객체지향의 사실과 오해

#스터디

> Quote
>
> 객체지향이 말 그대로 객체를 지향한다는 사실을 잘 알고 있으면서도 많은 분들이 여전히 클래스나 상속을 중심으로 객체지향을 바라보고 있습니다.

내가 이 책을 읽어야 하는 이유가 이 하나의 문장에 잘 드러나 있다. '객체지향'이라는 단어에서 클래스가 먼저 생각났고 이어서 C++/Java가 떠올랐기 때문이다. 이 책을 통해 객체를 지향한다는 것의 의미를 제대로 알 수 있을 것이라 기대한다.

저자는 객체지향으로 향하는 단계를 아래와 같이 정의한다.

1. 클래스가 아니라 객체를 바라보기
2. 객체를 독립적인 존재가 아니라 기능을 구현하기 위해 협력하는 공동체의 존재로 바라보기
3. 협력에 참여하는 객체들에게 적절한 역할과 책임 부여하기
4. 앞에서 설명한 개념을 프로그래밍 언어에 담아내기

1장. 협력하는 객체들의 공동체

> Quote
>
> 객체란 현실 세계의 존재하는 사물에 대한 추상화라는 것이다.

객체지향을 처음 접하고 공부할 때 나는 절차지향과 비교하는 방식으로 학습하였다. 절차지향과 달리 객체지향은 현실세계를 잘 반영하기에 실생활에 관련된 문제를 잘 해결할 수 있다고 배웠다.

나는 의심의 여지도 없이 이 말을 믿고 있었지만 이것이 내가 객체지향에 대해 **첫 번째로 오해**하고 있던 것이다. 저자는 방화벽을 예시로 들어 현실 세계의 방화벽과 네트워크 방화벽이 비유로서는 적절할지 몰라도 실제 연관성은 거의 없고 새로운 것을 창조하는 것에 가깝다고 말한다.

그럼에도 이러한 비유를 사용하는 이유는 비록 실무적 관점에서는 부적합하더라도 객체지향의 다양한 측면을 이해하고 학습하는 데 효과적이라고 한다.

1. 협력하는 사람들

우리가 일상생활에서 커피 한 잔을 마시는 데도 **역할, 책임, 협력**이라는 개념이 조화를 이루어 숨겨져 있다. 손님, 캐시어, 바리스타라는 각자의 역할이 주어져 있기 때문에 맡은 바 책임을 다하고 암묵적으로 협력한다.

요청과 응답으로 구성된 협력

살다 보면 혼자서 해결할 수 없는 복잡한 문제가 있기 때문에 우리는 다른 사람에게 도움을 **요청**하고 그에 대한 **응답**을 제공받는다. (비록 거절당할지라도..)

그림 1.2 커피 주문을 위해 협력하는 사람들

하나의 문제에 관여하는 사람이 많기에 요청은 **연쇄적**으로 발생하고 그에 대한 응답 역시 반대 방향으로 연쇄적으로 전달된다. 이렇듯 우리는 요청과 응답을 통해 다른 사람과 **협력**할 수 있다.

역할과 책임

역할이 주어지면 당연히 그에 대한 책임을 암시하기 때문에 **역할은 책임이라는 개념을 내포**하고 있다.

1. 여러 사람이 동일한 역할을 수행할 수 있다.
2. 역할은 대체 가능성을 의미한다.
3. 책임을 수행하는 방법은 자율적으로 선택할 수 있다.
4. 한 사람이 동시에 여러 역할을 수행할 수 있다.

2. 역할, 책임, 협력

앞서 언급한 4가지 개념을 약간의 변형 과정만 거치면 객체지향이라는 틀 안에서 이해할 수 있다.

1. 사람 → 객체
2. 요청 → 메시지
3. 요청의 처리 방법 → 메서드

역할과 책임을 수행하며 협력하는 객체들

> Quote
>
> 협력의 핵심은 특정한 책임을 수행하는 역할들 간의 연쇄적인 요청과 응답을 통해 목표를 달성한다는 것이다.

