6장. 객체 지도

> ***“ 유일하게 변하지 않는 것은 모든 것이 변한다는 사실뿐이다. “***
>
> 기능이 아니라 구조를 바탕으로 시스템을 분할하는 객체지향의 다른 측면을 알아보자.

지도
- 🗺️ 길을 찾는 방법
- 🤔 지도는 왜 범용적인가?
- 🎲 객체지향

기능 설계 vs 구조 설계
- 👍🏻 훌륭한 소프트웨어
- ♻️ 요구사항은 변경된다.
- 🏭 기능 설계
- 👷🏻 구조 설계

도메인 모델
- 🖼️ 도메인의 모습을 담을 수 있는 객체지향
- ⚖️ 표현적 차이
- 🧩 불안정한 기능을 담는 안정적인 도메인 모델

유스케이스
- 🏴‍☠️ 가치
- 🦸🏻 특성

기능과 구조의 통합
- 💡 책임-주도 설계
- 💡 기능 변경을 흡수하는 안정적인 구조
  - 도메인 모델이 안정적인 이유 ?
  - 연결완전성

지도

🗺️ 길을 찾는 방법

- 길을 직접 알려주기

    - 기능적
    - 해결 방법 지향적 접근법
    - 현재의 요구만을 만족 가능
- 지도를 이용하기
    - 구조적
    - 문제 지향적 접근법
    - 다양한 목적으로 재사용 가능

🤔 지도는 왜 범용적인가?

- 지도 사용자가 원하는 `기능`에 비해 지도에 표시된 `구조`가 더 **안정적**
- 지형은 거의 변하지 않기 때문에 과거의 지도가 현재에도 유용함

→ 객체지향 개발 방법과 유사

🎲 객체지향

- 안정적인 구조에 변경이 빈번하게 발생하는 기능을 종속시킴
- 범용적
- 높은 재사용성
- 변경에 안정적

안정적인 구조를 따라 역할, 책임, 협력을 구성하라.

기능 설계 vs 구조 설계

👍🏻 훌륭한 소프트웨어

훌륭한 기능을 제공하는 동시에 사용자가 원하는 새로운 기능을 빠르고 안정적으로 추가할 수 있다.

- 충분 조건 : 훌륭한 기능
- 필요 조건 : 훌륭한 구조

♻️ 요구사항은 변경된다.

- 예측 불가한 요구사항 변경에 유연하게 대처할 수 있는 안정적인 구조 제공 필요
    - 설계가 어려운 이유
- 변경을 예측하지 말고 변경을 수용할 수 있는 선택의 여지를 설계에 마련할 것
- 설계의 일차적 목표는 변경에 소요되는 비용을 낮추는 것

→ 답은 안정적인 구조에 있다.

🏭 기능 설계

제품이 사용자를 위해 무엇을 할 수 있는가?

기능 : 사용자가 자신의 목표를 달성하기 위해 사용할 수 있는 시스템의 서비스

- 사용자의 목표를 만족시키기 위해 책임을 수행하는 시스템의 행위로 표현
- 유스케이스 모델링

👨🏻‍🎨 구조 설계

제품의 형태가 어떠해야 하는가?

구조 : 시스템의 기능을 구현하기 위한 기반

- 사용자나 이해관계자들이 도메인에 관해 생각하는 개념과 개념들간의 관계로 표현
- 도메인 모델링

도메인 모델

> 도메인 : 사용자가 프로그램을 사용하는 대상 분야
> 모델 : 대상을 단순화해서 표현한 것

- 사용자가 프로그램을 사용하는 대상 영역에 관한 지식을 선택적으로 단순화하고 의식적으로 구조화한 형태
- 소프트웨어가 목적하는 영역 내 개념과 개념 간 관계, 규칙, 제약 등을 주의 깊게 추상화한 것
- 소프트웨어 개발과 관련된 이해관계자들이 도메인에 대해 생각하는 관점

🚧 **도메인 모델은 단순한 다이어그램이 아닌, `멘탈 모델` 이다**

`멘탈 모델` : 사람들이 자기 자신, 다른 사람, 환경, 사물들에 대해 갖는 모형
- 사용자 모델, 디자인 모델, 시스템 이미지로 구분됨.
소프트웨어 사용자들이 도메인에 존재하는 현상을 이해하고 현상에 반응하기 위해 도메인과 관련된 멘탈 모델을 형성함.

→ 제품에 관한 모든 것이 사용자들이 가지고 있는 멘탈 모델과 정확하게 일치해야 한다.

