왜 객체지향을 사용해야 하는가?

객체지향 세계에서는 여러 객체가 서로 메시지를 주고 받는 형식으로 코드가 구성된다.
이것을 통해 생각을 끊어낼 수 있다. A, B, C 객체가 있고 나는 A 객체를 개발할 때, B와 C 객체에 대해서는 고민할 필요가 없다. B와 C 객체와는 메시지를 통해 통신하기 때문에 그 구현을 고민할 필요가 없으며, 그래서 A 객체 개발에 집중할 수 있게 도와준다.

이것은 내가 뛰어난 개발자가 아니기 때문에 더 도움이 되는데, 뛰어난 개발자라면 복잡한 코드를 한번에 생각해내고 작성할 수 있겠지만, 나는 그렇지 않다. 그래서 몇가지의 스탭으로 이를 나눌 필요가 있는데 이때 객체지향이 정말 많은 도움이 되는 것 같다.

근데 그러면 절차지향에서 메서드를 적절히 분리하는 것만으로 달성할 수 있지 않나?

미완성된 메서드 몇 개를 만들고, 해당 메서드들에 의존하는 코드를 작성하는 방식으로 현재 구현하는 기능에 집중할 수 있지 않을까?

인정한다. 그래서 여기서 추상화의 개념이 들어온다고 느낀다.
내부 구현을 숨길 수 있다는 것은, 해당 내부 구현이 어떻게 바뀌어도 외부에서는 알 수 없다는 것이고 그럴 필요도 없다는 것이다. 그래서 추상화를 통하여 다른 객체를 사용하는 방법을 통해 해당 행동을 유연하게 수정할 수 있다.

이 말은 해당 객체가 여러 상황에서 유용하게 쓰일 수 있다는 말이고, 코드 중복을 줄이고 재사용성을 늘릴 수 있다.
이렇게 재사용성이 높아진 코드는 한 곳에서만 수정하고 한 곳에서만 확장할 수 있기 때문에 편리해진다.

편리하면 좋은건가?

사실 진짜 편리한건 코드를 한 곳에서 빠르게 작성하는 것이지 않을까?

맞다. 하지만 그렇게 절차지향적인 코드 역시 항상 좋은 것이 아니다. 수정과 확장의 포인트가 여러 곳으로 늘어나고, 복잡한 서비스의 경우 코드가 읽기 매우 어려워진다는 단점이 있

다. 객체지향은 수정과 확장이 용이하고, 복잡한 서비스의 경우에는 오히려 코드가 읽기 쉬워진다는 장점이 있으나, 간단할수록 오히려 장황한 코드가 탄생하고 사용하기 어렵다는 단점이 존재한다.

내가 우아한테크코스에서 공부하는 것은 객체지향을 잘하는 것이다. 잘 만들어진 객체지향 코드는 절차지향의 모든 단점을 장점으로 가지면서도, 잘 읽히고 장황하지 않다. 즉, 고점이 높기 때문에 객체지향을 공부하고 사용하는 것이다.