내용정리

책내용이 ..

동부 주립대학(ESU)의 기존 수강신청 시스템은 교수의 강의 배정과 학생 수강신청 과정에서 과도한 수작업으로 인해 비효율적이다. 학생이 제출한 종이 신청서를 등록처 직원들이 수작업으로 입력하고, 일괄(batch)처리로 수업을 배정하기 때문에 시간과 노력이 많이 든다. 또한 데이터 관리 및 접근 효율성이 주요 위험 요소로 지적되어, 이를 해결하기 위해 데이터베이스와 하드웨어 성능을 평가하는 여러 프로토타입을 개발하여 온라인 수강신청 시스템으로의 전환을 준비 중이다.

너무 짧아서 해결과정을 객체지향적 관점에서 바라봤을때의 해결 가능한 점을 적었습니다 !

📌 ESU 수강신청 시스템과 객체지향의 관계

ESU 대학의 수강신청 시스템 사례는 객체지향 분석(Object-Oriented Analysis, OOA) 및 설계(Object-Oriented Design, OOD)를 실제로 적용하기 좋은 대표적인 사례

객체지향 패러다임은 현실 세계를 객체(Object)라는 독립적인 요소들로 나누어 분석하고 설계하는 방법으로, ESU 사례와 같이 복잡한 업무 프로세스를 명확하게 구조화하고, 유지보수와 확장이 용이한 시스템을 구축할 수 있도록 돕습니다.

🔍 1. 객체지향적 접근의 필요성

기존 ESU 시스템의 문제점:

- 과도한 수작업 (종이 서류 작성, 데이터 입력 반복)
- 데이터 관리 및 접근 비효율성
- 충돌 발생 시 별도의 수작업 추가로 업무 증가
- 시스템 유지보수 어려움

이러한 문제점을 객체지향적으로 접근하면 효율적으로 해결할 수 있습니다.

객체지향 패러다임이 주는 이점:

- 현실 세계의 개념을 객체로 직접 매핑하여 이해하기 쉬운 시스템 구축

- 반복되는 작업을 객체의 메소드로 추상화하여 자동화 가능
- 유지보수 및 확장 용이

📦 2. ESU 시스템의 객체지향적 분석(Object-Oriented Analysis)

ESU 수강신청 시스템을 분석하면 다음과 같은 객체(Object)와 클래스(Class)를 추출할 수 있습니다.

클래스(Class)	객체(Object)의 예시	설명
Student	학생 (이름, 학번, 학과 등)	수강신청서를 작성하여 과목을 신청하는 주체
Professor	교수 (이름, 교수번호, 전공 등)	자신이 담당할 과목을 결정하고 학생을 지도하는 주체
Course	과목 (과목명, 과목코드 등)	학생들이 등록하고 교수들이 담당하는 강의 정보
Registrar	등록처 (관리직원)	학생의 수강신청 관리, 입력, 배정 관리 주체
RegistrationForm	수강신청서 (신청과목 리스트)	학생이 작성하여 등록처에 제출하는 서류 객체
Schedule	수업 시간표	학생과 교수에게 배부되는 수업 배정 정보 객체

각 객체는 **속성(attribute)**과 **행위(behavior, method)**를 가집니다.

- 예시:
  - **Student** 클래스의 행위(method): `fillRegistrationForm()`, `submitForm()`
  - **Registrar** 클래스의 행위(method): `enterStudentData()`, `runBatchRegistration()`, `resolveConflicts()`

이러한 분석은 **현실 세계와 시스템 간의 간격을 줄이고, 업무 프로세스를 명확하게 모델링**할 수 있도록 돕습니다.

🛠️ 3. ESU 시스템의 객체지향적 설계(Object-Oriented Design)

객체지향 분석을 통해 얻은 클래스 및 객체를 바탕으로, 객체지향 설계를 진행합니다.

✅ 주요 객체 간 관계 설정 (객체지향 설계의 핵심 개념)

- **상속(Inheritance)**

    - 교수(Professor)와 학생(Student)은 모두 "Person" 클래스를 상속받아 공통적인 속성(이름, 주소 등)을 공유할 수 있음.
- **연관(Association)**
    - 학생(Student)은 수강신청서(RegistrationForm)를 작성한다.
    - 교수(Professor)는 강의(Course)를 담당한다.
- **집합(Aggregation)**
    - 등록처(Registrar)는 여러 수강신청서(RegistrationForm)를 모아서 관리하는 역할 수행.
- **의존성(Dependency)**
    - 학생(Student)은 등록처(Registrar)에 수강신청을 제출할 때 일시적으로 Registrar 객체에 의존함.

✅ 설계 시 중요한 객체지향 원칙 적용 예시

- **캡슐화(Encapsulation)**
    - 각 클래스는 데이터를 보호하고 메소드를 통해서만 접근 가능하도록 설계.
    - 예시: 학생의 개인정보는 학생 클래스 내에서만 관리.
- **다형성(Polymorphism)**
    - 수업 배정 알고리즘을 여러 방식으로 구현할 수 있으며, 시스템 요구에 따라 유연하게 변경 가능.
    - 예시: `assignCourses()` 메소드를 다양한 배정 알고리즘으로 오버라이딩 가능.

🔧 4. 프로토타입 개발과 객체지향 패러다임 활용

ESU 개발팀이 수행한 프로토타입(prototype) 개발 또한 객체지향 패러다임과 밀접한 관련이 있습니다.

- 프로토타입을 통해 **데이터베이스 객체(DB객체)**, **시스템 객체**, **사용자 인터페이스(UI) 객체** 등을 설계하고 테스트
- 객체지향 설계를 통해 DB 접근 및 관리 효율성을 평가하여 리스크를 효과적으로 관리 가능

🌐 5. 객체지향 패러다임 적용 후 기대 효과

- 반복적이고 복잡한 업무를 객체지향 방식으로 자동화하여 효율성 증가
- 시스템의 유연성 증가로 인해 변화하는 환경에 대응이 쉬워짐
- 유지보수 비용 절감 및 시스템 확장 용이성 확보
- 실시간 데이터 처리 및 빠른 응답성 제공 가능
