SPRING BOOT & JPA 활용

▼ 어노테이션들!

- @Entity : JPA를 사용해서 테이블과 매핑할 클래스에 붙여주는 어노테이션
  - 엔티티 안에서 사용할 수 있는 추가 기능

    `@Id` : 기본키 매핑

    `@GeneratedValue` : 기본키에 값을 자동 생성

    `@Column(name = "DB의 열이름")` : 엔티티의 필드를 DB의 열에 매핑

`@Embedded` : 다른 클래스(`@Embeddable`을 사용해야 함)를 임베드 (캡슐화 느낌..?)

`@Enumerated(EnumType.` *`STRING`* `)` : enum을 DB에 저장할 수 있게 함

→ `EnumType.` *`STRING`*과 `EnumType.ORDINAL` 이 존재

→ string은 문자열로 저장되어 여러값 도출 가능

→ ordinal은 0,1로 들어가 두가지 결과만 도출 가능

`@OneToMany(mappedBy = "객체명")` : 1인 쪽에 붙으며 owner가 아님을 보여줌

`@ManyToOne` : Owner (Many인 쪽 / 외래키를 갖고 있는 부분) 쪽에 붙음

```
//Member class 내
@OneToMany(mappedBy = "member")
private List<Order> orders = new ArrayList<>();

//Order class 내
@ManyToOne
```

```
@JoinColumn(name = "member_id")
private Member member;
```

`@JoinColumn(name = "저장할 컬럼명")` : name 속성은 말 그대로 Order 엔티티에 존재하는 member 라는 필드를 어떤 이름으로 Order 테이블에 컬럼명으로 설정할 것인지를 나타내주는 것 → Order 테이블에 member_id 라는 컬럼으로 member 필드가 들어감

- `@Table(name= "테이블명")` : 엔티티와 DB의 table을 매핑
- `@Inheritance(strategy = InheritanceType.SINGLE_TABLE )` : 상속관계의 클래스들 구조 지정

  → `InheritanceType.SINGLE_TABLE` : 부모 클래스에 모든 자식 클래스의 속성들을 몰빵

  → `InheritanceType.JOINED` : 자식클래스에 부모 클래스의 기본키 배정
- `@DiscriminatorColumn(name = "dtype")` : 부모 클래스에서 사용하고 dtype이라는 컬럼을 생성해 상속받는 클래스들의 키를 저장
- `@DiscriminatorValue("A")` : 자식 클래스에서 사용하고 dtype에 들어가는 값으로 자식 클래스들을 구별하는 데에 쓰임