5. 책임과 메시지

↗ Books	객체지향의 사실과 오해

1️⃣ 자율적인 책임

자율적인 객체란 스스로의 의지와 판단에 따라 각자 맡은 책임을 수행하는 객체를 의미한다.
객체가 책임을 자율적으로 수행하기 위해서는 객체에게 할당되는 책임이 자율적이어야 한다.

객체에게 할당되는 책임이 자율적이어야 한다는 의미는 예를 들면 다음과 같다.
“음식을 먹다”라는 책임이 주어졌을 때 젓가락으로 집어 먹을 수도 있고 손으로 주워 먹을 수도 있다.
음식을 먹다라는 책임만 수행한다면 어떤 방법을 사용하냐는 객체에 달려있다는 것이다.

그렇다고 포괄적이고 추상적인 책임을 선택한다고 해서 무조건 좋은 것은 아니다.
”설명하라”와 같은 책임을 수행해야 할 때 어떤 것에 대해 설명해야 하는지 파악할 수 없기 때문이다.

정리하자면, 자율적인 책임의 특징은 객체가 ‘무엇’을 해야 하는가를 설명한다는 것이다.

2️⃣ 메시지와 메서드

하나의 객체는 메시지를 전송함으로써 다른 객체에 접근한다.

메시지 전송은 수신자와 메시지의 조합이며 메시지는 메시지 이름과 인자의 조합이라고 할 수 있다.

```
모자장수.증언하라(어제, 왕국)
```

수신자가 메시지를 변경하지만 않는다면 책임을 수행하는 방법을 변경하더라도 송신자는 알 수 없다.
다른 말로 하면 객체의 외부와 내부가 메시지를 기준으로 분리된다는 것이다.

이렇게 메시지를 처리하기 위해 내부적으로 선택하는 방법을 메서드라고 한다.
객체는 메시지를 수신하면 해당 메시지를 처리할 수 있는지 여부를 확인하고, 처리할 수 있다고 판단되면 메서드를 선택하게 된다.

객체가 실행 시간에 메서드를 선택할 수 있다는 사실은 절차적인 언어와 확연히 구분되는 특징이다.

메시지와 메서드의 차이와 관계를 이해하고 나면 객체지향의 핵심 개념인 다형성을 쉽게 이해할 수 있다.
다형성이란 서로 다른 유형의 객체가 동일한 메시지에 대해 다르게 반응하는 것을 의미한다.

더 구체적으로 말하면 서로 다른 타입에 속하는 객체들이 동일한 메시지를 수신할 경우 서로 다른 메서드를 이용해 메시지를 처리할 수 있는 매커니즘을 가리킨다.

중요한 것은 메시지 송신자의 관점에서 수신자들이 다른 방식을 선택하더라도 결국 동일한 책임을 수행한다는 것이다.
이 말은 동일한 역할을 수행할 수 있는 객체들 사이의 대체 가능성을 의미한다.

이 모든 것이 다형성의 축복처럼 느껴지지만 결국 메시지가 존재하기 때문에 가능한 것이다.
메시지는 송신자와 수신자 사이의 결합도를 낮춤으로써 설계를 유연하고, 확장 가능하고, 재사용 가능하게 만든다.

송신자는 수신자가 메시지를 이해하고 처리할 것이라는 사실만 알아도 되고, 수신자는 메시지를 처리하기 위해 자율적으로 메서드를 선택할 수 있지만 메서드 자체는 송신자에게 노출시키지 않는다.

3️⃣ 메시지를 따라라

클래스는 객체의 속성과 행위를 담는 틀일 뿐이며 심지어 클래스를 사용하지 않고도 객체의 속성과 행위를 표현할 수 있다.

클래스가 아니라 메시지를 주고 받는 동적인 객체들의 집합으로 바라볼 때, 더 나아가 개별적인 객체가 아니라 메시지를 주고 받는 객체들 사이의 커뮤니케이션에 초점을 맞출 때 비로소 객체지향에 가까워질 수 있다.

