Homework 01: 24시간 학교 편의점 스마트 재고 관리 시스템

📖 프로젝트 배경 스토리

🎓 주인공: 탄지로 (컴퓨터소프트웨어공학과 3학년)

탄지로는 일본에서 한국으로 유학을 온 컴퓨터소프트웨어공학과 3학년 학생입니다. 한국어를 매우 유창하게 구사해서 이름을 묻지 않으면 일본인인지 모를 정도이지만, 얼굴에 태어날 때부터 작은 붉은 반점이 있는 것이 특징입니다. 한국에서의 유학 생활비를 마련하기 위해 학교 내 24시간 편의점에서 아르바이트를 시작했습니다. 하지만 첫 주부터 여러 문제점을 발견했습니다.

첫 번째 문제 발견

> "어? 새우깡이 벌써 떨어졌네? 어제 확인했을 때는 충분했는데..."

탄지로가 근무하던 어느 날, 인기 상품인 새우깡이 갑자기 품절되었습니다. 사장님께 문의해보니 **수기로 재고를 관리**하다 보니 정확한 재고량을 파악하기 어렵다고 했습니다.

두 번째 문제 발견

> "이 도시락들 유통기한이 내일까지네... 이걸 언제 다 팔지?"

다음 날에는 냉장고 안에서 **유통기한이 임박한 도시락 15개**를 발견했습니다. 이런 상품들은 할인해서 빨리 팔아야 하는데, 언제 할인해야 하는지, 얼마나 할인해야 하는지에 대한 체계적인 기준이 없었습니다.

세 번째 문제 발견

> "사장님, 이번 주에 뭐가 제일 많이 팔렸나요?" "음... 글쎄, 새우깡이랑 콜라가 많이 나간 것 같기는 한데... 정확히는 잘 모르겠네."

매출 분석이 전혀 되지 않고 있었습니다. 어떤 상품이 인기가 있는지, 언제 발주를 해야 하는지, 얼마나 주문해야 하는지에 대한 데이터가 전혀 없었습니다. 모든 것이 감 과 경험 에 의존하고 있었습니다.

탄지로의 결심

> "내가 지금까지 배운 Kotlin 지식으로 이 문제들을 해결할 수 있지 않을까? 일본에서도 편의점을 많이 봤는데, 한국도 비슷한 문제가 있구나."

탄지로는 수업시간에 배운 **객체지향 프로그래밍**과 **함수형 프로그래밍** 개념들을 활용해서 **스마트 재고 관리 시스템**을 만들기로 결심했습니다.

🎯 프로젝트 발전 과정

1단계: 기본 시스템 구축 📦

탄지로가 가장 먼저 해결하고자 한 것은 **체계적인 상품 정보 관리**였습니다.

탄지로의 생각

> "우선 상품들을 제대로 분류하고 관리할 수 있는 기본 틀을 만들어보자. 음료, 과자, 도시락, 생활용품... 이렇게 카테고리별로 나누고, 각 상품의 정보를 체계적으로 저장할 수 있도록 하자. 일본 편의점과 비슷한 상품들이 많으니 분류하기 어렵지 않을 것 같은데."

이 단계에서 탄지로는 다음과 같은 상황들을 마주쳤습니다:

- **상품 분류의 필요성**: "콜라는 음료, 새우깡은 과자, 도시락은 식품... 어떻게 체계적으로 분류하지?"
- **재고 상태 관리**: "재고가 충분한지, 부족한지, 품절인지... 상태를 명확히 구분해야겠어."
- **상품별 특성 관리**: "유통기한이 있는 상품과 없는 상품을 어떻게 구분하지?"
- **가격 정책**: "정가, 할인가, 할인율... 이런 정보들을 어떻게 관리하지?"

2단계: 스마트 분석 기능 추가 🧠

기본 시스템이 잘 작동하자, 편의점 사장님이 더 고급 기능을 요청했습니다.

