

객체지향의 오해와 진실 - 조영호

특제: 역할, 책임, 협력의 관점에서 본 객체지향

클래스 리얼 인터페이스

다형성 - 동일한 요청에 대해 서로 다른 방식으로 응답할 수 있는 능력

동일한 클래스나 인터페이스를 공유하고 있는 서로 다른 객체. 상속

객체지향의 덕목

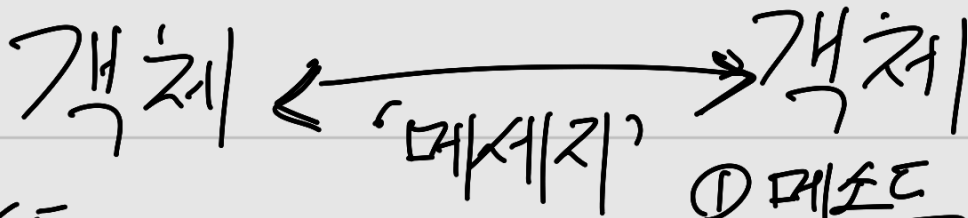
자율적! 협력적!

내외부를 분리 캡슐화

캡슐화 → 상속 → 다형성

책임 → 역할 → 협력

협력적이라는 것은 수동적으로 명령에 따라 행동하지 않고, 요청에 응답하는 것을 의미한다.



- ① 메소드
- ② 메소드
- ⋮

- ① 메소드
- ② 메소드
- ⋮

메시지로 요청하고, 메서드로 응답한다.

가장 보편적인 예시는 지하철 노선도

추상화

추상화는 두 단계 (차원으로 이뤄진다)

1. 공통점은 취하고 차이점은 버리는 일반화

2. 중요한 부분을 강조하기 위해

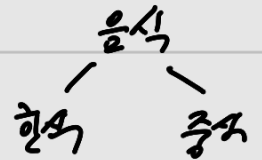
현실의 복잡도를 ← 불필요한 부분 제거
극복

추상화 기법

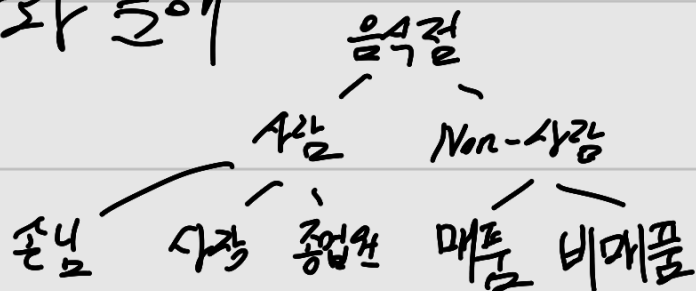
1. 분류와 인스턴스화

A 자동차, B 자동차

2. 일반화와 특수화



3. 결합과 분해



'분류' 및 '개념'

분류 (classification) 은 객체에 특정한 개념을 적용하는 작업이다.



개념의 기초적인 예는

Type 이 될 수 있다.

OOP에서는 객체의 타입이

클래스가 되는 것으로 보임.

→ 캡슐화

'객체 타입은 내부구조와 무관하다. 내부포함 방식이 다르더라도 어떤 객체들이 동일하게 행동한다면, 그 객체들은 동일한 타입에 속한다.

같은 타입에 속한 객체는 동일한 책임을 가진다.

↳ 객체가 메시지 수신.

단, 내부구조는 다르므로

메시지를 처리하는 방식이 다르다 => 다형성

상태 (= 속성, 멤버 변수)

상태 (속성)을 이용하면, 객체의 History 없이 쉽게
행동의 결과를 예측하고 실행할 수 있다.

객체의 행동은 상태를 변경시키고, 행동의 결과는
상태 의존적이다.

객체는 데이터가 아니다. 객체에서 중요한 건 행동
상태는 행동의 결과로 초래된 복수적인 효과를
효과적으로 표현하기 위한 추상적 개념일 뿐이다.
이전까지 이야기하네.

상태보다는 객체간 협력에 집중해서 고려하라.

클래스

각 물리적 객체는 공용 인터페이스를 수정하지 않는 한
외부 객체에 영향을 미치지 않고 내부 구현을 자유롭게
수정할 수 있다.