OOP 4대 특징

Giftbot

Four Primary Concepts

- 추상화 (Abstraction)
- 캡슐화 (Encapsulation)

  ※ 은닉화 (Information Hiding)
- 상속성 (Inheritance)
- 다형성 (Polymorphism)

Abstraction

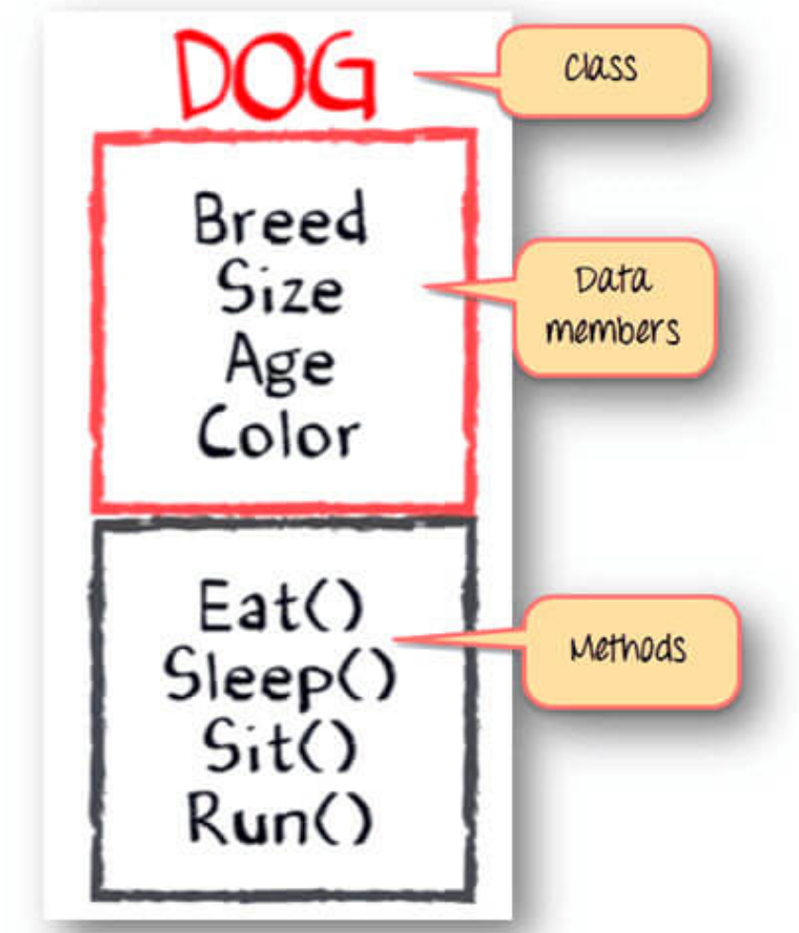
대상의 불필요한 부분을 무시하여 복잡성을 줄이고 목적에 집중할 수 있도록 단순화 시키는 것 (디자인 레벨)

