Chapter 02. 디자인 패턴

디자인 패턴

Chapter 02. 디자인 패턴

디자인 패턴 이란?

디자인 패턴

자주 사용하는 설계 패턴을 정형화 해서 이를 유형별로 가장 최적의 방법으로 개발을 할 수 있도록 정해둔 설계

알고리즘과 유사 하지만, 명확하게 정답이 있는 형태는 아니며, 프로젝트의 상황에 맞추어 적용 가능 하다.

Gof 디자인 패턴

소프트웨어를 설계 할 때는 기존에 경험이 매우 중요하다. 그러나 모든 사람들이 다양한 경험을 가지고 있을 수는 없다.

이러한 지식을 공유하기 위해서 나온 것이 GOF (Gang of Four) 의 디자인 패턴이다. 객체지향 개념에 따른 설계 중 재사용할 경우 유용한 설계를 디자인 패턴으로 정리 해둔 것이다.

Gof 의 디자인 패턴은 총 23개 이며, 이를 잘 이해하고 활용한다면, 경험이 부족하더라도 좋은 소프트웨어 설계가 가능하다.

디자인 패턴의 장점

- 개발자(설계자) 간의 원활한 소통
- 소프트웨어 구조 파악 용이
- 재사용을 통한 개발 시간 단축
- 설계 변경 요청에 대한 유연한 대처

디자인 패턴의 단점

- 객체지향 설계 / 구현
- 초기 투자 비용 부담

생성 패턴

객체를 생성하는 것과 관련된 패턴으로, 객체의 생성과 변경이 전체 시스템에 미치는 영향을 최소화 하고, 코드의 유연성을 높여 준다.

- Factory Method
- Singleton
- Prototype
- Builder
- Abstract Factory
- Chaining

구조 패턴

프로그램 내의 자료구조나 인터페이스 구조 등 프로그램 구조를 설계하는데 활용 될 수 있는 패턴
클래스, 객체들의 구성을 통해서 더 큰 구조를 만들 수 있게 해준다.
큰 규모의 시스템에서는 많은 클래스들이 서로 의존성을 가지게 되는데, 이런 복잡한 구조를 개발 하기 쉽게 만들어 주고, 유지 보수 하기 쉽게 만들어 준다.

- Adapter
- Composite
- Bridge
- Decorator
- Facade
- Flyweight
- Proxy

행위 패턴

반복적으로 사용되는 객체들의 상호작용을 패턴화한 것으로, 클래스나 객체들이 상호작용하는 방법과 책임을 분산하는 방법을 제공 한다. 행위 패턴은 행위 관련 패턴을 사용하여 독립적으로 일을 처리하고자 할 때 사용.

- Template Method
- Interpreter
- Iterator
- Observer
- Strategy
- Visitor
- Chain of responsibility
- Command
- Mediator
- State
- Memento

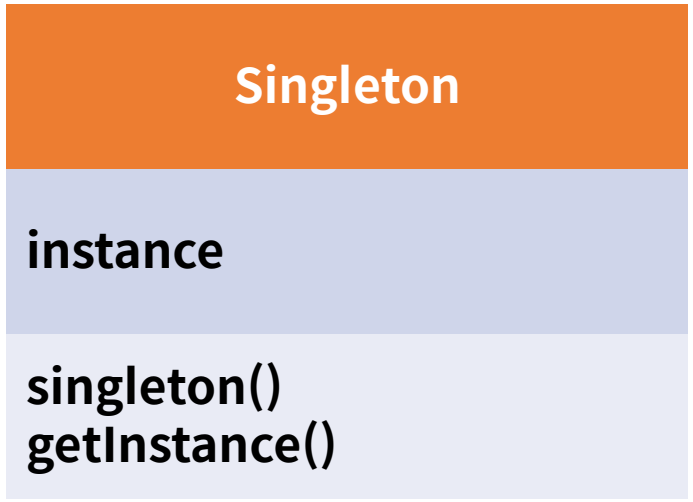
Chapter 02. 디자인 패턴

Singleton Pattern

Singleton pattern

Singleton 패턴은 어떠한 클래스(객체)가 유일하게 1개만 존재 할 때 사용한다.

