3️⃣

타입과 추상화

> 💡 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

추상화를 통한 복잡성 극복

진정한 의미에서 추상화란 현실에서 출발하되 불필요한 부분을 도려내가면서 사물의 놀라운 본질을 드러나게 하는 과정임

추상화의 목적은 불필요한 부분을 무시함으로써 현실에 존재하는 복잡성을 극복하는 것ㅇ

추상화는 복잡한 현실을 단순화하기 위해 사용하는 인간의 가장 기본적인 인지 수단이라고 할 수 음

어떤 추상화도 의도된 목적이 아닌 다른 목적으로 사용된다면 오도될 수 있다

추상화의 수준, 이익, 자치는 목적에 의존적이다

```
[ 추상화 ]
어떤 양상, 세부 사항, 구조글 좀 더 명확하게 이해하기 위해 특정 절차나 물체
감춤으로써 복잡도를 극복하는 방법이다

복잡성을 다루기 위해 추상화는 두 차원에서 이뤄진다
  - 첫 번째 차원은 구체적인 사물들 간의 공통점은 취하고 차이점은 버리는 일빈
  - 두 번째 차원은 중요한 부분을 강조하기 위해 불필요한 세부 사항을 제거함으
```

```
모든 경우에 추상화의 목적은 복합성을 이해하기 쉬운 수준으로 단순화하는 것이다
```

객체지향 패러다임은 객체라는 추상화를 통해 현실의 복잡성을 극복함

객체지향 패러다임을 이용해 유용하고 아름다운 애플리케이션을 개발하기 위한 첫걸음을 추상화의 두 차원을 올바르게 이해하고 적용하는 것

객체지향과 추상화

개념

- 공통점을 기반으로 객체들을 묶기 위한 그릇을 개념이라고 함
- 개념이란 일반적으로 우리가 인식하고 있는 다양한 사물이나 객체에 적용할 수 있는 아이디어나 관념을 뜻함
    - 거리를 빠른 속도로 누비는 교통수단에 대해서는 '자동차'라는 개념을 적용
    - '책상' 위에 놓여있는 '모니터'에 수놓아지는 다양한 '글자'들을 바라보면 '키보드'를 이용해 '책'을 써내려 가고 있다
    - 앨리스에 나오는 정원사, 병사, 왕자와 공주, 하객으로 참석한 왕과 왕비, 하트 잭, 하트 왕과 하트 여왕 = '트럼프'라는 개념을 적용
- 개념을 이용하면 객체를 여러 그룹으로 분류할 수 있음
    - 개념은 객체를 분류할 수 있는 틀을 제공함
    - 앨리스가 정원에 존재하는 객체를 '트럼프'와 '토끼'라는 두 개념으로 나누고, 두 개념에 적합한 객체가 각 그룹에 포함되도록 분류함
- 객체에 어떤 개념을 적용하는 것이 가능해서 개념 그룹의 일원이 될 때 객체를 그 개념의 인스턴스라고 함
    - 앨리스가 수많은 군상들을 단지 트럼프일 뿐이라고 일축했던 것처럼 주변의 복잡한 객체들은 단지 몇 가지 개념의 인스턴스일 뿐

```
[ 객체 ]
객체란 특정한 개념을 적용할 수 있는 구체적인 사물을 의미한다
```

```
개념이 객체에 적용됐을 때 객체를 개념의 인스턴스라고 함
```

개념의 세 가지 관점

- 개념은 특정한 객체가 어떤 그룹에 속할 것인지를 결정함
- 개념의 세 가지 관점
  - **심볼**: 개념을 가르키는 간략한 이름이나 명칭
  - **내연**: 개념의 완전한 정의이며, 내연의 의미를 이용해 객체가 개념에 속하는지 여부 확인 가능
  - **외연**: 개념에 속하는 모든 객체의 집합
- 개념을 구성하는 심볼, 내연, 외연은 객체의 분류 방식에 대한 지침을 제공함

```
심볼 - 트럼프
내연 - 몸이 납작하고 두 손과 두 발은 네모 귀퉁이에 달려 있는 등장인물
외연 - 정원사, 병사, 신하, 왕자와 공주, 하객으로 참석한 왕과 왕비들, 하트
```

