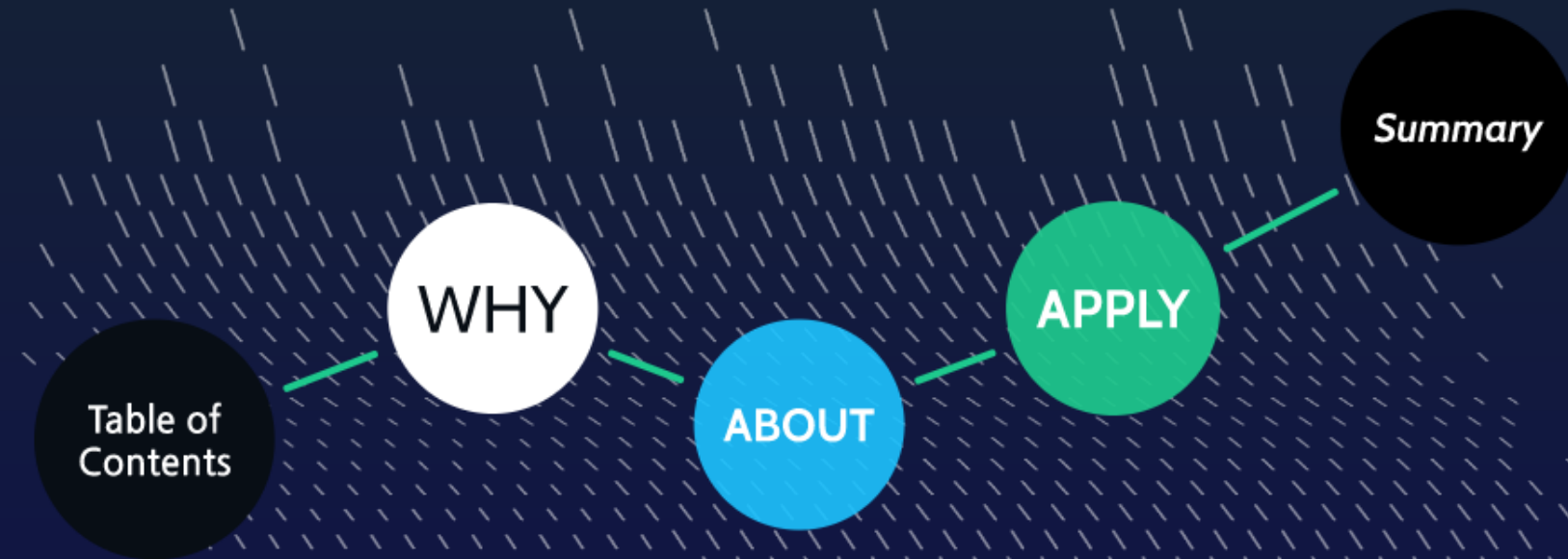


# Design Pattern

네버리스트

# Table of Contents

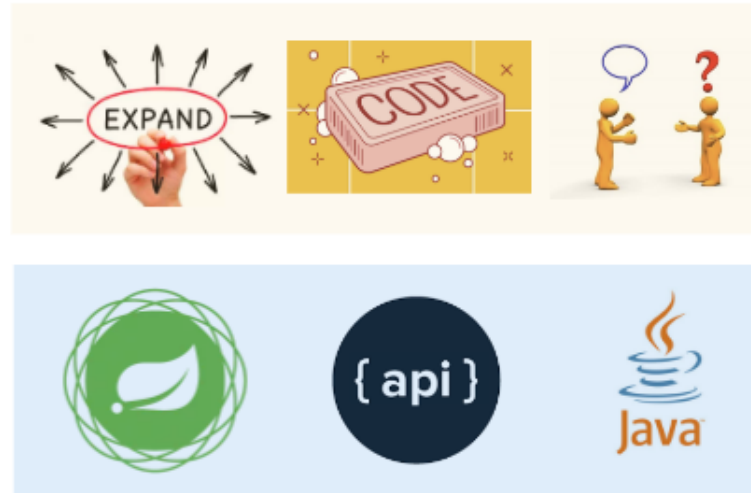
- **WHY** design pattern
- **ABOUT** design pattern
- **APPLY** design pattern