아래 4가지 개념은 객체지향 패러다임의 **다형성**과 연관되어 있다.

1. 여러 객체가 동일한 역할을 수행할 수 있다.
2. 역할은 대체 가능성을 의미한다.
3. 각 개체는 책임을 수행하는 방법을 자율적을 선택할 수 있다.
4. 하나의 객체가 동시에 여러 역할을 수행할 수 있다.

예를 들어 학교에 등교할 때 자전거를 타고 갈 수도 있고 차를 타고 갈 수도 있다. 비록 그 방법은 다를지라도 동일한 역할을 수행하기에 대체될 수 있다. 그리고 등교라는 역할 이외에 레이싱(?) 역할로 사용될 수도 있다.

3. 협력 속에 사는 객체

역할, 책임, 협력의 중심에는 항상 객체가 존재한다. 객체는 요청에 대해 **협력적**으로 응답해야 한다. 누군가 말을 걸었는데 무시한다면 협력의 성공과 실패를 떠나서 협력 자체가 성립할 수 없다.

또한 객체는 **자율적**으로 판단하고 결정하여 요청에 응답해야 한다.

상태와 행동을 함께 지닌 자율적인 객체

객체가 자율적으로 판단하고 결정하기 위해서는 어떠한 **행동**을 하여야 하며 그 행동을 위한 **상태**가 필요하다.

또한 그러한 자율성을 지키기 위해서는 외부의 간섭이 없어야 한다. 외부에서는 그 객체가 **무엇**을 하는지만 알면 되고 **어떻게** 하는지는 몰라야 한다.

따라서 객체는 **상태와 행위를 하나의 단위로 묶는 자율적인 존재**이며 **유지보수와 재사용**이 용이하다.

협력과 메시지

객체지향의 의사소통 수단은 오직 **메시지**로만 존재하며 메시지를 전송하는 객체를 **송신자**, 메시지를 수신하는 객체는 **수신자**라고 부른다.

메서드와 자율성

수신된 메시지를 처리하는 방법을 **메서드**라 하며 메시지를 수신한 객체가 **실행 시간에 메서드를 선택**할 수 있다.

메시지와 메서드를 분리하는 것은 자율성을 높일 수 있으며 **캡슐화** 개념과 관련되어 있다.

4. 객체지향의 본질

지금까지의 내용을 모두 종합하면 아래와 같다.

1. 객체지향이란 시스템을 상호작용하는 **자율적인 객체들의 공동체**로 바라보고 객체를 이용해 시스템을 분할하는 방법이다.
2. 자율적인 객체란 **상태와 행위**를 함께 지니며 스스로 자기 자신을 책임지는 객체를 의미한다.
3. 객체는 시스템의 행위를 구현하기 위해 다른 객체와 **협력**한다. 각 객체는 협력 내에서 정해진 **역할**을 수행하며 역할은 관련된 **책임**의 집합이다.
4. 객체는 다른 객체와 협력하기 위한 **메시지**를 전송하고, 메시지를 수신한 객체는 메시지를 처리하는 데 적합한 **메서드**를 자율적으로 선택한다.

객체를 지향하라

> **Quote** ›
>
> 클래스가 객체지향 프로그래밍 언어의 관점에서 매우 중요한 구성요소인 것은 분명하지만 객체지향의 핵심을 이루는 중심 개념이라고 말하기에는 무리가 있다.

클래스는 객체들의 협력 관계를 코드로 옮기는 도구에 불과하며 클래스 관점에서 메시지를 주고 받는 **객체의 관점**으로 사고의 중심을 전환해야 한다. 중요한 점은 클래스들의 정적인 관계가 아니라 메시지를 주고받는 **객체들의 동적인 관계**다.

> **Note** ›
>
> 목차를 보면 1장에서 언급된 개념들이 뒤에서도 자주 등장하는 것으로 보인다. 다형성, 캡슐화 등 언급되었지만 구체적으로 설명되어 있지 않은 개념들은 뒤에서 조금씩 채워나갈 수 있을 것 같다.