스프링 핵심 원리 이해1 - 예제 만들기

💡 **인텔리제이 단축키**

ctrl + d : 한 줄 복사

ctrl + alt + l : 코드 정리

ctrl + shift + enter : 자동완성

alt + insert : 생성자, getter setter, 클래스 생성 등

sout : sysout 대신

enum : 열거형 타입

: 값들을 이름으로 묶어 표현할 때 사용되는 자료형

```
public class Main {
    // 1. Enum 정의
    public enum Grade {
        FIRST,    // 1학년
        SECOND,   // 2학년
        THIRD,    // 3학년
        FOURTH    // 4학년
    }

    public static void main(String[] args) {
        // 2. Enum 사용
```

```
        Grade myGrade = Grade.THIRD; // 3학년

        // 3. Enum 값 확인
        if (myGrade == Grade.THIRD) {
            System.out.println("현재 학년은 3학년입니다!");
        }

        // 4. 모든 Enum 값 순회
        for (Grade grade : Grade.values()) {
            System.out.println("학년: " + grade);
        }
    }
}
```

```
현재 학년은 3학년입니다!
학년: FIRST
학년: SECOND
학년: THIRD
학년: FOURTH
```

- 장점 (by. GPT)
  - **코드 가독성**: `Grade.SECOND`처럼 이름이 있으니까, 숫자로 표현하는 것보다 알아보기 쉬워.
  - **안전성**: `Grade`에 정의된 값만 사용할 수 있어서 실수를 줄일 수 있어. 예를 들어, `Grade.FIFTH` 같은 값은 정의되지 않았으니까 컴파일 에러가 나!
  - **유지보수 용이**: 새로운 값(예: `FIFTH`)을 추가하거나 수정할 때, 관련된 모든 코드가 깔끔하게 관리돼.

Test를 사용하는 이유?

회원 도메인 설계의 문제점

- 다른 저장소로 변경할 때 OCP 원칙을 잘 준수하는지?
- DIP는 잘 지키고 있는지?

→ 의존관계가 인터페이스 뿐만 아니라 구현까지 모두 의존하는 문제점이 있음.