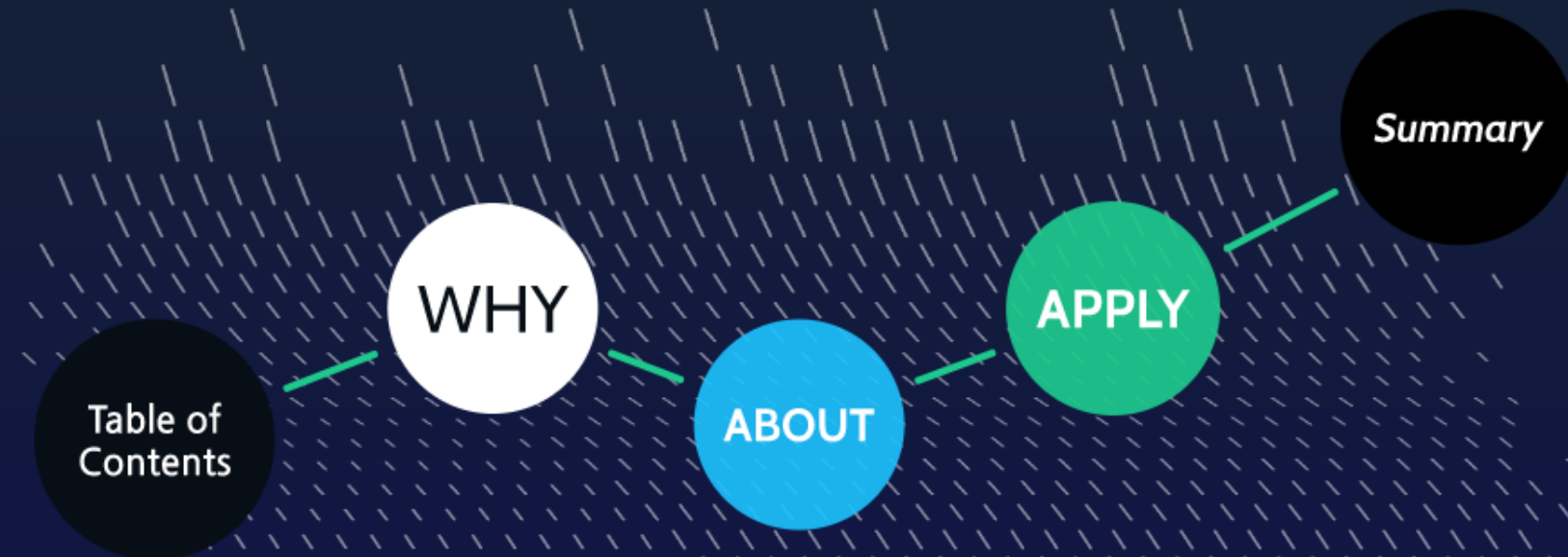


# Design Pattern

네버리스트

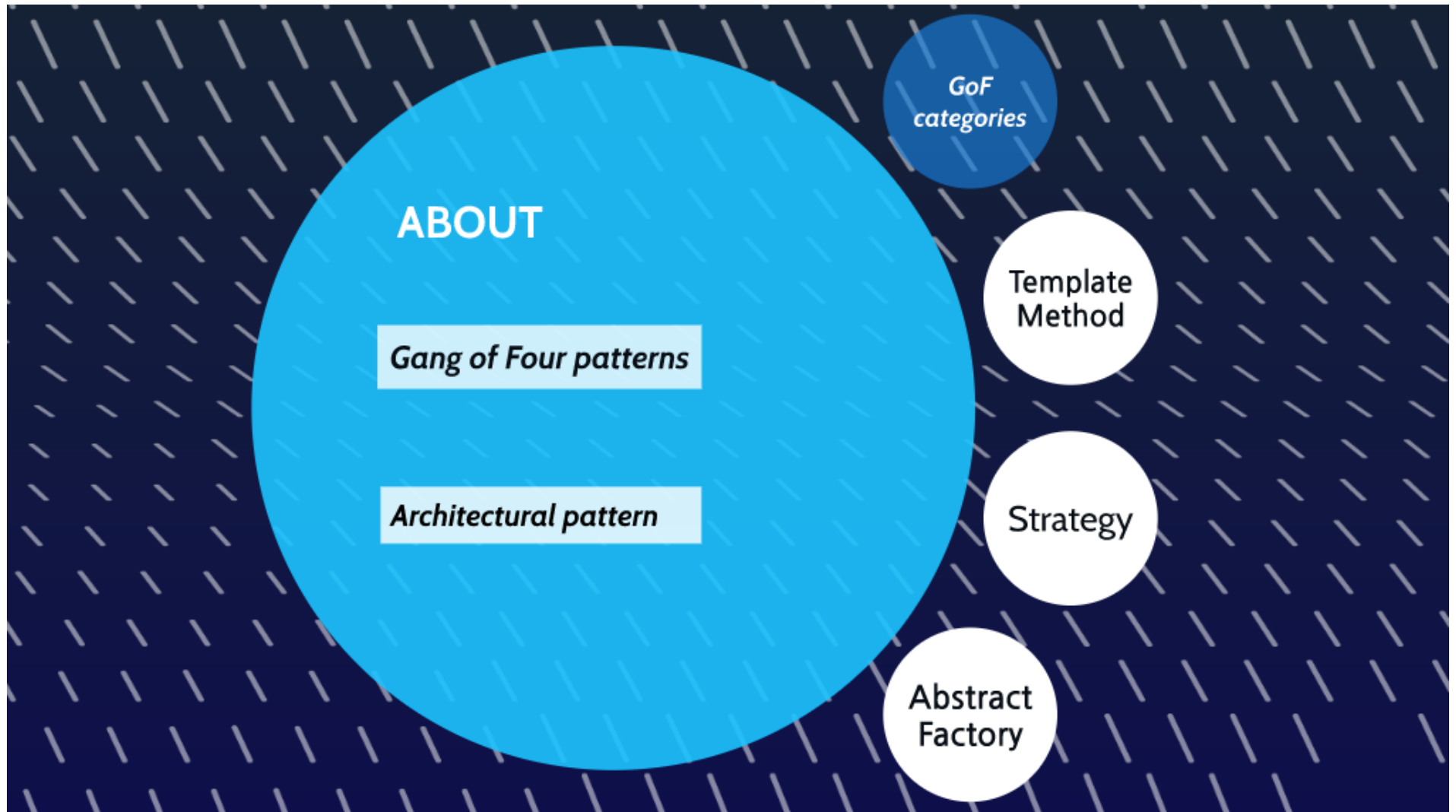
# WHY





# Design Pattern

네버리스트



# ***Gang of Four patterns***

# *GoF categories*

## Creational Patterns

Factory Method, **Abstract Factory**, Builder, Prototype, Singleton

## Structural Patterns

Decorator, Adapter, Composite, Facade, Flyweight, Proxy ...

