3. 타입과 추상화

↗ Books	객체지향의 사실과 오해

> *일단 컴퓨터를 조작하는 것이 추상화를 구축하고, 조작하고, 추론하는 것에 관한 모든 것이라는 것을 깨닫고 나면 (훌륭한) 컴퓨터 프로그램을 작성하기 위한 중요한 전제 조건은 추상화를 정확하게 다루는 능력이라는 것이 명확해진다.*
> *- 키스 데블린(Keith Devlin)*

이동 수단으로써 지하철을 큰 어려움 없이 이용할 수 있는 이유는 단순하고 직관적으로 역 간의 네트워크를 표현하는 지하철 노선도가 있기 때문이다. ~~(길치는 여려운데요 .. ?)~~

해리 벡이 창조한 지하철 노선도는 모든 역의 위치와 거리도 부정확하고 경로를 표시한 선은 실제 지하철이 이동하는 경로와는 전혀 상관이 없었지만 이동이라는 '목적'에 집중했기 때문에 이해하기 쉽고 단순하게 창조할 수 있었다.

1️⃣ 추상화를 통한 복잡성 극복

진정한 의미에서 추상화란 현실에서 출발하되 불필요한 부분을 도려내가면서 사물의 놀라운 본질을 드러나게 하는 과정이라고 할 수 있다.
추상화의 목적은 불필요한 부분을 무시함으로써 현실에 존재하는 복잡성을 극복하는 것이다.

이 책에서는 추상화를 다음과 같이 정의한다.

> 💡 추상화
> 어떤 양상, 세부 사항, 구조를 좀 더 명확하게 이해하기 위해 특정 절차나 물체를 의도적으로 생략하거나 감춤으로써 복잡도를 극복하는 방법이다.
> 복잡성을 다루기 위해 추상화는 두 차원에서 이뤄진다.
>
> - 첫 번째 차원은 구체적인 사물들 간의 공통점은 취하고 차이점은 버리는 일반화를 통해 단순하게 만드는 것이다.
> - 두 번째 차원은 중요한 부분을 강조하기 위해 불필요한 세부 사항을 제거함으로써 단순하게 만드는 것이다.
>
> 모든 경우에 추상화의 목적은 복잡성을 이해하기 쉬운 수준으로 단순화하는 것이라는 점을 기억하라.

2️⃣ 객체지향과 추상화

모두 트럼프일 뿐

앨리스는 하얀 토끼를 제외한 모든 객체를 '트럼프'라는 하나의 개념으로 단순화해서 바라본다.

계급, 나이, 성격 등의 차이점은 무시한 채 '트럼프'라는 유사성을 기반으로 추상화해서 바라보고 있는 것이다.

그룹으로 나누어 단순화하기

정원사, 병사, 신하, 하트 왕, 하트 여왕 등의 다양한 인물들을 '트럼프'라는 한 단어로 지칭할 수 있는 이유는 공통적으로 '트럼프'라고 했을 때 떠오르는 일반적인 외형과 행동 방식을 지니고 있기 때문이다.

앨리스는 정원에 있는 인물들을 트럼프 그룹과 토끼 그룹으로 나눴다.
토끼 그룹에 속하는 등장인물이 하나뿐이라고 해도 그룹으로 나누어 바라보는 것은 복잡성을 감소시킨다.

개념

객체와 인스턴스의 차이가 이 부분에서 잘 나와 있다고 생각한다.

이처럼 공통점을 기반으로 객체들을 묶기 위한 그릇을 개념(concept)이라고 한다.

위에서 설명한 앨리스의 이야기로 살펴보면 몸이 납작하고 두 손과 두 발이 네모난 몸 모서리에 달려 있는 객체만을 트럼프라는 개념으로 추상화한 것이다.

개념을 이용하면 객체를 여러 그룹으로 분류(classification)할 수 있다. 즉, 개념은 공통점을 기반으로 객체를 분류할 수 있는 일종의 체라고 할 수 있다.

결국 각 객체는 특정한 개념을 표현하는 그룹의 일원으로 포함되는데 이 때 객체를 그 개념의 인스턴스(instance)라고 한다.

개념의 세 가지 관점

일반적으로 객체의 분류 장치로서 개념을 이야기할 때는 아래 세 가지 관점을 함께 언급한다.

- 심볼(symbol): 개념을 가리키는 간략한 이름이나 명칭
- 내연(intension): 개념의 완전한 정의를 나타내며 내연의 의미를 이용해 객체가 개념에 속하는지 여부를 확인할 수 있다.
- 외연(extension): 개념에 속하는 모든 객체의 집합(set)

앨리스의 이야기로 살펴보면 아래와 같이 정리할 수 있다.

- 심볼: 트럼프
- 내연: 몸이 납작하고 두 손과 두 발은 네모 귀퉁이에 달려 있는 등장인물

- 외연: 정원사, 병사, 신하, 왕자와 공주, 하트 왕과 하트 여왕 등

객체를 분류하기 위한 틀

어떤 객체를 어떤 개념으로 분류할지가 객체지향의 품질을 결정한다. 객체를 적절한 개념에 따라 분류하지 못한 애플리케이션은 유지보수가 어렵고 변화에 쉽게 대처하지 못한다.

3️⃣ 타입

타입은 개념이다

타입의 정의는 개념의 정의와 완전히 동일하지만 컴퓨터 내부로 들어오면 다르다.

