1. 협력하는 객체들의 공동체

> **“시너지를 생각하라. 전체는 부분의 합보다 크다.”**

🕑 협력하는 사람들

커피 공화국의 아침

커피숍에서 이루어지는 모든 과정에는 손님, 캐셔, 바리스타 사이의 **협력, 역할, 책임**이 존재한다.
이 세가지가 바로 객체지향에서 가장 중요한 개념이다.

요청과 응답으로 구성된 협력

사람들은 스스로 해결하지 못하는 문제와 마주치면 도움을 **요청(request)**하며, 요청을 받은 사람은 주어진 책임을 다하면서 **응답(response)**을 제공한다.

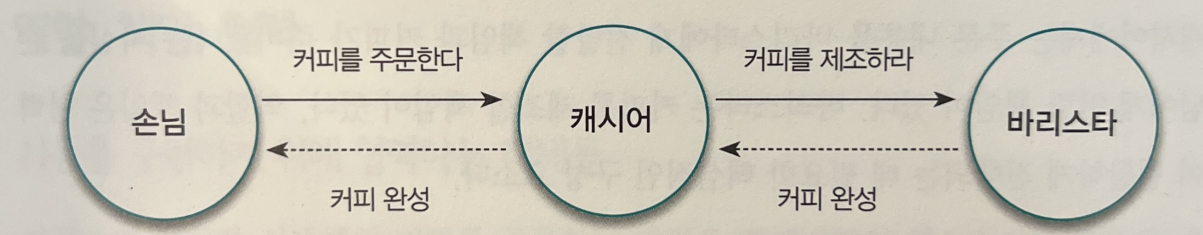

그림 1.2 커피 주문을 위해 협력하는 사람들

요청과 응답은 연쇄적으로 발생하며, 이를 통해 사람들은 서로 **협력(collaboration)**할 수 있다.

역할과 책임

사람들은 다른 사람과 협력하는 과정 속에서 특정한 **역할(role)**을 가진다.
역할은 어떤 협력에 참여하는 특정한 사람이 차지하는 **책임(responsibility)**이나 의무를 의미한다.

사람들이 협력을 위해 특정한 역할을 맡고 역할에 적합한 책임을 수행한다는 사실은 몇 가지 중요한 개념을 제시한다.

- **여러 사람이 동일한 역할을 수행할 수 있다.**
  - 어떤 캐셔가 주문을 받든, 어떤 바리스타가 커피를 제조하든 손님에게는 중요하지 않다.
    손님 입장에서는 캐셔와 바리스타가 각자의 책임을 성실히 수행하면 그만이다.
- **역할은 대체 가능성을 의미한다.**
  - 두 명이 동일한 역할을 수행할 수 있다면, 요청자 입장에서 둘 중 누가 수행하더라도 문제가 되지 않는다.
- **책임을 수행하는 방법은 자율적으로 선택할 수 있다.**
  - 커피를 제조하라는 동일한 요청을 받더라도 바리스타의 역할을 수행하는 사람들마다 다른 방식으로 요청을 처리(응답)할 수 있다. **(다형성)**
- **한 사람이 동시에 여러 역할을 수행할 수 있다.**
  - 한 사람이 동시에 둘 이상의 역할을 수행하는 것도 가능하다.

🕐 역할, 책임, 협력

기능을 구현하기 위해 협력하는 객체들

앞에서 설명한 내용에서 사람이라는 단어를 **객체**로, 에이전트의 요청을 **메시지**로, 에이전트가 요청을 처리하는 방법은 **메서드**로 바꾸면 대부분의 설명을 객체지향이라는 문맥으로 옮

겨올 수 있다.

→ 객체지향을 설명하기 위해 실세계의 모방이라는 은유를 차용하는 이유

역할과 책임을 수행하며 협력하는 객체들

시스템은 역할과 책임을 수행하는 객체로 분할되고, 시스템의 기능은 객체 간의 연쇄적인 요청과 응답의 흐름으로 구성된 협력으로 구현된다.

객체지향 설계라는 예술은 적절한 객체에게 적절한 책임을 할당하는 것에서 시작된다.

역할은 관련성 높은 책임의 집합으로, 유연하고 재사용 가능한 협력 관계를 구축하는 데 중요한 설계 요소다.

🕗 협력 속에 사는 객체

객체지향 애플리케이션의 윤곽을 결정하는 것은 역할, 책임, 협력이지만 실제로 협력에 참여하는 주체는 **객체**다.

패러다임의 중심에 객체가 있기 때문에 '객체지향'이라고 부르는 것이다.

협력 공동체의 일원으로서 객체는 다음과 같은 두 가지 덕목을 갖춰야 하며, 두 덕목 사이에서 균형을 유지해야 한다.

