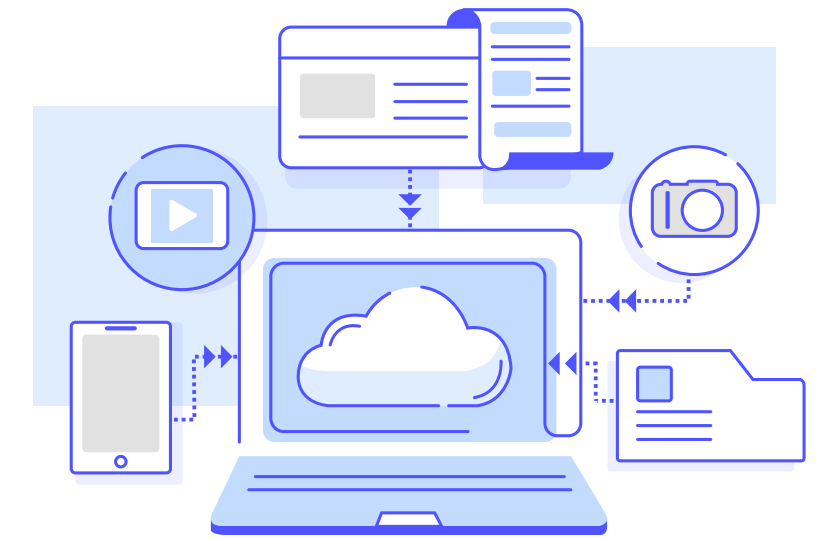


편의점 관리 소프트웨어 개발 프로젝트

콘솔 기반 시스템 구현을 통한 효율적인 편의점 운영
솔루션



목차

01 | 프로젝트 개요

- 프로젝트 목표 및 기대 효과

02 | 시스템 설계

- 전체 시스템 구조 설계
- 데이터베이스 설계
- 사용자 인터페이스 설계

03 | 요구사항에 맞는 동영상 설명

04 | 핵심 기능 구현 설명

- 물품 입력
- 물품 계산

05 | 향후 보완 계획 및 느낀 점

- 데이터 입출력 방식
- 파일 시스템 활용
- 데이터 백업 및 복구 방안

프로젝트 개요

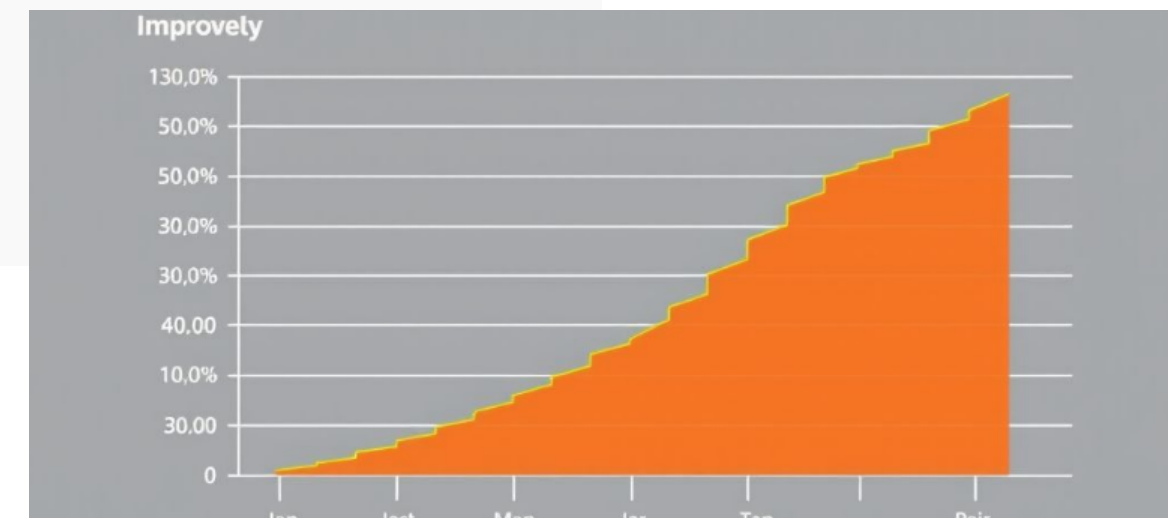
요구사항에 맞춘 철저한 개발 완성

- 전체 기능을 요구사항 문서 기준으로 구현
- 제품 입력, 재고 확인, 물품 입고, 계산, 검색
- 예외 처리 반영



프로젝트 목표 및 기대 효과

- 빠른 결과보다 이해 중심으로 천천히 구현
- 각 기능의 구조와 흐름을 직접 설계하고 구현
- 콘솔 개발을 통해 Java, DB, 구조에 대한 이해도 향상



