

소프트웨어 디자인 패턴_퍼사드

Facade란?



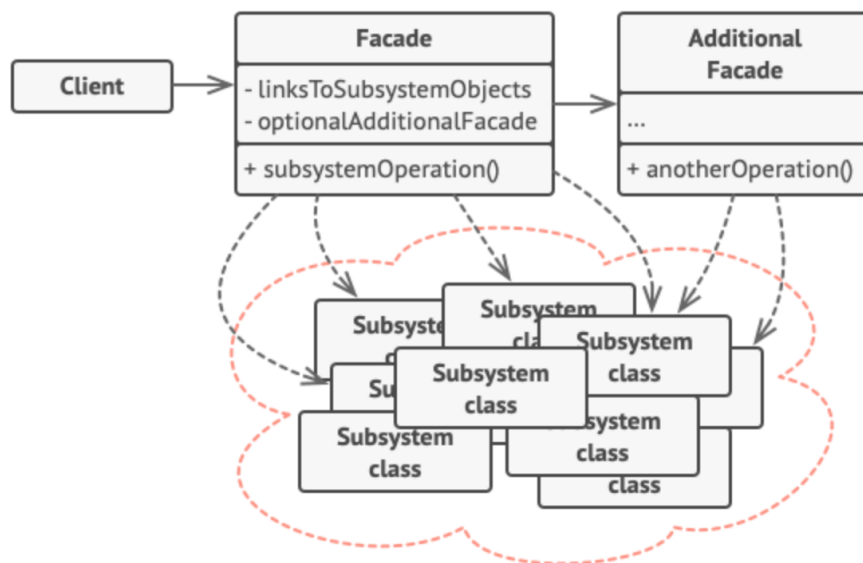


Facade (퍼사드 : 건축물의 정면)

건축물의 정면은 보통 건축물의 이미지와 건축 의도를 나타내기 때문에 오래 전부터 특별한 디자인을 적용하여 의미를 부여했다. 이처럼 건축물 정면만 봐도 이 건물이 어떤 목적을 하는지 단번에 알 수 있다는 특징을 차용하여 명명했다.

퍼사드 패턴이란?

퍼사드 패턴 구조



- 퍼사드 패턴은 복잡하게 짜여있는 라이브러리나 API 서비스 또는 패키지에서 사용하는 패턴으로 사용자가 복잡하게 짜여있는 코드를 하나하나 알 필요 없이 제공되는 사용법에 대해 설명되어있는 인터페이스만을 보고 사용할 수 있도록 한 패턴이다.(like 깃허브의 readme / 레고의 조립설명서)
- 퍼사드 패턴은 메인 시스템과 서브 시스템 중간에 위치하는데, 새로운 인터페이스 계층을 추가하며 시스템 간 의존성을 해결한다. 인터페이스 계층은 메인 시스템과 서브 시스템의 연결 관계를 대신 처리한다.
- 서브 시스템은 호출, 결합할 수 있는 인터페이스를 제공한다. 인터페이스는 한 개 일 수 있고 여러 개 일 수도 있고 또는 이를 함수 형태로 제공하기도 한다.

- 객체의 내부 구조를 사용자가 상세히 알 필요 없다. 퍼사드 패턴은 시스템의 연결성과 종속성을 최소화하는 것을 목적으로 한다.

퍼사드 패턴 특징

▼ 패턴 사용 시기

- 시스템이 너무 복잡할때 (⇒ 의존성을 낮춰 간결하게 만듦)
- 그래서 간단한 인터페이스를 통해 복잡한 시스템을 접근하도록 하고 싶을때
- 시스템을 사용하고 있는 외부와 결합도도 너무 높을때 의존성을 낮추기 위할때

▼ 패턴 장점

- 하위 시스템의 복잡성에서 코드를 분리하여, 외부에서 시스템을 사용하기 쉬워진다.
- 하위 시스템 간의 의존 관계가 많을 경우 이를 감소시키고 의존성을 한 곳으로 모을 수 있다.
- 복잡한 코드를 감춤으로써, 클라이언트가 시스템의 코드를 모르더라도 Facade 클래스만 이해하고 사용 가능하다.



Tip

외부에서 내부 로직을 직접 사용하기 때문에 내부 로직의 구조를 변경한다고 하거나 파라미터나 리턴값 등을 변경할 경우 직접적으로 영향을 받아 수정이 힘들거나 불가능한 경우가 종종 있다. 하지만 중간에 매개체 역할을 해주는 퍼사드 객체가 있기 때문에 실제 내부 로직이 어떻게 변경이 되더라도 상관이 없어지므로 의존성이 감소된다

▼ 패턴 단점

- 퍼사드가 앱의 모든 클래스에 결합된 God 객체가 될 수 있다.
- 퍼사드 클래스 자체가 서브시스템에 대한 의존성을 가지게 되어 의존성을 완전히는 피할 수 없다.

- 어찌되었건 추가적인 코드가 늘어나는 것이기 때문에 유지보수 측면에서 공수가 더 많이 들게 된다.
- 따라서 추상화 하고자하는 시스템이 얼마나 복잡한지 퍼사드 패턴을 통해서 얻게 되는 이점과 추가적인 유지보수 비용을 비교해보며 결정하여야 한다.

▼ 최소 지식 원칙 (디미터/데메테르 원칙)

- 최소 지식만 적용해 객체의 상호 작용을 설정하면 유지 보수가 용이해진다.
- 불필요한 객체의 생성 루틴과 재호출을 코드에 삽입해 코드의 가독성과 복잡성을 증가시키지 않도록 한다.
- 자기 자신만의 객체 사용
- 매서드에 전달된 매개변수 사용
- 매서드에서 생성된 객체 사용
- 객체에 속하는 매서드 사용

▼ 예제 코드

<https://hirlawldo.tistory.com/172>

[Java][디자인 패턴] 11. 파사드 패턴 (Facade Pattern)

디자인패턴 [Java][디자인 패턴] 11. 파사드 패턴 (Facade Pattern) 파사드는 요즘과 같이 협업과 대형 시스템을 개발하고 배포하는 데 자주 응용되는 패턴이다. (API 등) 파사드 패턴은 강


 <https://hirlawldo.tistory.com/172>



<https://inpa.tistory.com/entry/GOF-퍼사드Facade-패턴-제대로-배워보자>

퍼사드(Facade) 패턴 - 완벽 마스터하기

Facade Pattern 퍼사드 패턴(Facade Pattern)은 사용하기 복잡한 클래스 라이브러리에 대해 사용하기 편하게 간편한 인터페이스(API)를 구성하기 위한 구조 패턴 이다. 예를들어 라이브러리의

 <https://inpa.tistory.com/entry/GOF-퍼사드Facade-패턴-제대로-배워보자>