## Behavioral Patterns

**Template method**, Iterator, Observer, **Strategy**, State ...



# Template Method

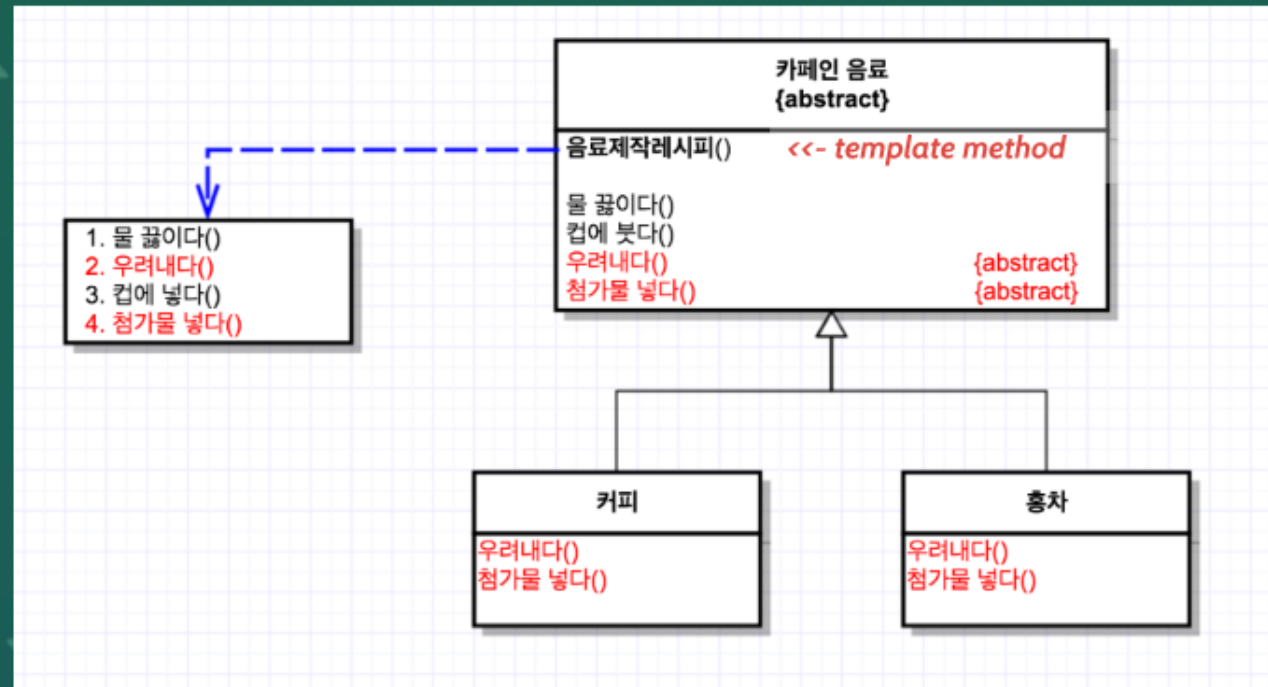
알고리즘의 구조를 유지하면서  
서브클래스에서 특정 단계를 재정의



*Example*

# Example

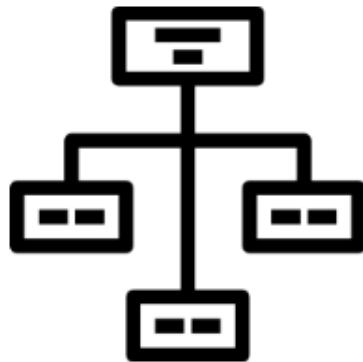
알고리즘의 구조를 유지하면서  
서브클래스에서 특정 단계를 재정의



Head First Design Pattern 335p

# Strategy

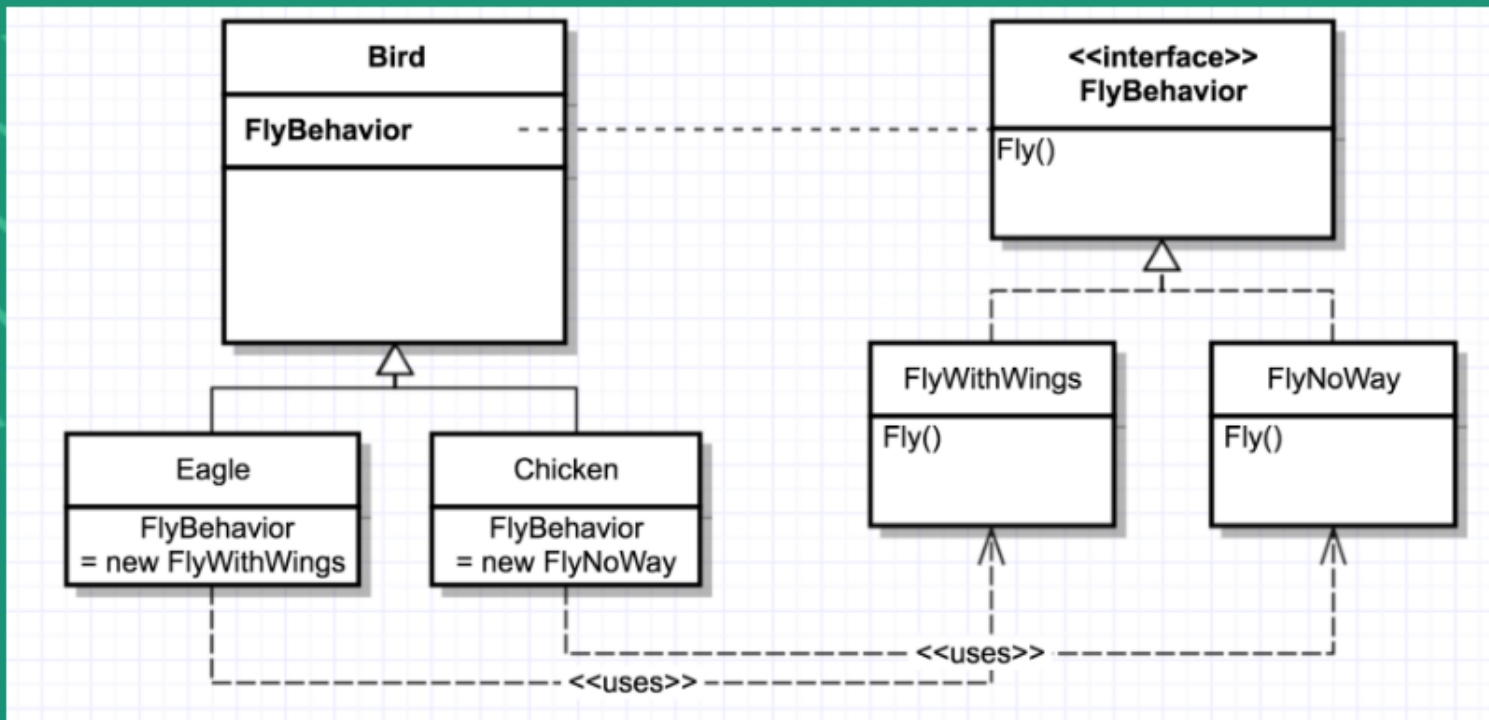
알고리즘 군을 정의하고  
각각을 캡슐화하여 교환할 수 있도록 하는 패턴



