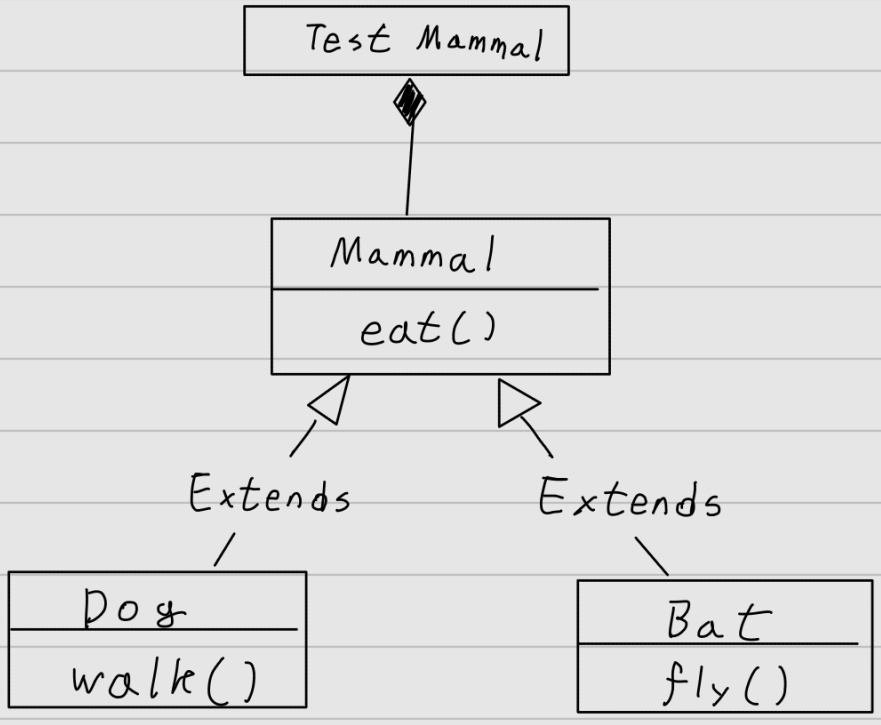
객체 지향 프로그래밍 기본 개념

- 캡슐화
- 상속
- 다형성
- 합성

상속 : 클래스들 간에 의존체가 있게 함

합성 : 상속과 마찬가지로 묶임 문제가 발생할 수 있음

1) 상속

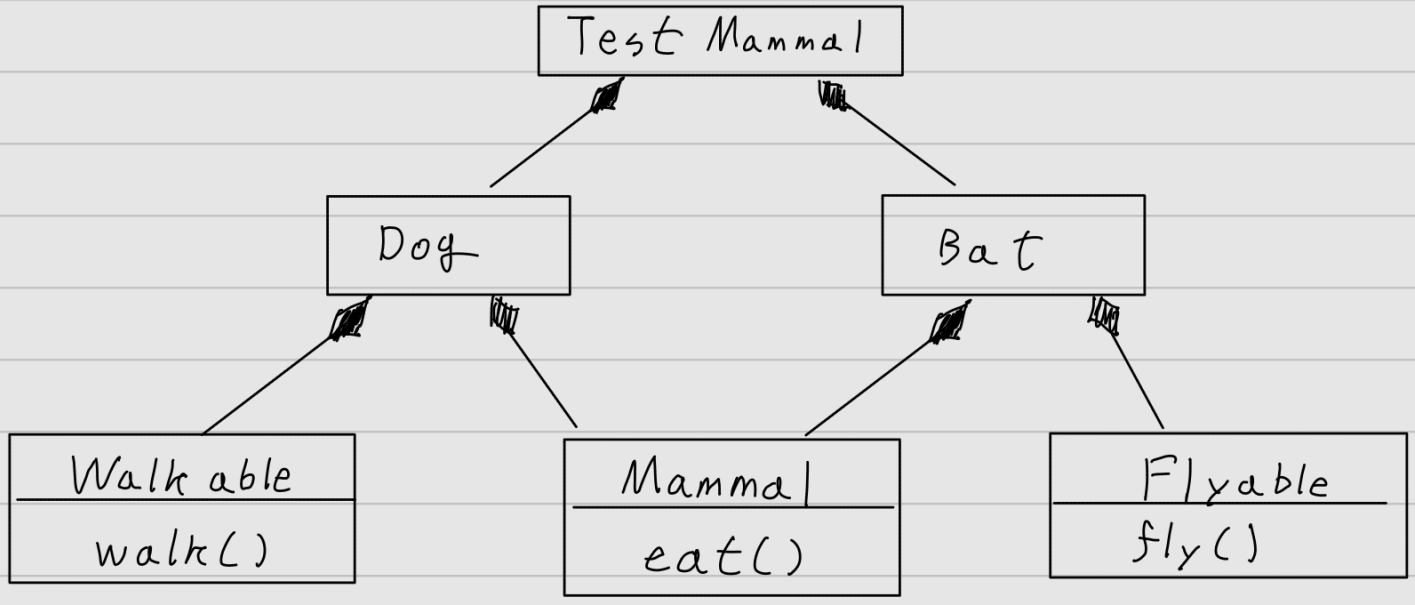


상속 위 계층 구조에서 메서드가 어디에 배치되어야 할지 결정하는 것이 까다로움

ex) 모든 포유류가 걷는 것은 아님!

