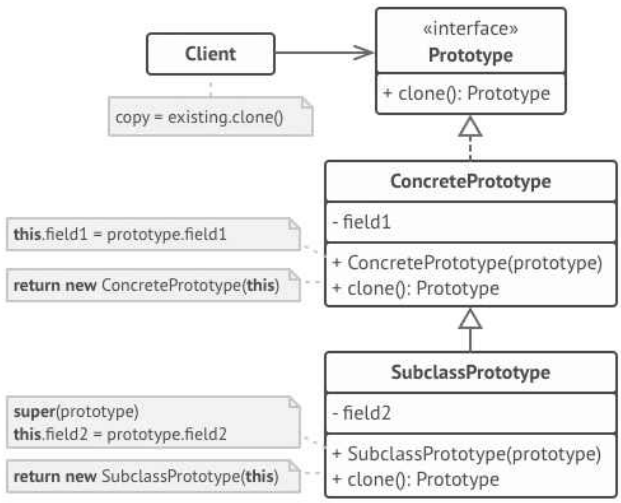
김현주
Prototype (생성 패턴)

prototype(원형)을 만들어놓고, 그것을 복사한 후 필요한 부분만 수정하면
new Object() 메서드로 객체를 생성하는 것보다 편리하다.
간단하게 객체를 이용해서 다른 객체를 생성하는 패턴이다.

예시 -
이미 만들어둔 적을 이용해서 같은 적을 계속 생산해낼 수 있는 것
하지만 트레이드 오프는 피할 수 없음. 기준이 되는 객체의 크기만큼 메모리의 용량을 차지하게 된다.

Prototype이 없어도 복사 생성자를 이용해서
객체 생성은 가능하다
하지만 Prototype을 이용하면 가독성이 올라간다.



▲ Prototype 클래스 다이어그램

Prototype: 인터페이스를 제공합니다.

Concrete Prototype: 부모 위치에 있는 클래스입니다. 복사 생성자를 정의해야 합니다.

Subclass Prototype: 자식 위치에 있는 클래스입니다. 복사 생성자를 정의해야 하고 부모 복사 생성자를 호출해야 합니다.

Prototype에서 핵심이 되는 부분은 clone()입니다.

소스 코드 예제들
https://github.com/ruvendix/CPP-GameCodingStep1/tree/master/Example/032-Prototype

확인할 부분은 복사 생성자가 제대로 호출되느냐 아니냐

이 패턴이 주로 필요한 상황은
1. 객체를 이용해서 객체를 생성해야 하는데 코드의 일관성을 지키고 싶을 때
2. 상속 관계에서 객체의 복사 생성자 계층을 자동으로 호출하고 싶을 때

장점	단점
별다른 비용 없이 객체로 객체를 생성할 수 있음. 게임처럼 비슷한 객체가 많을 때 유리한 패턴.	기본 객체가 메모리 공간을 차지하게 됨 (해제하는 시점은 자유)
코드 중복 없이 Clone() 하나로 인터페이스를 통일할 수 있음	복사 생성자와 Clone()을 무조건 구현해야 함... 상속 구조에서는 부모 복사 생성자 호출을 잊으면 망함...
	순환 참조가 있는 객체인 경우 복사 생성자 구현이 까다로움

도움 사이트

https://refactoring.guru/design-patterns/prototype

https://blog.naver.com/yoochansong/222102759167

https://sourcemaking.com/design_patterns/prototype

https://ko.wikipedia.org/wiki/%ED%94%84%EB%A1%9C%ED%86%A0%ED%83%80%EC%9E%85_%ED%8C%A8%ED%84%B4