JAVA Term Project

상위 설계서

지금까지 컴퓨터 프로그래밍을 해오면서 어떠한 프로젝트를 직접 설계하고 구현하는 경험은 이번이 처음이었다. 따라서, 무엇부터 시작해야 할지 매우 막막하였다. 교수님께서 자바 텀 프로젝트를 주셨을 시기에 마침 “자기주도 진로설계” 시간이나 “자바 프로그래밍(이론)” 시간에 교수님들께서 항상 공통적으로 하시는 말씀이 있었다. 바로 “설계”였다. 모든 소프트웨어 프로그램들 뿐 아니라 어떠한 서비스, 제품을 완성도 있게 만들기 위해서는 그만큼 완성도 있고 구체적인 설계가 필요하다. 하지만 지금까지 수업시간에 배우거나 연구실에서 배우는 과제들을 할 때에는 그만큼 규모가 작은 코딩이기 때문에 어떠한 계획을 수립하거나, 구상을 미리 하지 않고 시작부터 코드를 짜는 경우가 대다수였다. 코드를 짜다가 논리 오류가 발생하면 그제서야 노트를 꺼내는 습관이 뿌리깊게 박혀 있었다. 교수님들의 말씀을 듣고 이 습관을 고쳐야겠다라고 다짐하게 되었고, 자바 텀 프로젝트를 그 계기로 삼아 프로젝트 계획, 설계의 방법을 배우고 직접 해보기 위하여 로버트 C.마틴의 **『**클린 아키텍처, 소프트웨어 구조와 설계의 원칙**』**을 발췌 독 하였다.

로버트 C.마틴의 **『**클린 아키텍처, 소프트웨어 구조와 설계의 원칙**』** 에서는 기본적으로 아키텍처란 무엇인지, 아키텍처 설계의 필요성과 목적, 이의 방법에 대하여 상세하게 설명한다. 올바른 소프트웨어 아키텍처의 설계는 유지보수에 용이하고 가독성을 높이며, 프로그램 구조의 수정을 간단하게 만든다. 즉, 올바른 설계를 토대로 작성된 프로그래밍은 코드를 갈아엎는 일이 발생하지 않는다. 이 책에 설명되어 있는 설계 원칙을 요약하면 다음과 같다.

1. SRP(Single Responsibility Principle): 단일 책임 원칙

단일 책임 원칙은 클래스의 메서드가 한가지 역할만 수행해야 함을 의미한다고 볼 수 있다. 이를 효율적으로 구현하기 위하여 나온 것이 바로 퍼사드Facade 패턴 이라고 한다.

2. OCP(Open-Closed Principle): 개방-페쇄 원칙

3. LSP(Liskov Substitution Principle): 리스코프 치환 원칙

4. ISP(Interface Segregation Principle): 인터페이스 분리 원칙

5. DIP(Dependency Inversion Principle): 의존성 역전 원리

클래스 다이어그램을 통한 클래스 설계

클래스 다이어그램이란 클래스 내부의 구성요소를 표기하고, 클래스 간의 관계를 도식화 함으로써 프로그램의 구성을 표시할 수 있는 단순화 방법이다. 클래스 다이어그램을 표기하는 방법중 UML은 표준화된 표기 방법이다.

//UML 기호 정리

//문제조건 분석

텍스트, 도표, 평면도, 개략도이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.