데이터 타입

메모리의 상의 모든 데이터는 일련의 비트열로 구성되기 때문에 '10010001'이라는 값이 숫자인지, 문자인지 혹은 특정한 메모리 상의 주소인지 구분하기 어렵다.
이런 문제를 해결하기 위해 데이터의 용도와 행동에 따라 분류하는 타입 시스템(type system)이 생겼다.
결과적으로 타입 시스템의 목적은 데이터가 잘못 사용되지 않도록 제약사항을 부과하는 것이다.

지금까지 이야기한 내용을 통해 타입에 관련된 두 가지 중요한 사실을 알 수 있다.

1. 타입은 데이터가 어떻게 사용되느냐에 관한 것이다.
   어떤 연산자를 적용할 수 있느냐가 그 데이터의 타입을 결정한다.
2. 타입에 속한 데이터를 메모리에 어떻게 표현하는지는 외부로부터 철저하게 감춰진다.
   개발자는 해당 데이터 타입을 사용하기 위해 적용할 수 있는 연산자만 알고 있으면 된다.

객체와 타입

객체를 타입에 따라 분류하고 그 타입에 이름을 붙이는 것은 결국 프로그램에서 사용할 새로운 데이터 타입을 선언하는 것과 같다.

앞서 데이터 타입에 관해 언급한 두 가지 조언은 객체의 타입을 이야기할 때도 동일하게 적용된다.

1. 어떤 객체가 어떤 타입에 속하는지를 결정하는 것은 객체가 수행하는 행동이다.
   어떤 객체들이 동일한 행동을 수행할 수 있다면 그 객체들은 동일한 타입으로 분류될 수 있다.
2. 객체의 내부적인 표현은 외부로부터 철저하게 감춰진다.
   객체의 행동을 효과적으로 수행할 수만 있다면 객체 내부의 상태를 어떤 방식으로 표현하더라도 무방하다.

정리하자면 객체의 타입을 결정하는 것은 행동 뿐이며 어떤 데이터를 보유하고 있는지는 타입을 결정하는 데 아무런 영향도 미치지 않는다.

이 두 가지 조언으로부터 객체지향 설계에 대한 중요한 원칙을 이끌어낼 수 있다.

행동이 우선이다

타입이 데이터가 아니라 행동에 의해 결정된다는 사실은 객체지향 패러다임을 특정 짓는 중요한 몇 가지 원리와 원칙에 의미를 부여한다.

첫 번째는 다형성이다.
동일한 타입에 속한 객체는 내부의 데이터 표현 방식이 다르더라도 동일한 메시지를 수신하고 이를 처리할 수 있다. 다만 내부의 표현 방식이 다르기 때문에 처리하는 방식은 서로 다를 수밖에 없다.

두 번째는 캡슐화다.
행동만이 고려 대상이라는 사실은 외부에 데이터를 감춰야 한다는 것을 의미한다.

이렇게 행동을 먼저 생각하고 적합한 데이터를 나중에 결정하는 방법을 책임-주도 설계(RDD, Responsibility-Driven Design)라고 한다.

4️⃣ 타입의 계층

트럼프 계층

지금까지 '트럼프'라고 분류했던 인물들은 사실 '트럼프 인간'에 가깝다.
트럼프 타입의 객체가 할 수 있는 행동에 추가적으로 걸어다니는 등 인간이 할 수 있는 행동이 더해졌기 때문이다.

즉, 트럼프는 트럼프 인간을 포괄하는 좀 더 일반적인 개념이고 이러한 관계를 일반화/특수화 관계라고 한다.

일반화/특수화 관계

일반화/특수화 관계를 결정하는 것은 행동이다.
특수한 타입의 행동 = 일반 타입의 행동 **+ 특수한 행동**

슈퍼타입과 서브타입

일반적인 타입 = 슈퍼타입, 특수한 타입 = 서브타입

> 💡 슈퍼-서브타입과 상속이 같은 개념이라고 생각이 들어서 찾아봤다.
> 같은 개념이지만 사용하는 목적이 다르다고 이해했다.
> 상속은 객체지향 프로그래밍에서, 슈퍼-서브타입은 관계형 데이터베이스에서 사용한다고 하는데 왜 이 책에서 슈퍼-서브타입을 설명하는지 궁금하다.

5️⃣ 정적 모델

타입의 목적

타입을 사용하면 동적으로 변하는 객체의 상태를 정적인 관점에서 표현할 수 있다.
시간에 따라 변하는 앨리스의 키 값을 나열하는 대신 가변값인 키를 가지고 있다고 나타내는 것이다.

그러면 시간이라는 요소가 제거됨으로써 정적인 모습의 앨리스를 표현할 수 있다.

동적 모델과 정적 모델

스냅샷처럼 객체의 상태가 어떻게 변하고 어떻게 행동하는지를 포착하는 것을 동적 모델이라고 하며,

객체가 가질 수 있는 모든 상태와 모든 행동을 시간에 독립적으로 표현하는 것을 정적 모델이라고 한다.

클래스

객체지향 프로그래밍 언어에서 정적인 모델은 클래스를 이용해 구현된다.
타입을 구현하는 가장 보편적인 방법이 클래스를 이용하는 것이다.

클래스와 타입은 동일한 개념이 아니다. 타입을 구현할 수 있는 방법 중 하나가 클래스인 것이다.

점점 읽을수록 아- 이런거구나 하고 이해하기보다 머리에 그냥 넣게 된다.
내가 책을 읽는 방법이 잘못된걸까? 아니면 아예 모르는 개념들에 대해 읽다보니 자연스럽게 외우는 걸까?