule2();

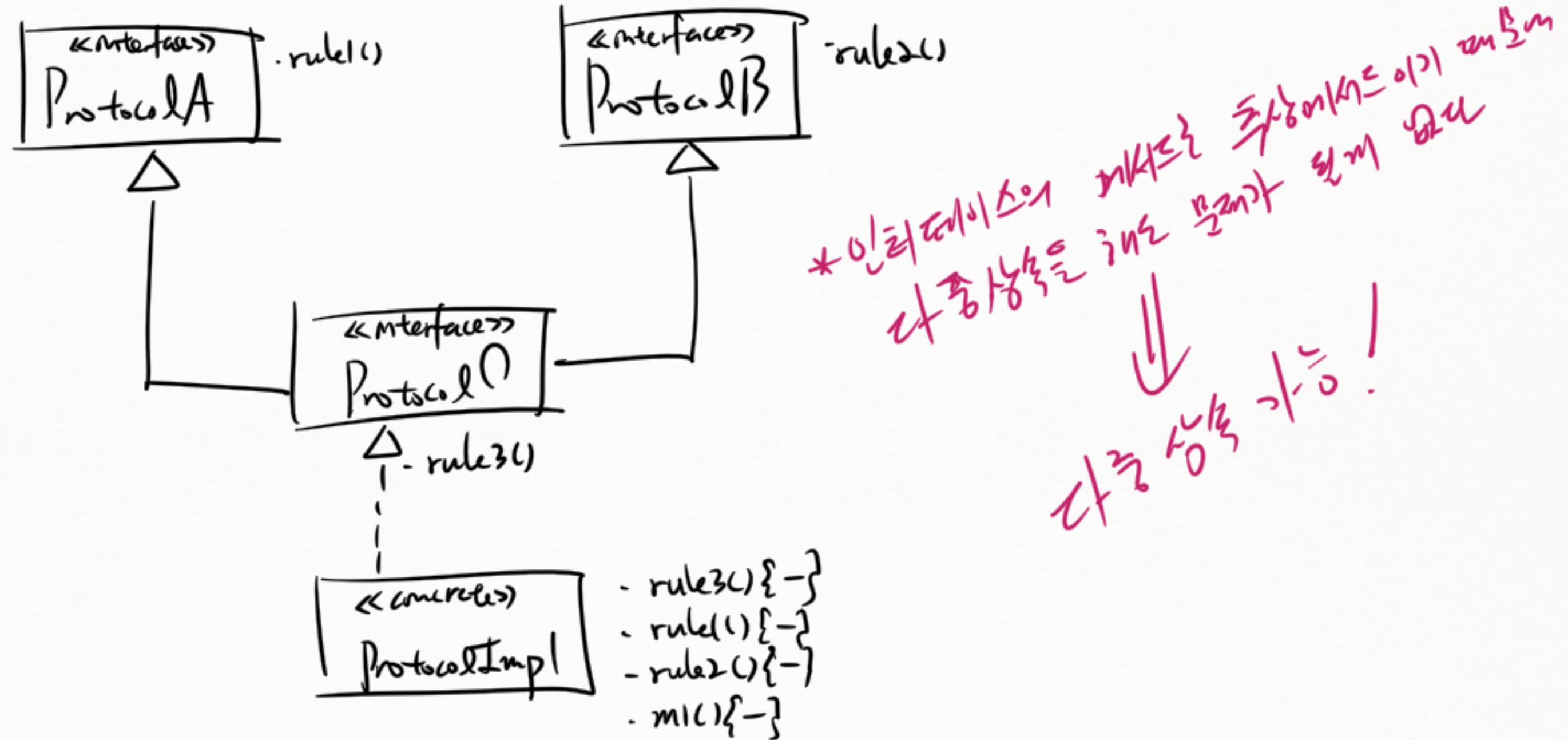
ProtocolB obj2 = obj;