*Example*

# Example

알고리즘 군을 정의하고  
각각을 캡슐화하여 교환할 수 있도록 하는 패턴



# Abstract Factory

인터페이스를 이용하여 서로 연관되거나 의존하는 객체를 구상 클래스를 지정하지 않고 생성하게 하는 패턴



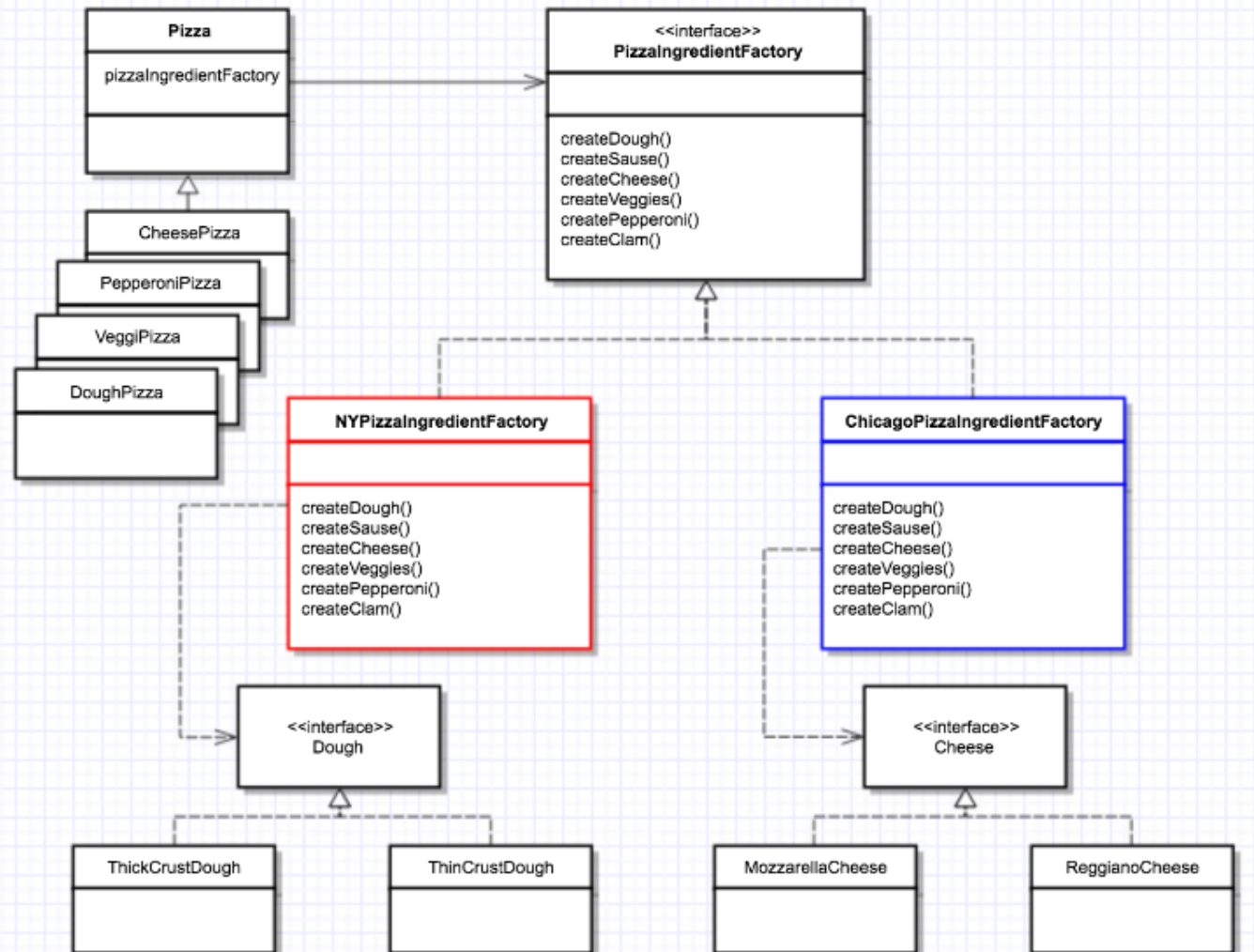
{abstract}

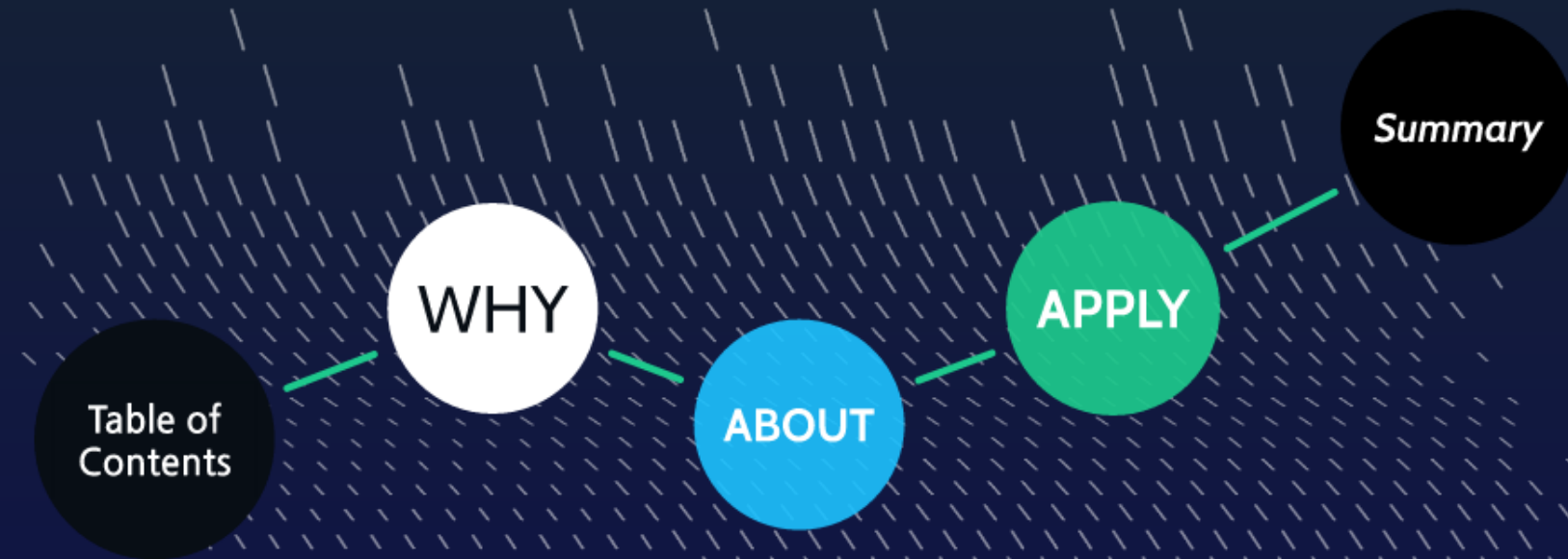
*Example*

# Example

인터페이스를 이용하여  
서로 연관되거나 의존하는 객체를  
구상 클래스를 지정하지 않고  
생성하게 하는 패턴

