역할, 책임, 협력

최혁

객체지향 패러다임의 핵심은 역할, 책임, 협력이다.

협력을 구성하기 위해 적절한 객체를 찾고 적절한 책임을 할당하는 과정을 통해 객체지향의 본질이 들어난다.

-> 위를 고려하지 않고 구현에 초점을 맞추면 변경하기 어렵고 유연하지 못한 코드를 만들게 된다.

협력

객체들이 애플리케이션의 기능을 구현하기 위해 수행하는 상호작용

- 두 객체 사이의 협력은 **메시지 전송** 이라는 유일한 커뮤니케이션 수단을 통해 이루어진다
- 메시지를 수신한 객체는 **메서드** 를 실행해 요청에 응답한다
- 외부의 객체는 메시자만 전송할 수 있고, 메시지를 어떻게 처리할지는 **메시지를 수신할 객체가 직접 결정** 한다

객체

상태와 행동을 함께 캡슐화하는 실행 단위

객체의 행동을 결정하는 것은 객체가 참여하고 있는 협력이고, *(정확히 협력 안에서 객체가 처리할 메시지로 결정된다)* 객체의 상태를 결정하는 것은 행동이다.

객체의 상태는 행동을 수행하는 데 필요한 정보에 따라 결정된다.

협력은 객체를 설계하는 데 필요한 문맥(context)를 제공한다.
(영화 예매를 위한 협력 / Movie 객체)

협력 -> 행동 -> 상태

책임

협력에 필요한 객체를 찾을 때, 협력에 참여하기 위해 객체가 수행해야 하는 행동

- 객체가 유지해야 하는 정보와 수행할 수 있는 행동에 대해 개략적으로 서술한 문장
- [doing] [knowing]으로 세분화할 수 있다
- 협력 안에서 객체에게 할당한 책임이 외부의 인터페이스와 내부의 속성을 결정한다.

책임과 메서드

- 책임은 객체가 수행할 수 있는 행동을 종합적이고 간략하게 서술하기에 메시지 보다 추상적이고 개념적으로 더 크다.
- 하나의 메서드로 해결될 것 같은 단순한 책임이 여러 메서드로 분할되기도 함
- 하나의 객체가 수행할 수 있다고 생각했던 책임이 나중에는 여러 객체들이 협력해야만 하는 커다란 책임으로 자라기도 함

책임 할당

자율적인 객체를 만드는 가장 기본적인 방법은 책임을 수행하는 데 필요한 정보를 가장 잘 알고 있는 전문가에게 그 책임을 할당하는 것이다. (정보 전문가)패턴

1. 시스템이 사용자에게 제공하는 기능을 하나의 책임으로 본다
2. 시스템의 책임을 완료하는 데 필요한 더 작은 책임을 찾아낸다
3. 이를 적절한 객체들에게 할당한다

객체가 책임을 수행하는 유일한 방법은 메시지를 전송하는 것이다.

다만, 어떤 경우에는 응집도와 결합도의 관점에서 정보 전문가가 아닌 객체에게 책임을 할당하는 것이 더 적절한 경우도 있다.(ch1 step2 -> step3)

책임 주도 설계

> 책임을 찾고 책임을 수행할 적절한 객체를 찾아 책임을 할당하는 방식으로 협력을 설계하는 방식

책임을 할당할 때 고려해야 할 요소

1. **객체가 메시지를 선택하는 것이 아닌, 메시지가 객체를 선택하게 한다.**
   최소한의 인터페이스를 가질 수 있게 된다.
   충분히 추상적인 인터페이스를 가질 수 있게 된다.

2. **행동이 상태를 결정하게 한다.**
   만약 상태에 초점을 맞춰 행동을 결정할 경우 객체의 내부 구현이 퍼블릭 인터페이스에 노출되기에 캡슐화를 저해한다.

역할

객체가 어떤 특정한 협력 안에서 수행하는 **책임의 집합**

역할을 통해 유연하고 재사용 가능한 협력을 얻을 수 있기 때문에 역할이 중요하다.

할인 요금을 계산하라(책임) -> 금액 할인 정책을 사용하는 협력 and 퍼센트 할인 정책을 사용하는 협력 -> 많은 중복!!

역할은 다양한 종류의 객체를 수용할 수 있는 일족의 슬롯이자 구체적인 객체들의 타입을 캡슐화하는 추상화이다.

역할?

오직 한 종류의 객체만 협력에 참여하는 상황에서 역할이라는 개념을 고려하는 것이 유용할까?

-> 협력에 참여하는 후보가 여러 종류의 객체에 의해 수행될 필요가 있다면 그 후보는 역할이 되지만, 단 한 종류의 객체만이 협력에 참여할 필요가 있다면 후보는 객체가 된다.

결국 역할과 객체를 명확히 구분하는 것은 그렇게 중요하지 않고, 적절한 책임과 협력의 큰 그림을 탐색하는 것이 중요하다!

배우와 배역

연극 안에서 배역을 연기하는 배우라는 은유는 협력 안에서 역할을 수행하는 객체라는 관점이 가진 입체적인 측면들을 훌륭하게 담아낸다. 배우가 여러 연극을 참여하면서 여러 배역을 연기할 수 있는 것처럼 객체 역시 여러 협력에 참여하면서 다양한 역할을 수행할 수 있다. 객체는 여러 역할을 가질 수 있지만 특정한 협력 안에서는 일시적으로 오직 하나의 역할만 보여진다는 것을 주의해야 한다.

왜 역할, 책임, 협력을 통해 설계해야 하지?

객체의 책임에 초점을 맞춘 설계는 낮은 결합도와 높은 응집도를 가진 구조를 갖게 되기에 유연하고 변경하기 쉽다. 책임은 객체의 상태에서 행동으로, 나아가 객체와 객체 사이의 상호작용으로 설계 중심을 이동시키고, 결합도가 낮고 응집도가 높으며 구현을 효과적으로 캡슐화하는 객체들을 창조할 수 있는 기반을 제공한다.