시스템 설계

전체 시스템 구조

- 콘솔 기반 흐름
- 제품 입력 및 확인
 - 제품 입고
 - 제품 계산
- 매출 확인 및 종료
- 종료시 일당 계산
- 각 기능 독립 구현
- 재사용성 고려 로직 분리

데이터베이스 설계

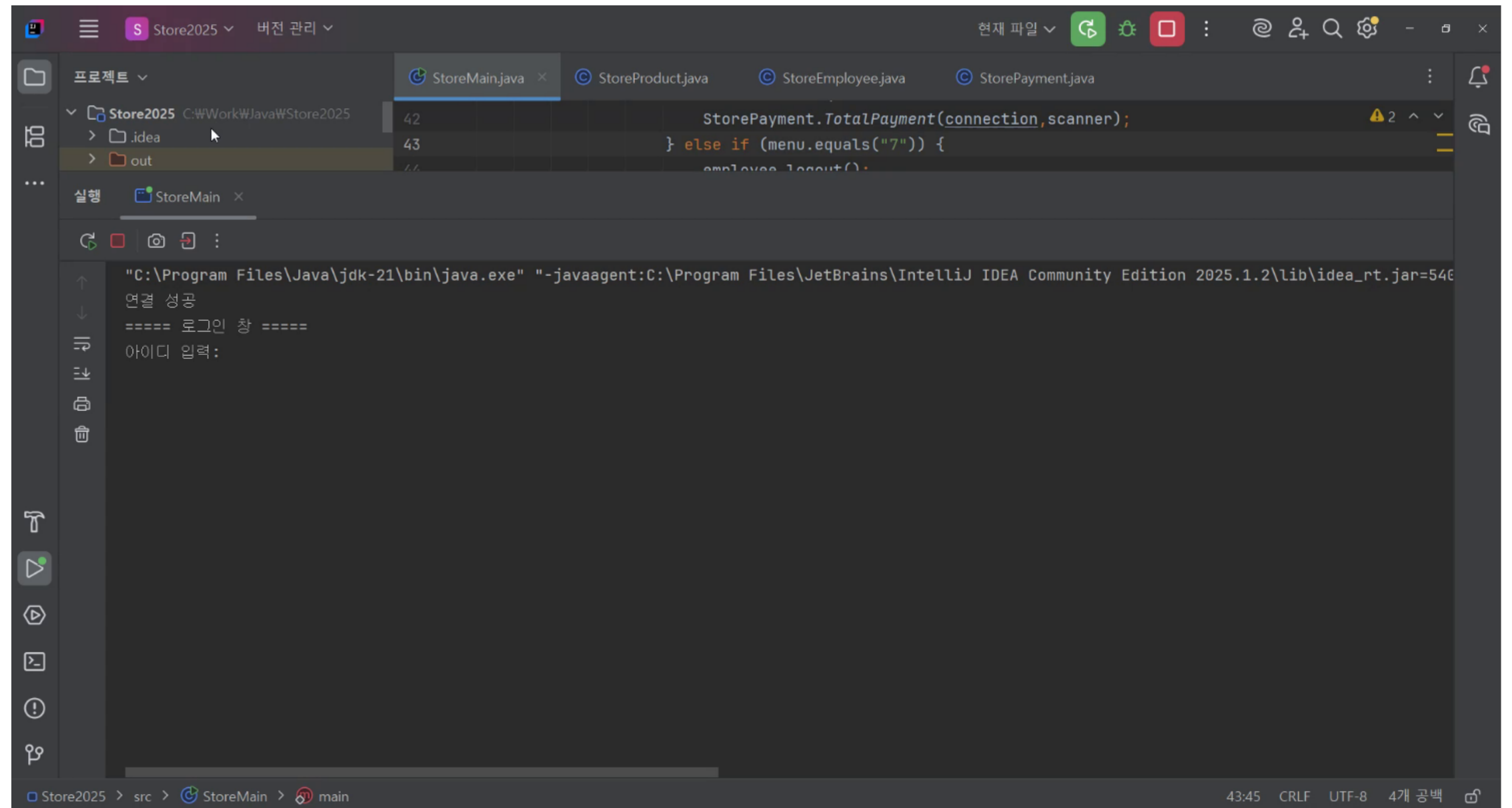
- 직원 테이블
- 제품 테이블
- 결제 테이블
- 잔고 테이블

사용자 인터페이스 설계

- 콘솔 출력 메뉴
- 숫자 선택 방식
- 유효하지 않은 입력 시 오류
 - 유통기한
 - 19금 여부
- 거스름돈 계산 등 안내
- 종료 시 인사말