상속은 다형성이 꼭 필요한 경우에만 사용하는 것이 좋음

2) 합성



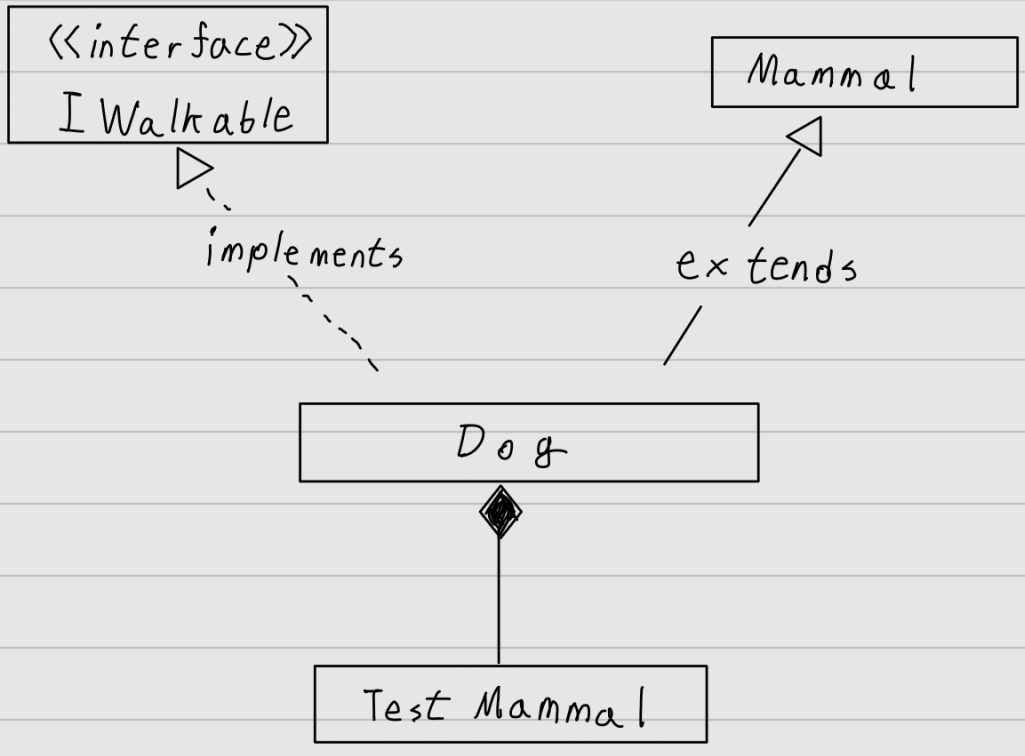
각 행위에 대해 개별 클래스를 만들어
응집을 통해 포함함으로 포유류를 만들 수 있다.

합성을 사용시 기존의 클래스를 변경하지 않고
새로운 클래스를 추가할 수 있다.

3) 의존성 주입 (P. 229 ~ 232)

: 객체를 생성하는 대신에 매개 변수 목록을 통해 다른 객체 내부로 외부 객체를 주입 하는 것

* 제어 역전 (IoC) : 다른 사람이 의존체를 인스턴스화 해 전달



위 와 같은 구조에서 Dog 클래스의 생성자 와 setter 는 IWalkable w 를 매개 변수로 받아 this.walker 를 Dog에 삽입 한다.

* 원칙적으로는 생성자에서 주입하고 setter 은 선택적으로 제공한다.

결론

의존성 주입은 여러분이 만든 클래스의 구성을 클래스 의존체들의 구성으로부터 분리한다. 이는 여러분이 매번 자신만의 것을 직접 제작하는 대신에 (공급업체의) 상품 진열대에서 무언가를 구입해 오는 일과 같다.

이게 상속과 합성에 대한 토론의 핵심이다. 이것이 단순히 토론이라는 점에 유의해야 한다. 이번 장의 목적은 클래스를 설계하기에 '최적인' 방법을 설명하기 위한 것이 아니라 상속과 합성 사이의 결정과 관련된 문제에 대해 생각하는 것이다.