5️⃣

의존 관계 역전 원칙 (DIP - Dependency Inversion Principle)

정의

> 프로그래미는 추상화에 의존해야지 구체화에 의존하면 안된다. 의존성 주입은 이 원칙을 따르는 방법 중 하나다.

의존 관계 역전을 하지 않은 경우 문제점

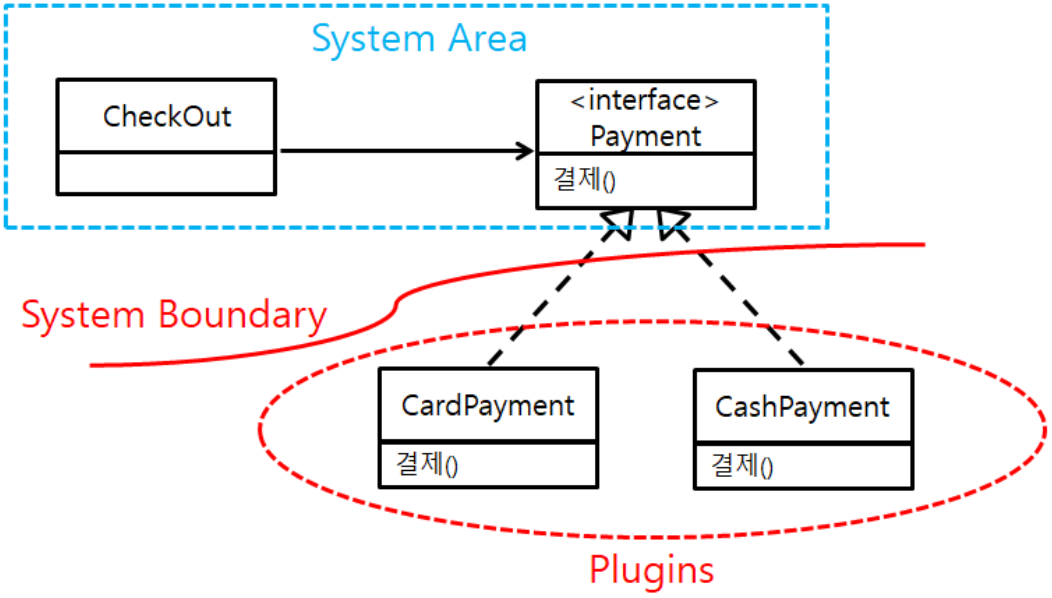
💡 OCP와 발생하는 문제는 동일하다.

해결 방법

💡 역시 OCP와 동일하다.

Appendix

OCP와 DIP의 차이점



OCP는 요구사항 변화에 대응하는 즉, 변경의 영향에 초점이 맞추어져 있다. 반면 DIP는 의존성을 역전시키는 것으로 Plugin을 만들 때 Boundary를 설정하고자 하는 영역의 의존성을 역전시킨다.

DIP는 같은 레이어에 포함된 의존성을 역전하는 것은 아니다. 바운더리를 정하고는 것이 중요하다. 따라서 시스템의 Boundary를 설정하고자 할 때 Boundary를 교차하는 부분의 의존성의 방향이 역전되도록 한다. 시스템쪽에서 바운더리 바깥으로 의존성이 생성되면 소스코드를 수정해야 할 일이 발생하게 된다.

플러그인은 대체 가능하거나 추가 제거가 용이한 형태로 되어 있으며 의존성 역전이 위 그림과 같은 플러그인 방식이 가능하도록 해준다. 시스템 영역을 만들고 플러그인으로 분리를 한 것은 다른 플러그인에 따라 다른 시스템처럼 동작할 수 있으며, 재사용 가능한 프레임워크를 만드는 것이다.