객체지향 철학 그리고 5대 개념

Sunny Kwak
sunnykwak@daum.net

Agenda

- **객체지향 철학**
  - 원리의 근본 = 철학
  - 서양철학과 동양철학
  - 인간 = 주체, 나머지 = 객체
  - 객체지향 프로그래밍

- **객체지향 5대 개념**
  - 객체지향의 5가지 개념
  - 객체 (Object)
  - 클래스 (Class)
  - 캡슐화 (Capsulation)
  - 상속 (Inheritance)
  - 다형성 (Polymorphism)

철학에서 비롯된 객체지향 원리

객체지향 철학

원리의 근본 = 철학

- **모든 원리의 근본**
  - 세상이 어떻게 동작하는지 이해하는 것입니다.

- **객체지향 프로그래밍의 원리는...**
  - 나는 무엇을 아는가?
  - 외부의 사물(物)은 어떻게 인식되는가?
  - 인간의 인식은 어떻게 "거기 밖(out there)"에 있는 실재에 대응할 수 있는가?

위와 같은 질문에 대한 이해에서 비롯된 것입니다.

서양철학과 동양철학

- **서양철학**
  - 인간은 인간의 불완전 하지만, 神의 형상(형질)를 본받음.
  - 따라서, 인간은 神에게 위임받아 세계와 자연을 보살필 의무와 권리를 가짐.

- **동양철학**
  - 음(陰) 양(陽)의 조화
  - 인간은 자연의 일부이며, 자연을 소유하는 것이 아니라 구성 요소임.

미켈란젤로의 천지창조

다빈치의 인체도

음양

복희와 여와

인간 = 주체, 나머지 = 객체

- **서양철학에서 '인간'은...**
  - 神을 부정하건, 받아들이건 간에 세상의 나머지 모든 것을 주관하고 관리하는 주체입니다.

- **따라서, 인간 = 주체 (Subject)**
  - 인간을 제외한 모든 것 혹은 인간 중심(시점)에서 바라보고 다루는 모든 것(thing)은 '주체의 반대 개념'인 객체 (Object)입니다.

객체지향 프로그래밍

- **절차적 프로그래밍**
  - 프로그래밍이란, 입력을 받아 명시된 순서대로 처리한 다음, 그 결과를 내는 것이라고 정의함. 프로그래밍이란 어떻게 어떤 논리를 컴퓨터가 이행하는 명령으로 서술하는 것.

- **구조적 프로그래밍**
  - 에츠허르 다익스트라가 1968년, "GOTO문의 해로움"(Go To Statement Considered Harmful)이라는 논문에서 프로그램을 함수(프로시저) 단위로 나누고 프로시저끼리 호출을 하는 방식을 제안.

- **객체지향 프로그래밍**
  - 비 수학적인 사고로 문제를 해결하는 방식을 지향, 달리 말해 사람들이 세상의 사물(객체)을 인식하는 방식으로 프로그램을 설계하려는 의지. (다분히 서양철학의 사상에서 비롯됨)
  - 큰 문제를 작게 쪼개는 것이 아니라, 먼저 작은 문제들을 해결할 수 있는 객체들을 만든 뒤, 이 객체들을 조합해서 큰 문제를 해결하는 상향식(Bottom-up) 해결법

개념을 확실히 잡아야 합니다.

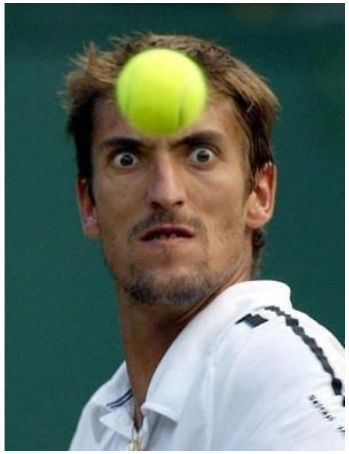
객체지향 5대 개념

객체지향의 5가지 개념

- **객체 (Object)**
  - 상태(state)와 행위(behavior)를 가지는 것(thing)
- **클래스 (Class)**
  - 동일한 형질의 객체의 타입 혹은 객체들을 만들어내는 틀.
- **캡슐화 (Capsulation)**
  - 객체의 내면(상태와 행위)을 감추어 단순화 시키는 것.
- **상속 (Inheritance)**
  - 계층 구조(hierachy)의 표현. (재사용은 덤으로 얻는 것)
- **다형성 (Polymorphism)**
  - 하나의 계층에 속한 객체들이 같은 지시(명령)에 대해 다른 행위를 수행하는 것.

객체 (Object)

- **Object = Thing**
  - 인간 중심에서 바라보는 모든 대상 혹은 사물



- **Object = State + Behavior**
  - 모든 객체는 상태(state)와 행위(behavior)를 가진다.