Head First Design Pattern 195.p





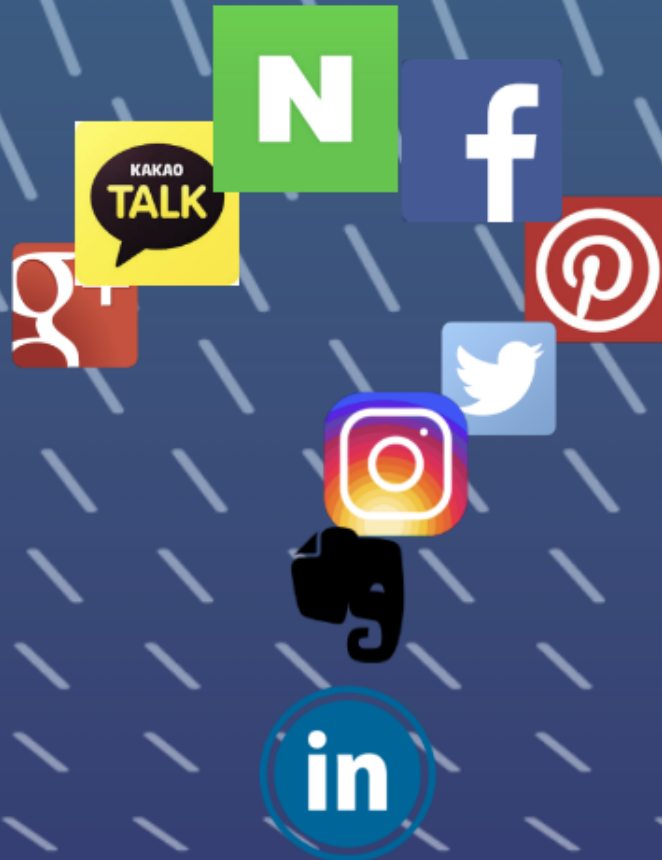
# Design Pattern

네버리스트





# OAuth 2.0 Provider 확장시..



# APPLY Before

## LoginController

- NaverLoginService
- GoogleLoginService
- .....
- N-th LoginService

NaverLoginService

GoogleLoginService

FacebookLoginService

KakaoLoginService