이를 주로 사용하는 곳은 서로 자원을 공유 할 때 사용하는데, 실물 세계에서는 프린터가 해당되며,
실제 프로그래밍에서는 TCP Socket 통신에서 서버와 연결된 connect 객체에 주로 사용한다.

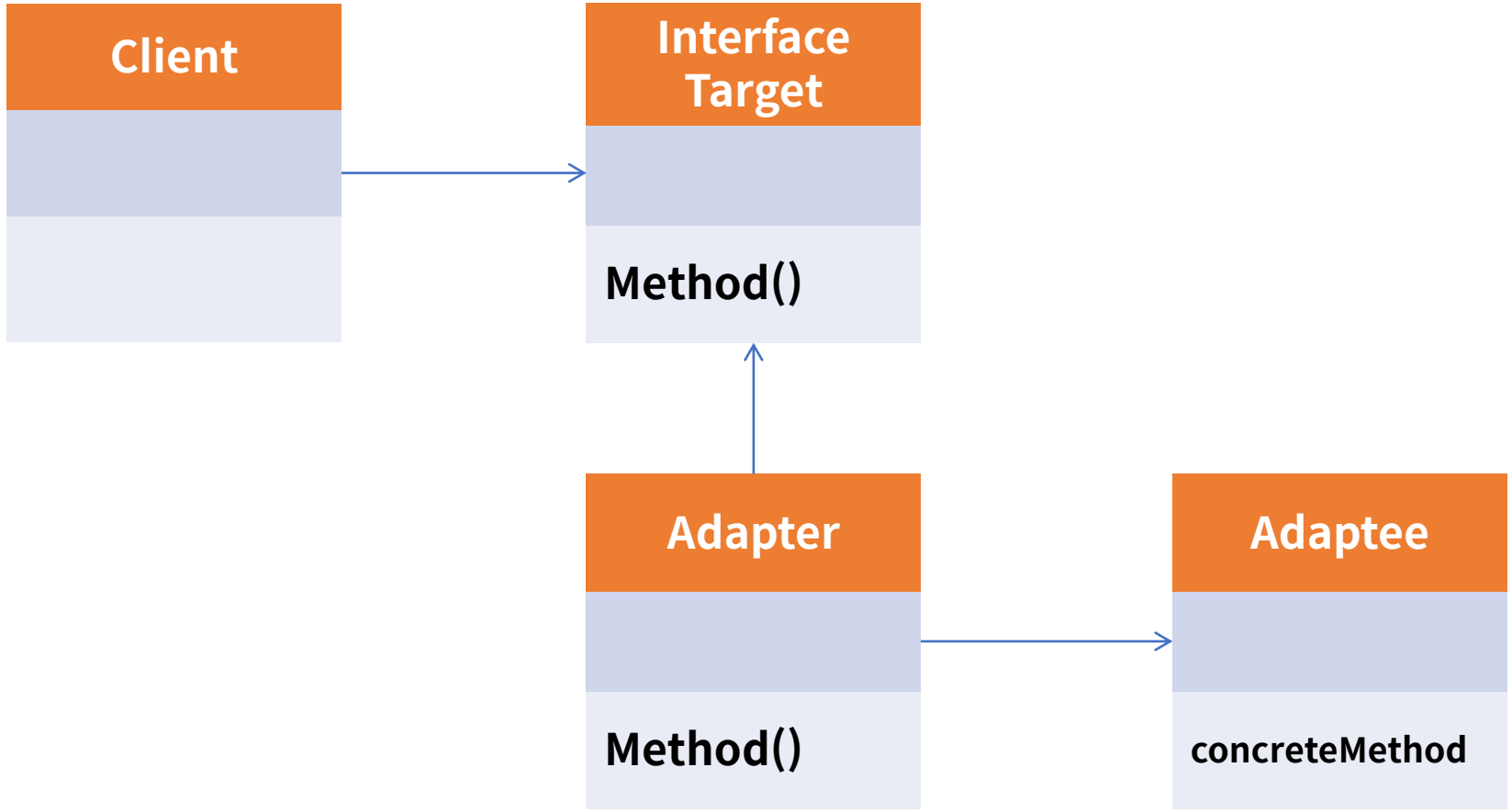


Chapter 02. 디자인 패턴

Adapter Pattern

Adapter pattern

Adapter는 실생활에서는 100v 를 220v로 변경해주거나, 그 반대로 해주는 흔히 돼지코 라고 불리는 변환기를 예로 들 수 있다.
호환성이 없는 기존 클래스의 인터페이스를 변환하여 재사용 할 수 있도록 한다.
SOLID중에서 개방폐쇄 원칙 (OCP)를 따른다.