사장님의 요청

> "이거 정말 좋네! 그런데 이제 좀 더 똑똑하게 만들 수 있을까? 유통기한 임박한 것들 자동으로 찾아주고, 뭐가 제일 잘 팔리는지도 알려주고, 언제 발주해야 하는지도 알려주면 좋겠는데..."

탄지로는 이번에는 **데이터 분석과 자동화** 기능을 추가하기로 했습니다:

- **자동 분석 기능**: "기존 상품 클래스에 분석 기능을 어떻게 자연스럽게 추가하지?"
- **대량 데이터 처리**: "수백 개의 상품 데이터를 효율적으로 처리하려면?"
- **복잡한 계산 로직**: "할인율 계산, 회전율 분석, 발주량 계산... 이런 복잡한 로직들을 어떻게 깔끔하게 구조화하지?"
- **다양한 분석 요구사항**:
  - "3일 내 유통기한 임박 상품은?"
  - "이번 주 매출 1위는?"
  - "재고 회전율이 가장 낮은 상품은?"
  - "자동 발주가 필요한 상품들은?"

3단계: 확장 가능한 시스템 설계 🏗️

시스템이 성공적으로 운영되자, 사장님이 놀라운 제안을 했습니다.

사장님의 새로운 제안

> "탄지로 학생, 이 시스템이 정말 훌륭하네! 내 친구가 다른 지역에서 편의점을 운영하는데, 거기에도 이 시스템을 적용해줄 수 있을까? 그런데 거기는 좀 다른 종류의 상품들도 많이 팔거든..."

탄지로는 이제 **확장 가능하고 재사용 가능한 구조**를 설계해야 했습니다:

- **다양한 상품 타입**: "음료는 용량별로, 식품은 유통기한별로, 생활용품은 브랜드별로... 각각 다른 관리 방식이 필요한데?"
- **공통 기능 정의**: "모든 상품들이 공통으로 가져야 하는 기능들은 무엇일까?"
- **시스템 전체 관리**: "여러 편의점의 데이터를 통합 관리하려면 어떤 구조가 필요할까?"
- **확장성 고려**: "나중에 새로운 상품 타입이 추가되거나 새로운 기능이 필요해져도 쉽게 확장할 수 있으려면?"

💡 탄지로가 해결해야 할 과제들

위의 스토리를 읽으면서, 탄지로가 마주한 각 상황에서 어떤 기능들이 필요한지 생각해보세요. 명시적인 요구사항 대신, **상황을 분석하여 필요한 기능들을 직접 도출**해보는 것이 이 과제의 핵심입니다.

🤔 생각해볼 질문들

1. **상품 정보 관리**: 어떤 정보들을 저장해야 할까요?
2. **카테고리 분류**: 상품들을 어떻게 체계적으로 분류할 수 있을까요?
3. **재고 상태**: 재고 수준을 어떻게 자동으로 판단할 수 있을까요?
4. **유통기한 관리**: 유통기한 임박 상품을 어떻게 자동으로 찾고 할인을 적용할까요?
5. **매출 분석**: 판매 데이터에서 어떤 정보들을 추출할 수 있을까요?
6. **발주 최적화**: 언제, 얼마나 주문해야 하는지 어떻게 계산할까요?
7. **시스템 확장**: 다른 편의점에서도 사용할 수 있도록 어떻게 설계할까요?

🛠️ 구현 가이드라인

Phase 1: 기본 시스템 구축 (Week 2 개념 활용)

목표: 상품 정보를 체계적으로 관리할 수 있는 기본 구조 만들기

탄지로의 첫 번째 도전:

- 상품을 어떻게 표현할까?
- 카테고리를 어떻게 분류할까?
- 재고 상태를 어떻게 구분할까?
- 각 상황별로 어떤 처리를 해야 할까?