객체를 분류하기 위한 틀

- 외연의 관점에서 어떤 객체에 어떤 개념을 적용할 수 있다는 것은 동일한 개념으로 구성된 객체 집합에 해당 객체를 포함시킨다는 것을 의미함
- 어떤 객체와 마주했을 때 객체에게 적용할 개념을 결정하는 것은 결국 해당 객체를 개념이 적용된 객체 집합의 일원으로 맞아들인다는 것을 의미함
- 객체에 어떤 개념을 적용할 것인지를 결정하는 것은 결국 객체들을 개념에 따라 분류하는 것과 동일함
- 분류란 특정한 객체를 특정한 개념의 객체 집합에 포함시키거나 포함시키지 않는 작업을 의미함

```
[ 분류 ]
분류란 객체에 특정한 개념을 적용하는 작업이다
```

```
객체에 특정한 개념을 적용하기로 결심했을 때 우리는 그 객체를 특정한 집합의
```

- 어떤 객체를 어떤 개념으로 분류할지가 객체지향의 품질을 결정함
- 객체를 적절한 개념에 따라 분류하지 못한 애플리케이션은 유지보수가 어렵고 변화에 쉽게 대처하지 못함

분류는 추상화를 위한 도구다

- 개념을 통해 객체를 분류하는 과정을 추상화의 두 가지 차원을 모두 사용함
  - 정원사, 병사 등을 '트럼프' 라는 개념으로 묶은 것 = 객체 간의 차이점은 무시하고 공통점을 취한 결과
  - '트럼프'에 속하는 객체들의 공통점 중에서도 우리가 중요하다고 생각하는 특징은 납작하고 두 손과 두 발이 네모난 몸 모서리에 달려 있다는 것이고, 나머지 사항들은 무시함 = 불필요한 세부 사항을 제거함
- 개념은 객체들의 복잡성을 극복하기 위한 추상화 도구
- 추상화를 사용함으로써 우리는 극도로 복잡한 이 세상을 그나마 제어 가능한 수준으로 단순화할 수 있음

타입

타입은 개념이다

- 공학자들이 개념을 대체할 수 있은 좀 더 세련돼 보이는 용어를 수학적으로 차용해 왔는데 그게 타입임
- 타입의 정의는 개념과 동일함
- 공통점을 기반으로 객체들을 묶기 위한 틀이며, 심볼, 내연, 외연을 이용해 서술할 수 있고, 타입에 속하는 객체도 타입의 인스턴스라고 함

```
[ 타입 ]
타입은 개념과 동일하다
```

따라서 타입이란 우리가 인식하고 있는 다양한 사물이나 객체에 적용할 수 있는 어떤 객체에 타입을 적용할 수 있을 때 그 객체를 타입의 인스턴스라고 한다 타입의 인스턴스는 타입을 구성하는 외연인 객체 집합의 일원이 된다

데이터 타입

데이터 타입이 생겨난 계기

- 메모리에 불러들여진 데이터들은 무수히 많은 0과 1로 치장되어 메모리에 저장됨
- 메모리의 세상에는 타입이라는 질서가 존재하지 않음
    - 비트열로 구성됨
- 타입 없는 무질서가 초래한 혼돈의 세상에 질려버린 사람들은 메모리 안의 데이터에 특정한 의미를 부여함
- 데이터의 용도와 행동에 따라 분류
- 컴퓨터 안에 살아가는 데이터를 목적에 따라 분류하기 시작하면서 프로그래밍 언어 안에는 서서히 타입 시스템이 자라나기 시작
    - 메모리 안의 모든 데이터가 비트열로 보임으로써 야기되는 혼란을 방지하는 것

데이터 타입의 특징

- 타입은 데이터가 어떻게 사용되느냐에 관한 것
    - 숫자형 데이터가 숫자형인 이유는 데이터를 더하거나 빼거나 곱하거나 나눌 수 있기 때문
    - 데이터가 어떤 타입에 속하는지를 결정하는 것은 데이터에 적용할 수 있는 작업임
    - 연산자의 종류가 아닌 어떤 연산자를 적용할 수 있느냐가 그 데이터의 타입을 결정함
- 타입에 속한 데이터를 메모리에 어떻게 표현하는지는 외부로부터 철저하게 감춰짐
    - 데이터 타입의 표현은 연산 작업을 수행하기에 가장 효과적인 형태임
    - 개발자는 해당 데이터 타입의 표현 방식을 몰라도 데이터를 사용하는 데 지장이 없음
    - 개발자는 해당 데이터 타입을 사용하기 위해 단지 데이터 타입에 적용할 수 있는 연산자만 알고 있으면 됨

```
[ 데이터 타입 ]
데이터 타입은 메모리 안에 저장된 데이터의 종류를 분류하는 데 사용하는 메모리
데이터에 대한 분류는 암시적으로 어떤 종류의 연산이 해당 데이터에 수행될 수
```