~~obj2. m1();~~
~~obj2. rule2();~~
~~obj2. rule1();~~

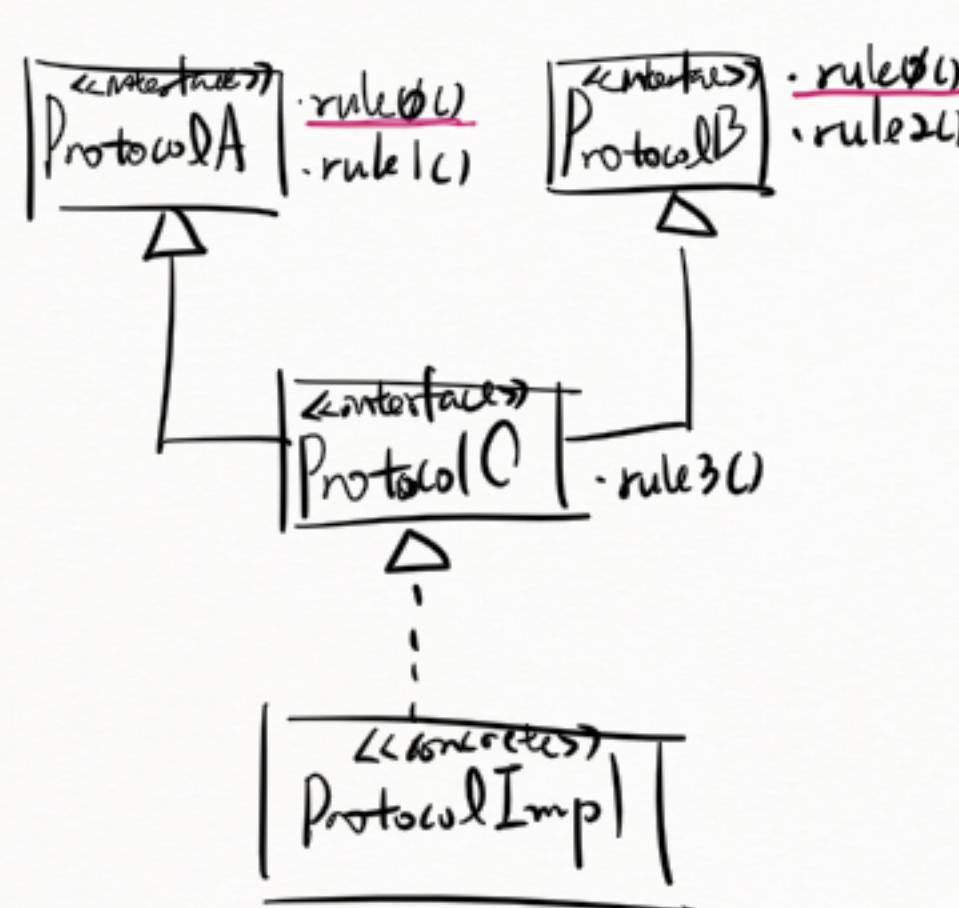
~~obj3. m1();~~
~~obj3. rule2();~~
~~obj3. rule1();~~

ProtocolA obj3 = obj;

