객체지향 5가지 설계 원칙

정보컴퓨터공학과 201924548 이풍헌

『객체지향 프로그래밍의 5가지 설계 원칙, 실무 코드로 살펴보는 SOLID[[1]](#footnote-1)』

C++에서 클래스의 기능을 알려주지만 왜 클래스를 사용해 구현해야 하는지에 대한 이유가 자세히 적혀 있지 않다. 학습내용에서 접하게 되는 소형 프로그래밍의 경우 객체지향적으로 설계하면 오히려 복잡해질 수 있기 때문에 객체지향을 처음 접하는 사람에게는 막막한 것이 사실이다. 우리가 실제로 해야 할 프로그래밍은 유지, 보수를 필요로 하는 대형 프로그래밍이 대부분이기 때문에 앞으로 설명할 객체지향 5가지 설계 원칙을 바탕으로 설계하면 도움이 될 것이다.

객체지향 5가지 설계 원칙은 다음과 같다.

1. 첫 번째 원칙은 단일 책임의 원칙이다. “하나의 클래스는 한 가지 책임을 가져야 한다는 원칙으로 클래스가 변경되는 이유는 한 가지여야 한다.” 변경이 필요한 경우 수정할 대상이 명확해지기 때문이다.
2. 두 번째 원칙은 개방 폐쇄 원칙이다. “프로그램의 기능을 확장에 대해서는 개방적이고 기존의 코드를 수정하는 것에 대해서는 폐쇄적이어야 한다.” 기존의 코드를 수정하면 의존하는 모든 코드를 수정해야 되기 때문이다.
3. 세 번째 원칙은 리스코프 치환 원칙이다. “하위 클래스는 상위 클래스를 대체할 수 있어야 한다.” 하위 클래스를 파라미터로 전달할 때 메서드가 이상하게 작동할 수 있으며 상위 클래스에 작성된 메서드가 서브 클래스에서 작동되지 않을 수 있기 때문이다.
4. 네 번째 원칙은 인터페이스 분리 원칙이다. “목적과 용도에 적합한 인터페이스만을 제공하여 불필요한 간섭을 없애는 것이다.” 첫 번째 원칙에서 클래스가 단일 책임을 가지듯이 인터페이스도 단일 책임을 가져야 한다. 인터페이스를 분리함으로써 의존성을 약화시켜 변경이 쉽게 만들 수 있기 때문이다.
5. 다섯 번째 원칙은 의존 역전 원칙이다. “상위 클래스는 하위 클래스의 구현에 의존해서는 안 되며 하위 클래스는 상위 클래스의 추상 타입에 의존해야 한다.” 하위 레벨 클래스의 변경이 상위 레벨 클래스까지 전파되는 것을 막기 위해서다.

위의 내용들은 결국 추상화를 통해 수정, 확장이 용이한 프로그램을 만드는 것을 목표로 한다. 소형 프로그래밍은 대부분 주어진 문제를 해결하는 것만 목표로 작성했기 때문에 기능을 추가하거나 단점을 보완한 경험이 없었을 것이다. 또한 개발 도중에 발견한 오류를 수정하기 위해서 코드의 처음부터 끝까지 디버깅을 했던 경우도 있었을 것이다. 그렇기에 객체지향 프로그래밍의 개념과 원칙을 습득하여 대형 프로그래밍 개발 환경에 익숙해져야 한다.

1. https://mangkyu.tistory.com/194 [↑](#footnote-ref-1)