6. 객체 지도

💭

> **“유일하게 변하지 않는 것은 모든 것이 변한다는 사실뿐이다.”**

기능 설계 대 구조 설계

- **기능 측면의 설계**는 제품이 사용자를 위해 무엇을 할 수 있는지에 초점을 맞춘다.
- **구조 측면의 설계**는 제품의 형태가 어떠해야 하는지에 초점을 맞춘다.
- 훌륭한 **기능**이 훌륭한 소프트웨어를 만드는 **충분조건**이라고 한다면, 훌륭한 **구조**는 훌륭한 소프트웨어를 만드는 **필요조건**이다.
- **설계를 하는 목적**은 나중에 설계하는 것을 허용하는 것이며, 설계의 일차적인 목표는 **변경에 소요되는 비용을 낮추는 것**이다.

> “사람들에게 길을 묻지 마라. 객체를 이용해 지도를 만들어라. 기능은 지도에 표시된 길을 따라 자연스럽게 흘러갈 것이다.”

두 가지 재료: 기능과 구조

- 객체지향 세계를 구축하려면 '기능'과 '구조'라는 재료가 필요하다.
  - **기능:** 사용자가 자신의 목표를 달성하기 위해 사용할 수 있는 시스템의 서비스
  - **구조**: 시스템의 기능을 구현하기 위한 기반으로, 기능 변경을 수용할 수 있도록 안정적이어야 함
- 기능과 구조를 표현하는 두 가지 기법
  - 구조는 사용자나 이해관계자들이 도메인에 관해 생각하는 개념과 개념들 간의 관계로 표현한다.
  - 기능은 사용자의 목표를 만족시키기 위해 책임을 수행하는 시스템의 행위로 표현한다.
- **유스케이스 모델링**: 기능을 수집하고 표현하기 위한 기법
- **도메인 모델링**: 구조를 수집하고 표현하기 위한 기법

안정적인 재료: 구조

도메인 모델

- 도메인 모델은 도메인에 대한 사용자 모델, 디자인 모델, 시스템 이미지를 포괄하도록 추상화한 소프트웨어 모델이다. 따라서 **도메인 모델은 소프트웨어에 대한 멘탈 모델**이다.
  - **사용자 모델**: 사용자가 제품에 대해 가지고 있는 개념들의 모습
  - **디자인 모델**: 설계자가 마음 속에 갖고 있는 시스템에 대한 개념화
  - **시스템 이미지**: 최종 제품

> ✏️ **멘탈 모델(Mental Model)이란?**
> 사람들이 자기 자신, 다른 사람, 환경, 자신이 상호작용하는 사물들에 대해 갖는 모형이다.
> 사람들은 세상에 존재하는 현상을 이해하고 반응하기 위해 자신의 마음 속에 멘탈 모델을 구축한다.

도메인의 모습을 담을 수 있는 객체지향

- 객체지향 패러다임은 사용자의 관점, 설계자의 관점, 코드의 모습을 모두 유사한 형태로 유지할 수 있게 하는 유용한 사고 도구와 프로그래밍 기법을 제공한다.

- 객체지향을 이용하면 도메인에 대한 사용자 모델, 디자인 모델, 시스템 이미지 모두가 유사한 모습을 유지하도록 만드는 것이 가능하며, 이러한 특징을 **연결 완전성**, **표현적 차이**라고 한다.

표현적 차이

- 소프트웨어 객체는 현실 객체의 특성을 토대로 구축되며, 소프트웨어 객체와 현실 객체 사이의 의미적 거리를 가리켜 **표현적 차이** 또는 **의미적 차이**라고 한다.
- 소프트웨어 객체를 창조하기 위해 은유해야 하는 대상은 바로 도메인 모델이다.
- 도메인 모델을 기반으로 설계하고 구현하는 것은 사용자가 도메인을 바라보는 관점을 그대로 코드에 반영할 수 있게 한다. → **표현적 차이가 줄어들고, 사용자의 멘탈 모델이 그대로 코드에 녹아 스며들게 될 것이다.**
- 도메인 모델은 코드 안에 존재하는 미로를 헤쳐나갈 수 있는 지도를 제공한다.

불안정한 기능을 담는 안정적인 도메인 모델

- 도메인에 대한 사용자의 관점을 반영해야 하는 이유?
  - 사용자들이 누구보다도 도메인의 '본질적인' 측면을 가장 잘 이해하고 있기 때문
- 사용자 모델에 포함된 개념과 규칙은 비교적 변경될 확률이 적기 때문에 사용자 모델을 기반으로 설계와 코드를 만들면 변경에 쉽게 대처할 수 있는 가능성이 커진다.

