7장. 함께 모으기

마틴 파울러의 세 가지 관점

개념 관점

- 설계는 도메인 안에 존재하는 개념과 개념들 사이의 관계를 표현한다.
    - 실제 도메인의 규칙과 제약을 최대한 유사하게 반영하는 것이 핵심

명세 관점

- 도메인의 개념이 아니라 객체들의 책임에 초점을 맞춘다.
    - 객체가 협력을 위해 '**무엇**'을 할 수 있는가

구현 관점

- 객체들이 책임을 수행하는 데 필요한 동작하는 코드를 작성한다.
    - 객체의 책임을 '**어떻게**' 수행할 것인가

→ 클래스는 세 가지 관점을 모두 수용할 수 있도록 개념, 인터페이스, 구현을 함께 드러내야 한다.

커피 전문점 도메인

커피 전문점이라는 세상

- 메뉴 항목, 메뉴판, 손님, 바리스타, 커피는 각각 자신만의 경계를 가진 객체이다.
    - 손님 → 손님 타입의 인스턴스
    - 바리스타 → 바리스타 타입의 인스턴스
    - 아메리카노 커피, 에스프레소 커피, 카푸치노 커피 등등 → 커피 타입의 인스턴스
    - 메뉴판 → 메뉴판 타입의 인스턴스
    - 아메리카노, 에스프레소, 카푸치노 등등의 메뉴 → 메뉴 항목 타입의 인스턴스

객체들 간의 관계

- 하나의 메뉴판 객체는 다수의 메뉴 항목 객체로 구성되어 있다.
    - **메뉴 항목은 메뉴판과 포함 또는 합성 관계**

- 포함 관계는 아니지만 서로 연관 관계인 것
    - **손님은 메뉴판, 바리스타와 연관 관계**
    - **바리스타는 손님, 커피와 연관 관계**

설계하고 구현하기

커피를 주문하기 위한 협력 찾기

- 협력을 설계할 땐, 메시지를 먼저 선택하고 그 후에 메시지를 수신하기에 적절한 객체를 선택해야 한다.
- 객체가 수신한 메시지가 객체의 인터페이스를 결정한다.
    - **손님 → 커피를 주문하라.**
    - **메뉴판 → 메뉴 항목을 찾아라.**
    - **바리스타 → 커피를 제조하라.**
    - **커피 → 커피를 생성하라**

구현하기

- **구현 도중 객체의 인터페이스는 변경될 수 있다.** 💡
    - 설계는 코드로 구현하는 단계에서 대부분 변경된다.
    - 따라서 협력 구상 단계에 너무 오랜 시간을 쏟지 말고 최대한 빨리 코드를 구현해서 구현 가능성을 판단해야 한다.
- **객체의 속성은 객체의 내부 구현에 속하므로 캡슐화되어야 한다.** 💡
    - 인터페이스에는 객체의 내부 속성에 대한 어떠한 힌트도 제공되어서는 안된다.
    - 인터페이스를 정하는 단계에서는 객체가 어떤 속성을 가지는지 고려하지 않는 것이 좋다.
    - 객체가 어떤 책임을 수행해야 하는지(인터페이스) 먼저 결정하고, 필요한 속성을 결정해라.

코드와 세 가지 관점

코드는 세 가지 관점을 모두 제공해야 한다

- 개념 관점

    - 클래스가 도메인 개념의 특성을 최대한 수용하면 변경을 관리하기 쉽고 유지보수성을 향상시킬 수 있다.
- 명세 관점 (인터페이스)
    - 최대한 변화에 안정적인 인터페이스를 만들기 위해서는 구현 세부사항이 드러나지 않아야 한다.
- 구현 관점 (클래스 내부구현)
    - 메서드와 속성이 철저하게 클래스 내부로 캡슐화되어야 한다.

도메인 개념을 참조하는 이유

- 메시지를 수신할 객체를 선택하는 기준 - 도메인 개념 중에서 가장 적절한 것을 선택한다.
    - 도메인에 대한 지식을 기반으로 코드의 구조와 의미를 쉽게 유추할 수 있다.
    - 클래스가 도메인 개념을 따르면 변화에 쉽게 대응할 수 있다

인터페이스와 구현을 분리하라

- 명세 관점 - 클래스의 안정적인 측면을 드러내야 한다.
- 구현 관점 - 클래스의 불안정한 측면을 드러내야 한다.

→ 인터페이스가 구현 세부사항을 노출하기 시작하면 아주 작은 변동에도 전체 협력이 요동치는 취약한 설계가 된다.