N-th LoginService

- private fields (apiKey, apiSecret...)
- login/getProfile 관련 메소드

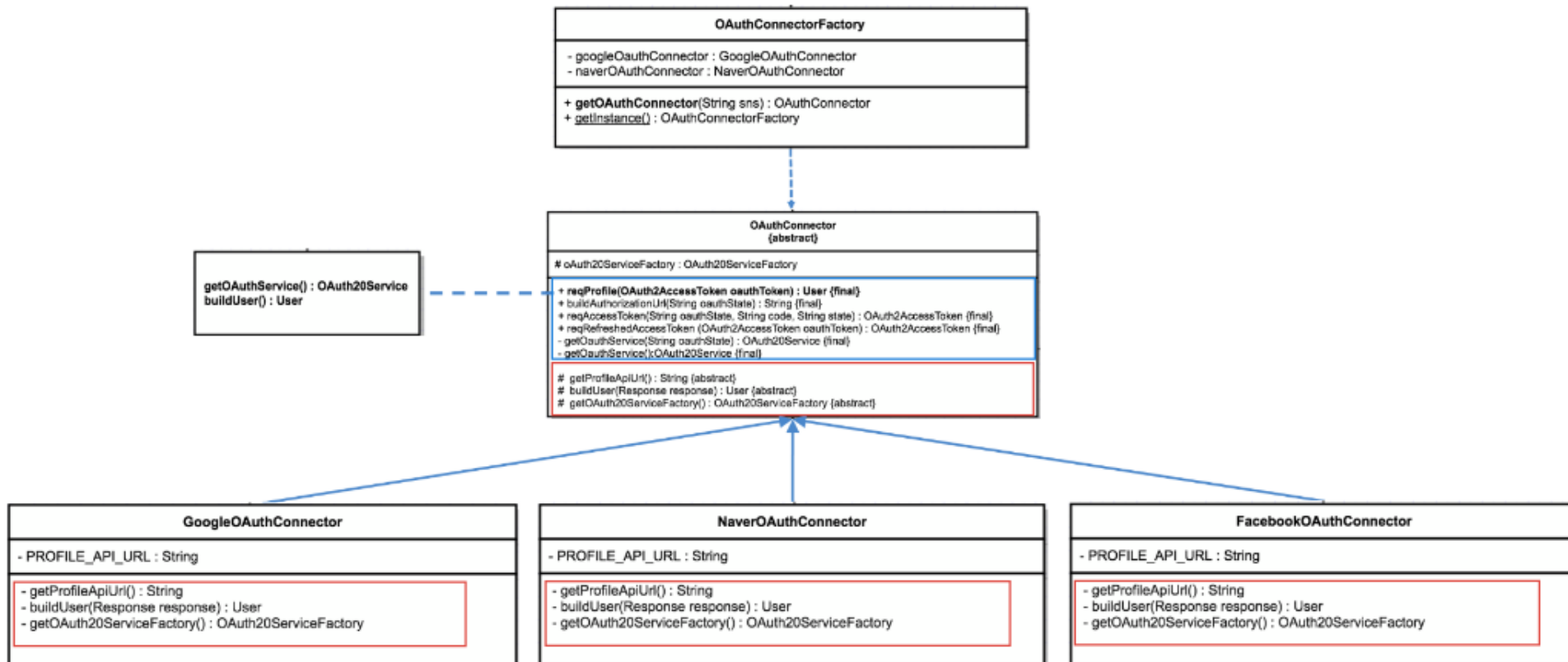
**OCP**

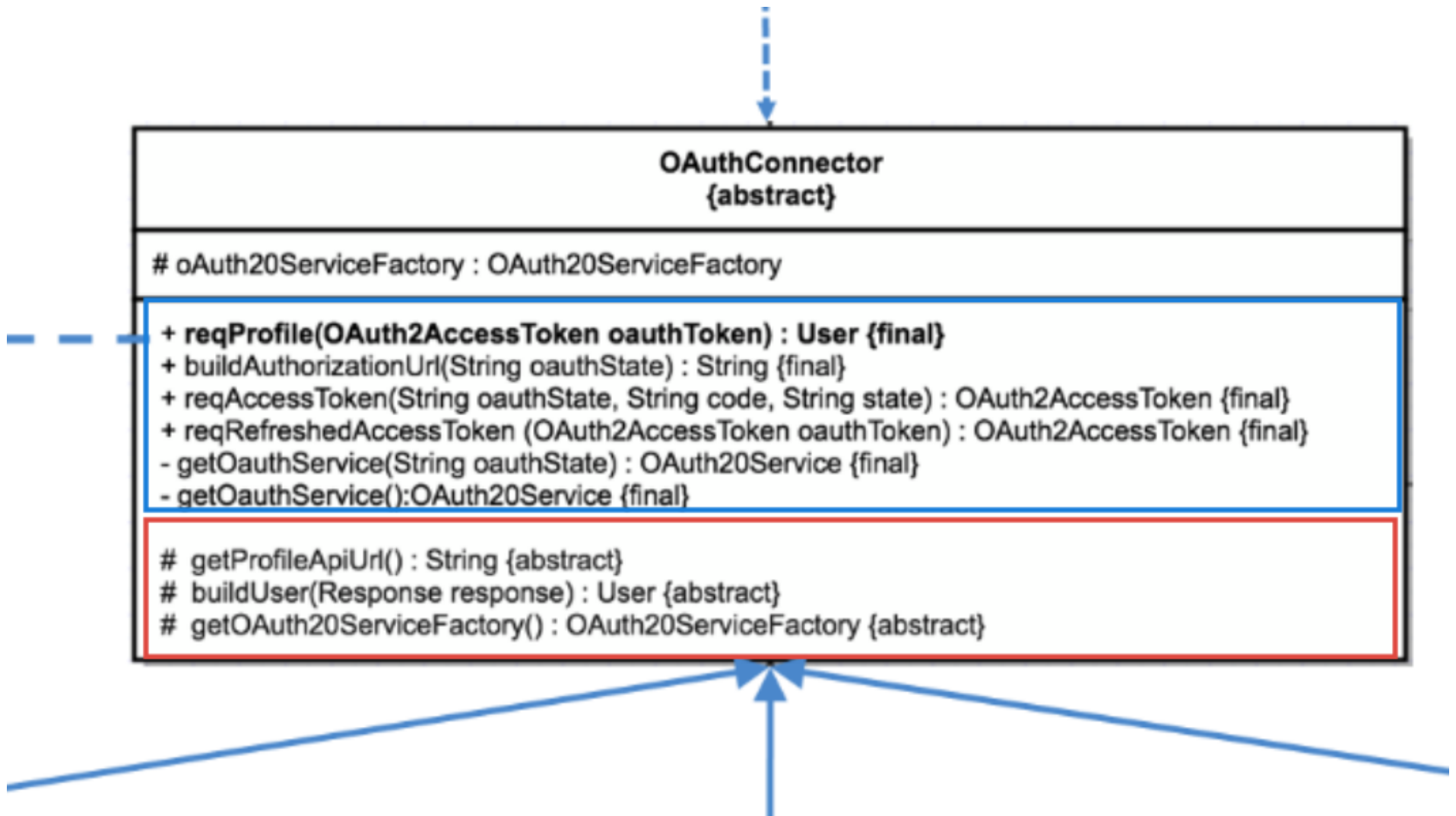
## Apply After

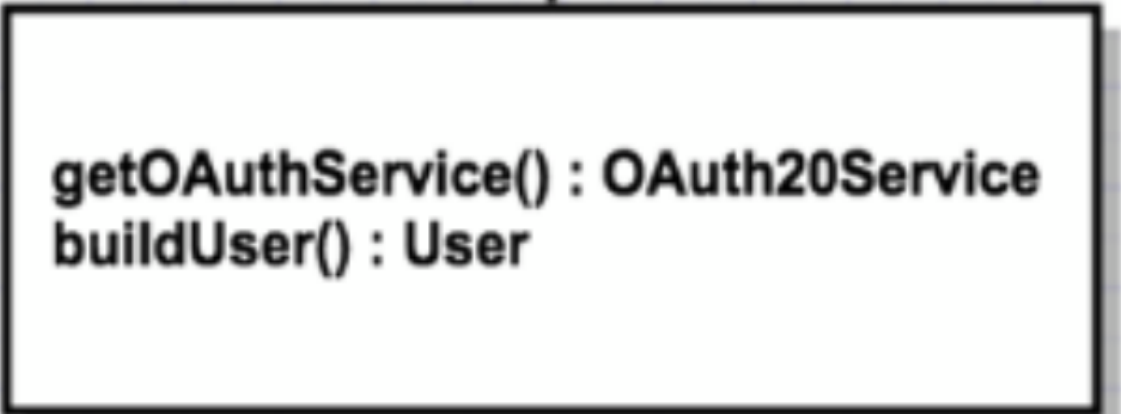
*Template  
method  
pattern*

*Abstract  
Factory  
pattern*

# Template method pattern



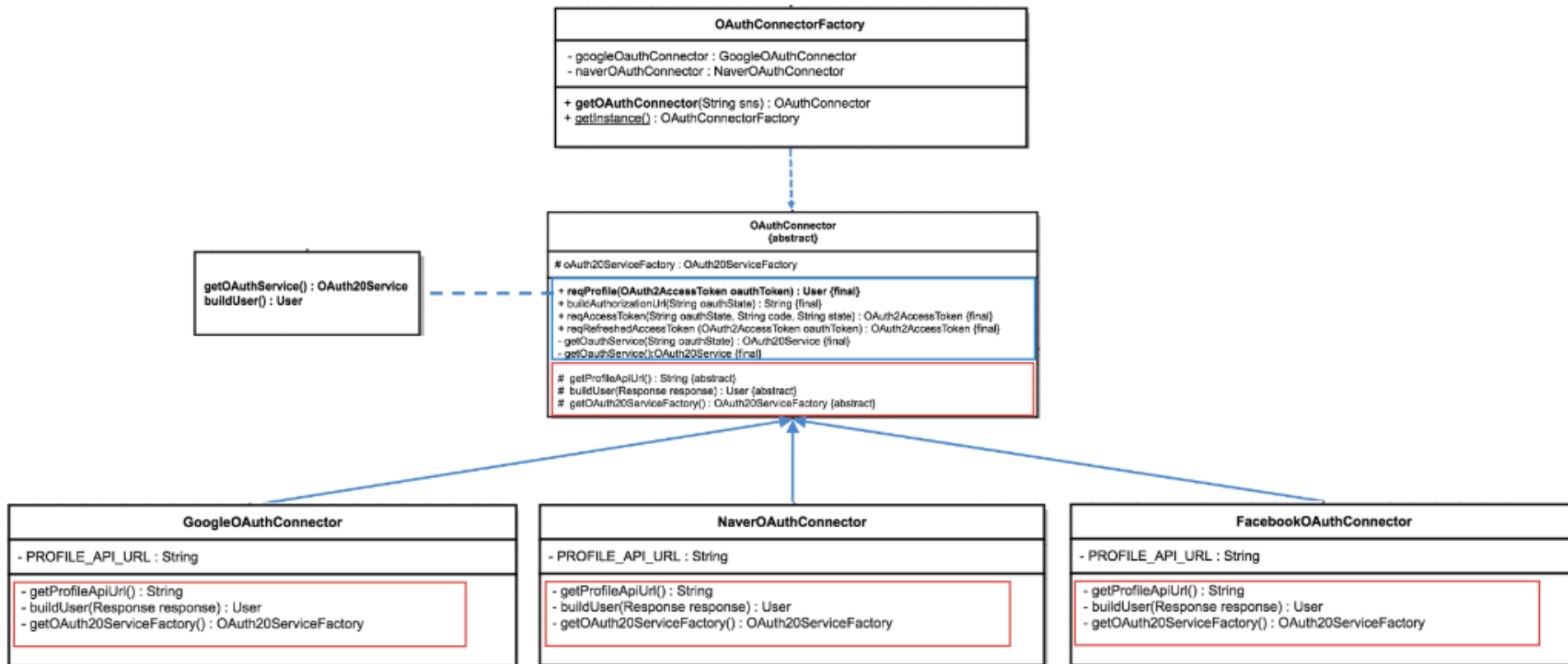




