**프로젝트 제안: 온라인 쇼핑몰 시스템**

**프로젝트 개요**

콘솔 기반의 온라인 쇼핑몰 시스템으로, 상품 조회부터 장바구니 추가, 주문 처리까지의 핵심 기능을 구현합니다. 자바의 객체 지향 개념을 다양하게 활용할 수 있는 구조입니다.

**주요 기능 및 필요 개념**

1. **상품 관리** (Object 클래스, 클래스 설계)
   * 상품 클래스 설계: 상품ID, 이름, 가격, 재고량, 카테고리 등 속성 정의
   * toString() 오버라이딩: 상품 정보 출력
   * equals(), hashCode() 오버라이딩: 상품ID 기반 식별
2. **상품 카테고리 관리** (Enum 활용)
   * 카테고리 정의: 전자제품, 의류, 식품, 도서 등
   * Enum 타입으로 카테고리 관리 및 카테고리별 할인율 설정
3. **사용자 관리** (상속, 다형성)
   * 기본 User 클래스 설계
   * 일반회원/VIP회원으로 상속 구현
   * 회원 등급에 따른 할인율, 포인트 적립률 차별화
4. **장바구니 기능** (컬렉션 프레임워크)
   * 장바구니에 상품 추가/삭제/수량 변경
   * 장바구니 내 상품 목록 조회
   * 총 금액 계산 (할인 적용)
5. **주문 및 결제 처리** (날짜와 시간 API)
   * 주문 생성, 주문 날짜 기록
   * 결제 방식 선택 (신용카드, 계좌이체, 포인트 사용 등)
   * 영수증 생성
6. **상품 검색 및 필터링** (String 클래스, 컬렉션)
   * 이름, 카테고리 등으로 상품 검색
   * 가격 범위로 필터링
   * 정렬 기능 (가격 오름차순/내림차순)
7. **리뷰 및 평점 시스템** (중첩 클래스)
   * 구매 후 리뷰 작성 기능
   * 평점 계산 및 상품별 평균 평점 표시
8. **예외 처리** (예외 처리 개념)
   * 재고 부족 시 예외 처리
   * 유효하지 않은 결제 정보 처리
   * 입력값 검증 및 예외 처리

**구현 단계별 가이드**

1. **1단계: 기본 클래스 설계**
   * Product 클래스 및 하위 클래스 구현
   * Category Enum 구현
   * User 클래스 및 하위 클래스 구현
   * Cart, CartItem 클래스 구현
2. **2단계: 상품 관리 시스템 구현**
   * ProductService 클래스 구현 (싱글톤 패턴 적용)
   * 상품 등록, 수정, 삭제, 조회 기능
3. **3단계: 사용자 관리 시스템 구현**
   * UserService 클래스 구현
   * 사용자 등록, 로그인, 정보 수정 기능
4. **4단계: 장바구니 시스템 구현**
   * CartService 클래스 구현
   * 장바구니 관리 기능 구현
5. **5단계: 주문 및 결제 시스템 구현**
   * Order 및 Payment 클래스 구현
   * 주문 처리 및 결제 로직 구현
6. **6단계: 검색 및 필터링 기능 구현**
   * 다양한 조건의 상품 검색 및 필터링 구현
7. **7단계: 리뷰 시스템 구현**
   * Review 클래스 구현
   * 리뷰 등록 및 조회 기능
8. **8단계: 콘솔 UI 구현**
   * 사용자 인터페이스 구현
   * 메뉴 시스템 구현

**클래스 구조 예시**

com.shop

├── model

│ ├── product

│ │ ├── Product.java

│ │ ├── ElectronicProduct.java

│ │ ├── ClothingProduct.java

│ │ ├── Category.java (Enum)