객체와 타입

객체는 데이터가 아님

- 객체에서 중요한 것은 객체의 행동임
- 객체가 협력을 위해 어떤 책임을 지녀야 하는지를 결정하는 것이 객체지향 설계의 핵심임

객체의 타입의 특징

- 어떤 객체가 어떤 타입에 속하는지를 결정하는 것은 객체가 수행하는 행동임
  - 어떤 객체들이 동일한 행동을 수행할 수만 있다면 그 객체들은 동일한 타입으로 분류될 수 있음
- 객체의 내부적인 표현은 외부로부터 철저하게 감춰짐
  - 객체의 행동을 가장 효과적으로 수행할 수만 있다면 객체 내부의 상태를 어떤 방식으로 표현하더라도 무방함

행동이 우선이다

- 첫 번째, 두 번째 특징에 따르면 객체의 내부 표현 방식이 다르더라도 어떤 객체들이 동일하게 행동한다면 그 객체들은 동일한 타입에 속함
  - 동일한 책임을 수행하는 일련의 객체는 동일한 타입의 속한다고 할 수 있음
- 객체가 어떤 데이터를 가지고 있는지는 관심사가 아님

어떤 객체를 다른 객체와 동일한 타입으로 분류하는 기준이 무엇인가?

⇒ 그 객체가 타입에 속한 다른 객체와 동일한 행동을 하기만 하면 됨

타입이 데이터가 아니라 행동에 의해 결정된다는 사실은 객체지향 패러다임을 특징 짓는 중요한 몇가지 원리와 원칙에 의미를 부여함

- 같은 타입에 속한 객체는 행동만 동일하다면 서로 다른 데이터를 가질 수 있음
  - 동일한 행동 = 동일한 책임 = 동일한 메시지 수진
  - 동일한 타입에 속한 객체는 내부의 데이터 표현 방식이 다르더라도 동일한 메시지를 수신하고 이를 처리할 수 있음
  - 다만 내부의 표현 방식이 다르기 때문에 동일한 메시지를 처리하는 방식은 서로 다름

⇒ 다형성에 의미를 부여함

다형성

- 동일한 요청에 대해 서로 다른 방식으로 응답할 수 있는 능력
- 동일한 메시지를 서로 다른 방식으로 처리하기 위해서는 객체들은 동일한 메시지를 수신할 수 있어야 하기 때문에 결과적으로 다형적인 객체들은 동일한 타입에 속하게 됨

캡슐화

- 데이터의 내부 표현 방식과 무관하게 행동만이 고려 대상이라는 사실 = 외부 데이터를 감춰야 한다는 것
- 훌륭한 객체지향 설계는 외부에 행동만을 제공하고 데이터는 행동 뒤로 감춰야 함
- 이 원칙을 캡슐화라고 함

책임-주도 설계

- 행동에 따라 객체를 분류하기 위해서는 객체가 내부적으로 관리해야 하는 데이터가 아니라 객체가 외부에 제공해야 하는 행동을 먼저 생각해야 함
- 객체가 외부에 제공해야 하는 책임을 먼저 결정하고 그 책임을 수행하는 데 적합한 데이터를 나중에 결정한 후, 데이터를 책임을 수행하는 데 필요한 외부 인터페이스 뒤로 캡슐화해야 함
- 데이터-주도 설계 방법의 단점을 개선하기 위해 고안됨

객체를 결정하는 것은 행동임

데이터는 단지 행동을 따를 뿐임

⇒ 객체를 개체답게 만드는 가장 핵심적인 원칙

타입의 계층

트럼프 계층

일반적인 트럼프는 걸어다닐 수 없음

앞에서 트럼프 타입이라고 불렀던 객체들을 좀 더 정확하게 트럼프 인간이라는 타입으로 분류할 것

트럼프 타입을 '납작 엎드릴 수 있고 뒤집어질 수 있다'고 정의함

트럼프 인간 타입을 '납작 엎드릴 수 있고 뒤집어질 수 있으며 걸을 때마다 몸이 종이처럼 좌우로 펄럭일 수 있다'고 정의함

트럼프는 트럼프 인간을 포괄하는 좀 더 일반적인 개념

트럼프 인간은 트럼프보다 좀 더 특화된 행동을 하는 특수한 개념

⇒ 이 두 개념 사이의 관계를 일반화/특수화 관계라고 함

일반화/특수화 관계

- 일반화와 특수화는 동시에 일어남
- 일반화/특수화 관계를 결정하는 것은 객체의 상태를 표현하는 데이터가 아니라 행동임
    - 중요한 것은 객체가 내부에 보관한 데이터가 아니라 객체가 외부에 제공하는 행동

행동의 관점에서 더 일반적인 타입이란 무엇이고 더 특수한 타입은 무엇인가?