  → **도메인 모델이 기능을 담을 수 있는 안정적인 구조를 제공할 수 있음**을 의미

불안정한 재료: 기능

유스케이스

- **유스케이스**란
  - 사용자의 목표를 달성하기 위해 사용자와 시스템 간에 이뤄지는 상호작용의 흐름을 텍스트로 정리한 것
- 유스케이스는 이해관계자들 중에서 일차 액터라 불리는 행위자의 요청에 대한 시스템의 응답으로서, 다양한 조건하에 있는 시스템의 행위를 서술한다.

✏️ **일차 액터(primary actor)**란?
시스템의 서비스 중 하나를 요청하는 이해관계자로, 하나의 목표를 가지고 유스케이스를 시작하는 액터.
일반적으로 시스템과 연동하는 외부 시스템 역시 일차 액터의 범주에 포함시킨다.

> **“사용자의 목표가 유스케이스의 핵심이다.**
> **유스케이스는 공통의 사용자 목표를 통해 강하게 연관된 시나리오의 집합이다.”**

유스케이스의 특성

1. **유스케이스는 다이어그램이 아니라 사용자와 시스템 간의 상호작용을 보여주는 ‘텍스트’다.**

   다이어그램에 노력을 쏟지 말자.

   유스케이스의 핵심은 사용자와 시스템 간의 상호작용을 일련의 이야기 흐름으로 표현하는 것이다.

2. **유스케이스는 하나의 시나리오가 아니라 여러 시나리오들의 집합이다.**

   시나리오는 유스케이스를 통해 시스템을 사용하는 하나의 특정한 이야기 또는 경로이며, **유스케이스 인스턴스**라고도 한다.

3. **유스케이스는 단순한 피처(feature) 목록과 다르다.**

   피처(feature): 시스템이 수행해야 하는 기능의 목록을 단순하게 나열한 것

   예를 들어 ‘시스템은 정기예금 정보를 보여준다’와 ‘시스템은 당일이나 현재 일자의 이자를 계산한다’라는 피처가 있을 때, 두 피처를 서로 연관이 없는 독립적인 기능으로 보이게끔 만든다는 단점이 있다.

   두 피처를 ‘중도 해지 이자액을 계산한다’라는 유스케이스로 묶음으로써 시스템의 기능에 대해 의사소통할 수 있는 문맥을 얻을 수 있다.

4. **유스케이스는 사용자 인터페이스와 관련된 세부 정보를 포함하지 말아야 한다.**

   자주 변경되는 사용자 인터페이스 요소는 배제하고, 사용자 관점에서 시스템의 행위에 초점을 맞춘다.

   이처럼 사용자 인터페이스를 배제한 유스케이스 형식을 **본질적인 유스케이스**라고 한다.

5. **유스케이스는 내부 설계와 관련된 정보를 포함하지 않는다.**

유스케이스의 목적은 연관된 시스템의 기능을 이야기 형식으로 모으는 것이지 내부 설계를 설명하는 것이 아니다.