- 객체는 충분히 '**협력적**'이어야 한다.
    - 만약 객체가 외부의 도움을 무시한 채 모든 것을 스스로 처리하려고 하면, 내부적인 복잡도로 인해 자멸할 것이다.
    - 객체는 다른 객체의 요청에 응할지, 응한다면 어떤 방식으로 응답할지 스스로 판단하고 결정한다.
    - 객체는 다른 객체의 명령에 복종하는 수동적인 존재가 아님에 주의하자.
- 객체는 충분히 '**자율적**'이어야 한다.
    - 공동체에 속한 객체들은 공동의 목표를 달성하기 위해 협력에 참여하지만, 스스로의 결정과 판단에 따라 행동하는 자율적인 존재다.

객체지향 설계의 묘미는 다른 객체와 조화롭게 협력할 수 있을 만큼 충분히 개방적인 동시에 협력에 참여하는 방법을 스스로 결정할 수 있을 만큼 충분히 자율적인 객체들의 공동체를 설계하는 데 있다.

상태와 행동을 함께 지닌 자율적인 객체

"객체는 **상태(state)**와 **행동(behavior)**을 함께 지닌 실체"
→ 객체가 협력에 참여하기 위해 어떤 행동을 해야 한다면, 그 행동을 하는 데 필요한 상태도 함께 지니고 있어야 한다는 것을 의미한다.

객체의 자율성은 객체의 내부와 외부를 명확하게 구분하는 것으로부터 나온다.
객체는 다른 객체가 '무엇(what)'을 수행하는지는 알 수 있지만, '어떻게(how)' 수행하는지에 대해서는 알 수 없다.

전통적인 개발 방법과 객체지향을 구분 짓는 가장 핵심적인 차이는 **데이터와 프로세스를 객체라는 하나의 틀 안에 함께 묶어 놓음으로써 객체의 자율성을 보장**한다는 것이다.

자율적인 객체로 구성된 공동체는 유지보수가 쉽고 재사용이 용이한 시스템을 구축할 수 있는 가능성을 제시한다.

협력과 메시지

객체지향의 세계에서는 **메시지**라는 오직 한 가지 의사소통 수단만이 존재한다.

객체지향의 세계에서 협력은 메시지를 전송하는 객체(**송신자**)와 메시지를 수신하는 객체(**수신자**) 사이의 관계로 구성된다.

메서드와 자율성

수신자는 메시지를 이해할 수 있는지 여부를 판단한 후 미리 정해진 자신만의 방법에 따라 메시지를 처리한다.
이처럼 객체가 수신된 메시지를 처리하는 방법을 **메서드(method)**라고 부른다.

메시지를 수신한 객체가 실행 시간에 메서드를 선택할 수 있다는 점은 다른 프로그래밍 언어와 객체지향 프로그래밍 언어를 구분 짓는 핵심적인 특징 중 하나다.

외부의 요청이 무엇인지를 표현하는 **메시지**와 요청을 처리하기 위한 구체적인 방법인 **메서드**를 분리하는 것은 객체의 자율성을 높이는 핵심 메커니즘이다. → **캡슐화(encapsulation)**와도 깊이 관련되어 있다.

🕐 객체지향의 본질

- 객체지향이란 시스템을 상호작용하는 **자율적인 객체들의 공동체**로 바라보고 객체를 이용해 시스템을 분할하는 방법이다.
- 자율적인 객체란 **상태**와 **행위**를 함께 지니며 스스로 자기 자신을 책임지는 객체를 의미한다.
- 객체는 시스템의 행위를 구현하기 위해 다른 객체와 **협력**한다. 각 객체는 메시지를 처리하는 데 적합한 **메서드**를 자율적으로 선택한다.

객체를 지향하라

클래스가 객체지향 프로그래밍 언어의 관점에서 매우 중요한 구성요소인 것은 분명하지만, 지나치게 클래스를 강조하는 관점은 객체의 캡슐화를 저해하고 클래스를 서로 강하게 결합시킨다.

객체지향의 핵심은 적절한 책임을 수행하는 역할 간의 유연하고 견고한 협력 관계를 구축하는 것이다. 클래스의 구조와 메서드가 아니라 객체의 역할, 책임, 협력에 집중하자.

객체지향은 객체를 지향하는 것이지, 클래스를 지향하는 것이 아니다.

💭 후기

나는 줄곧 클래스와 상속을 중심으로 한 붕어빵과 같은 비유로 객체지향을 공부해왔고, 최근 프로젝트를 진행하면서 직접 설계해보니 생각보다 개념을 적용하기가 쉽지 않았다.

그 이유를 이번 장을 통해서 깨닫게 되었고, 객체지향을 어떻게 바라봐야 할지 감을 잡은 것 같다.

객체지향은 결국 객체에 역할, 책임을 적절하게 부여해서 견고한 협력 관계를 구축하는 것이다.

하지만 아직은 "**객체지향의 목표는 실세계를 모방하는 것이 아니다. 오히려 새로운 세계를 창조하는 것이다.**"라는 말의 의미를 이해하기 힘들다.

다음 장부터 이 말의 이해할 수 있을까?