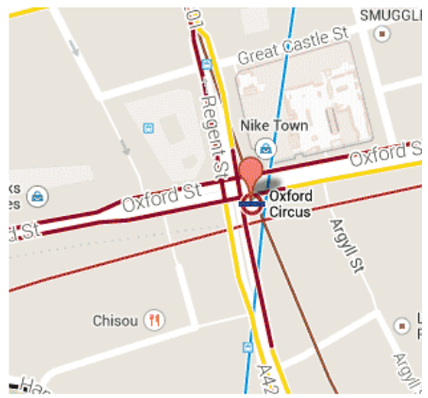
- 사물들 간의 공통점만 취하고 차이점을 버리는 일반화를 통한 단순화

- 중요한 부분의 강조를 위해 불필요한 세부 사항을 제거하는 단순화

관심 영역 = 도메인 = 컨텍스트 / 추상화 = 모델링 = 설계

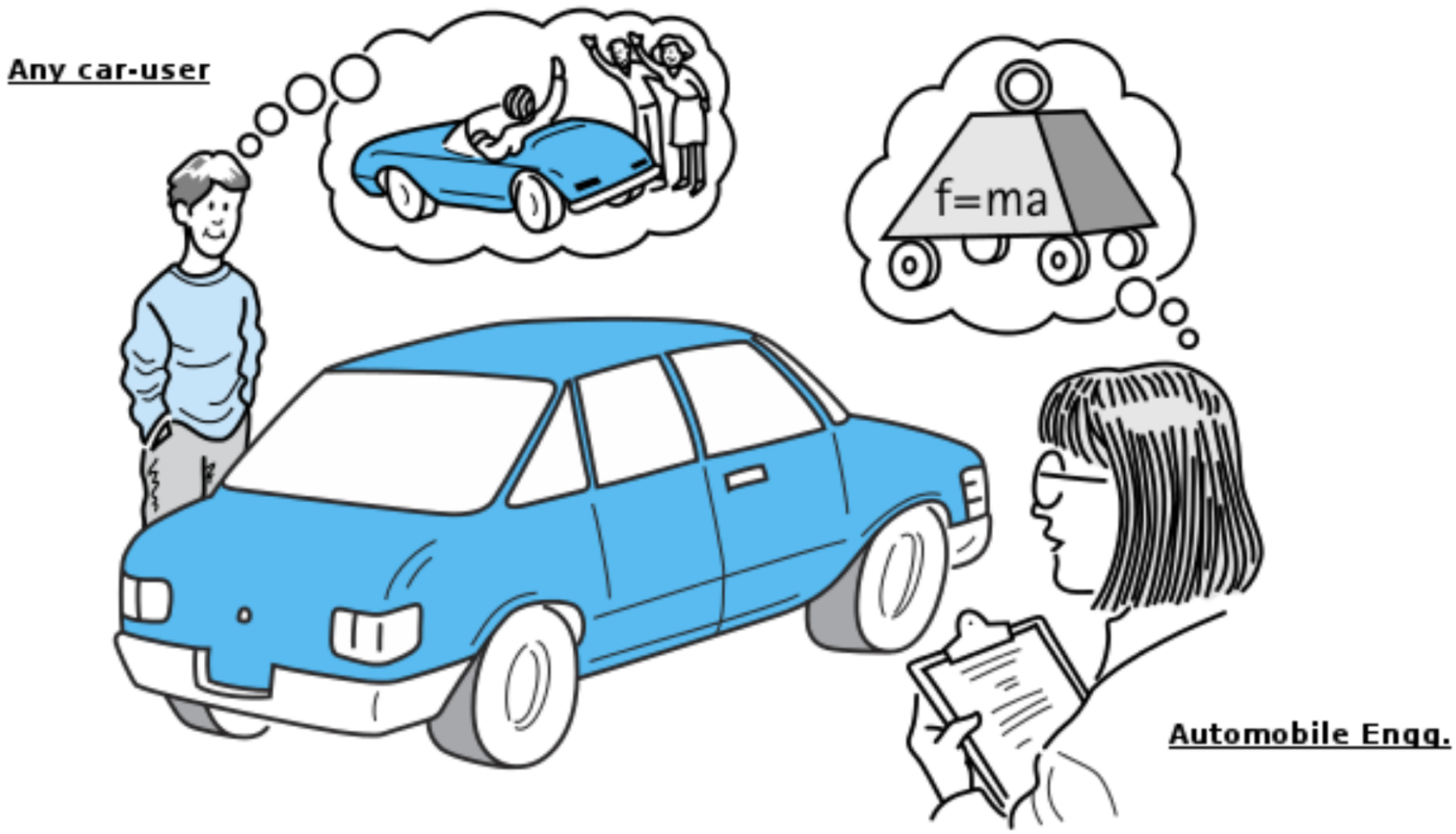
e.g. 지하철 노선도, 비상구 이미지, 이모지, 캐리커쳐 등





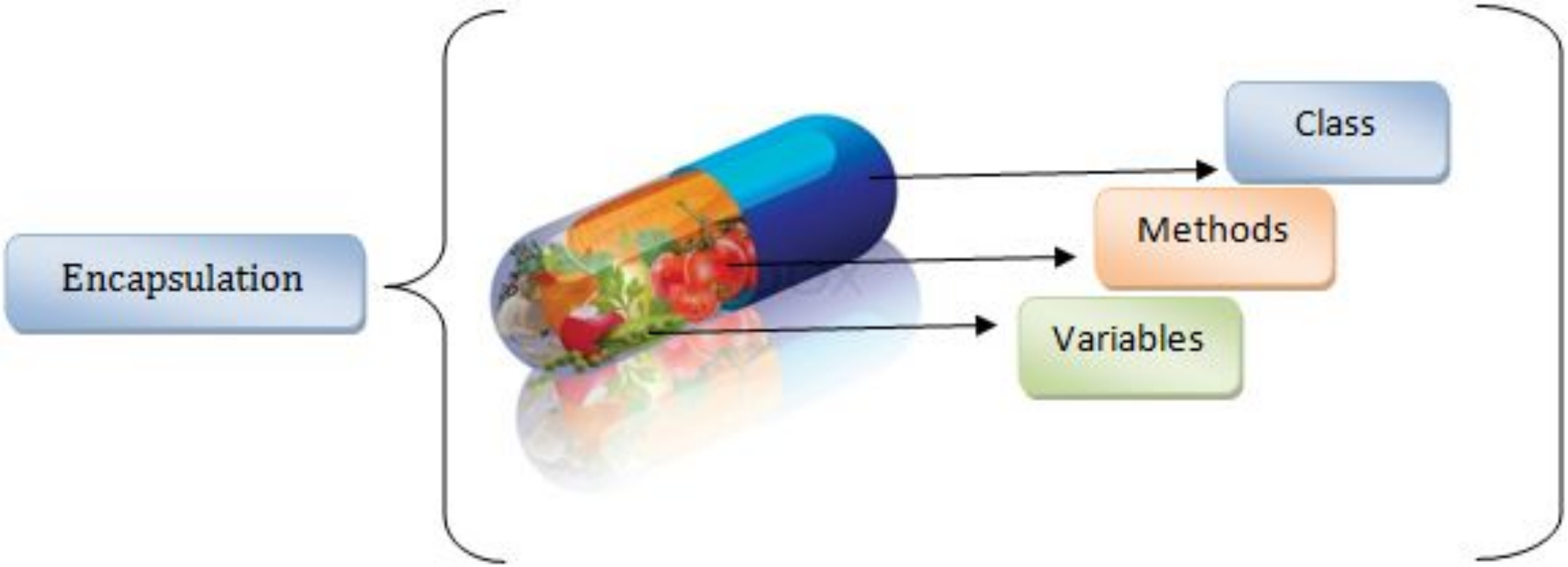
Abstraction

추상화는 대상에 대한 관점과 사용 목적에 따라 달라질 수 있음



An abstraction includes the essential details relative to the perspective of the viewer

Encapsulation



관련있는 정보와 기능을 묶는 것을 캡슐화라고 함.
알약내부에 무슨 성분이 있는지 다 알고 복용하지 않듯이 중요하지 않는 것은 드러내지 않기도 함.
필요한 내용만 노출시킴.

Encapsulation

추상화가 디자인 레벨에 해당하는 개념이라면 캡슐화는 구현 레벨에서의 개념

- 데이터 캡슐화 (Data Encapsulation): 연관된 상태와 행동을 하나의 단위 (객체) 로 캡슐화
- 정보 은닉화 (Information Hiding): 외부에 필요한 것만 알리고 불필요하거나 감출 정보는 숨김

