4. 역할, 책임, 협력

> **“우리 모두를 합친 것보다 더 현명한 사람은 없다.”**

협력

요청하고 응답하며 협력하는 사람들

- **협력**은 한 사람이 다른 사람에게 도움을 **요청**할 때 시작된다.
- 요청을 받은 사람은 요청한 사람에게 필요한 지식이나 서비스를 제공하는 것으로 요청에 **응답**한다.
- 협력은 다수의 연쇄적인 요청과 응답의 흐름으로 구성된다.

누가 파이를 훔쳤지?

- “이상한 나라의 앨리스” 中 파이를 훔친 하트 잭에 대한 공판이 열리고 있는 법정 모습
    - *파이를 훔친 하트 잭을 재판하는 법정에는 왕과 하얀 토끼, 모자 장수, 하트 잭이 있다.*
    - *왕은 하얀 토끼에게 증인 모자 장수를 불러오라고 명령한다.*
    - *하얀 토끼는 모자 장수를 불러오고, 모자 장수는 자신이 알고 있는 내용을 증언한다.*
- 이야기에 등장하는 객체들은 하트 잭의 재판이라는 **동일한 목적을 달성하기 위해 협력**하고 있다.

재판속의 협력

- 왕이 모자 장수로부터 증언을 듣는 과정
    1. 누군가가 왕에게 재판을 **요청**함으로써 재판이 시작된다.
    2. 왕이 하얀 토끼에게 증인을 부를 것을 **요청**한다.
    3. 왕의 요청을 받은 토끼는 모자 장수에게 증인석으로 입장할 것을 **요청**한다.
        a. 모자 장수는 증인석에 입장함으로써 토끼의 요청에 **응답**한다.
        b. 모자 장수의 입장은 왕이 토끼에게 요청했던 증인 호출에 대한 **응답**이기도 하다.
    4. 이제 왕은 모자 장수에게 증언할 것을 **요청**한다.
        a. 모자 장수는 자신이 알고 있는 내용을 증언함으로써 왕의 요청에 **응답**한다.
- 어떤 등장인물들이 **특정한 요청을 받아들일 수 있는 이유는 그 요청에 대해 적절한 방식으로 응답하는 데 필요한 지식과 행동 방식을 가지고 있기 때문**이다.
- 요청과 응답은 협력에 참여하는 객체가 수행할 책임을 정의한다.

책임

- 어떤 객체가 어떤 요청에 대해 대답해줄 수 있거나, 적절한 행동을 할 의무가 있는 경우 해당 객체가 **책임**을 가진다고 말한다.
- 어떤 대상에 대한 요청은 그 대상이 요청을 처리할 책임이 있음을 암시한다.

책임의 분류

- 크레이그 라만은 객체의 책임을 크게 ‘하는 것’과 ‘아는 것’의 두 가지 범주로 분류한다.

    - **하는 것(doing)**
        - 객체를 생성하거나 계산을 하는 등의 스스로 하는 것
        - 다른 객체의 **행동**을 시작시키는 것
        - 다른 객체의 **활동**을 제어하고 조절하는 것
    - **아는 것(knowing)**
        - 개인적인 정보에 관해 아는 것
        - 관련된 객체에 관해 아는 것
        - 자신이 유도하거나 계산할 수 있는 것에 관해 아는 것
- 책임은 객체의 외부에 제공해줄 수 있는 정보(아는 것)와 서비스(하는 것)의 목록이다.
- 따라서 책임은 객체의 **공용 인터페이스(public interface)**를 구성한다.

책임과 메시지

- **메시지 전송**: 객체가 다른 객체에게 주어진 책임을 수행하도록 요청을 보내는 것
    - **송신자**: 메시지를 전송함으로써 협력을 요청하는 객체
    - **수신자**: 메시지를 받아 처리하는 객체
    - 메시지는 협력을 위해 한 객체가 다른 객체로 접근할 수 있는 유일한 방법
- 왕과 모자 장수가 협력할 수 있는 이유는 왕이 모자 장수가 이해할 수 있는 메시지를 전송할 수 있고, 모자 장수는 왕이 전송하는 메시지에 대해 적절한 책임을 수행할 수 있기 때문이다.
- 책임과 메시지의 수준이 같지는 않다.
    - 책임은 객체가 협력에 참여하기 위해 수행해야 하는 행위를 상위 수준에서 개략적으로 서술한 것이기 때문에 **하나의 책임이 여러 메시지로 분할되는 것이 일반적**이다.

역할

책임의 집합이 의미하는 것

- 어떤 객체가 수행하는 책임의 집합은 객체가 협력 안에서 수행하는 역할을 암시한다.

판사와 증인

- “이상한 나라의 앨리스” 中 파이를 훔친 하트 잭에 대한 공판이 열리고 있는 법정 모습

    - *모자 장수의 증언이 끝난 후 하얀 토끼는 다음 증인으로 요리사를 부른다.*
    - *요리사가 증언을 마치고나서 왕은 여왕에게 재판을 대신할 것을 요청한다.*
    - *여왕은 앨리스를 재판하며, 앨리스는 알고 있는 내용을 증언한다.*
- "왕" 대신 "여왕이", "모자 장수" 대신 "요리사"나 "앨리스"가 협력에 참여한다는 차이를 제외하면 각각의 재판 과정은 모두 동일하다.