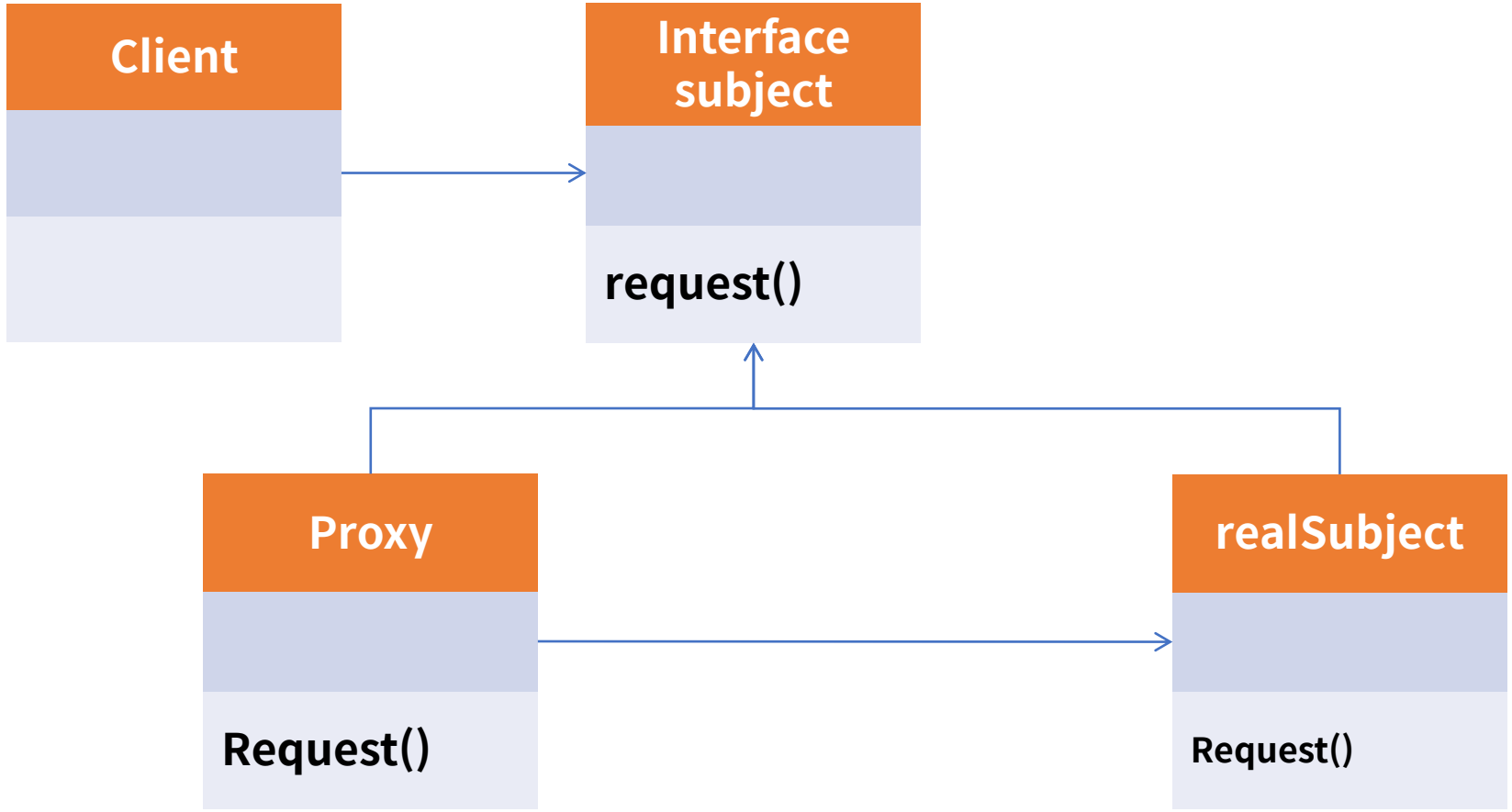


Chapter 02. 디자인 패턴

Proxy Pattern

Proxy pattern

Proxy는 대리인 이라는 뜻으로써, 뭔가를 대신해서 처리하는 것
Proxy Class를 통해서 대신 전달 하는 형태로 설계되며, 실제 Client는 Proxy 로 부터 결과를 받는다.
Cache의 기능으로도 활용이 가능 하다.
SOLID중에서 개방폐쇄 원칙 (OCP)과 의존 역전 원칙 (DIP)를 따른다.