힌트:

- 상품에는 어떤 정보들이 필요할까요? (이름, 가격, 재고량, 카테고리, 유통기한...)
- 카테고리는 몇 가지로 나눌 수 있을까요?
- 재고 상태를 어떻게 정의할 수 있을까요? (충분, 부족, 품절...)
- 상황별 처리 로직을 어떻게 깔끔하게 작성할 수 있을까요?

Phase 2: 스마트 분석 기능 (Week 3 개념 활용)

목표: 기존 시스템에 분석 기능을 자연스럽게 추가하기

탄지로의 두 번째 도전:

- 기존 상품 클래스를 수정하지 않고 새 기능을 어떻게 추가할까?
- 많은 상품 데이터를 어떻게 효율적으로 처리할까?
- 복잡한 분석 로직을 어떻게 깔끔하게 구조화할까?

힌트:

- 클래스를 수정하지 않고 기능을 추가하는 방법이 있을까요?
- 컬렉션 데이터를 처리하는 함수형 접근법은?
- 함수 안에 작은 함수들을 만들어서 로직을 정리할 수 있을까요?

Phase 3: 확장 가능한 구조 (Week 4 개념 도입)

목표: 다양한 상품 타입과 확장을 지원하는 구조 설계

탄지로의 세 번째 도전:

- 상품 타입별로 다른 특성을 어떻게 표현할까?
- 공통 기능들을 어떻게 정의할까?
- 시스템 전체를 관리하는 구조는?

힌트:

- 비슷한 클래스들 사이의 관계를 어떻게 표현할 수 있을까요?
- 서로 다른 클래스들이 공통으로 지켜야 할 약속은?
- 전체 시스템을 관리하는 하나의 객체가 있다면?

📊 예상 시스템 결과물

📥 시스템 입력 데이터 예제

탄지로의 시스템이 처리해야 할 기본 데이터입니다:

상품 정보 입력

```
// 예시: 편의점 상품 데이터
val products = listOf(
    Product("새우깡", 1500, ProductCategory.SNACK, 30, 5, null),
    Product("콜라 500ml", 1500, ProductCategory.BEVERAGE, 25, 8, null),
    Product("김치찌개 도시락", 5500, ProductCategory.FOOD, 20, 3,
LocalDate.now().plusDays(2)),
    Product("참치마요 삼각김밥", 1500, ProductCategory.FOOD, 15, 12,
LocalDate.now().plusDays(1)),
    Product("딸기 샌드위치", 2800, ProductCategory.FOOD, 10, 2,
LocalDate.now()),
    Product("물 500ml", 1000, ProductCategory.BEVERAGE, 50, 25, null),
    Product("초코파이", 3000, ProductCategory.SNACK, 20, 15,
LocalDate.now().plusDays(1)),
    Product("즉석라면", 1200, ProductCategory.FOOD, 40, 45,
LocalDate.now().plusDays(30))
)
```

판매 기록 입력

```
// 예시: 오늘의 판매 데이터
val todaySales = mapOf(
    "새우깡" to 15,            // 15개 판매
    "콜라 500ml" to 12,        // 12개 판매
    "참치마요 삼각김밥" to 10, // 10개 판매
    "초코파이" to 8,           // 8개 판매
    "물 500ml" to 7,           // 7개 판매
```

```
    "딸기 샌드위치" to 3,      // 3개 판매
    "김치찌개 도시락" to 2      // 2개 판매
)
```

시스템 설정

```
// 재고 관리 설정
val stockThreshold = 0.3  // 재고 30% 이하 시 발주 알림
val expiryWarningDays = 3 // 3일 이내 유통기한 임박 시 할인

// 할인 정책
val discountPolicy = mapOf(
    3 to 0.0,   // 3일 이상: 할인 없음
    2 to 0.3,   // 2일 남음: 30% 할인
    1 to 0.5,   // 1일 남음: 50% 할인
    0 to 0.7    // 당일: 70% 할인
)
```

📤 시스템 출력 결과 (예시)