🖼️ 도메인의 모습을 담을 수 있는 객체지향

- 최종 코드는 사용자가 도메인을 바라보는 관점을 반영해야 한다.
  - 애플리케이션이 도메인 모델을 기반으로 설계돼야 한다.
- 객체지향은 ...
  - 사용자들이 이해하는 도메인의 구조와 최대한 유사하게 코드를 구조화하게 해준다.
  - 실체를 시스템 안의 객체로 재창조할 수 있게 해준다.
  - 동적인 객체의 복잡성을 정적인 타입으로 단순화해준다.
  - **사용자/설계자의 관점, 코드의 모습을 모두 유사한 형태로 유지할 수 있게 하는 사고 도구와 프로그래밍 기법을 제공한다.**

⚖️ 표현적 차이

소프트웨어 객체 - 현실 객체간 의미적 거리

- 소프트웨어 객체 ≠ 현실 객체에 대한 추상화
  - 소프트웨어 객체 - 현실 객체 간 관계를 가장 효과적으로 표현할 수 있는 단어는 **은유** !
  - 은유를 통해 표현적 차이를 줄이자.
- 은유를 통해 투영해야 하는 대상
  - 사용자가 도메인에 대해 생각하는 개념들 (=도메인 모델)

    - 도메인 모델을 기반으로 설계/구현하는 것은 사용자가 도메인을 바라보는 관점을 코드에 반영할 수 있게 함
- 소프트웨어를 이해하고 수정하기 쉽게 만들어 줌
    - 코드의 구조 → 도메인의 구조를 반영
    - 도메인을 이해하면 코드를 이해하기 수월해짐

🧩 불안정한 기능을 담는 안정적인 도메인 모델

도메인 모델이 제공하는 구조는 상대적으로 안정적이다.

- 도메인 모델의 핵심
    - 사용자가 도메인을 바라보는 관점을 반영해 소프트웨어를 설계하고 구현하는 것
    - 사용자들은 누구보다도 도메인의 본질적인 측면을 가장 잘 이해하고 있다.
    - 본질적인 것은 변경이 적다. → 개발에 이득
- 변경에 유연하게 대응할 수 있는 탄력적인 소프트웨어를 만들자.

유스케이스

> 사용자의 목표를 달성하기 위해 사용자 - 시스템 간 이뤄지는 상호작용의 흐름을 텍스트로 정리한 것

🏴‍☠️ 가치

- 사용자들의 목표를 중심으로 시스템의 기능적 요구사항을 이야기 형식으로 묶을 수 있음
- 산발적으로 흩어져 있는 기능에 `사용자 목표`라는 문맥 제공
- 각 기능이 유기적 관계를 지닌 체계를 이룸
- 요구사항을 관리하는 데 필요한 정신적 과부하를 줄임

🦹 특성

1. 사용자와 시스템 간 상호작용을 보여주는 **텍스트**다
2. 하나의 시나리오가 아니라 여러 시나리오의 집합이다.

3. 단순한 피처 목록과는 다르다.
4. 사용자 인터페이스와 관련된 세부 정보는 포함하지 말아야 한다.
5. 내부 설계와 관련된 정보는 포함하지 않는다.
6. 설계 기법도, 객체지향 기법도 아니다.
    - 단순히 사용자가 바라보는 시스템의 외부 관점만을 표현한다.
    - 시스템의 내부 구조나 실행 메커니즘에 관한 어떤 정보도 제공하지 않는다.
    - 사용자가 무엇을 얻을 수 있고 어떻게 상호작용할 수 있느냐만 기술된다.

기능과 구조의 통합

- 도메인 모델 : 안정적인 구조를 개념화
- 유스케이스 : 불안정한 기능을 서술

→ 유스케이스에 정리된 시스템의 기능을 도메인 모델을 기반으로 한 객체들의 책임으로 분배하자.

💡 책임-주도 설계

- 시스템의 기능을 역할과 책임을 수행하는 객체들의 협력 관계로 바라보게 함
- 유스케이스와 도메인 모델을 통합
- 사용자의 관점에서 시스템 기능을 명시 (유스케이스)
- 사용자와 설계자가 공유하는 안정적인 구조(도메인 모델)를 기반으로 기능을 책임으로 변환

💡 기능 변경을 흡수하는 안정적인 구조

도메인 모델이 안정적인 이유 ?

- 도메인 모델을 구성하는 개념은 비즈니스가 없어지지 않는 한 안정적으로 유지된다.
- 도메인 모델을 구성하는 개념 간관계는 비즈니스 규칙을 기반으로 하기 때문에 정책이 크게 변경되지 않는 한 안정적으로 유지된다.

연결완전성

- 객체지향의 가장 큰 장점은 도메인 모델링 기법과 도메인 프로그래밍 기법이 동일하다는 점
- 도메인 모델링에서 사용한 객체와 개념을 프로그래밍으로 변환할 수 있다.
- **객체지향에서는 연결완전성의 역방향도 설립한다 !**

> 🔑 **쉬운 유지보수와 유연한 객체지향 시스템으로 가는 길 ...**
>
> 안정적인 도메인 모델을 기반으로 시스템의 기능을 구현하라.
> 도메인 모델과 코드를 밀접하게 연관시켜라.