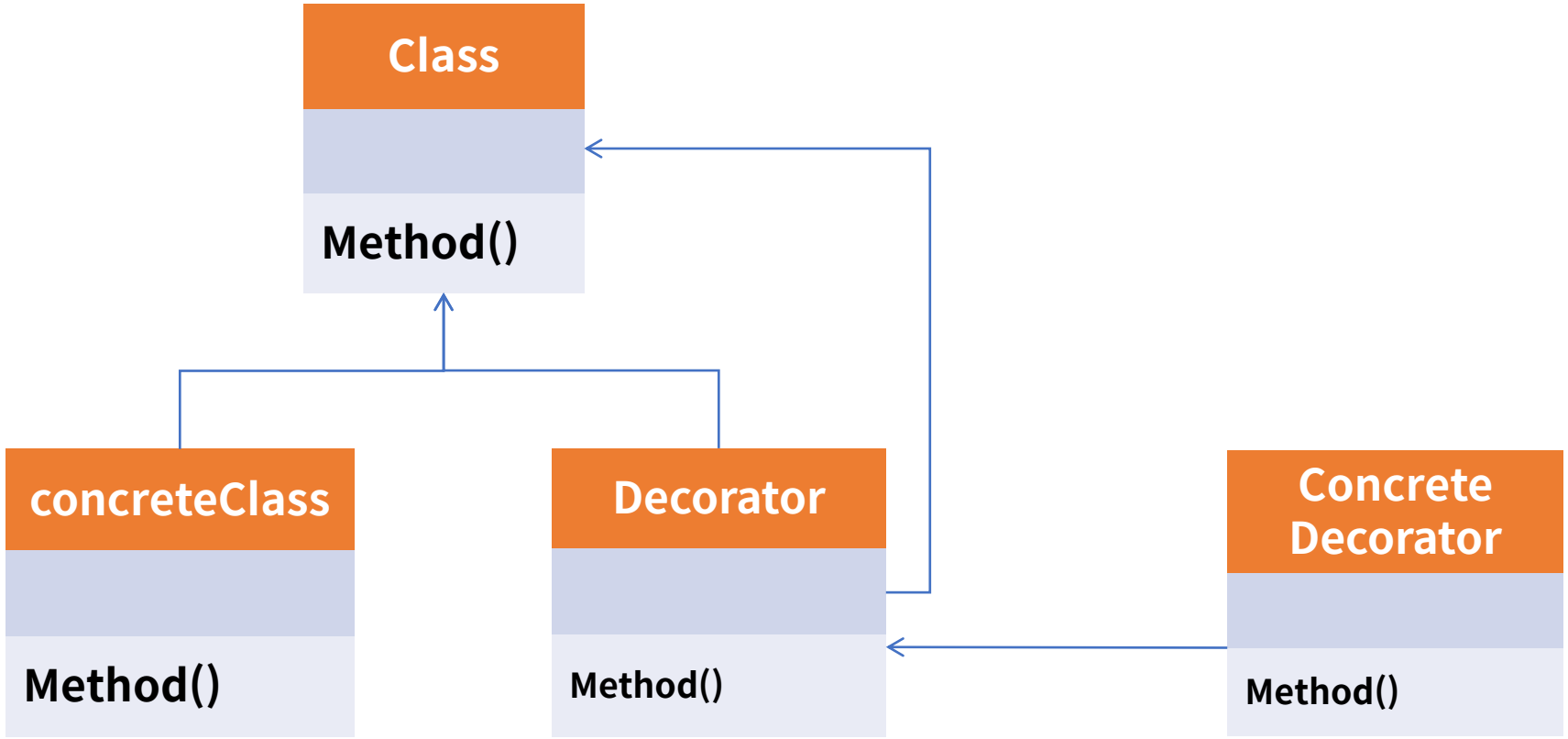


Chapter 02. 디자인 패턴

Decorator Pattern

Decorator pattern

데코레이터 패턴은 기존 뼈대 (클래스)는 유지하되, 이후 필요한 형태로 꾸밀 때 사용한다. 확장이 필요한 경우 상속의 대안으로도 활용 한다. SOLID중에서 개방폐쇄 원칙 (OCP)과 의존 역전 원칙 (DIP)를 따른다.

