02 이상한 나라의 객체

Tags	Subin kim
날짜	@May 25, 2022

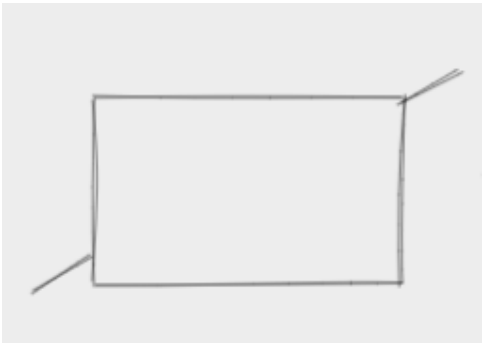
목차

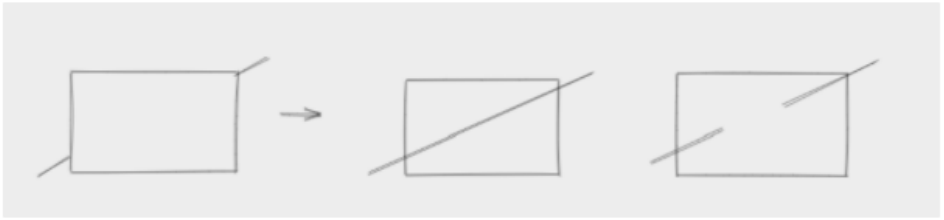
---

1. 객체지향과 인지능력

Q. 한개일까 두개일까?

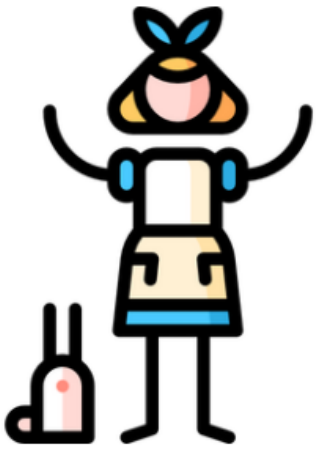
3개월 애기는 이걸 어떻게 판단할까?





결론 : 3개월 애기도 `함께 행동하는 물체`를 `하나의 개념`으로 인지

2. 객체 그리고 이상한 나라



```
[System] 이상한 나라에 오신걸 환영합니다.

[System] 앨리스가 되어 이상한 나라를 여행해 보세요!

앨리스 상태 >> Alice{name='alice', height=130, place=STREAM, itemList=[]}

[System] 앨리스가 할 '행동'을 선택해 주세요.
1. 아이템을 사용한다
2. 음식을 먹는다
3. 이동한다
4. 대화한다
5. 종료한다
>> 1
```

- 앨리스의 행동은 → 앨리스의 상태를 변경시킨다

```
[System] 상호작용할 아이템을 선택해 주세요
1. Fan
2. Table
>> 1
[System] 부채를 부쳤더니 10만큼 앨리스가 작아집니다.

앨리스 상태 >> Alice{name='alice', height=120, place=STREAM, itemList=[]}
```

```
[System] 상호작용할 아이템을 선택해 주세요
1. Fan
2. Table
>> 2
[System] 테이블 위에서 정원으로 가는 키를 획득했습니다.

앨리스 상태 >> Alice{name='alice', height=120, place=STREAM, itemList=[Key{place=GARDEN}]}
```

3. 객체 그리고 소프트웨어

- **상태**

▼ 상태란?

> 상태는 특정 시점에 객체가 가지고 있는 정보의 집합으로 객체의 구조적 특징을 표현한다. 객체의 상태는 객체에 존재하는 정적인 프로퍼티와 동적인 프로퍼티 값으로 구성된다. 객체의 프로퍼티는 단순한 값과 다른 객체를 참조하는 링크로 구분할 수 있다.

▼ 왜 상태가 필요한가?

▼ `행동`과 `과정`과 `결과`를 **단순하게 기술하기 위해**

- 상태를 이용하면 행동의 결과를 쉽게 예측하고 설명할 수 있다
  - 과거에 얽매이지 않고 현재를 기반으로 객체의 행동 방식을 이해

▼ ex) 객체가 주변 환경과의 상호작용에 어떻게 반응하는가?

- → 그 시점까지 객체에 어떤일이 발생했느냐에 좌우됨
  - 행동의 결과를 설명하는 것을 매우 어렵게 만들음
  - (과거에 했던 모든 행동을 기억해야만 가능하기 때문)

▼ ex) 앨리스

- 앨리스가 과거에 어떤 행동을 했었는지 모르더라도
- 앨리스의 `키`와 문의 `높이`라는 두가지 **상태만 알면**
- 문을 통과하는 **행동의 결과를 쉽게 예측할 수 있음**

- **행동**

  ▼ 행동이란?

