코딩 교육 메뉴얼 (1)

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.

Why 객체지향?

처음 객체지향에 대한 개념을 학습하게 되면 많은 것이 낯설고 어색하게 느껴지곤 합니다. 저도 처음 자바를 통해 객체지향 프로그래밍을 공부하기 시작했을 때를 떠올리면, 여러 부분에서 이전에 배웠던 C언어가 더 직관적이고 분명하게 느껴졌던 기억이 납니다. 그 중에서 특히 제가 객체지향을 처음 공부하며 가장 많이 느꼈던 것은 “대체 이런 기능을 왜 만들었을까??” 였던 것 같습니다. 캡슐화, 상속, 추상화 … 이런 개념들이 저에게는 다소 불필요하고 번거롭게만 느껴졌던 기억이 납니다.

그러나 안타깝게도 당장의 설명만으로 모든 궁금증을 해소하긴 어렵습니다. 왜냐하면 이러한 것들의 대부분은 실제 개발에 참여하고 어떤 애플리케이션을 구현해보는 과정에서 객체지향의 필요성을 체감하면서 터득되기 때문입니다. 그럼에도 객체지향을 깊게 공부해야 하는 이유는 **“현재의 객체지향 패러다임은 수많은 개발자들이 오랜 기간의 연구로 축적된 애플리케이션 개발에 가장 잘 어울리는 프로그래밍 방식”**이기 때문입니다. 이 메뉴얼 하나만으로 많은 궁금증들을 해소시킬 순 없겠지만, 앞으로 객체지향의 원리를 공부하고 프로그래밍하는데 있어 자그마한 도움이 되길 바라겠습니다.

객프는 결국 자바 문법 공부?

객체지향의 개념을 처음 학습하시는 분들이라면 자바 문법이 주는 낯선 느낌으로 인해 충분히 그렇게 느끼실 수 있을 것이라 생각이 됩니다. 자바라는 언어에 익숙해지는 것은 당장에 객프 과목을 수강하는 데 있어서 당연히 매우 중요한 부분입니다. 그러나 중요한 것은 자바 문법을 공부하고 코딩을 하면서도 객체지향 패러다임에 대한 지속적인 관심과 궁금증을 가지고 있어야 합니다. 그래야 공부를 해나가며 조금씩 그 궁금증이 해소되고 객체지향의 원리에 차츰 가까워질 수 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 단언컨데, 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.

Why Java?

이번 메뉴얼에서 마지막으로 드리고 싶은 이야기는 자바라는 언어에 대한 이야기입니다. 이 이야기는 객체지향 프로그래밍을 수강하는 여러분께 꼭 드리고 싶었던 주제였는데, 객체지

향 프로그래밍을 공부하며 너무 자바라는 언어에만 집중하고 있지는 않은지 꼭 점검했으면 좋겠다는 생각이 들어 이렇게 주제를 잡아보았습니다.

세상에는 객체지향 패러다임을 적용한 수많은 언어들이 존재합니다. 제 주변에는 굉장히 유능한 개발자이지만 자바는 거의 다루지 못하는 분도 있습니다. 대부분의 언어는 특정 분야에서의 필요성에 의해 탄생하기 때문에 각 언어는 각자 특화된 분야가 존재합니다. 그 말은 여러분이 미래에 취업하게 될 분야에서는 자바를 전혀 사용하지 않을 수도 있다는 것입니다. 물론 당장에 자바 공부를 하지 말라는 이야기는 아니지만, 이 과목을 처음 접하는 학생 여러분은 자바가 지닌 세세한 기능을 외우는데 집중하는 것이 아닌 객체지향의 원리와 자바가 그 패러다임을 어떻게 구현했는지에 집중해서 공부를 이어가셨으면 좋겠습니다. 여러분이 미래에 자바를 사용하지 않을 수는 있지만, 객체지향 개념을 모르고 좋은 엔지니어로 성장하긴 힘들기 때문입니다. 틀에 박힌 이야기일 수 있지만, 코딩 언어의 나무보다는 객체지향이라는 거대한 숲을 바라보며 공부하시기 바랍니다.