데이터를 중심으로 객체를 설계하는 방식은 객체의 내부 구조를 객체 정의의 일부로 만들기 때문에 객체의 자율성을 저해한다.
데이터에 대한 결정을 뒤로 미루면서 객체의 행위를 고려하기 위해서는 협력이라는 문맥 안에서 생각해야 한다.
협력이라는 문맥에서 벗어나 독립적인 객체에 관해 고민하는 것은 클래스에 초점을 맞추는 것과 별다른 차이가 없다.

책임-주도 설계 다시 살펴보기

객체지향 설계는 적절한 책임을 적절한 객체에게 할당하면서 메시지를 기반으로 협력하는 객체들의 관계를 발견하는 과정이다.
이처럼 책임을 완수하기 위해 협력하는 객체들을 이용해 시스템을 설계하는 방법을 책임-주도 설계라고 한다.

책임-주도 설계 방법은 협력 관계를 시작할 적절한 객체를 찾아 시스템의 책임을 객체의 책임으로 할당하고, 다른 객체의 도움이 필요하면 어떤 메시지가 필요한지 결정한다. 메시지를 결정한 후 메시지를 수신하기에 적합한 객체를 선택한다.

What/Who 사이클

책임-주도 설계의 핵심은 어떤 행위가 필요한지 먼저 결정한 후 이 행위를 수행할 객체를 결정하는 것인데, 이 과정을 What/Who 사이클이라고 한다.

어떤 행위를 수행할 것인지가 먼저란 의미인데 여기서 어떤 행위가 바로 메시지다.

묻지 말고 시켜라

이렇게 메시지를 먼저 결정하고 객체가 따르게 하는 방식은 객체가 외부에 제공하는 인터페이스가 독특한 스타일을 따르게 한다. 이 스타일을 묻지 말고 시켜라(Tell, Don’t Ask) 또는 데메테르 법칙이라고 한다.

‘묻지 말고 시켜라’스타일은 자율적인 객체들의 공동체를 강조한다. 객체는 다른 객체의 결정에 간섭하지 말아야 하며, 모든 객체는 자신의 상태를 기반으로 스스로 결정을 내려야 한다.

객체는 다른 객체의 상태를 묻지 말아야 한다. 객체가 다른 객체의 상태를 묻는다는 것은 메시지를 전송하기 이전에 객체가 가져야 하는 상태에 관해 너무 많이 고민하고 있었다는 증거다.

4️⃣ 객체 인터페이스

인터페이스

일반적으로 인터페이스는 다음 세 가지 특징을 지닌다.

1. 인터페이스의 사용법을 익히기만 하면 내부 구조나 동작 방식을 몰라도 쉽게 대상을 조작하거나 의사를 전달할 수 있다.
   → 운전자는 자동차의 구조나 원리를 몰라도 운전하는데 아무런 문제가 없다.
2. 인터페이스 자체는 변경하지 않고 단순히 내부 구성이나 작동 방식만을 변경하는 것은 인터페이스 사용자에게 어떤 영향도 미치지 않는다.
   → 자동차의 엔진을 교체하거나 타이어를 교체한다고 해서 운전하는 방법을 새로 배울 필요가 없다.
3. 대상이 변경되더라도 동일한 인터페이스를 제공하기만 하면 아무런 문제 없이 상호작용할 수 있다.
   → 운전하는 방법만 알고 있으면 차종이 바뀌어도 운전이 가능하다.

인터페이스는 외부에서 접근 가능한 인터페이스와 내부에서만 접근할 수 있는 감춰진 인터페이스로 구분된다.
외부에 공개된 인터페이스는 공용 인터페이스라고 한다.

객체가 협력에 참여하기 위해 수행하는 메시지가 객체의 공용 인터페이스의 모양을 암시한다.
메시지를 결정하고 객체를 나중에 결정하기 때문에 메시지가 수신자의 인터페이스를 결정할 수밖에 없다.

공용 인터페이스는 객체의 외부와 내부를 명확하게 분리한다.
객체가 외부로부터 수신할 수 있는 메시지가 공용 인터페이스를 구성하기 때문이다.