- **Object = Component**
  - 객체지향 프로그램에서 프로그램의 구성 요소(component)는 함수(function) 혹은 프로시져(proceure)가 아니라 객체.
  - 앞서 설명한 바, 세상의 중심이 '자아'라는 인식이 자연스러운 사람들은 평소 사용하는 언어 및 사고 습관에 의해 '객체지향'이 자연스럽게 받아들여진다.

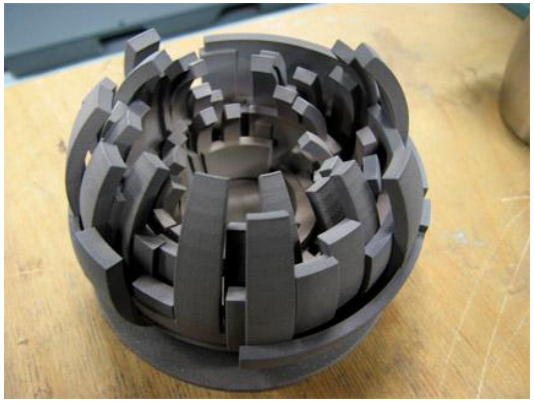
클래스 (Class)

- **Class = Type of object**
  - 동일한 행위(behavior)와 속성(attribute)을 가지는 객체들의 청사진.

- **Class = Template**
  - 객체들을 만들어내는 틀(template).
  - 객체는 특정 클래스에서 만들어진 후 분리되는 것이지,
    클래스에 포함되어 있는 것은 아니다.

캡슐화 (Capsulation)

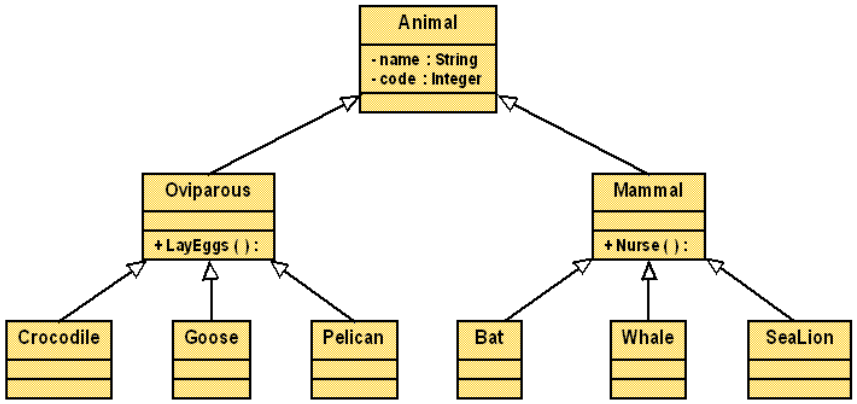
- **Capsulation = Packaging**
  - 객체의 상태(state) 혹은 내부에서 일어나는 행위(behavior)를 외부에서 알 수 없게 감추는 것.
  - 대다수의 버그(bug)는 데이터가 적절한 범위에서 벗어나기 때문에 발생하므로, (예를 들어, 사람의 나이는 음수가 될 수 없다.) 외부에서 함부로 내부의 상태와 행위를 제어할 수 없게 한다.



- **Capsulation = Abstraction**
  - 겉으로 드러나는 특성을 적게 유지하면, 클래스 혹은 객체를 쉽게 이해할 수 있다.
  - Abstraction 은 복잡함을 감춘다는 의미도 있으며, '요약'으로 번역할 수도 있다.
  - 다른 표현은 정보 은폐 (Information Hiding).

상속 (Inheritance)

- **Inheritance**
  - 클래스들을 동일한 특징을 가진 것으로 분류하는 것.
  - 복잡한 문제를 나누어서 공통된 부분부터 구현(정의)하기 위함.
  - 또한, 복잡한 구조를 한 눈에 파악할 수 있게 하기 위함.
  - 재사용(reuse)은 부가적으로 얻는 효과!
    재사용은 상속의 핵심 목표가 절대 아니다.



다형성 (Polymorphism)

- **Polymorphism = Many shapes**
  - 동일한 상속 구조의 하위에 속한 클래스들의 서로 다른 행위를 하나의 명칭(메소드 이름)으로 통일하는 것.
  - 흐름 제어(flow control)을 단순화하기 위한 기법.
    (복잡한 문제를 최대한 단순하게 표현하자는 의도)

참조 (Reference)

- **철학**
  - http://ko.wikipedia.org/wiki/%EC%B2%A0%ED%95%99
- **객체-지향 프로그래밍 이란 무엇인가? (OOP)**
  - http://vandbt.tistory.com/10
- **객체-지향 프로그래밍 이란 무엇인가? : 다섯개의 기반 개념**
  - http://vandbt.tistory.com/39