4장. 역할, 책임, 협력

객체들 간의 협력에 집중하라

- 인간처럼 객체의 세계에서도 **협력**이라는 문맥이 객체의 행동 방식을 결정한다.
- 객체지향 설계의 전체적인 품질은 여러 객첸르이 모여 이뤄내는 협력의 품질이다.
- 훌륭한 객체지향 설계는 서로 조화를 이루며 적극적으로 상호작용하는 협력적인 객체를 창조하는 것이다.

협력

- 협력은 다수의 연쇄적인 요청과 응답의 흐름으로 구성된다.
    - 스스로 해결하기 어려운 일은 다른 누군가에게 요청 → 요청받은 일을 처리한 후 응답

🥧 누가 파이를 훔쳤지?

> *__앨리스__가 재판장에 도착했을 때 __하트 왕__과 __하트 여왕__은 옥좌에 앉아 있었다.*
> *그리고 주위에는 온갖 종류의 새와 짐승과 한 벌의 카드가 모여 있었으며 그들 앞에 파이를 훔쳤다는 혐의를 받고 있는 하트 잭이 사슬에 묶인 채*
> *__병사들__의 감시를 받으며 서 있었다.*
> *하트 잭의 옆에는*
> *__하얀 토끼__가 한 손에는 트럼펫을, 다른 한 손에는 양피지 두루마리를 들고 서 있었다.*

→ 모든 객체들(등장인물)은 '하트 잭의 재판'이라는 동일한 목적을 달성하기 위해 협력하고 있다.

🎩 **왕이 모자 장수로부터 증언을 듣는 과정**

- **[요청]** 누군가 → 왕에게 재판을 요청
- **[요청]** 왕 → 하얀토끼에게 증인을 부를 것을 요청
- **[요청]** 하얀토끼 → 모자장수에게 증인석으로 입장할 것을 요청
- **[응답]** 모자장수 → 하얀토끼에게 증인석에 입장함으로써 응답
- **[요청]** 왕 → 모자장수에게 증언할 것을 요청
- **[응답]** 모자장수 → 왕에게 내용을 증언함으로써 응답

- 특정한 요청을 받아들일 수 있는 이유는 그 요청에 대해 적절하게 응답하는 데 필요한 지식과 행동 방식을 가지기 때문이다.
- 그리고 요청과 응답은 협력에 참여하는 객체가 수행할 책임을 정의한다.

책임

- 객체지향 세계에서 '책임을 가진다'의 의미
  - 어떤 객체가 어떤 요청에 대해 대답해 줄 수 있다.
  - 적절한 행동을 할 의무가 있다.

책임의 분류

- 객체의 책임은 '**객체가 무엇을 알고 있는가(knowing)**'와 '**무엇을 할 수 있는가(doing)**'로 구성된다.
  - **아는 것(knowing)**
    - 개인적인 정보에 관해 아는 것
    - 관련된 객체에 관해 아는 것
    - 자신이 유도하거나 계산할 수 있는 것에 관해 아는 것
  - **하는 것(doing)**
    - 객체를 생성하거나 계산을 하는 등 스스로 하는 것
    - 다른 객체의 행동을 시작시키는 것
    - 다른 객체의 화동을 제어하고 조절하는 것

- 객체의 책임을 이야기할 때는 일반적으로 외부에서 접근 가능한 공용 서비스의 관점에서 이야기한다.
  - 즉, 책임은 외부에 제공해줄 수 있는 정보(아는 것), 외부에 제공해줄 수 있는 서비스(하는 것)의 목록이다.
  - 책임은 객체의 **공용 인터페이스**를 구성한다.

책임과 메시지

- 협력 안에서 객체는 다른 객체로부터 요청이 전송되었을 경우에만 자신에게 주어진 책임을 수행한다.
  - 다른 객체에 요청을 보내는 것을 **메시지 전송**이라고 한다.
  - 따라서 두 객체 간의 협력은 **메시지**를 통해 이뤄진다.
- 메시지는 협력에 참여하는 두 객체 사이의 관계를 강조한다.
- 책임과 메시지의 수준이 같지는 않다.
  - 책임을 결정한 후 이를 메시지로 변환할 때는 하나의 책임이 여러 메시지로 분할되는 것이 일반적이다.
  - 설계 초반에는 책임을 구현하는 방법까지는 섣부르게 고민하지 않도록 한다.

