

1장. 협력하는 객체들의 공동체

시너지를 생각하라. 전체는 부분의 합보다 크다.

 객체지향이란 ?

 적합하지 않은 이유 ?

 객체지향의 목표

협력하는 사람들

 협력

 역할과 책임

 객체의 역할

협력 속에 사는 객체

 객체의 덕목

1. 충분히 '협력적'일 것

2. 충분히 '자율적'일 것

 자율적인 객체

 협력과 메시지

 메서드와 자율성

객체지향 프로그래밍 언어의 특징

객체지향의 본질

 정리

 객체를 지향하라

메시지를 주고받는 객체의 관점으로 사고 중심을 전환하라.

시너지를 생각하라. 전체는 부분의 합보다 크다.

객체지향이란 ?

실세계를 직접적이고 직관적으로 모델링할 수 있는 패러다임

- 객체지향의 기반을 이루는 철학적 개념을 설명하는 데는 적합
- 유용하고 실용적인 관점에서는 **적합하지 않음**

적합하지 않은 이유 ?

1. 애플리케이션 개발 중 객체에 직접적으로 대응되는 실세계 사물을 발견할 확률이 낮음
2. 소프트웨어가 반영해야 하는 객관적인 실세계가 존재하는가 ?
3. 소프트웨어 객체와 실세계 사물 간 연관성은 희미하다.

→ 실세계에 대한 비유가 그래도 객체지향의 다양한 측면을 학습하는데 효과적이긴 함 !

객체지향의 목표



연결완전성

- 새로운 세계를 창조하는 것
 - 고객 만족
- 상태와 행위를 캡슐화하는 소프트웨어 객체의 자율성
- 객체 간 메시지를 주고받으며 공동의 목표를 달성하기 위해 협력

협력하는 사람들

모든 과정은 **역할**, **책임**, **협력**이라는 사람의 일상속의 세 가지 개념이 조화를 이루며 만들어 낸 것이다.



협력

- 스스로 해결하지 못하는 문제는 그걸 아는 사람에게 도움을 **요청**한다.
- 요청을 받은 사람은 책임을 다하여 지식을 제공하기 위해 **응답**한다.



역할과 책임

- 사람들은 협력하는 과정 속에서 특정한 **역할**을 부여받는다.
 - 역할: 어떤 협력에 참여하는 특정한 사람이 협력 안에서 차지하는 **책임**이나 의무
- **역할은 의미적으로 책임을 내포한다.**



객체의 역할

역할은 유연하고 재사용 가능한 협력 관계를 구축하는 중요한 설계 요소

- 여러 사람(객체)이 동일한 역할을 수행할 수 있다.
 - 요청하는 사람(객체)이 원하는 일을 할 수만 있다면 누가 해주는 지 중요하지 않다.
- 역할은 대체 가능하다.
- 책임을 수행하는 방법(메서드)은 자율적으로 선택할 수 있다.

- 요청을 받은 사람(객체)은 처리해 줄 수만 있다면 어떻게 하는지 중요하지 않다. (다형성)
- 한 사람(객체)이 동시에 여러 역할을 수행할 수 있다.

협력 속에 사는 객체

객체의 덕목

1. 충분히 '협력적'일 것

- 혼자 다 하려고 하면 복잡해진다.
- 다른 객체의 요청에 충실히 귀 기울인다.
- 다른 객체에게 적극적으로 도움을 요청한다.

2. 충분히 '자율적'일 것

- 자기 스스로의 원칙에 따라 일을 한다.
- 자기 스스로를 통제하여 절제한다.

자율적인 객체

객체란 상태와 행동을 함께 지닌 실체

- 객체의 내부와 외부를 명확하게 구분하는 것부터 자율성이 나온다 !
 - 사적인 부분은 외부에서 간섭하지 못하게 차단할 것
 - 외부에서는 허락된 수단만을 통해서 소통할 것

협력과 메시지


객체지향의 세계에서는 협력을 위해 오직 한 가지 의사소통 수단만이 존재한다. **메시지 !**

- 송신자 : 메시지를 전송하는 객체
- 수신자 : 메시지를 수신하는 객체

메서드와 자율성

객체가 수신된 메시지를 처리하는 방법. **메서드 !**

객체지향 프로그래밍 언어의 특징

- 메시지를 수신한 객체가 런타임에 메서드를 선택할 수 있다.
 - 절차적 언어는 프로시저 호출에 대한 실행 코드를 컴파일 시간에 결정
- 외부의 요청을 표현하는 메시지, 요청을 처리하기 위한 메서드를 구분 가능
 - 메서드 분리는 객체의 자율성을 높이는 핵심 메커니즘
 - 캡슐화 개념과도 연관 

객체지향의 본질

정리

- 객체지향이란 시스템을 상호작용하는 **자율적인 객체들의 공동체**로 바라보고 객체를 이용해 시스템을 분할하는 방법이다.
- 자율적인 객체란 **상태**와 **행위**를 함께 가지고 스스로 자기 자신을 책임진다.
- 객체는 시스템의 행위를 구현하기 위해 다른 객체와 **협력**한다.
- 각 객체는 협력 내 정해진 **역할**을 수행하고 역할은 관련된 **책임**의 집합이다.
- 객체는 다른 객체와 협력하기 위해 메시지를 전송하고 **메시지**를 수신한 객체는 이를 처리하기 위한 **메서드**를 자율적으로 선택한다.

객체를 지향하라

객체지향 설계의 목적은 훌륭한 클래스를 식별하는 것이라고 배웠나요 ?

- 객체지향 언어의 핵심은 오직 객체 !
 - 프로토타입 기반 객체지향 언어(ex: JS)는 오직 객체만 존재
 - 상속도 객체 간 위임 메커니즘을 기반으로 한다.
- 클래스 강조 관점은 객체의 캡슐화를 저해하고 클래스를 강결합시킴
- 유연하고 확장 가능한 애플리케이션 구축을 방해함 !

메시지를 주고받는 객체의 관점으로 사고 중심을 전환하라.

🤔 ❌ 어떤 클래스가 필요한가 ?

✅ 어떤 객체들이 어떤 메시지를 주고받으며 협력하는가 ?

- 클래스는 협력에 참여하는 객체를 만드는 데 필요한 구현 메커니즘일 뿐
- 핵심은 적절한 책임을 수행하는 역할 간 유연하고 견고한 협력 관계 구축
- 객체의 **역할**, **책임**, **협력**에 집중하라