Encapsulation

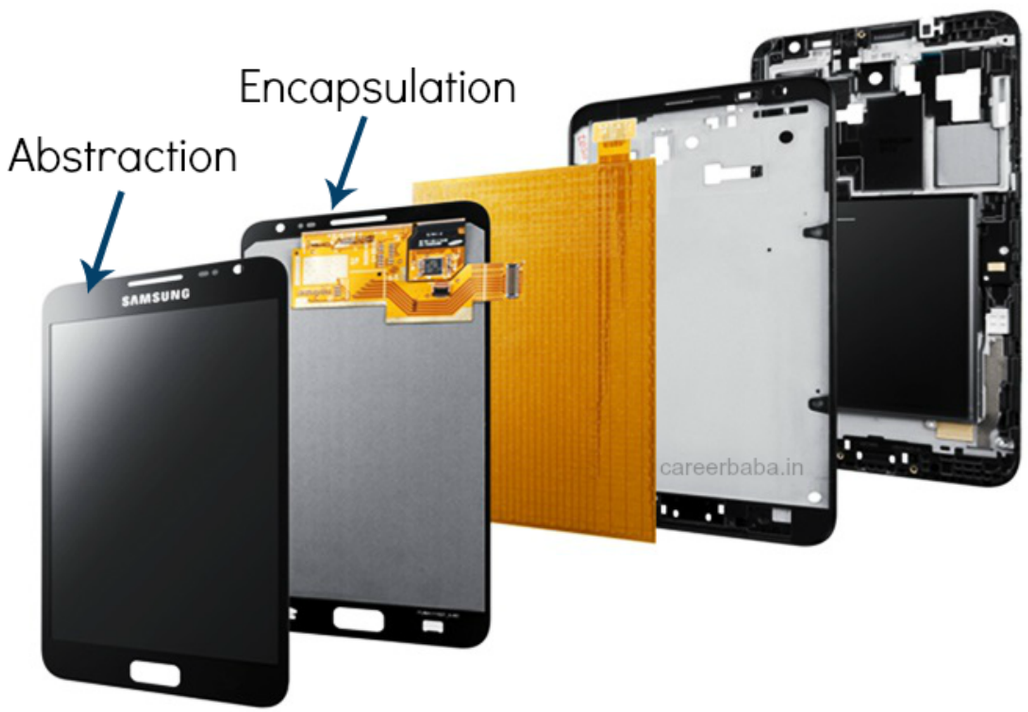
객체가 독립적으로 자신의 상태와 역할을 책임지고 수행할 수 있도록 자율성 부여

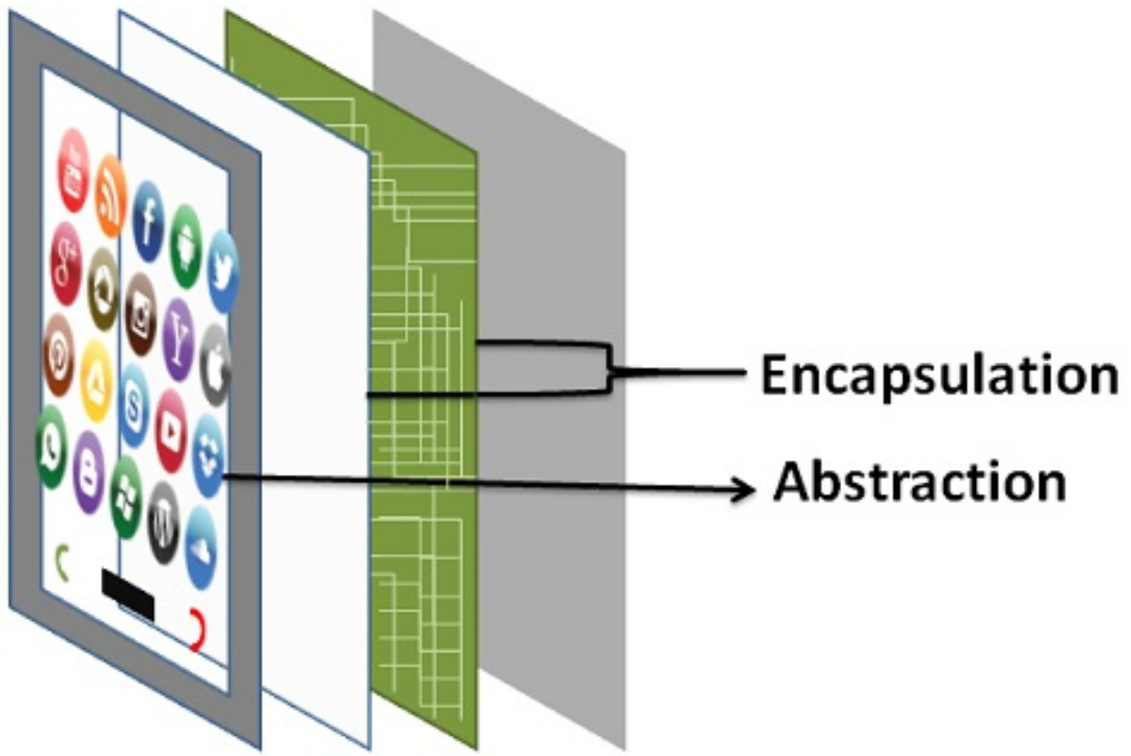
접근 제한자(private)를 이용해 데이터를 외부로부터 보호하여 무결성을 강화하고 변화에 유연하게 대응