역할

- 협력의 관점에서 어떤 객체가 어떤 책임의 집합을 수행한다는 것이 무엇을 의미할까?
  - 어떤 객체가 수행하는 책임의 집합은 객체가 협력 안에서 수행하는 **역할**을 암시한다.
  - **역할**은 재사용 가능하고 유연한 객체지향 설계를 낳는 매우 중요한 구성요소이다.

👨🏻‍⚖️ 판사와 증인

> *"다음 증인을 불러라"*
> *왕이 하얀 토끼에게 명령했다. 다음 증인은 공작 부인의 요리사였다.*
> *"증언하라."*
> *왕이 말했다.*
> *"싫어요"*
> *왕이 크게, 안심한 목소리로 하얀 토끼에게 다시 명령했다.*

> *"신경 쓸 것 없다! 다음 증인을 불러라"*
> *그리고 왕은 목소리를 낮춰서 여왕에게 말했다.*
> *"그런데 여보, 다음 증인은 당신이 심문해야만 되겠고. 나는 머리가 너무 아파졌다오."*

증언을 하는 과정

- 누군가 **왕**에게 재판을 요청함으로써 재판이 시작되었다.
- **왕**이 하얀 토끼에게 증인을 부를 것을 요청했고, **모자 장수**가 증인석으로 입장했다.
- **모자 장수**는 자신이 알고 있는 내용을 증언했다.

협력에 참여하는 사람이 바뀐다면

- 이 이야기에서 왕 대신 여왕이, 모자장수 대신 요리사나 앨리스가 협력에 참여하게 된다.
- 그렇다면 이 세 개의 협력은 각각 별도로 관리하고 유지해야하는가?

  → 역할이 답이다.

역할이 답이다

- 위 증언 과정에서 객체를 역할로 치환하면 하나의 협력으로 **추상화**할 수 있다.
  - 누군가 **판사**에게 재판을 요청함으로써 재판이 시작되었다.
  - **판사**가 하얀 토끼에게 증인을 부를 것을 요청했고, **증인**이 증인석으로 입장했다.
  - **증인**은 자신이 알고 있는 내용을 증언했다.
- 역할은 협력 내에서 다른 객체로 대체할 수 있음을 나타내는 표식이다.
  - 역할을 이용해 협력을 추상화했기 때문에 어떤 객체라도 협력에 참여할 수 있다.
- 어떤 객체라도 동일한 역할을 대체할 수 있을까? 🤦‍♀️
  - 역할을 대체하기 위해서는 동일한 메시지를 이해하고 처리할 수 있는 객체여야 한다.

협력의 추상화

- 역할을 사용하면 유사한 협력을 추상화함으로써 단순화할 수 있다.
  - 다양한 객체들이 협력에 참여할 수 있기 때문에 협력이 유연해진다.
  - 다양한 객체들이 동일한 협력에 참여할 수 있기 때문에 재사용성이 높아진다.

- 역할은 객체지향 설계의 **단순성(simplicity)**, **유연성(flexibility)**, **재사용성(reusability)**을 뒷받침한다.