  행동이란 외부의 요청 또는 수신된 메시지에 응답하기 위해 동작하고 반응하는 활동이다. 행동의 결과로 객체는 자신의 상태를 변경하거나 다른 객체에게 메시지를 전달할 수 있다. 객체는 행동을 통해 다른 객체와의 협력에 참여하므로 행동은 외부에 가시적이어야 한다.

  ▼ 상태와 행동

  - 객체가 취하는 **행동은** 객체 자신의 **상태를 변경**시킨다.
  - 객체 행동에 의해 객체 상태가 변경된다는 것
    - == 행동이 **부수효과(side effect)**를 초래한다는 것을 의미

  ▼ 상태와 행동사이에는 다음과 같은 관계가 있음

  - **객체의 행동**은 상태에 영향을 받는다
    - ex) 앨리스의 `키가 40센티 이하(상태)` 면 `문을 통과(행동)`할 수 있다.
  - 객체의 행동은 상태를 변경시킨다.
    - ex) `문을 통과한 후(행동)`에 앨리스의 `위치는 정원(상태)`으로 바뀌어야 한다.

  ▼ **협력**과 행동

  - 객체의 행동은 객체가 협력에 참여할 수 있는 유일한 방법

  > 어떤 객체도 섬이 아니다.

  - 객체는 다른 객체와 상호작용하며 `협력하는 객체들의 공동체` 에 참여하기 위해 노력

- **식별자**

  ▼ 식별자란?

  식별자란 어떤 객체를 다른 객체와 구분하는 데 사용하는 객체의 프로퍼티다. 값은 식별자를 가지지 않기 때문에 상태를 이용한 동등성 검사를 통해 두 인스턴스를 비교해야 한다. 객체는 상태가 변경될 수 있기 때문에 식별자를 이용한 동일성 검사를 통해 두 인스턴스를 비교할 수 있다.

▼ 값

- 값객체(Value Object) 라고 불림
- 불변(immutable) -변하지 않는 양을 모델링
  - ex) 숫자, 문자열, 날짜, 시간, 금액
- 두 값이 같은지 판단 ⇒ 동등성(Equality)
  - 상태가 같은지를 판단 O → 왜냐면 값의 상태가 변하지 않기 때문에

▼ 객체

- 참조객체(reference object) || 엔티티(entity) 라고도 불림
- 가변(mutable) - 시간, 행동에 따라 변경되는 상태를 포함
- 두 객체가 같은지를 판단 ⇒ 동일성(identical)
  - 상태가 같은지로 판단 X → 왜냐면 시간의 흐름에 따라 객체의 상태가 변하기 때문

4. 기계로서의 객체

▼ **AliceMachine**

그림 2.6 기계로서의 앨리스 객체

```
public class AliceMachine {
    private final Alice alice;
    @Link
    private final DrinkMachine drinkMachine;

    public AliceMachine(Alice alice, DrinkMachine drinkMachine) {
        this.alice = alice;
        this.drinkMachine = drinkMachine;
    }

    @Query
    public int 키(){ return alice.getHeight();}
    @Query
    public Place 위치(){ return alice.getPlace();}

    @Command
    public void 음료를_마신다(){drinkMachine.마셔지다();}
    @Command
    public void 케이크를_먹다(Cake cake){...}
    @Command
    public void 버섯을_먹다(Mushroom mushroom){...}
    @Command
    public boolean 문을_통과한다(Door door){...}
}
```

