Templete Method Pattern

결론 및 정의

정의

Template Method Pattern은 Behavioral Pattern 중 하나이다.
알고리즘의 흐름 / 단계를 정의하는 Superclass 가 있다 하자. 그 **각 단계는 개별적인 Method 로 구분**되어 있다.
이때, 이 **개별적인 Method - 개별적인 알고리즘의 구현을 SubClass 에게 위임하여 각 단계의 구현을 변경할 수 있도록 하는 Pattern** 이다.
이때, Subclass 에게 위임된 아직 구현되지 않았거나 바꿀 수 있는 Algoritm의 각 단계에 대응되는 Method 를 "**Hook Method**" 라 한다.

- **AbstractClass**: 알고리즘의 단계를 정의하며, 이 중 일부 단계는 추상 메서드로서 서브클래스에게 구현을 위임한다.
- **ConcreteClass**: AbstractClass를 상속받아 추상 메서드들의 구체적인 구현을 제공합니다.

구현

1. 알고리즘의 각 단계를 메서드로 정의하는 AbstractClass를 만든다. 이 중 변경될 수 있는 단계는 추상 메서드로 정의한다.
2. AbstractClass를 상속받아, 추상 메서드들의 구체적인 구현을 제공하는 ConcreteClass를 만든다.

Context & Problem & Merit

- 여러 알고리즘 또는 프로세스에서 공통적인 구조를 가지지만, 일부 단계에서는 다양한 구현이 필요할 때 Template Method Pattern을 사용합니다.
- 알고리즘의 구조는 변경되지 않지만, 특정 단계의 구현을 다양화하거나 확장하고 싶을 때 이 패턴이 유용합니다.

1. **코드 재사용**: 공통된 알고리즘의 구조를 한 곳에서 관리하므로 중복 코드를 줄일 수 있습니다.
2. **확장성**: 새로운 변형이나 구현을 추가하기 위해선 새로운 ConcreteClass만 추가하면 됩니다.
3. **제어 역전**: 고수준의 로직을 AbstractClass에서 관리하며, 저수준의 로직은 서브클래스에서 구현합니다. 이를 통해 코드의 흐름 제어가 슈퍼클래스에게 있게 되어 제어의 역전(Inversion of Control)이 발생합니다.

Design Pattern Principles

> ***Design Principle 8 : The Hollywood Principle*** = *Don't call us, we'll call you.*
>
> - *high level component 와 low level component 가 서로를 dependency 하는 상황을 철저히 지양한다.*
> - *low level components 는 그들 스스로 연결한다. high level component 는 언제, 어디에 필요한지 결정한다.*
> - *high level component 가 필요할 때, low level component 를* ***호출하는 방식****으로 구성한다.*