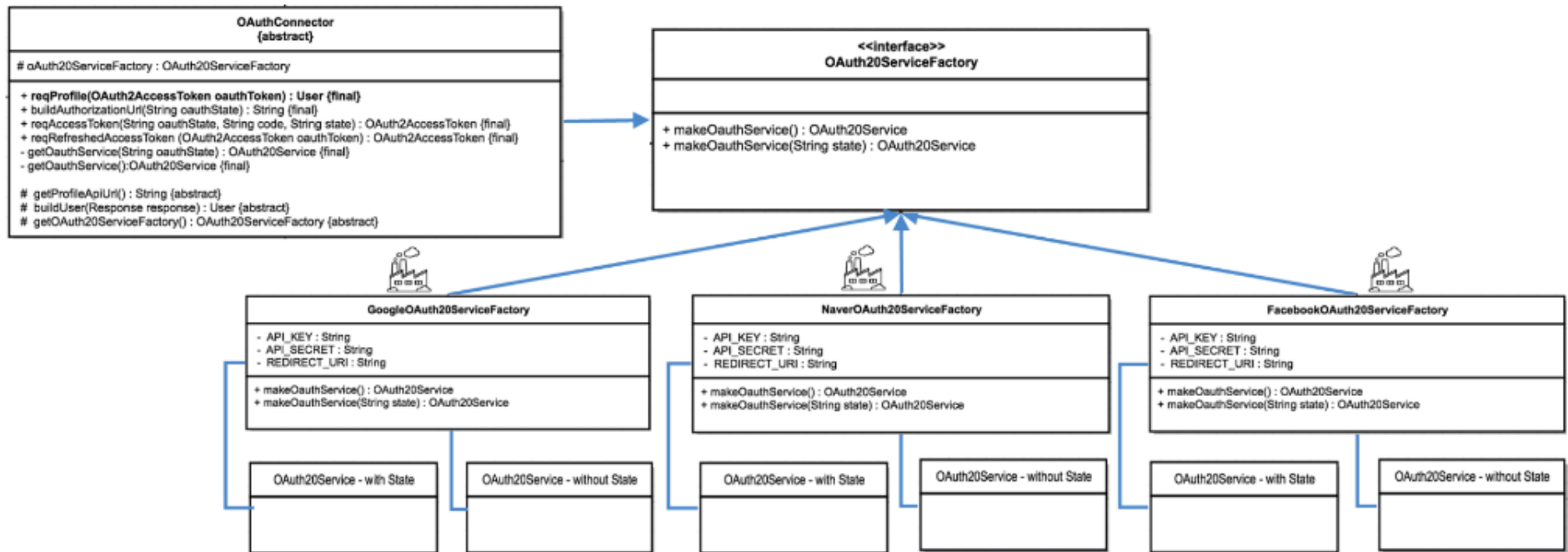
A UML class diagram element consisting of a rectangular box with a black border. Inside the box, two methods are listed: `getOAuthService() : OAuth20Service` and `buildUser() : User`. To the right of the box, there are two short, horizontal blue lines.

