코딩 교육 메뉴얼 (2)

코딩 교육 메뉴얼은 현재 객체지향 프로그래밍을 수강하고 있는 학생들에게 프로그래밍 공부를 헤매지 않고 지속적으로 공부할 수 있을 지 고민하여, 주기적으로 코딩 공부에 대한 가이드라인이나 간략한 노하우를 알려주기 위해 작성되고 있습니다.

객체지향을 공부하며 자주 하는 궁금증

Q. 메서드와 함수의 차이가 뭐죠?

컴퓨터 프로그래밍 과목에서 C를 공부하고 나서, 처음 자바를 접하게 되면 **"메서드"**라는 용어에 당황스러움을 느낄 수 있습니다.

함수랑 비슷한 용도에 형태도 유사하게 생겼는데 왜 굳이 헷갈리게 메서드라는 다른 단어를 사용하는지 의문이 생기기도 합니다.

전통적으로 프로그래밍에서 함수라는 이름은 수학에서 따온 것입니다. 수학의 함수와 개념이 유사하기 때문이죠.

그러나 객체지향 개념에서는 함수(function) 대신 **객체의 행위나 동작을 의미하는 메서드(method)라는 용어를 사용합니다**.

메서드는 함수와 같은 의미이지만, 특정 클래스에 반드시 속해야 한다는 제약이 있기 때문에 기존의 함수와 같은 의미의 다른 용어를 선택해서 사용한 것입니다.

> *참고로 자바 문법을 계속해서 공부하다 보면 "람다식"이라는 개념을 학습할 수 있는데, 이는 다시 메서드가 하나의 독립적인 기능을 하기 때문에 함수라는 의미에서 "식"이라는 용어를 사용하게 되었습니다. 람다식의 도입으로 인해, 이제 자바는 객체지향 언어인 동시에 함수형 언어의 기능을 갖추게 되었습니다.*

Q. 객체지향 프로그래밍의 핵심은 결국 클래스 아닌가요?

객체지향 프로그래밍의 핵심이 클래스라는 생각은 객체지향을 공부하며 가장 많이 하는 오해입니다. 이는 어떻게 보면 자바 문법 그 자체를 학습하는데 너무 집중하기 때문에 나타나는 오해라고 생각됩니다.

그러나 객체지향의 핵심은 클래스가 아니며 당연히 객체지향도 클래스 지향은 아닙니다.

객체지향의 핵심을 한 마디로 얘기하자면, **적절한 책임을 수행하는 역할 간의 유연하고 견고한 협력 관계를 구축하는 것**입니다.

클래스는 협력에 참여하는 객체를 만드는 데 필요한 구현 메커니즘 중 하나일 뿐입니다.

지금은 이러한 말들이 너무 추상적이고 동떨어지게 느껴질 수 있겠지만, 앞서 얘기하였듯 중요한 것은 이런 생각들을 가진 채로 지속적으로 프로그래밍 공부를 해나가는 것입니다.

Q. 그래서 객체지향 프로그래밍이 다른 프로그래밍 방식과 달리 가지는 장점이 무엇이죠?

과거 전통적인 개발 방법은 데이터와 프로세스를 엄격하게 구분합니다. 이에 반해 객체지향에서는 데이터와 프로세스를 객체라는 하나의 틀 안에 함께 묶어 놓음으로써 객체의 자율성을 보장하죠.

이것이 전통적인 개발방법과 객체지향을 구분 짓는 가장 핵심적인 차이라고 할 수 있습니다.

자율적인 객체로 구성된 공동체는 유지보수가 쉽고 재사용이 용이한 시스템을 구축할 수 있는 가능성을 제시한다는 장점을 갖고 있습니다.

또한, 메시지를 수신한 객체가 실행 시간에(runtime) 메서드를 선택할 수 있다는 점은 다른 프로그래밍 언어와 객체지향 프로그래밍 언어를 구분 짓는 핵심적인 특징 중 하나라고 할 수 있습니다.

이것은 프로시저 호출에 대한 실행 코드를 컴파일 시간에 결정하는 절차적인 언어와 확연히 구분되는 특징이죠.