코딩 교육 메뉴얼 (4)

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.

지난 시간에 객체의 추상화와 은유에 대한 개념을 이야기하며 객체지향 프로그래밍이 단순히 현실 세계의 모방하는 것이 아님을 설명하였습니다.

이번 시간에는 은유에 대한 추가 설명과 함께 굉장히 유명하면서도 까먹기 쉬운 객체지향의 네 가지 특징과 객체지향 설계 원칙을 소개하도록 하겠습니다.

은유(metaphor)

은유라고 하면 흔히 '내 마음은 호수' 와 같이 시에서 사용하는 것이라 생각되기 마련입니다. 이러한 단어 간의 은유 관계는 객체지향 모델링에서 발견되는 현실 객체와 소프트웨어 객체의 관계와 일치합니다.

현실 속의 객체의 의미 일부가 소프트웨어 객체로 전달되기 때문에

프로그램 내의 객체는 현실 속의 객체에 대한 은유가 되기 때문입니다.

은유는 **표현적 차이**(representational gap)라는 논점과 관련성이 깊은데요.

여기서 차이란 소프트웨어에 대해 사람들이 생각하는 모습과 실제 소프트웨어의 표현 사이의 차이를 의미합니다.

예를 들어, 은유 관계에 있는 실제 객체의 이름을 소프트웨어 객체의 이름으로 사용하면 표현적 차이를 줄여 소프트웨어의 구조를 쉽게 예측할 수 있죠.

따라서 소프트웨어 객체에 대한 현실 객체의 은유를 효과적으로 사용할 경우 표현적 차이를 줄일 수 있으며, 이해하기 쉽고 유지보수가 용이한 소프트웨어를 만들 수 있는 것입니다.

바로 이러한 이유로 모든 객체지향 지침서에서는 현실 세계인 도메인에서 사용되는 이름을 객체에게 부여하라고 가이드하는 것이죠.

이러한 개념들이 너무 낯설거나 당장의 프로그래밍에 불필요하게 느껴질 수 있습니다. 사실 위의 내용을 굳이 외우거나 애써 기억할 필요는 없습니다. 아래의 이야기만 마음에 담아두시고 앞으로 객체지향 설계를 할 때 기억하시기만 하면 됩니다.

> 객체지향 설계자로서 우리의 목적은 현실을 모방하는 것이 아니다. 단지 새로운 세계를 창조하기만 하면 된다. 현실을 닮아야 한다는 어떤 제약이나 구속도 없다. 여러분이 창조한 객체의 특성을 상기시킬 수 있다면 현실 속의 객체의 이름을 이용해 객체를 묘사하라. 그렇지 않다면 깔끔하게 현실을 무시하고 자유롭게 여러분만의 새로운 세계를 창조하기 바란다.
> - 객체지향의 사실과 오해 p.71

객체지향 프로그래밍(OOP) 주요 특징

1. 추상화 (Abstraction)

   추상화는 앞선 메뉴얼에서 언급한 추상화의 개념이 그대로 적용된 OOP의 특징입니다. 객체들의 공통의 속성이나 기능을 묶어 정의하는 것이죠. 그 과정에서 불필한 부분은 숨기고 공통적인 부분만을 정리함으로써 객체지향 모델링을 단순화하고 코드의 재사용성을 높일 수 있습니다.

2. 캡슐화 (Encapsulation)

   객체에서 데이터 구조와 데이터, 즉 필드 변수와 메서드를 하나로 묶는 것입니다. 이런 설명은 애매할 수 있어 흔히 데이터 은닉화를 떠올리게 됩니다. 이를 조금 더 풀어 써본다면, 객체가 맡은 역할을 수행하기 위한 하나의 목적을 하나로 묶는 특징입니다. 이를 통해 데이터를 직접적으로 외부에 노출하지 않고 오로지 메서드(행위)를 통해서만 접근 가능하도록 구현하는 것이죠.

3. 상속 (Inheritance)

   상속은 객체지향의 매력이 아주 잘 느껴지는 부분이죠. 상속을 통해 코드의 재사용성과 유지 보수를 굉장히 향상 시킬 수 있는데 이는 바로 다음 설명하는 다형성을 빼놓고 설명할 수 없습니다. 상속과 다형성을 통해 객체지향 프로그래밍은 프로그램 설계에 엄청난 강점을 발휘할 수 있습니다.

4. 다형성 (Polymorphism)

   정의대로면, 부모클레스에서 물려받은 가상 함수를 자식 클래스 내에서 오버라이딩 되어 사용되는 것입니다. 이를 통해 어떤 변수, 메서드가 실행 시간(Runtime)에서 동적으로 다른 결과를 낼 수 있게 만들어 줍니다.