Prototype Pattern

결론 및 정리

복잡한 Object 가 존재하고, 이를 복사하는 과정이 필요할 때.
Prototype Interface 를 만들어 두고, 해당 Interface 를 구현하는 Class 의 경우, 복사하기 쉽도록 clone() method 를 만들어 두어 관리하는 것이 이 Pattern 의 중점.

정의

Prototype Pattern은 Creational Pattern 중 하나로, **기존 객체를 복제(clone)하여 새로운 객체를 생성하는 패턴**. 이 패턴은 객체 생성에 관련된 비용이 크거나 시스템이 동적으로 객체를 생성하고자 할 때 유용함.

- **Prototype**: 복제될 객체의 원형이 되는 인터페이스나 추상 클래스. `clone()` 메서드를 선언 / 정의.
- **ConcretePrototype**: Prototype 인터페이스나 추상 클래스를 구현하는 클래스. `clone()` 메서드에서 실제 복제 로직을 구현.

구현

1. Prototype 인터페이스나 추상 클래스에서 `clone()` 메서드를 선언 / 정의한다.
2. ConcretePrototype 클래스에서 Prototype을 구현하며, `clone()` 메서드에서 객체의 복제 로직을 구현한다.
3. 클라이언트는 필요에 따라 `clone()` 메서드를 호출하여 새로운 객체를 생성한다.

Context & Problem & Merit

- 객체 생성의 비용이 크거나 복잡한 경우에 Prototype Pattern을 고려함.
- 클래스가 런타임에 동적으로 로드되거나, 객체의 상태가 다양한 조합으로 생성될 수 있을 때 유용함.

1. **성능 향상**: 기존 객체를 복제하는 것은 새로운 객체를 처음부터 생성하는 것보다 비용이 적을 수 있음.
2. **동적 생성**: 런타임에 특정 객체를 선택하고 복제하여 동적으로 새로운 객체를 생성할 수 있음.
3. **변형의 유연성**: 기존 객체를 기반으로 약간의 변형을 가한 새로운 객체를 쉽게 생성할 수 있음.