5️⃣ 인터페이스와 구현의 분리

맷 와이스펠드는 객체지향적인 사고 방식을 이해하기 위해서는 다음 세 가지 원칙이 중요하다고 주장한다.

1. 좀 더 추상적인 인터페이스
   → '목격했던 장면을 떠올려라', "말로 간결하게 표현하라'와 같은 지나치게 상세한 수준의 메시지가 아닌 '증언하라'라는 좀 더 추상적인 수준의 메시지를 수신할 수 있는 인터페이스를 제공하면 수신자의 자율성을 보장할 수 있다.
2. 최소 인터페이스
   → 외부에서 사용할 필요가 없는 인터페이스는 최대한 노출하지 말아야 한다.
3. 인터페이스와 구현 간에 차이가 있다는 점을 인식
   → 객체지향에서 내부 구조와 작동 방식을 가리키는 고유의 영어는 구현(implementation)이다.
   객체를 구성하지만 공용 인터페이스에 포함되지 않는 모든 것이 구현에 포함된다.
   객체의 외부와 내부를 분리하라는 것은 결국 객체의 공용 인터페이스와 구현을 명확하게 분리하라는 말과 동일하다.

인터페이스와 구현의 분리 원칙

훌륭한 객체란 구현을 모른 채 인터페이스만 알면 쉽게 상호작용할 수 있는 객체를 의미한다.
이것은 외부에 노출되는 인터페이스와 내부에 숨겨지는 구현을 명확하게 분리해야 한다는 것을 의미하며 이를 인터페이스와 구현의 분리 원칙이라고 한다.

인터페이스와 구현의 분리 원칙이 왜 중요한가? 소프트웨어는 항상 변경되기 때문이다!

모든 것이 외부에 공개돼 있다면 아무리 작은 부분을 수정하더라도 객체의 구석구석까지 파고들 것이다.

인터페이스와 구현을 분리한다는 것은 변경될 만한 부분을 내부에 숨겨 놓는 것을 의미하며 일반적으로 이 원칙을 수행하기 위한 객체 설계 방법을 캡슐화라고 한다.

캡슐화

객체의 자율성을 보존하기 위해 구현을 외부로부터 감추는 것을 캡슐화라고 한다.

- 상태와 행위의 캡슐화
  객체는 스스로 자신의 상태를 관리하며 상태를 변경하고 외부에 응답할 수 있는 행동을 내부에 함께 보관한다.

객체는 상태와 행동을 하나의 단위로 묶는 자율적인 실체라는 관점에서의 캡슐화를 데이터 캡슐화라고 한다.

- 사적인 비밀의 캡슐화
  객체는 외부의 객체가 자신의 내부 상태를 직접 관찰하거나 제어할 수 없도록 막기 위해 의사소통 가능한 특별한 경로만 외부에 노출한다.

6️⃣ 책임의 자율성이 협력의 품질을 결정한다

객체들이 동일한 목적을 달성하기 위해 협력하는 방법의 가짓수는 수도 없이 많을 수 있다. 하지만 어떤 협력이 다른 협력보다 좀 더 나은 설계라고 이야기한다. 그 기준이 무엇일까?

첫째, 자율적인 책임은 협력을 단순하게 만든다.

둘째, 자율적인 책임은 객체의 외부와 내부를 명확하게 분리한다.

셋째, 책임이 자율적일 경우 책임을 수행하는 내부적인 방법을 변경하더라도 외부에 영향을 미치지 않는다.

넷째, 자율적인 책임은 협력의 대상을 다양하게 선택할 수 있는 유연성을 제공한다.

다섯째, 객체가 수행하는 책임들이 자율적일수록 객체의 역할을 이해하기 쉬워진다.

7️⃣ 마무리

객체지향은 이런 부분들이 장점이야! 라고 했을 때 왜 장점인데? 라는 생각이 들면 5장을 읽으면 된다.

근데 같은 맥락의 얘기가 너무 반복되는 느낌이긴 하다..!