- 일반적인 타입: 특수한 타입이 가진 모든 행동들 중에서 일부 행동만 가지는 타입
    - 특수한 타입에 비해 더 적은 행동을 가짐
- 특수한 타입: 일반적인 타입이 가진 모든 행동을 포함하지만 거기에 더해 자신만의 행동을 추가하는 타입

    - 일반적인 타입에 비해 더 많은 행동을 가짐
    - 일반적인 타입이 할 수 있는 모든 행동을 동일하게 수행할 수 있어야 함

슈퍼타입과 서브타입

- 슈퍼타입: 일반화/특수화 관계에서 좀 더 일반적인 타입
    - 슈퍼타입의 행동은 서브타입에게 자동으로 상속됨
- 서브타입: 일반화 객체에서 좀 더 특수한 타입
    - 어떤 타입의 서브타입이 되기 위해서는 행위적 호환성을 만족시켜야 함
    - 서브타입은 슈퍼타입의 행위와 호환되기 때문에 서브타입은 슈퍼타입을 대체할 수 있어야 함

일반화는 추상화를 위한 도구다

추상화의 두 번째 차원은 중요한 부분을 강조하기 위해 불필요한 세부 사항을 제거시켜 단순하게 만드는 것

- 트럼프 인간의 특수한 능력은 제거하고 종이 조각처럼 쉽게 뒤집어지는 트럼프의 특성에 집중

두 가지 추상화 기법이 함께 사용됐다는 점을 주목

- 정원에 있던 등장인물들의 차이점은 배제하고 공통점만을 강조함으로써 이들을 공통의 타입인 트럼프 인간으로 분류함
- 트럼프 인간을 좀 더 단순한 관점에서 바라보기 위해 불필요한 특성을 배제하고 좀 더 포괄적인 의미를 지닌 트럼프로 일반화함

정적 모델

타입의 목적

- 인간의 인지 능력으로는 시간에 따라 동적으로 변하는 객체의 복잡성을 극복하기 너무 어렵기 때문에

- 타입은 시간에 따라 동적으로 변하는 상태를 시간과 무관한 정적인 모습으로 다룰 수 있게 해줌
    - 앨리스의 키가 더 크더라도 앨리스는 앨리스

```
앨리스
키                  <--------        키 = n(계속 바뀜)
버섯을  먹다
음료를  마시다
부채질  하다
```

그래서 결국 타입은 추상화다

- 타입은 어떤 시점에 앨리스에 관해 생각할 때 불필요한 시간이라는 요소와 상태 변화라는 요소를 제거하고 철저하게 정적인 관점에서 앨리스의 모습을 묘사하는 것을 가능하게 해줌
- 타입을 이용하면 객체의 동적인 특성을 추상화 가능
- 타입은 시간에 따른 객체의 상태 변경이라는 복잡성을 단순화할 수 있는 효과적인 방법

동적 모델과 정적 모델

객체를 생각할 때는 두 가지 모델을 동시에 고려한다는 사실을 알 수 있음

- 동적 모델**:** 스냅샷처럼 실제로 객체가 살아 움직이는 동안 상태가 어떻게 변하고 어떻게 행동하는지 포착하는 것
- 정적 모델: 동적으로 변하는 객체의 상태가 아니라 객체가 속한 타입의 정적인 모습을 표현함

객체지향 어플리케이션을 설계하고 구현하기 위해서는 객체 관점의 동적 모델과 객체를 추상화한 타입 관점의 정적 모델을 적절히 혼용해야 함

- 클래스를 작성하는 시점에는 시스템을 정적인 관점에서 접근
- 실제로 애플리케이션을 실행해 객체의 상태 변경을 추적하고 디버깅하는 동안에는 객체의 동적인 모델을 탐험하고 있는 것

클래스

- 객체지향 프로그래밍 언어에서 정적인 모델을 클래스를 이용해 구현함
- 타입을 구현하는 가장 보현적인 방법 = 클래스
  - 클라스와 타입은 동일한 것이 아님
  - 타입: 객체를 분류하기 위해 사용하는 개념
  - 클래스: 타입을 구현할 수 있는 여러 구현 메커니즘 중 하나
- 객체를 분류하는 기준을 타입이며, 타입을 나누는 기준은 객체가 수행하는 행동임
- 프로그래밍 언어를 이용해 타입을 구현할 수 있는 한 가지 방법이 클래스
- 객체지향에서 중요한 것은 동적으로 변하는 객체의 **상태**와 상태를 변경하는 **행위**임