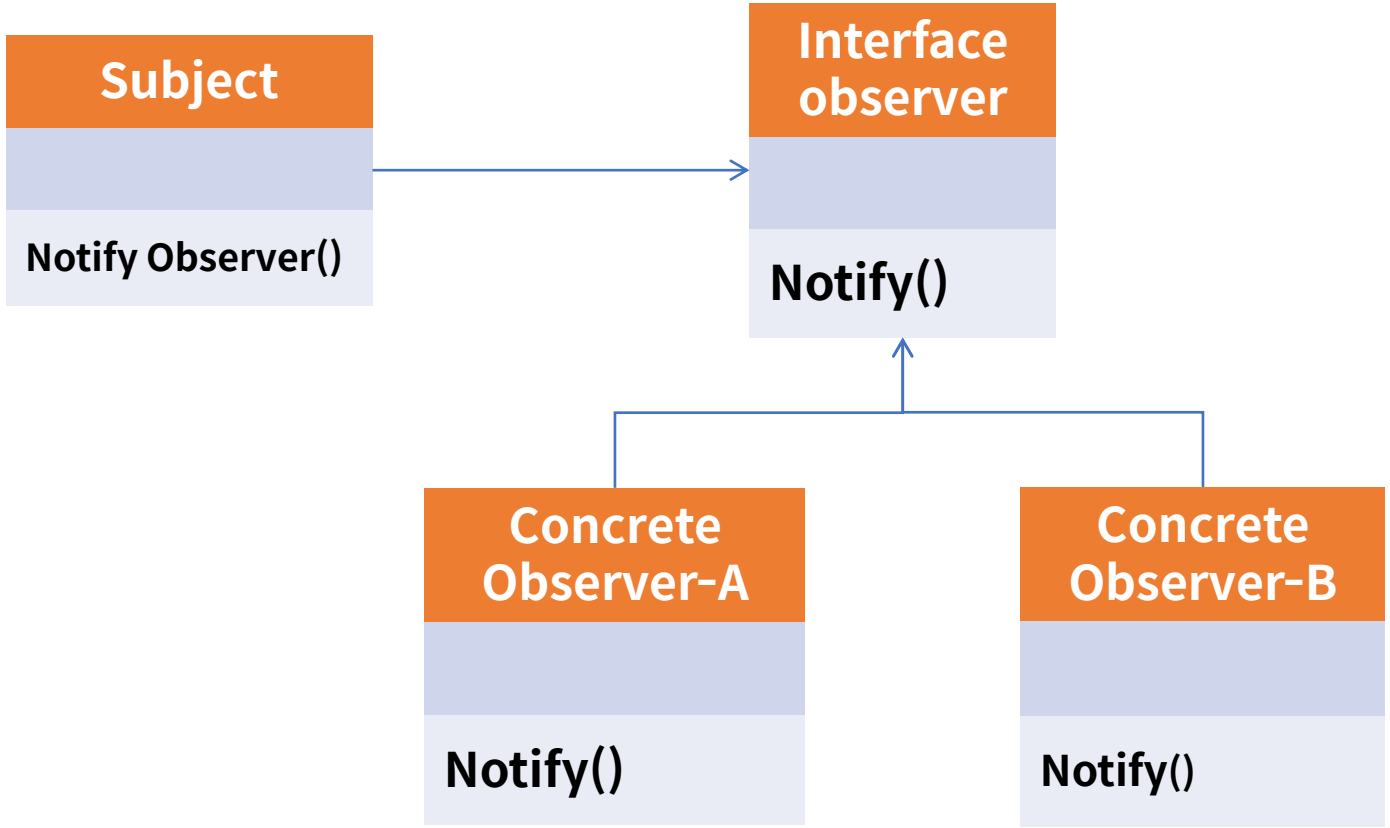


Chapter 02. 디자인 패턴

Observer Pattern

Observer pattern

관찰자 패턴은 변화가 일어 났을 때, 미리 등록된 다른 클래스에 통보해주는 패턴을 구현한 것이다.
많이 보이는 곳은 event listener 에서 해당 패턴을 사용 하고 있다.