* 인터페이스 다중 상속



* 이전에는 다음과 같은
가능한 경우

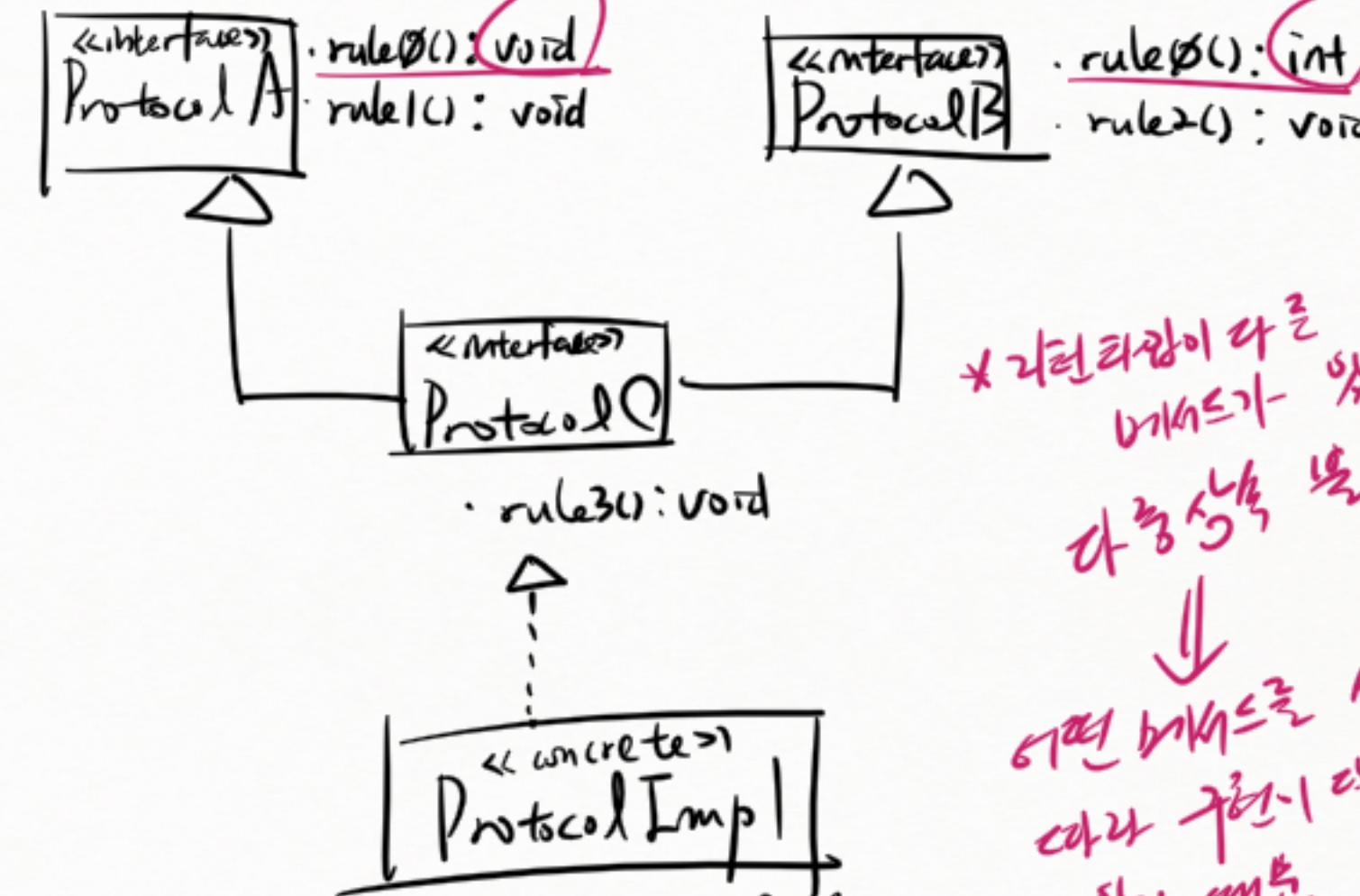


ProtocolA
rule0()
rule1()

ProtocolB
rule0()
rule2()

ProtocolC
rule3()

불가능한 경우



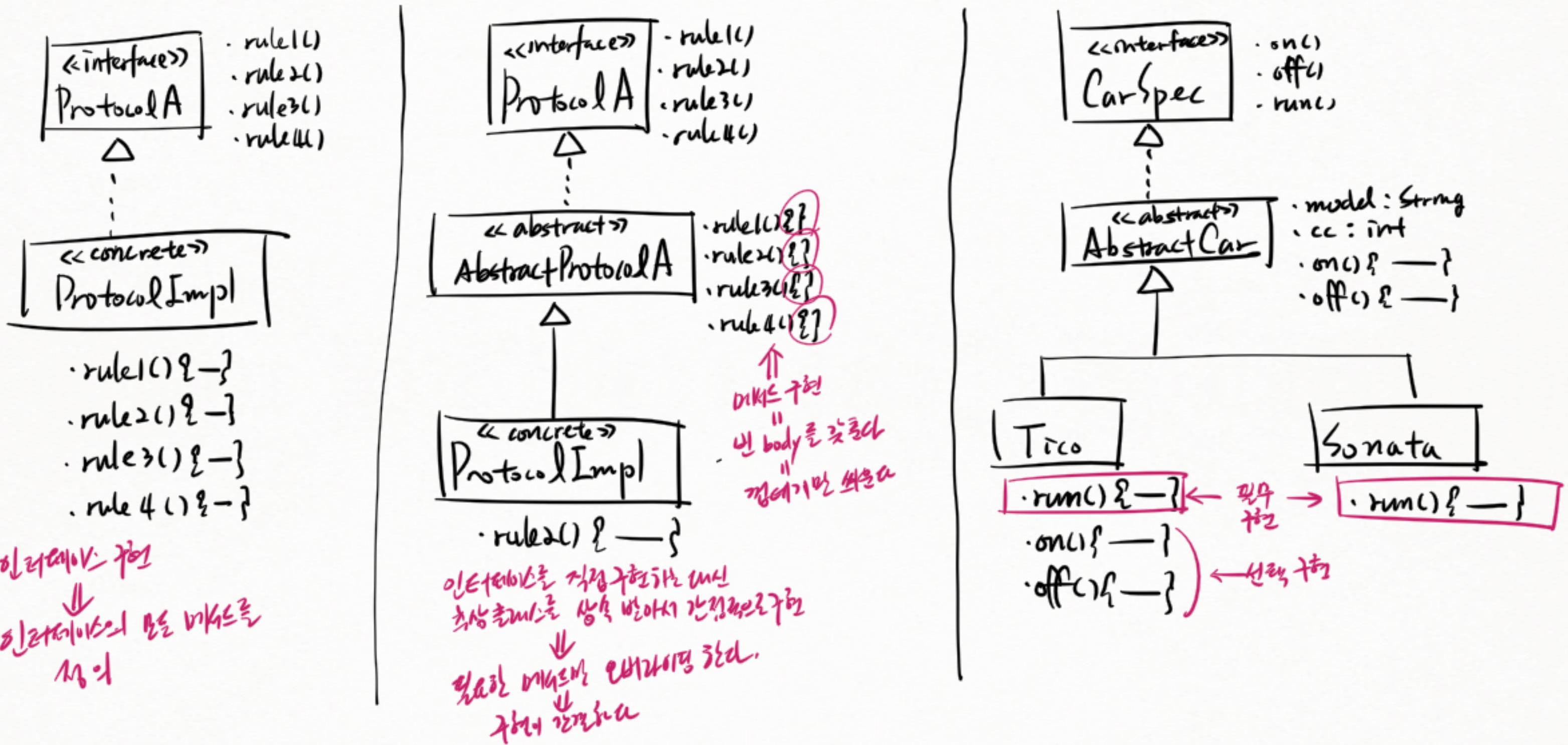
이전처럼 다를
수는 없지만
부모가 다른
것은 가능합니다.

• void rule0()
• int rule0()

* 이런 이유로 다를
수는 없지만
부모가 다른
것은 가능합니다!

여전히 부모는 상속 받습니다
그러나 구현이 달라질 수
있기 때문

* 인터페이스와 추상 클래스



* 인터페이스의 default 메소드