│ │ └── Review.java

│ ├── user

│ │ ├── User.java

│ │ ├── RegularUser.java

│ │ └── VIPUser.java

│ ├── cart

│ │ ├── Cart.java

│ │ └── CartItem.java

│ └── order

│ ├── Order.java

│ ├── OrderItem.java

│ └── Payment.java

├── service

│ ├── ProductService.java

│ ├── UserService.java

│ ├── CartService.java

│ └── OrderService.java

├── exception

│ ├── OutOfStockException.java

│ ├── InvalidPaymentException.java

│ └── ProductNotFoundException.java

└── ui

└── ShopConsoleUI.java

**학습한 개념별 활용 포인트**

1. **Object 클래스**
   * Product, User 클래스에서 toString(), equals(), hashCode() 오버라이딩
   * Object 타입으로 다양한 객체 다루기
2. **상속과 다형성**
   * User 클래스 상속 구조 (RegularUser, VIPUser)
   * Product 클래스 상속 구조 (ElectronicProduct, ClothingProduct 등)
   * 다형성을 활용한 제품별, 회원별 차별화된 서비스 제공
3. **String 클래스**
   * 상품 검색 시 contains(), startsWith() 등 활용
   * 문자열 처리 및 비교
4. **클래스 설계**
   * 캡슐화를 통한 데이터 은닉
   * getter/setter 메서드 구현
   * 생성자 오버로딩
5. **Enum 활용**
   * 상품 카테고리 정의 및 관리
   * 결제 방식 유형 정의
6. **날짜와 시간**
   * LocalDateTime을 활용한 주문 시간, 배송 예정일 계산
   * 주문 내역의 시간순 정렬
7. **예외 처리**
   * 재고 부족, 결제 실패 등의 상황에 대한 사용자 정의 예외
   * try-catch 블록으로 예외 상황 처리
8. **컬렉션 프레임워크**
   * ArrayList: 상품 목록, 장바구니 아이템 관리
   * HashMap: 상품ID로 빠른 검색, 사용자ID로 사용자 관리
   * TreeMap: 가격별 상품 정렬
9. **중첩 클래스**
   * 리뷰 시스템에 내부 클래스 활용
   * 결제 시스템에 내부 클래스 활용

**핵심 클래스 구현 예시**

**Product 클래스 (기본)**

public class Product {

private String id;

private String name;

private double price;

private int stockQuantity;

private Category category;

private List<Review> reviews;

// 생성자, getter, setter

// Object 클래스 메서드 오버라이딩

@Override

public String toString() {

return String.format("[%s] %s - ₩%.2f (재고: %d개)",

id, name, price, stockQuantity);

}

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (obj == null || getClass() != obj.getClass()) return false;

Product product = (Product) obj;

return id.equals(product.id);

}

@Override

public int hashCode() {

return Objects.hash(id);

}

// 기능 메서드

public boolean hasStock(int quantity) {

return stockQuantity >= quantity;

}

public void reduceStock(int quantity) throws OutOfStockException {

if (!hasStock(quantity)) {

throw new OutOfStockException("재고가 부족합니다: " + name);

}

stockQuantity -= quantity;

}

public double getAverageRating() {

if (reviews.isEmpty()) return 0;

return reviews.stream()

.mapToDouble(Review::getRating)

.average()

.orElse(0);

}

}

**Cart 클래스**

public class Cart {

private String userId;

private Map<String, CartItem> items; // 상품ID를 키로 사용

public Cart(String userId) {

this.userId = userId;

this.items = new HashMap<>();

}

public void addItem(Product product, int quantity) {

String productId = product.getId();

if (items.containsKey(productId)) {

// 이미 장바구니에 있는 상품이면 수량만 증가

CartItem existingItem = items.get(productId);

existingItem.increaseQuantity(quantity);

} else {

// 새 상품이면 CartItem 생성

items.put(productId, new CartItem(product, quantity));

}

}

public void removeItem(String productId) {

items.remove(productId);

}

public void updateQuantity(String productId, int newQuantity) {

if (items.containsKey(productId)) {

if (newQuantity <= 0) {

removeItem(productId);

} else {

items.get(productId).setQuantity(newQuantity);

}

}

}

public double getTotalPrice() {

return items.values().stream()

.mapToDouble(CartItem::getSubtotal)

.sum();

}

public List<CartItem> getItems() {

return new ArrayList<>(items.values());

}

public boolean isEmpty() {

return items.isEmpty();

}

public void clear() {

items.clear();

}

}