대체가능성

- 본질적으로 역할은 다른 객체에 의해 대체 가능함을 의미한다.
  - 객체가 역할을 대체 가능하기 위해서는 모든 책임을 동일하게 수행할 수 있어야 한다.
  - 객체는 역할에 주어진 책임 뿐 아니라 다른 책임을 수행할 수도 있다. (일반화/특수화 관계)
- 대체가능성 == 행위 호환성 == 동일한 책임의 수행

객체의 모양을 결정하는 협력

객체지향에 대한 선입견

- **객체는 데이터를 담기 위해 존재한다?!**
  - 데이터는 객체가 행위를 수행하는 데 필요한 재료일 뿐이다.
  - **객체의 존재 이유는 행위를 수행하며 협력에 참여하기 위함, 즉 책임이다.**
- **객체지향은 클래스와 클래스 간의 관계를 표현하는 시스템의 정적인 측면에 중점을 둔다?!**
  - 중요한 것은 동적인 객체이며, 클래스는 단지 프로그래밍 언어가 제공하는 구현 메커니즘일 뿐이다.
  - **객체지향의 핵심은 객체가 협력 안에서 어떤 책임과 역할을 수행할 것인지 결정하는 것이다.**

협력을 따라 흐르는 객체의 책임

- 올바른 객체를 설계하기 위해서는
  - 먼저, 견고하고 깔끔한 협력을 설계해야 한다.
    - 객체들이 주고받을 요청과 응답의 흐름을 결정한다는 의미
  - 그 다음, 객체에게 책임을 할당한다. (객체가 외부에 제공하게 될 행동)
  - 그 다음, 행동을 수행하는 데 필요한 데이터를 고민한다.
  - 마지막으로, 클래스의 구현 방법을 결정한다.

- 객체의 행위에 초점을 맞추기 위해서는 협력이라는 실행 문맥 안에서 책임을 분배해야 한다.
    - 객체를 충분히 협력적으로 만드는 것이 최우선이다.
    - 그 다음 협력이라는 문맥 안에서 객체를 충분히 자율적으로 만들어라.

객체지향 설계 기법

책임-주도 설계(Responsibilityb-Driven Design)

> 협력에 필요한 책임들을 식별하고, 적합한 객체에게 책임을 할당한다.

- 객체지향 설계란 기능을 구현하기 위해 협력 관계를 고안하고, 역할과 책임을 식별하여 적적한 객체를 식별하는 과정이다.
- 책임-주도 설계는 말 그대로 책임을 중심으로 시스템을 구축하는 설계 방법을 말한다.
    - 기능은 더 작은 규모의 책임으로 분할되고, 각 책임은 적절한 객체에게 할당된다.
    - 만약 책임을 여러 종류의 객체가 수행할 수 있다면 협력자는 객체가 아니라 추상적인 역할로 대체된다.
- 역할 책임, 협력은 유연하고 견고한 객체지향 시스템을 만드는 데 가장 중요한 재료다.
    - 그 외는 보조 재료일 뿐이다.

디자인 패턴

> 전문가들이 반복적으로 사용하는 해결 방법을 정의해 놓은 설계 템플릿의 모음이다.

- 디자인 패턴은 반복적으로 발생하는 문제와 그 문제에 대한 해법의 쌍으로 정의된다.

COMPOSITE 패턴

- 전체와 부분을 하나의 단위로 추상화해야 하는 경우 사용하는 패턴이다.
- 부분과 전체가 투명하고 동일한 인터페이스를 제공해야 한다는 제약하에서 식별된 역할, 책임, 협력을 제공한다.

테스트-주도 개발(Test-Driven Development)

> 테스트를 먼저 작성하고 테스트를 통과하는 구체적인 코드를 추가하면서 애플리케이션을 완성해나간다.

- 책임을 수행할 때 기대하는 역할이 메시지를 수신할 때 어떤 결과를 반환하고, 그 과정에서 어떤 객체와 협력할 것인지에 대한 기대를 테스트 코드의 형태로 작성하는 것이다.
- 테스트-주도 개발은 객체지향에 대한 깊이 있는 지식을 요구한다.
  - 테스트를 작성하기 위해 객체의 메서드를 호출하고 반환값을 검증하는 것 → 책임에 관해 생각하는 것
  - 스텁이나 목 객체를 사용하는 것 → 협력자에 관해 고민하는 것

마치며

- 역할, 책임, 협력 세 가지는 객체지향 설계에서 강력한 개념이다.
- 객체지향을 강력하게 만드는 비밀은 책임과 메시지에 숨겨져 있다. (커밍순)