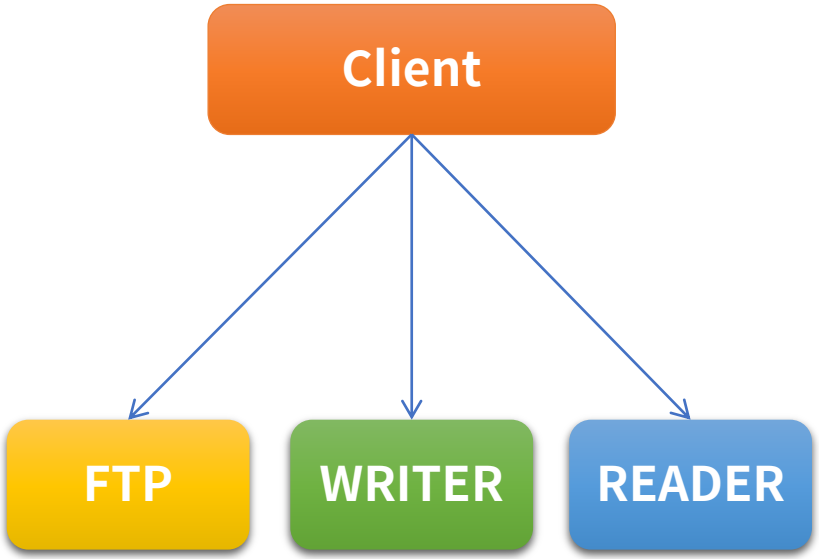


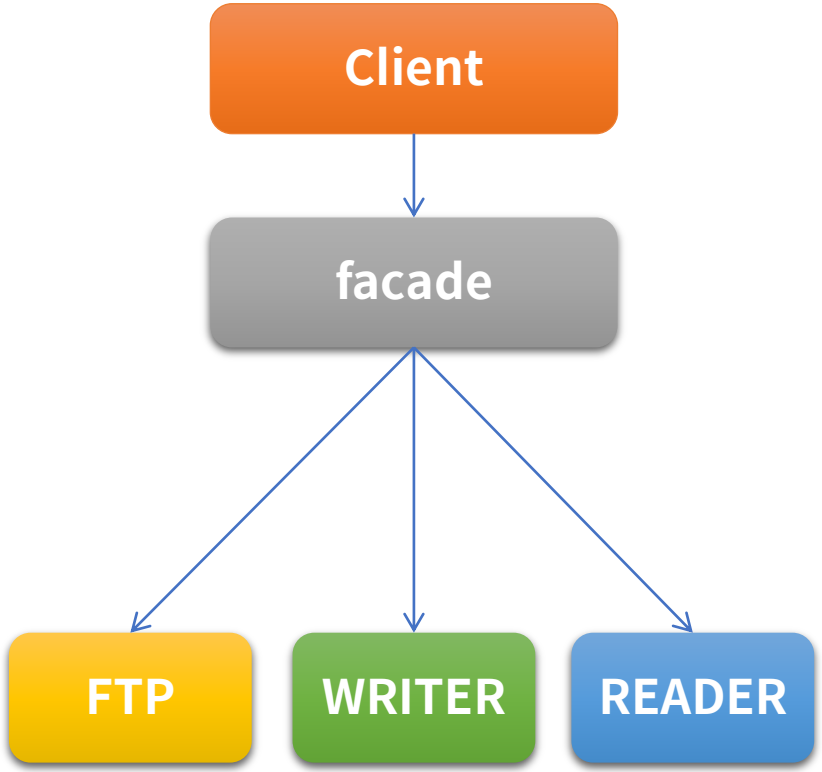
Chapter 02. 디자인 패턴

Facade Pattern

Facade pattern

Façade는 건물의 앞쪽 정면 이라는 뜻을 가진다. 여러 개의 객체와 실제 사용하는 서브 객체의 사이에 복잡한 의존간계가 있을 때, 중간에 facade 라는 객체를 두고, 여기서 제공하는 interface만을 활용하여 기능을 사용하는 방식이다. Façade는 자신이 가지고 있는 각 클래스의 기능을 명확히 알아야 한다.