▼ **DrinkMachine**

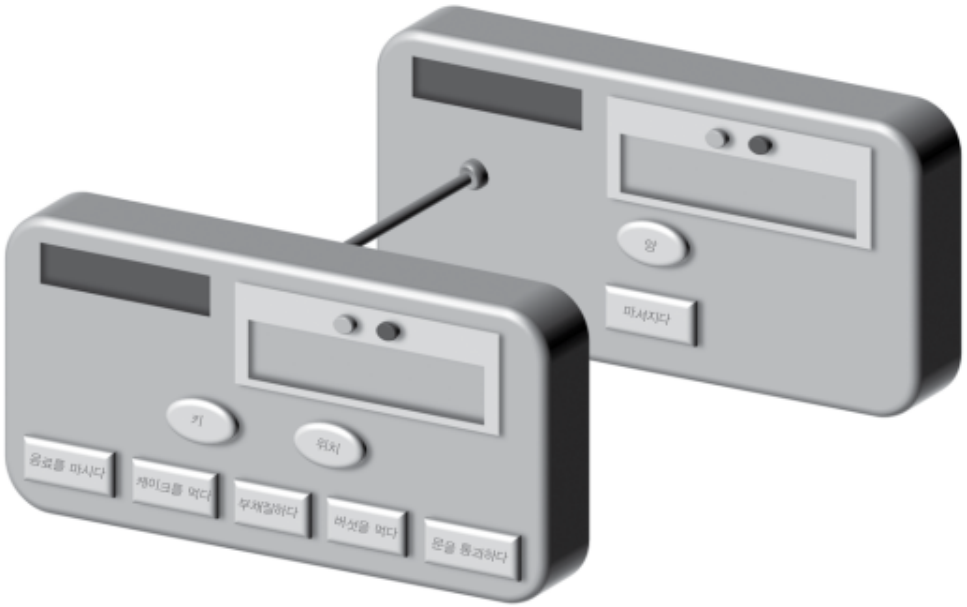


그림 2.8 협력하는 앨리스 기계와 음료 기계

```
public class DrinkMachine {
    private final Drink drink;
    public DrinkMachine(Drink drink) {
        this.drink = drink;
    }

    @Query
    public int 양(){return drink.getQuantity();}
    @Command
    public void 마셔지다(int quantity){
        drink.drunken(quantity);
    }
}
```

5. 행동이 상태를 결정한다 🌟

상태를 먼저 결정하고 행동을 나중에 결정하는 방법 → 설계에 나쁜 영향

▼ 1. 캡슐화가 저해된다

상태가 공용 인터페이스에 노출되어버릴 확률이 높아진다.

▼ 2. 객체를 협력자가 아닌 고립된 섬으로 만든다

- 객체가 필요한 이유는
  - 애플리케이션 문맥 내에서 다른 객체와 협력하기 위해서인데
  - 상태 먼저 고려하면 **협력 문맥에서 벗어난채 설계**하게 한다

▼ 3. 객체의 재사용성이 저하된다

- 재사용성은 다양한 협력에 참여할 수 있는 능력에서 나온다.
- 상태에 초점을 맞추면 다양한 협력에 참여하기 어렵다 → 재사용성 ↓

- 협력 안에서 **객체의 행동은** 결국 객체가 협력에 참여하면서 완수해야 하는 **책임**을 의미한다.
  - 객체의 적합성을 결정하는 것은 상태가 아니라 객체의 행동이다.
  - 따라서 어떤 책임이 필요한가를 결정하는 과정이 전체 설계를 주도해야 한다.

- **RDD(Responsibility-Driven Design)**
  - 협력이라는 문맥 안에서 객체의 행동을 생각하도록 도움
  - 응집도 높고 재사용 가능한 객체를 만들 수 있게 한다.

6. 은유와 객체

▼ 모방과 추상화

- 현실세계와 객체 사이의 관계를 깔끔하게 설명하지 못한다.
- → 현실 세계와 객체지향 세계 사이의 관계를 좀 더 정확하게 설명할 수 있는 단어는 `은유`다.

▼ 의인화

> 현실의 객체보다 더 많은 일을 할 수 있는 소프트웨어 객체의 특징이다
> - 레베카 위프스브록

객체지향 세계의 거리는 현실 속의 객체보다 더 많은 특징과 능력을 보유한 객체들로 넘쳐난다.

▼ 은유(metaphor)

- 은유 관계에 있는 `실제 객체의 이름` → `소프트웨어 객체의 이름`으로 사용
- 현실의 개념을 이용해 소프트웨어 객체를 잘 묘사하면 (은유를 효과적으로 사용할 경우)
  - ▼ `표현적 차이를 줄일 수 있다.`
    - 소프트웨어의 구조를 **쉽게 예측**할 수 있다.
    - 이해하기 쉽고 **유지보수가 용이**한 소프트웨어를 만들 수 있다.
    - 여러분이 창조한 객체를 **더욱 잘 이해**하고 **기억**할 수 있게 된다.
    - 이러한 이유로 객체지향 지침서에는 현실 세계인 도메인에서 사용되는 이름을 객체에게 부여하라고 가이드 함

▼ 일상 생활에서의 은유

- 그 여자는 양같아요 → 순한 양을 이용해 여자의 성격을 묘사

현실을 닮아야 한다는 어떤 제약이나 구속도 없다.

우리가 창조한 객체의 특성을 상기시킬 수 있다면 현실 속의 객체 이름을 이용해 객체를 묘사하라

그렇지 않다면 현실을 무시하고 **자유롭게 여러분만의 새로운 세계를 창조**하기를 바란다.

앨리스를 매혹시킨 이상한 나라가 그랬던 것 처럼..