자세히 몰라도 되는 내부 동작방법을 숨기고 사용하는 방법만을 외부로 노출

외부에서 요청을 전달하면 수신 객체는 '어떻게' 처리할 지를 결정. 외부에서 그 내용을 자세히 알 필요 없음

[예] - 선풍기, 핸드폰, 리모콘, 카메라, 캡슐 등





Inheritance

하나의 클래스의 특징(부모 클래스)을 다른 클래스가 물려받아 그 속성과 기능을 동일하게 사용하는 것

범용적인 클래스를 작성한 뒤 상속을 이용해 중복되는 속성과 기능을 쉽게 구현 가능

주요 목적 : 재사용과 확장 (상속은 수직 확장, Extension 은 수평 확장)

부모 클래스와 자식 클래스는 IS-A 관계. Bird is a Animal / Human is a Animal

Super Class

Animals

Child Class

Amphibians

Reptiles

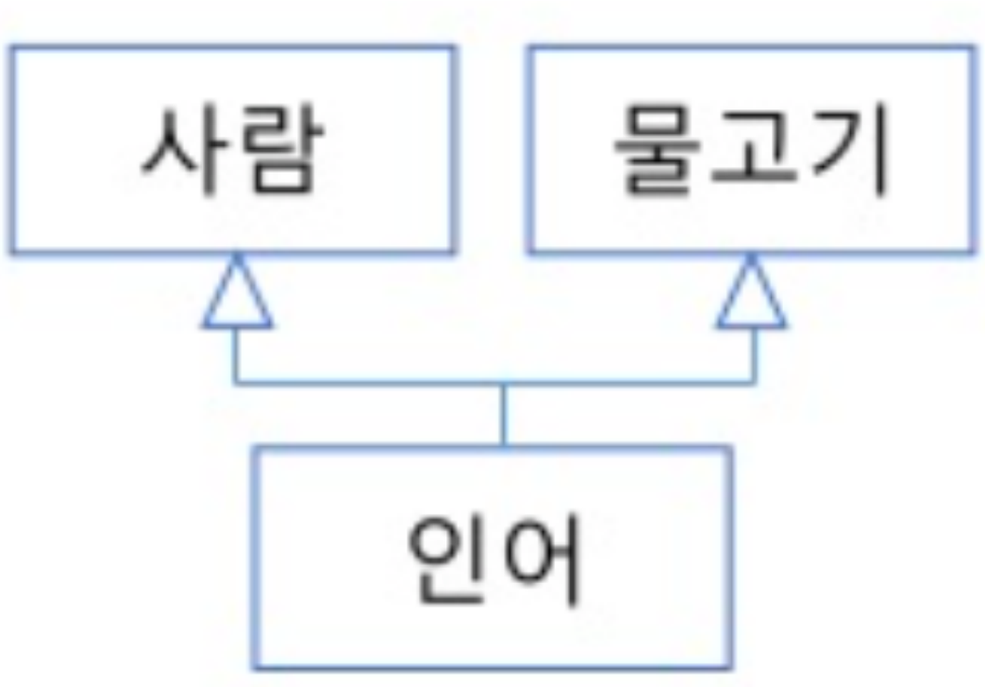
Mammals

Birds

Inheritance

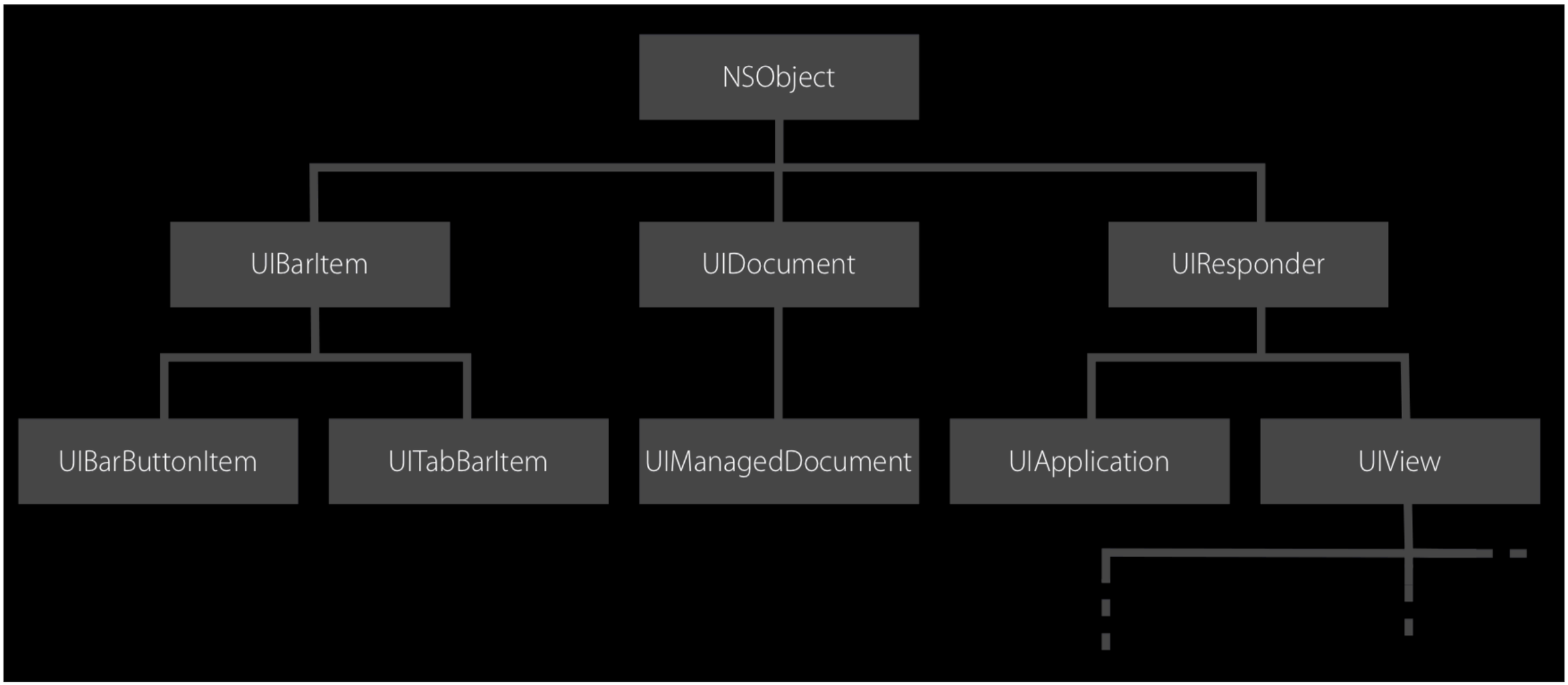
다이아몬드 상속 문제로 인해 언어에 따라 다중 상속을 허용하기도 하고 비허용 하기도 함