**getOAuthService() : OAuth20Service**  
**buildUser() : User**

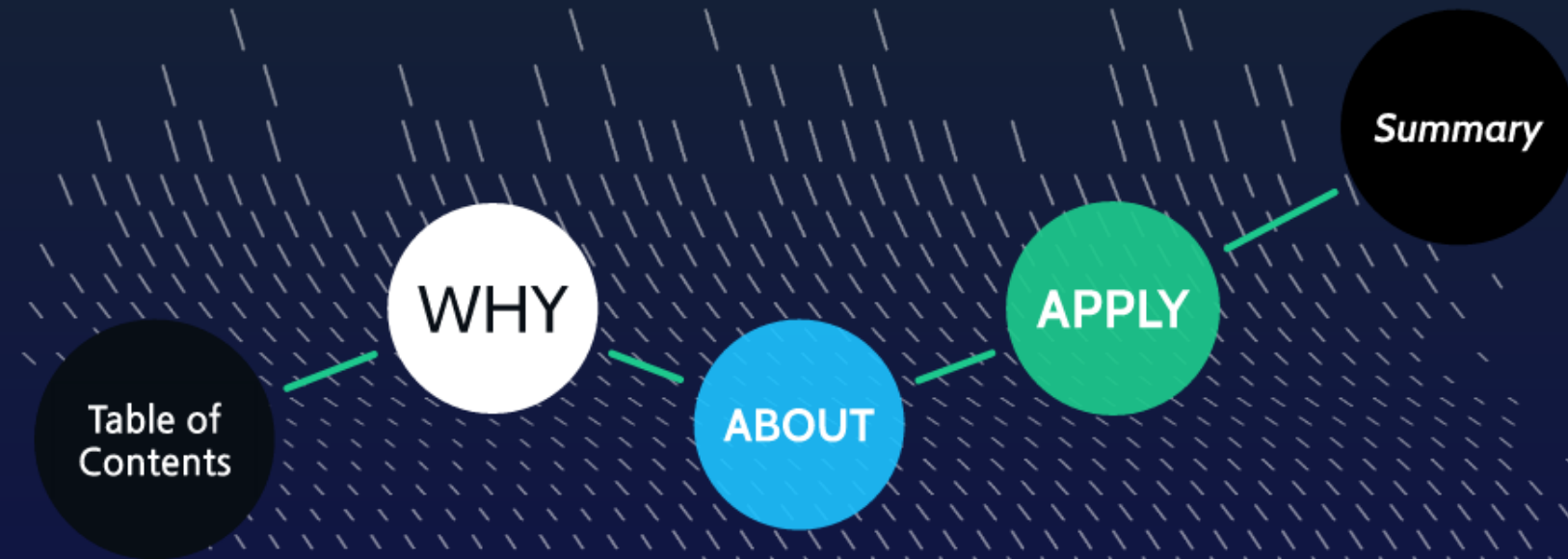
# Template method pattern



# Abstract Factory pattern







# Design Pattern

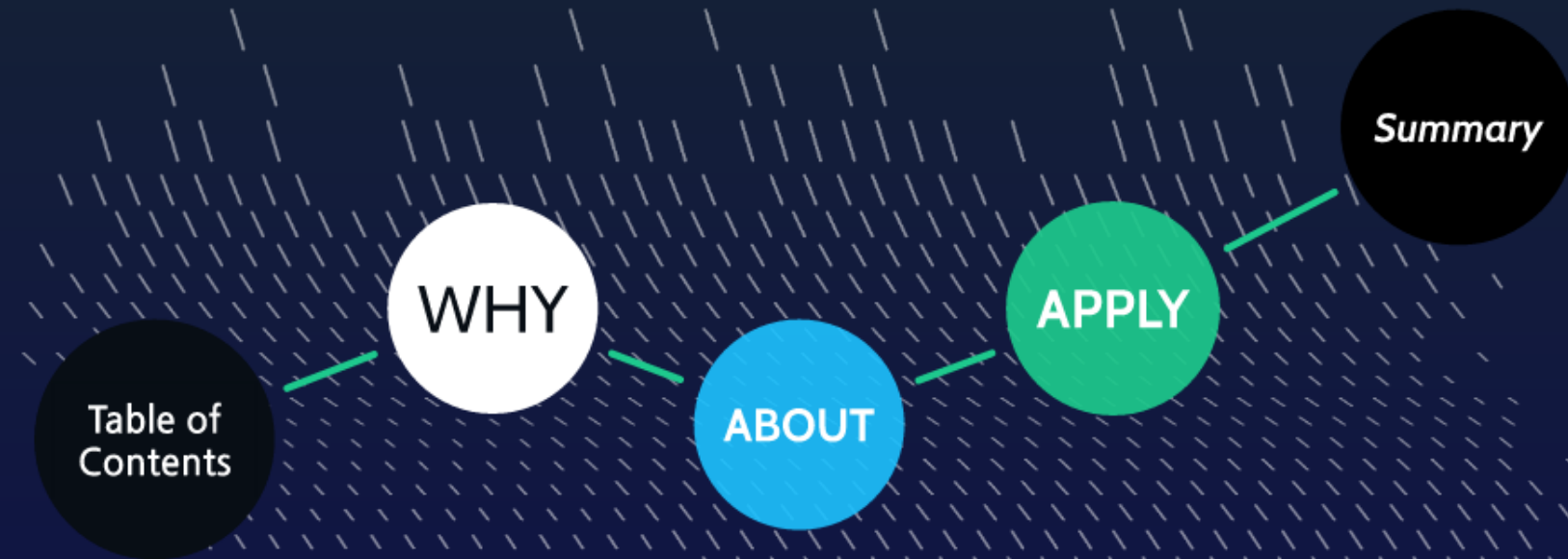
네버리스트

## *Summary*

반복되는 문제 해결을 위한 리팩토링에서 출발

변화의 시기가 다른 코드의 분리

정적인 구조보다는 패턴의 의도가 중요



# Design Pattern

네버레스트