**온라인 쇼핑몰 시스템 구현 가이드**

**ArrayList 개요 및 사용법**

ArrayList는 자바의 컬렉션 프레임워크 중 하나로, 크기가 동적으로 변하는 배열입니다. 기본 배열과 달리 크기를 미리 지정하지 않아도 되며, 요소 추가/삭제가 용이합니다.

**주요 메서드와 사용 상황:**

* add(E e): 요소 추가 - 상품 목록에 새 상품 추가할 때
* get(int index): 특정 위치의 요소 가져오기 - 특정 상품 정보 조회할 때
* remove(int index) 또는 remove(Object o): 요소 삭제 - 상품 목록에서 상품 제거할 때
* size(): 요소 개수 반환 - 장바구니에 있는 상품 개수 확인할 때
* contains(Object o): 요소 포함 여부 확인 - 이미 장바구니에 있는 상품인지 확인할 때
* isEmpty(): 비어있는지 확인 - 장바구니가 비어있는지 확인할 때
* clear(): 모든 요소 삭제 - 장바구니 비우기 기능 구현할 때

**클래스별 구현 가이드**

**1. Product 클래스**

/\*

\* 상품 정보를 나타내는 클래스

\*

\* 구현 방법:

\* 1. 필요한 멤버 변수(상품ID, 이름, 가격, 재고량, 카테고리)를 private으로 선언

\* 2. 생성자, getter, setter 메서드 구현

\* 3. toString() 메서드 오버라이딩: 상품 정보 출력 형식 지정

\* 4. equals(), hashCode() 메서드 오버라이딩: 상품ID 기반으로 동일성 판단

\* 5. 상품 재고 확인/감소 기능 메서드 구현

\*/

**2. Category Enum**

/\*

\* 상품 카테고리를 정의하는 열거형

\*

\* 구현 방법:

\* 1. 카테고리 값 정의 (ELECTRONICS, CLOTHING, FOOD, BOOKS 등)

\* 2. 각 카테고리별 할인율 필드 추가

\* 3. 생성자와 getter 메서드 구현

\* 4. 할인율을 적용한 가격 계산 메서드 구현

\*/

**3. User 클래스 (상위 클래스)**

/\*

\* 사용자 정보의 기본 클래스

\*

\* 구현 방법:

\* 1. 멤버 변수(ID, 이름, 이메일, 비밀번호, 주소) private으로 선언

\* 2. 생성자, getter, setter 메서드 구현

\* 3. toString() 메서드 오버라이딩: 사용자 정보 출력

\* 4. equals(), hashCode() 메서드 오버라이딩: 사용자ID 기반으로 동일성 판단

\* 5. 할인율 계산 메서드 구현 (하위 클래스에서 오버라이딩)

\* 6. 포인트 적립 메서드 구현 (하위 클래스에서 오버라이딩)

\*/

**4. RegularUser 클래스 (User 하위 클래스)**

/\*

\* 일반 회원 클래스

\*

\* 구현 방법:

\* 1. User 클래스 상속

\* 2. 일반 회원 특성(적립률, 할인율)을 상수로 정의

\* 3. 생성자 구현 (super() 활용)

\* 4. 할인율 계산 메서드 오버라이딩

\* 5. 포인트 적립 메서드 오버라이딩

\*/

**5. VIPUser 클래스 (User 하위 클래스)**

/\*

\* VIP 회원 클래스

\*

\* 구현 방법:

\* 1. User 클래스 상속

\* 2. VIP 회원 특성(높은 적립률, 높은 할인율, VIP 혜택)을 상수로 정의

\* 3. 생성자 구현 (super() 활용)

\* 4. 할인율 계산 메서드 오버라이딩 (더 높은 할인율 적용)

\* 5. 포인트 적립 메서드 오버라이딩 (더 높은 적립률 적용)

\* 6. VIP 전용 혜택 메서드 추가

\*/

**6. CartItem 클래스**

/\*

\* 장바구니 내 개별 상품 항목을 나타내는 클래스

\*

\* 구현 방법:

\* 1. 멤버 변수(상품, 수량) private으로 선언

\* 2. 생성자, getter, setter 메서드 구현

\* 3. 수량 증가/감소 메서드 구현

\* 4. 소계(상품가격 × 수량) 계산 메서드 구현

\* 5. toString() 메서드 오버라이딩: 장바구니 항목 정보 출력

\*/

**7. Cart 클래스**

/\*

\* 사용자의 장바구니를 나타내는 클래스

\*

\* 구현 방법:

\* 1. 멤버 변수(사용자ID, 상품 목록) 선언

\* - 상품 목록은 ArrayList<CartItem>으로 구현

\* 2. 생성자 구현 (빈 ArrayList 초기화)

\* 3. 상품 추가 메서드 구현 (이미 있는 상품이면 수량만 증가)

\* 4. 상품 제거 메서드 구현

\* 5. 상품 수량 변경 메서드 구현

\* 6. 장바구니 비우기 메서드 구현

\* 7. 총액 계산 메서드 구현

\* 8. 장바구니 내용 출력 메서드 구현

\*/

25.03.11 2차 여기부터 진행하기

**8. Order 클래스**

/\*

\* 주문 정보를 나타내는 클래스

\*

\* 구현 방법:

\* 1. 멤버 변수(주문ID, 사용자ID, 주문 항목 목록, 주문일시, 총액, 상태) 선언

\* - 주문 항목 목록은 ArrayList<OrderItem>으로 구현

\* 2. 생성자 구현 (현재 시간으로 주문일시 초기화)

\* 3. 주문 상태 Enum 내부 정의 (PENDING, PAID, SHIPPED, DELIVERED, CANCELLED)

\* 4. 주문 항목 추가 메서드 구현

\* 5. 총액 계산 메서드 구현

\* 6. 주문 상태 변경 메서드 구현

\* 7. 주문 취소 메서드 구현

\* 8. 영수증 생성 메서드 구현

\* 9. toString() 메서드 오버라이딩: 주문 정보 출력

\*/

**9. OrderItem 클래스**

/\*

\* 주문 내 개별 상품 항목을 나타내는 클래스

\*

\* 구현 방법:

\* 1. 멤버 변수(상품, 수량, 구매 당시 가격) 선언

\* 2. 생성자, getter 메서드 구현

\* 3. 소계 계산 메서드 구현

\* 4. toString() 메서드 오버라이딩: 주문 항목 정보 출력

\*/

**10. Review 클래스**

/\*

\* 상품 리뷰를 나타내는 클래스

\*

\* 구현 방법:

\* 1. 멤버 변수(리뷰ID, 사용자ID, 상품ID, 내용, 평점, 리뷰 날짜) 선언

\* 2. 생성자, getter, setter 메서드 구현

\* 3. 평점 검증 메서드 구현 (1~5 사이 값만 허용)

\* 4. toString() 메서드 오버라이딩: 리뷰 정보 출력

\*/

**11. ProductService 클래스**

/\*

\* 상품 관련 비즈니스 로직을 처리하는 서비스 클래스

\*

\* 구현 방법:

\* 1. 싱글톤 패턴 적용 (인스턴스를 하나만 생성하여 공유)

\* 2. 상품 목록을 ArrayList<Product>로 관리

\* 3. 상품 추가 메서드 구현

\* 4. 상품 삭제 메서드 구현

\* 5. 상품 업데이트 메서드 구현

\* 6. ID로 상품 검색 메서드 구현

\* 7. 이름으로 상품 검색 메서드 구현 (부분 일치 검색)

\* 8. 카테고리별 상품 필터링 메서드 구현

\* 9. 가격 범위로 상품 필터링 메서드 구현

\* 10. 상품 정렬 메서드 구현 (가격 오름차순/내림차순)

\*/

**12. UserService 클래스**

/\*

\* 사용자 관련 비즈니스 로직을 처리하는 서비스 클래스

\*

\* 구현 방법:

\* 1. 싱글톤 패턴 적용

\* 2. 사용자 목록을 ArrayList<User>로 관리

\* 3. 사용자 등록 메서드 구현

\* 4. 사용자 정보 업데이트 메서드 구현

\* 5. 사용자 삭제 메서드 구현

\* 6. 로그인 검증 메서드 구현

\* 7. ID로 사용자 검색 메서드 구현

\* 8. 회원 등급 업그레이드 메서드 구현

\*/

**13. CartService 클래스**

/\*

\* 장바구니 관련 비즈니스 로직을 처리하는 서비스 클래스

\*

\* 구현 방법:

\* 1. 싱글톤 패턴 적용

\* 2. 사용자별 장바구니를 ArrayList<Cart>로 관리

\* 3. 장바구니 생성/조회 메서드 구현

\* 4. 상품 추가 메서드 구현

\* 5. 상품 제거 메서드 구현

\* 6. 상품 수량 변경 메서드 구현

\* 7. 장바구니 비우기 메서드 구현

\* 8. 장바구니 항목 조회 메서드 구현

\* 9. 총액 계산 메서드 구현 (회원 할인 적용)

\*/

**14. OrderService 클래스**

/\*

\* 주문 관련 비즈니스 로직을 처리하는 서비스 클래스

\*

\* 구현 방법:

\* 1. 싱글톤 패턴 적용

\* 2. 주문 목록을 ArrayList<Order>로 관리

\* 3. 새로운 주문 생성 메서드 구현

\* 4. 장바구니 항목을 주문으로 변환 메서드 구현

\* 5. 주문 상태 업데이트 메서드 구현

\* 6. 주문 취소 메서드 구현

\* 7. 사용자별 주문 내역 조회 메서드 구현

\* 8. 주문ID로 주문 검색 메서드 구현

\* 9. 결제 처리 메서드 구현

\* 10. 영수증 생성 메서드 구현

\*/

**15. ShopConsoleUI 클래스**

/\*

\* 콘솔 기반 사용자 인터페이스 클래스

\*

\* 구현 방법:

\* 1. Scanner 객체로 사용자 입력 처리

\* 2. 각 서비스 클래스의 인스턴스 참조

\* 3. 메인 메뉴 표시 및 처리 메서드 구현

\* 4. 상품 목록 표시 메서드 구현

\* 5. 상품 상세 정보 표시 메서드 구현

\* 6. 장바구니 표시 및 관리 메서드 구현

\* 7. 주문 과정 안내 메서드 구현

\* 8. 결제 처리 메서드 구현

\* 9. 사용자 로그인/등록 메서드 구현

\* 10. 입력값 검증 및 예외 처리 메서드 구현

\*/

**16. 사용자 정의 예외 클래스**

/\*

\* 재고 부족 예외 클래스

\*

\* 구현 방법:

\* 1. Exception 클래스 상속

\* 2. 생성자 구현 (오류 메시지 전달)

\*/

/\*

\* 유효하지 않은 결제 예외 클래스

\*

\* 구현 방법:

\* 1. Exception 클래스 상속

\* 2. 생성자 구현 (오류 메시지 전달)

\*/

**ArrayList 활용 예시 상황**

1. **상품 목록 관리**

// ProductService 클래스 내부

private ArrayList<Product> products = new ArrayList<>();

// 상품 추가

public void addProduct(Product product) {

products.add(product);

}

// ID로 상품 찾기

public Product findProductById(String id) {

for (Product product : products) {

if (product.getId().equals(id)) {

return product;

}

}

return null;

}

// 가격으로 상품 필터링

public ArrayList<Product> filterByPriceRange(double min, double max) {

ArrayList<Product> filtered = new ArrayList<>();

for (Product product : products) {

double price = product.getPrice();

if (price >= min && price <= max) {

filtered.add(product);

}

}

return filtered;

}

1. **장바구니 관리**

// Cart 클래스 내부

private ArrayList<CartItem> items = new ArrayList<>();

// 상품 추가

public void addItem(Product product, int quantity) {

// 이미 있는 상품인지 확인

for (CartItem item : items) {

if (item.getProduct().getId().equals(product.getId())) {

item.increaseQuantity(quantity);

return;

}

}

// 새 상품이면 추가

items.add(new CartItem(product, quantity));

}

// 상품 제거

public void removeItem(String productId) {

for (int i = 0; i < items.size(); i++) {

if (items.get(i).getProduct().getId().equals(productId)) {

items.remove(i);

return;

}

}

}

이 가이드를 바탕으로 쇼핑몰 시스템을 구현하시면 됩니다. 클래스를 하나씩 구현하면서 자바의 객체 지향 개념들을 실습해보세요. 특히 ArrayList의 사용법을 익히는 데 중점을 두시면 좋겠습니다.