Swift 에서는 다중 상속을 비허용하고 대신 Protocol 을 이용하여 유사 기능 구현

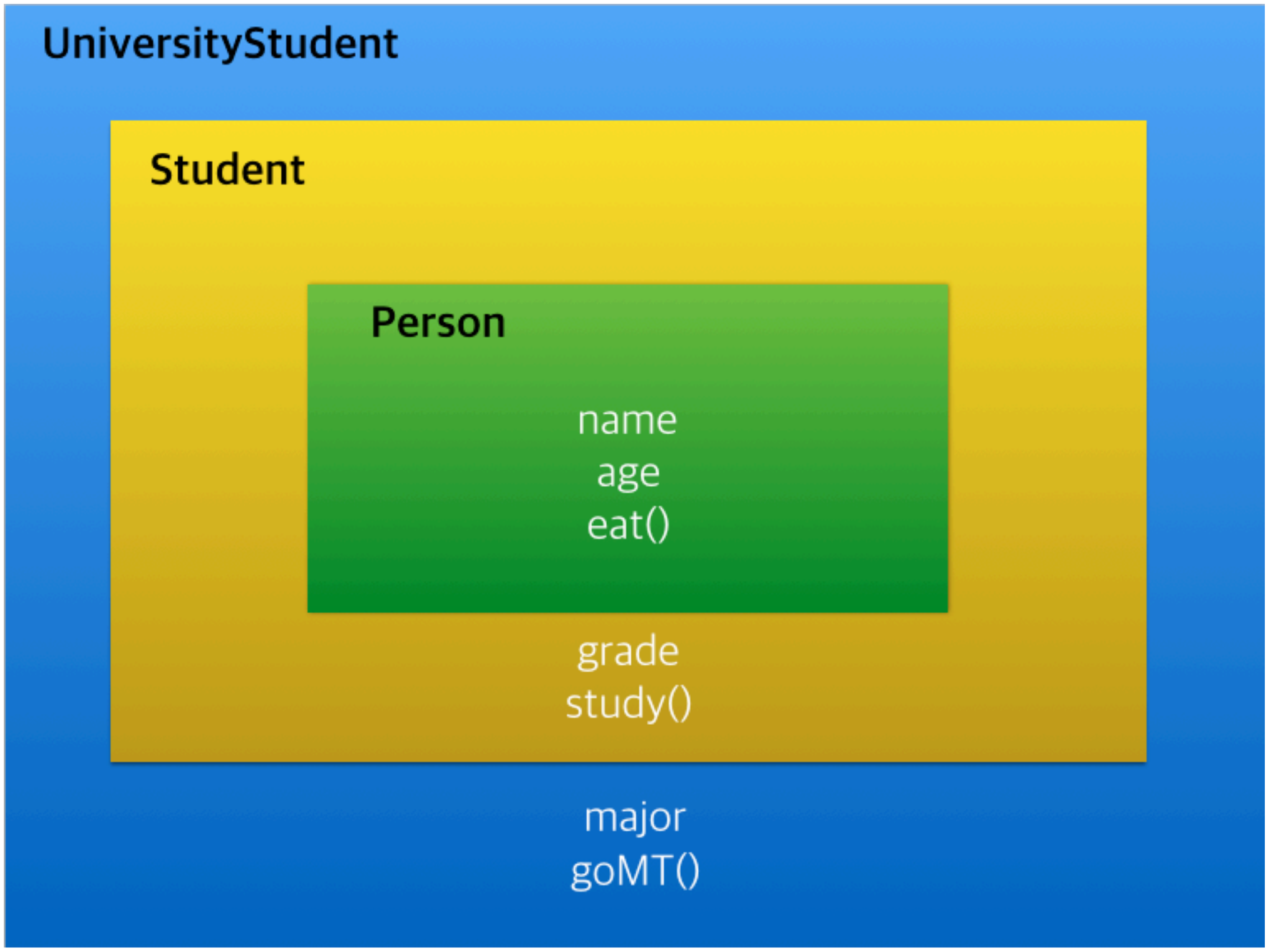


[다이아몬드 상속]

Inheritance



Inheritance



Polymorphism

다양한 형태로 나타날 수 있는 능력 / 여러 형태(many shapes)를 가진다는 의미의 그리스어에서 유래

동일한 요청에 대해 각각 다른 방식으로 응답할 수 있도록 만드는 것

오버라이딩(상속과 관련)과 오버로딩(상속과 무관)이 있으며 언어에 따라 오버라이딩만 지원하기도 함

Polymorphism

오버라이딩 (Overriding)

- 상위 클래스에서 상속 받은 메서드를 하위 클래스에서 필요에 따라 재정의하는 것
- 동일 요청이 객체에 따라 다르게 응답

```
class Shape {
  func draw() {}
}
class Circle: Shape {
  override func draw() { print("draw circle") }
}
class Triangle: Shape {
  override func draw() { print("draw triangle") }
}
```

Polymorphism

오버로딩 (Overloading)

- 동일한 이름의 메서드가 매개 변수의 이름, 타입, 개수 등의 차이에 따라 다르게 동작하는 것

- 동일 요청이 매개 변수에 따라 다르게 응답

```swift
func someFunction(param: Int) {
  print(param)
}
func someFunction(param: String) {
  print(param)
}

someFunction(param: 10)
someFunction(param: "10")
```