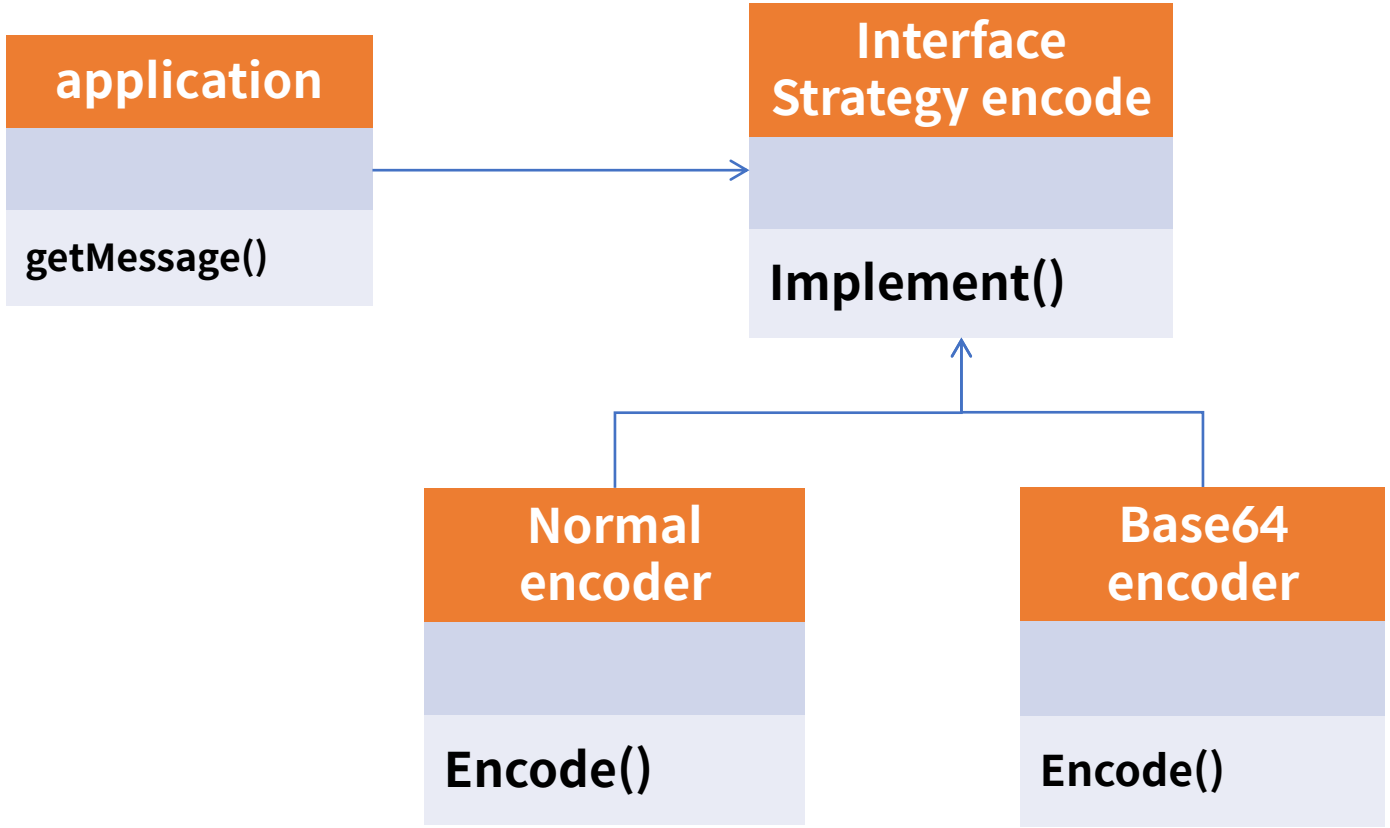


Chapter 02. 디자인 패턴

Strategy Pattern

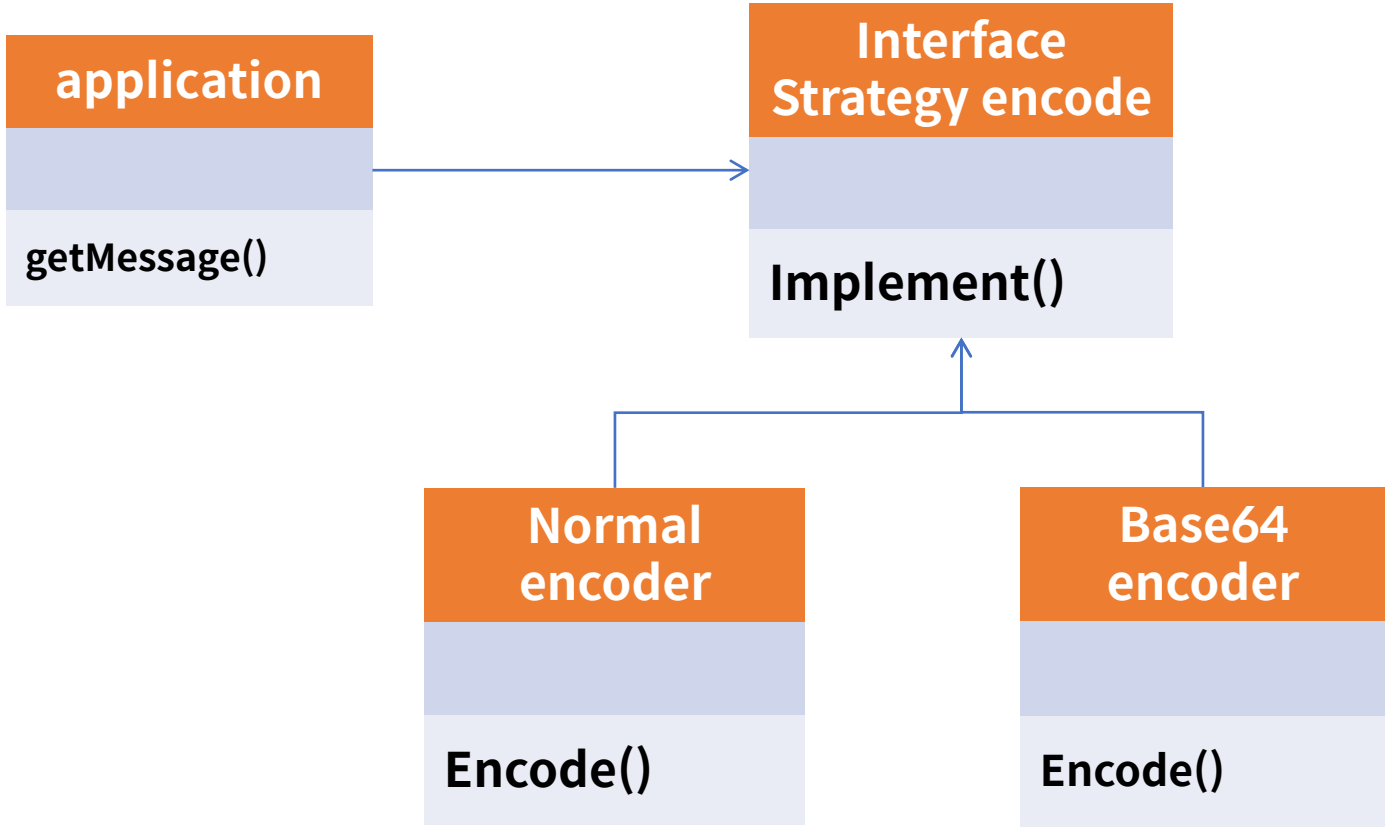
Strategy pattern

전략 패턴으로 불리며, 객체지향의 꽃이다.
유사한 행위들을 캡슐화하여, 객체의 행위를 바꾸고 싶은 경우 직접 변경하는 것이 아닌 전략만 변경 하여,
유연하게 확장 하는 패턴 SOLID중에서 개방폐쇄 원칙 (OCP)과 의존 역전 원칙 (DIP)를 따른다.



Strategy pattern

전략 메서드를 가진 전략 객체 (Normal Strategy , Base64 Strategy)
전략 객체를 사용하는 컨텍스트 (Encoder)
전략 객체를 생성해 컨텍스트에 주입하는 클라이언트



Chapter 02. 디자인 패턴

마치며···