역할이 답이다

- 재판 과정에서 '판사'와 '증인'이라는 **역할(role)**을 사용하면 위에서 이야기한 세 가지 협력을 모두 포괄할 수 있는 하나의 협력으로 추상화할 수 있다.
- 역할을 대체할 수 있는 객체는 동일한 메시지를 이해할 수 있는 객체로 한정된다.
- **동일한 역할을 수행하는 객체들이 동일한 메시지를 수신할 수 있기 때문에 동일한 책임을 수행할 수 있다.**
- 역할은 객체지향 설계의 **단순성(simplicity)**, **유연성(flexibility)**, **재사용성(reusability)**을 뒷받침하는 핵심 개념이다.

협력의 추상화

- 역할의 가장 큰 가치는 하나의 협력 안에 여러 종류의 객체가 참여할 수 있게 함으로써 협력을 **추상화**할 수 있다는 것이다.

대체 가능성

- 객체가 역할을 대체 가능하기 위해서는 협력 안에서 역할이 수행되는 모든 책임을 동일하게 수행할 수 있어야 한다.
- 객체는 역할에 주어진 책임보다 더 많은 책임을 가질 수 있다.
    - 따라서 대부분의 경우에 객체의 타입과 역할 사이에는 **일반화/특수화 관계**가 성립한다.
        - 객체의 역할 - 일반화
        - 객체의 타입 - 특수화

객체의 모양을 결정하는 협력

흔한 오류

1. ~~시스템에 필요한 데이터를 저장하기 위해 객체가 존재한다.~~

   → 객체가 존재하는 이유는 행위를 수행하며 협력에 참여하기 위해서다.

2. ~~객체지향의 핵심은 클래스와 클래스 간의 관계를 표현하는 시스템의 정적인 측면에 중점을 두는 것이다.~~

   → 객체지향의 핵심은 객체가 협력 안에서 어떤 책임과 역할을 수행할 것인지를 결정하는 것이다.

협력을 따라 흐르는 객체의 책임

- **올바른 객체 설계 과정**
  1. 견고하고 깔끔한 협력을 설계
     a. 객체들이 주고받을 요청과 응답의 흐름을 결정
  2. 결정된 요청과 응답의 흐름 ⇒ 객체가 협력에 참여하기 위해 수행될 책임
  3. 객체에게 할당한 책임 ⇒ 객체가 외부에 제공하게 될 행동
  4. 객체가 수행하게 될 책임, 즉 행동을 결정한 후 필요한 데이터를 고민
  5. 협력에 필요한 데이터와 행동이 결정된 후 클래스의 구현 방법을 결정
- 객체지향 시스템에서 가장 중요한 것은 **충분히 자율적인 동시에 충분히 협력적인 객체를 창조**하는 것이다.
- 이 목표를 달성할 수 있는 가장 쉬운 방법은 객체를 충분히 협력적으로 만든 후에 협력이라는 문맥 안에서 객체를 충분히 자율적으로 만드는 것이다.

객체지향 설계 기법

- 객체지향 설계란
  - 애플리케이션의 기능을 구현하기 위한 협력 관계를 고안하고,
  - 협력에 필요한 역할과 책임을 식별한 후
  - 이를 수행할 수 있는 적절한 객체를 식별해 나가는 과정

책임-주도 설계

- 레베카 워프스브록이 고안했으며, 말 그대로 **객체의 책임을 중심으로 시스템을 구축**하는 설계 방법이다.
- 개별적인 객체의 상태가 아니라 객체의 책임과 상호작용에 집중한다.
- **객체의 책임, 역할, 협력을 고안하기 위한 절차**
  1. 시스템이 사용자에게 제공해야 하는 기능인 시스템 책임을 파악한다.
  2. 시스템 책임을 더 작은 책임으로 분할한다.

3. 분할된 책임을 수행할 수 있는 적절한 객체 또는 역할을 찾아 책임을 할당한다.
4. 객체가 책임을 수행하는 중에 다른 객체의 도움이 필요한 경우 이를 책임질 적절한 객체 또는 역할을 찾는다.
5. 해당 객체 또는 역할에게 책임을 할당함으로써 두 객체가 협력하게 한다.

디자인 패턴

- 디자인 패턴은
  - 책임-주도 설계의 결과물인 동시에 지름길이다.
  - 모범이 되는 설계(example design)다.
  - 반복적으로 발생하는 문제와 그 문제에 대한 해법의 쌍으로 정의된다.
  - 반복해서 일어나는 특정 상황에서 어떤 설계가 왜 더 효과적인지에 대한 이유를 설명한다.
  - 공통으로 사용할 수 있는 역할, 책임, 협력의 템플릿이다.
- 디자인 패턴의 한 가지 예, **COMPOSITE** 패턴
  - 전체와 부분을 하나의 단위로 추상화해야 하는 경우에 사용할 수 있는 패턴

테스트-주도 개발

- 테스트-주도 개발은
  - 실패하는 테스트를 작성하고, 테스트를 통과하는 가장 간단한 코드를 작성한 후, 리팩토링을 통해 중복을 제거하는 것이다.
  - 테스트를 작성하는 것이 아니라 책임을 수행할 객체 또는 클라이언트가 기대하는 객체의 역할이 메시지를 수신할 때 어떤 결과를 반환하고, 그 과정에서 어떤 객체와 협력할 것인지에 대한 기대를 코드의 형태로 작성하는 것이다.

💭 후기

- 계속해서 “요청과 응답이 가능한 이유는 대상이 요청을 처리할 책임이 있기 때문”이라는 것을 언급한다.

  → 협력을 발견한다면, 책임은 자연스레 따라온다?
- 책임의 분류 중 **아는 것**은 **getter**를 통해 상태를 제공하는 것을 의미하는건가?
- 결국 좋은 객체지향 설계 순서는 **협력** → **책임** → **역할**
- 디자인 패턴과 TDD는 어렵다..