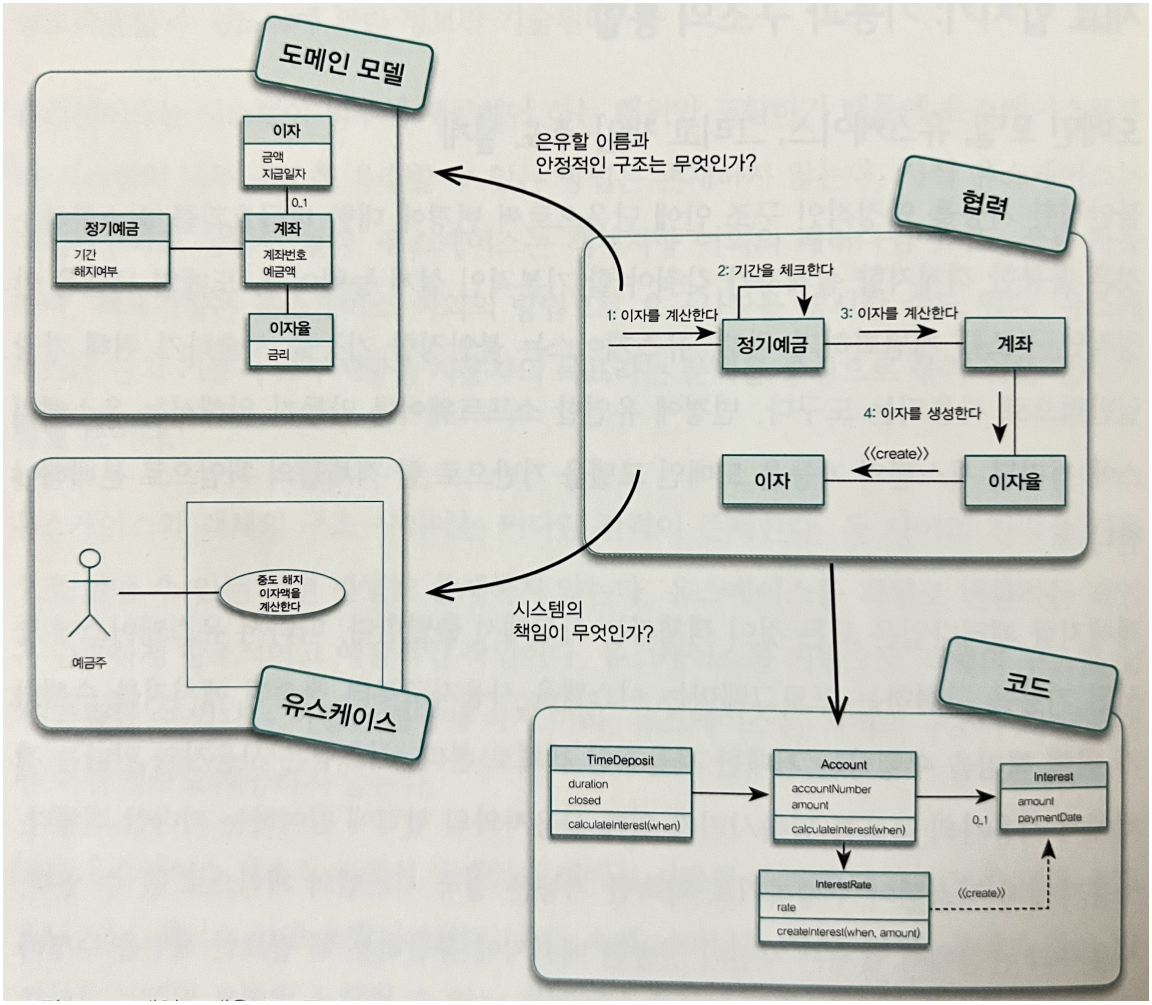
유스케이스는 설계 기법도, 객체지향 기법도 아니다

- 유스케이스 안에 도메인 모델을 구축할 수 있는 모든 정보가 포함돼 있지 않다. 그저 영감을 불러일으킬 수 있는 약간의 힌트만 들어 있을 뿐이다.

재료 합치기: 기능과 구조의 통합

도메인 모델, 유스케이스, 그리고 책임-주도 설계

- 시스템은 사용자로부터 전송된 메시지를 수행하기 위해 책임을 수행하는 거대한 자율적인 객체다.
- 도메인 모델은 구조를, 유스케이스는 협력의 출발점인 시스템 책임을 제공한다.



- 객체 설계는 '**요구사항들을 식별하고 도메인 모델을 생성한 후, 소프트웨어 클래스에 메서드들을 추가하고, 요구사항을 충족시키기 위해 객체들 간의 메시지 전송을 정의**'하는 것이다.
- 견고한 객체지향 애플리케이션을 개발하기 위해서는 **사용자의 관점에서 시스템의 기능을 명시**하고, 사용자와 설계자가 공유하는 **안정적인 구조를 기반으로 기능을 책임으로 변환**하는 체계적인 절차를 따라야 한다.

기능 변경을 흡수하는 안정적인 구조

- **도메인 모델이 안정적인 이유**
  - 도메인 모델을 구성하는 개념은 비즈니스가 없어지거나 완전히 개편되지 않는 한 안정적으로 유지된다.
  - 도메인 모델을 구성하는 개념 간의 관계는 비즈니스 규칙을 기반으로 하기 때문에 비즈니스 정책이 크게 변경되지 않는 한 안정적으로 유지된다.
- **연결완전성**
  - 도메인을 모델링하는 기법과 도메인을 프로그래밍하기 위해 사용하는 기법이 동일하기 때문에 도메인 모델링에서 사용한 객체와 개념을 프로그래밍 설계에서의 객체와 클래스로 매끄럽게 변환할 수 있는 특성
- **가역성(reversibility)**
  - 연결완전성과 반대로 코드의 변경으로부터 도메인 모델의 변경 사항을 유추할 수 있는 매끄러운 흐름
- **도메인 모델의 진정한 목표**
  - 사람들이 동일한 용어와 동일한 개념을 이용해 의사소통하고 코드로부터 도메인 모델을 유추할 수 있게 하는 것

- 객체를 이용해 지도를 만들어라. 기능은 지도에 표시된 길을 따라 자연스럽게 흘러갈 것이다.
  - 안정적인 구조가 있다면 기능은 자연스레 찾을 수 있게 된다고???
- 여지껏 무턱대고 도메인 모델을 만들고 참고하며 프로그래밍했고, 그래서 무수하게 변동되는 도메인 모델에 회의감을 느꼈다. 하지만 이번 챕터와 지금까지 읽은 내용을 통해서 도메인 모델링의 이유와 방식을 조오오오금은 알게 되었고 바로 적용해볼 생각이다.