위의 입력 데이터를 바탕으로 탄지로가 완성한 시스템은 다음과 같은 정보를 제공해야 합니다:

```
=== 24시간 학교 편의점 스마트 재고 관리 시스템 ===

🚨 긴급 재고 알림 (재고율 30% 이하)
- 새우깡(과자류): 현재 5개 → 적정재고 30개 (25개 발주 필요) [재고율: 16.7%]
- 김치찌개 도시락(식품류): 현재 3개 → 적정재고 20개 (17개 발주 필요) [재고율: 15%]
- 딸기 샌드위치(식품류): 현재 2개 → 적정재고 10개 (8개 발주 필요) [재고율: 20%]

⚠ 유통기한 관리 (3일 이내 임박 상품)
- 김치찌개 도시락: 2일 남음 → 할인률 30% 적용 (₩5,500 → ₩3,850)
- 참치마요 삼각김밥: 1일 남음 → 할인률 50% 적용 (₩1,500 → ₩750)
- 딸기 샌드위치: 당일까지 → 할인률 70% 적용 (₩2,800 → ₩840)

📈 오늘의 베스트셀러 TOP 5
1위: 새우깡 (15개 판매, 매출 ₩22,500)
2위: 콜라 500ml (12개 판매, 매출 ₩18,000)
3위: 참치마요 삼각김밥 (10개 판매, 매출 ₩15,000)
4위: 초코파이 (8개 판매, 매출 ₩12,000)
5위: 물 500ml (7개 판매, 매출 ₩7,000)

💰 매출 현황
- 오늘 총 매출: ₩57,100 (15+12+10+8+7+3+2 = 57개 판매)
  * 새우깡: ₩22,500 (15개 × ₩1,500)
  * 콜라 500ml: ₩18,000 (12개 × ₩1,500)
  * 참치마요 삼각김밥: ₩15,000 (10개 × ₩1,500)
  * 초코파이: ₩24,000 (8개 × ₩3,000)
  * 물 500ml: ₩7,000 (7개 × ₩1,000)
  * 딸기 샌드위치: ₩8,400 (3개 × ₩2,800)
```

```
   * 김치찌개 도시락: ₩11,000 (2개 × ₩5,500)

🎯 경영 분석 리포트 (입력 데이터 기반 분석)
- 재고 회전율 최고: 딸기 샌드위치 (재고 2개, 판매 3개 → 150% 회전)
- 재고 회전율 최저: 즉석라면 (재고 45개, 판매 0개 → 0% 회전)
- 판매 효율 1위: 새우깡 (재고 5개로 15개 판매 → 300% 효율)
- 재고 과다 품목: 즉석라면 (45개), 물 500ml (25개)
- 발주 권장: 총 3개 품목, 50개 수량

📋 종합 운영 현황 (시스템 처리 결과)
- 전체 등록 상품: 8종
- 현재 총 재고: 100개 (새우깡 5 + 콜라 8 + 김치찌개 3 + 삼각김밥 12 + 딸기샌드 2 + 물 25
+ 초코파이 15 + 즉석라면 45)
- 현재 재고가치: ₩171,100
- 재고 부족 상품: 3종 (30% 이하)
- 유통기한 임박: 3종 (3일 이내)
- 오늘 총 판매: 57개
- 시스템 처리 완료: 100%
```

📝 제출 방법

🛠️ 프로젝트 구조

Gradle 단일 모듈 Kotlin 프로젝트로 작성하여 제출합니다.

