Clean Architecture

R.C.Martin

Clean Architecture?

Architecture란?

스프링, 장고, ROR, Express의 구조는 Best인가?

남들이 다 쓰니깐?

Clean한 Architecture는 항상 옳은가?

왜 항상 clean해야 하지?

Clean의 의미가 뭔데?

Behavior & Structure

구조적 프로그래밍

순차 / 분기 / 반복

객체 지향적 프로그래밍

캡슐화? / 상속? / 다형성(easy)?

객체 지향적 프로그래밍

뭐를 위한 방법일까?

현실 세계의 반영?

객체 지향적 프로그래밍

배포 독립성 / 개발 독립성

함수형 프로그래밍

불변성 vs 가변성

Race condition / deadlock / concurrent update

Time lag 같은 문제를 어떻게 해결하지?

실제로 받았던 코드 리뷰

나는 개인적으로 꼭 let을 쓸 필요가 없는 상황이라면 const를 선호해

let을 쓰는 순간 그 아래 모든 코드에서 '변경 가능성'을 염두에 두어야 해서 코드가 불필요하게 복잡해진다고 생각하거든

SRP

cohension

서로 다른 Actor를 구분하자

Solution

OCP

기존 코드의 수정 = 0

Business Rule OR Business Logic

Level of protect

LSP

Type Substitution

Do Not Permit Extra Mechanism

ISP

언어 타입

Back-end에서 정적 타입?

DIP

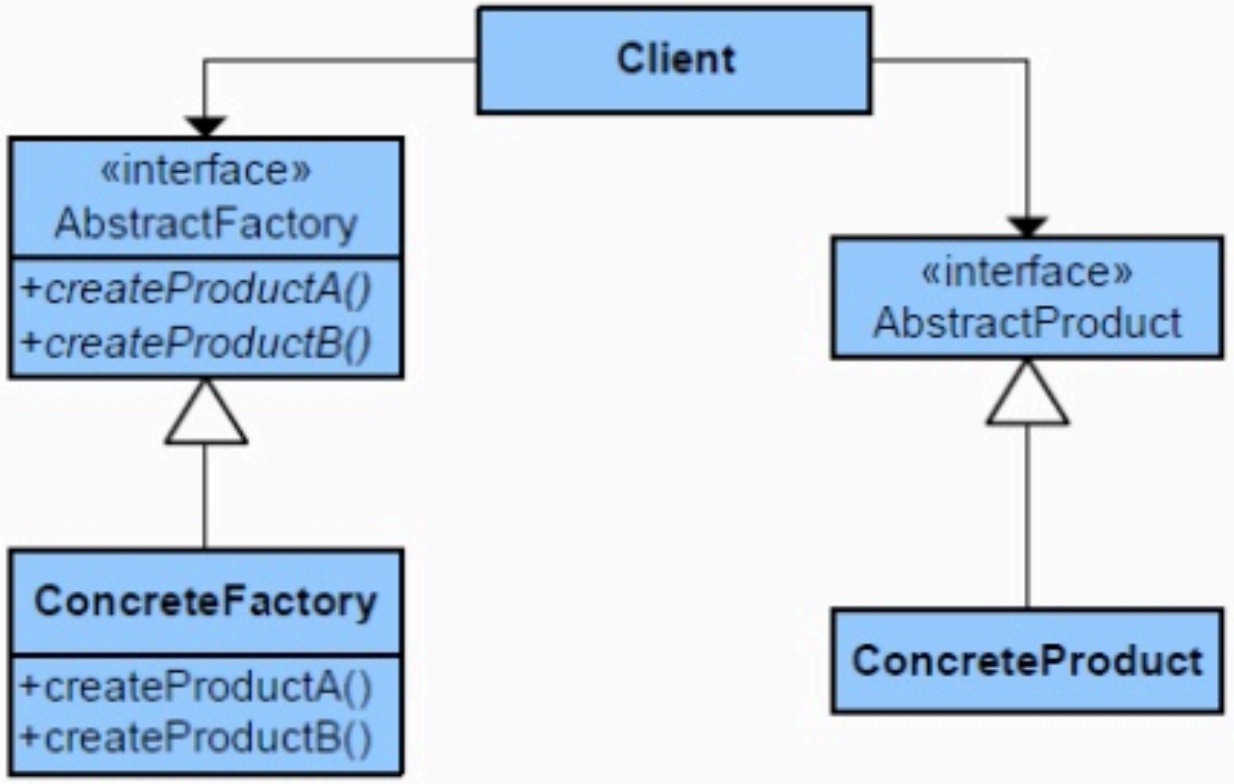
Abstraction에만 의존하자

제어 흐름의 반대로 의존 => 어쩌면 당연?

