디자인 패턴

디자인 패턴이란

디자인 패턴(Design Pattern)은 **소프트웨어 설계에서 자주 등장하는 문제를 해결하기 위해 검증된 설계 방법을 정리해놓은 재사용 가능한 설계 템플릿**이다.

즉, '좋은 설계 방법의 모음집'이라 할 수 있다.

디자인 패턴 특징

- 코드 자체가 아니라 설계 아이디어

  - 디자인 패턴은 구체적인 소스코드가 아니라, 문제를 해결하는 구조적 아이디어다.

- 반복되는 문제 해결

  - 개발 현장에서 자주 마주치는 문제 상황에 대해, 이미 많은 개발자들이 검증한 '정답에 가까운 해결책'을 제공한다.

- 표준화된 용어 제공

  - "싱글톤(Singleton)", "팩토리 메서드(Factory Method)"처럼 이름만 들어도 어떤 구조와 의도를 알 수 있게 해준다.

일상 속 예시

- 건축 설계
  - 집을 지을 때 매번 처음부터 설계하지 않고, 표준 도면이나 구조를 참고한다.
- 레시피
  - 요리할 때 검증된 레시피를 참고하면 빠르고 안정적으로 같은 결과를 낼 수 있다.
- 소프트웨어에서도 마찬가지로, 검증된 설계 패턴을 사용하면 안정적이고 유지보수가 쉬운 프로그램을 만들 수 있다.

디자인 패턴의 장점

- 재사용성 (Reusability)
  - 이미 검증된 구조를 재활용하여 설계 시간을 단축하고 품질을 보장할 수 있다.
- 유지보수성 (Maintainability)
  - 구조와 역할이 명확해 수정 시 영향 범위를 쉽게 파악할 수 있다.
- 가독성 및 의사소통 향상 (Readability & Communication)
  - “이 부분은 옵저버 패턴이야”라고 하면 팀원들이 설계 의도를 바로 이해할 수 있다.
- 유연성과 확장성 (Flexibility & Extensibility)
  - 변화에 대응하기 쉽도록 설계 구조가 유연해진다.
- 품질 향상 (Quality Improvement)
  - 시행착오를 줄이고 안정적인 프로그램을 만들 수 있다.

GOF란

- Gang of Four의 약자.
- 네 명의 소프트웨어 공학자:

  Erich Gamma

  Richard Helm

  Ralph Johnson

  John Vlissides
- 1994년,『Design Patterns: Elements of Reusable Object-Oriented Software』책을 출간.
- 객체지향 설계의 모범 사례를 23개의 패턴으로 정리.

GOF 디자인 패턴의 의의

- 객체지향 프로그래밍(OOP)에서 반복적으로 등장하는 문제 해결 방법을 체계화.
- 개발자들 사이의 표준 언어 역할 수행.
- 패턴 사용 시 코드 구조와 의도를 쉽게 공유 가능.

23가지 패턴의 분류

- 생성(Creational) 패턴
  - 객체 생성 방식에 관련된 패턴
  - Singleton, Factory Method, Abstract Factory, Builder, Prototype
- 구조(Structural) 패턴
  - 클래스나 객체를 조합해 더 큰 구조를 만드는 방법
  - Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy
- 행동(Behavioral) 패턴
  - 객체 간의 상호작용, 책임 분배에 관한 패턴
  - Observer, Strategy, Command, State, Template Method, Iterator, Mediator, Memento, Visitor, Interpreter, Chain of Responsibility