📁 프로젝트 디렉토리 구조 예시

```
convenience-store-system/
├── build.gradle.kts           // Gradle 빌드 설정
├── gradle.properties          // Gradle 속성 파일
├── settings.gradle.kts        // 프로젝트 설정
├── gradle/
│   └── wrapper/              // Gradle Wrapper 파일들
├── src/
│   └── main/
│       └── kotlin/
│           └── store/        // 패키지명 예시
│               ├── Product.kt                // Phase 1: 기본 상품 클래스
│               ├── ProductExtensions.kt      // Phase 2: 확장 함수들
│               ├── AdvancedProduct.kt        // Phase 3: 고급 상품 클래스들
│               ├── InventoryManager.kt       // Phase 3: 시스템 매니저
│               └── Main.kt                   // 실행 및 테스트
├── README.md                  // 프로젝트 설명, 사용시나리오 기술
├── REVIEW.md                  // 회고
└── PROMPT.md                  // 사용한 프롬프트 내용
```

🚀 실행 방법

프로젝트는 **Gradle 빌드 후 JAR 파일 실행** 방식으로 동작해야 합니다:

1. 프로젝트 빌드

```
# 프로젝트 빌드 (JAR 파일 생성)
./gradlew build

# 또는 Windows에서
gradlew.bat build
```

2. JAR 파일 실행

```
# 생성된 JAR 파일 실행
java -jar build/libs/convenience-store-system-1.0.0.jar
```

🔄 **개발 중 테스트용 (선택사항)**

개발 과정에서 빠른 테스트를 위해 `gradle run`도 지원합니다:

```
# 개발 중 빠른 실행 테스트
./gradlew run
```

최종 제출 시에는 반드시 JAR 파일이 정상 실행되는지 확인하세요!

📤 GitHub 제출 절차

1. GitHub 리포지토리 생성

1. **개인 GitHub 계정**에 로그인
2. 새 리포지토리 생성 (public)
3. 리포지토리 이름: `convenience-store-system`

2. 프로젝트 업로드

```
# 프로젝트 디렉토리에서 실행
git init
git add .
git commit -m "Initial commit: 편의점 재고 관리 시스템"

# GitHub 리포지토리와 연결 (본인의 GitHub 주소로 변경)
git remote add origin https://github.com/[본인계정]/convenience-store-
system.git

# main 브랜치로 업로드
```

```
git branch -M main
git push -u origin main
```

3. LMS 제출

LMS에는 GitHub 클론 주소만 제출합니다:

```
https://github.com/[본인계정]/convenience-store-system.git
```

🔧 기술적 요구사항

- **JDK 24** 사용 필수
- **Kotlin 2.2** 버전 사용
- **단일 모듈** 구조 (멀티 모듈 X)
- **외부 라이브러리 최소화** (Kotlin 표준 라이브러리 위주)
- **JAR 파일 실행** 가능해야 함

📊 평가 기준

- **실행 완정성 (50%)**: 오류 없이 실행되는 완성된 프로그램
- **알로리즘, 프로그램 내용(10%)** : 적정한 수준의 논리로 구현되었는지 확인, 주관적인 견해 있을수 있음
- **문서화(40%)**
  - Readme.md : 프로젝트 간단 설명, 구현방법, 사용방법과 사용 시나리오가 있어야 한다.
  - Review.md
    - 좋았던 것 : 긍정적이고 성공적이었던 경험이나 요소
    - 배운 것 : 새롭게 알게 된 지식, 기술, 통찰력 등
    - 놀라왔던 것 : 기대했던 것과 달랐던 점이나, 앞으로 더 시도해보고 싶거나 필요하다고 생각하는 것
    - 부족했던 것: 프로젝트를 진행하며 부족했던 부분, 개선이 필요한 부분
  - Prompt.md : 개발에 사용한 ai 프롬프트를 기술하고, 왜 이런 프롬프트를 만들어서 썼는지 기술
    - 사용한 모든 프롬프트를 다 쓸 필요는 없고, 중요하다고 생각되는 프롬프트만 써도 된다.

⚠ 제출 전 확인사항

1. `gradle build` 후 JAR 파일이 정상 실행되는지 확인
2. 모든 요구사항이 구현되었는지 체크리스트 확인
3. README.md가 충분히 상세하게 작성되었는지 확인
4. GitHub 리포지토리가 정상적으로 접근 가능한지 확인