Abstract Factory

Type: Creational

What it is:
Provides an interface for creating families of related or dependent objects without specifying their concrete class.



볼 수 있다. 좌측 하단에는 Presenter프레젠터와 View뷰를 담당하는 네 가지 컴포넌트가 위치한다.

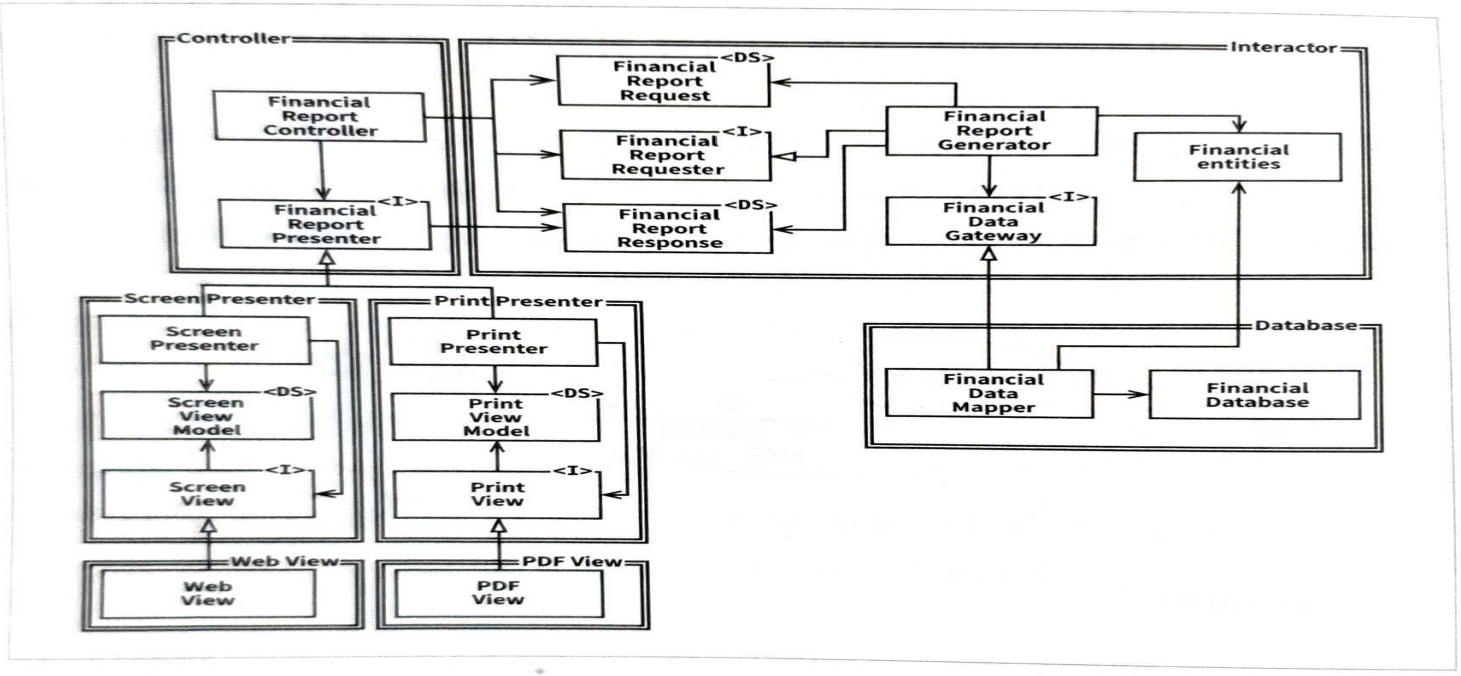


그림 8.2 처리 과정을 클래스 단위로 분할하고, 클래스는 컴포넌트 단위로 분리한다.

<I>로 표시된 클래스는 인터페이스이며, <DS>로 표시된 클래스는 데이터 구조다. 화살표가 열려 있다면 사용using 관계이며, 닫혀 있다면 구현implement 관계 또는 상속inheritance 관계다.

여기에서 주목할 점은 모든 의존성이 소스 코드 의존성을 나타낸다는 사실이다. 예를 들어 화살표가 A 클래스에서 B 클래스로 향한다면, A 클래스에서는 B 클래스를 호출하지만 B 클래스에서는 A 클래스를 전혀 호출하지

...서 그림 8.2에서 FinancialDataMapper는 그