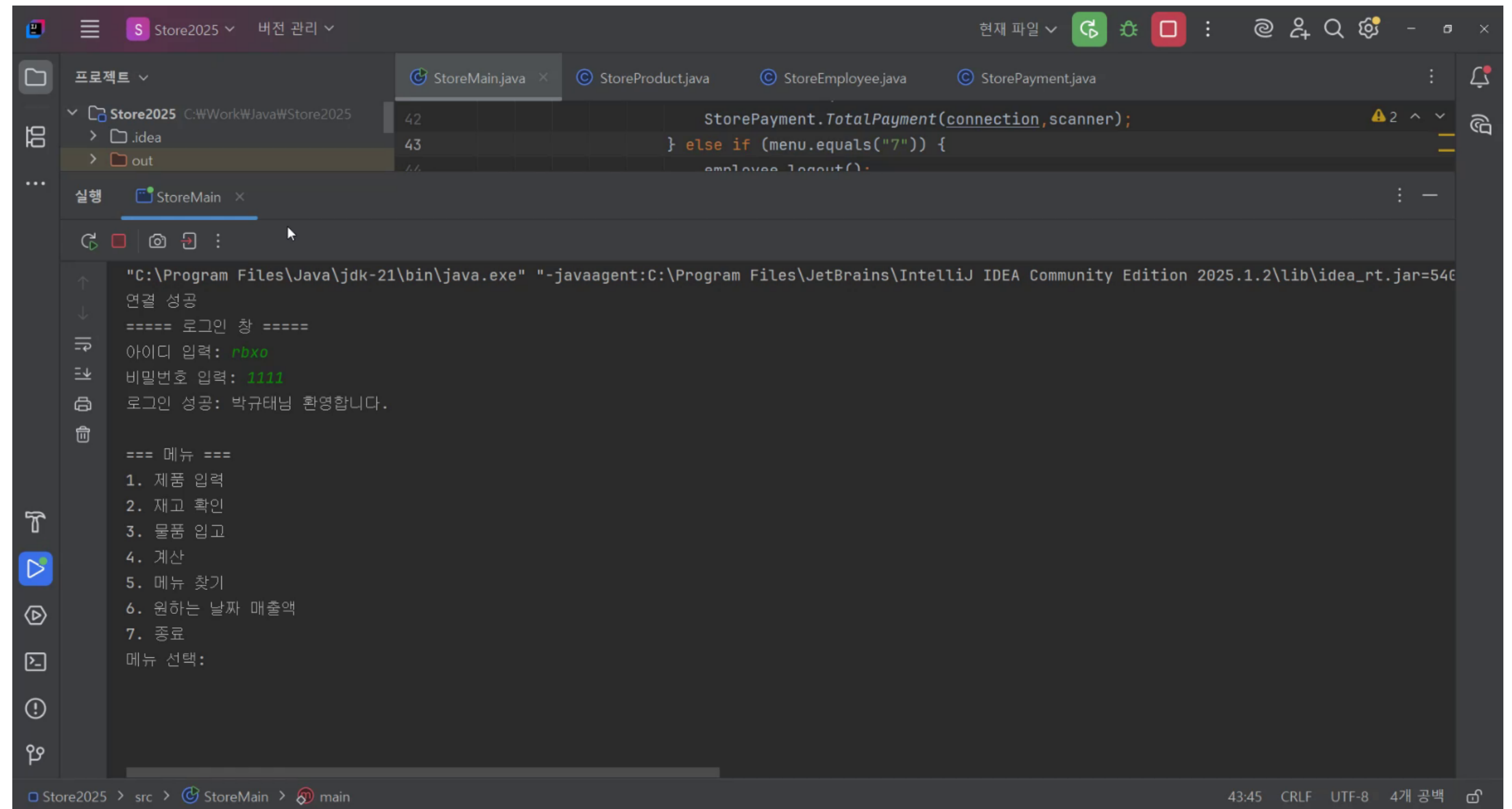
요구사항 1,2,3,4,5

1. 시작시 로그인 창
2. 아이디, 패스워드 입력
3. 개인정보 표시
4. 로그인 후 날짜, 시간 저장



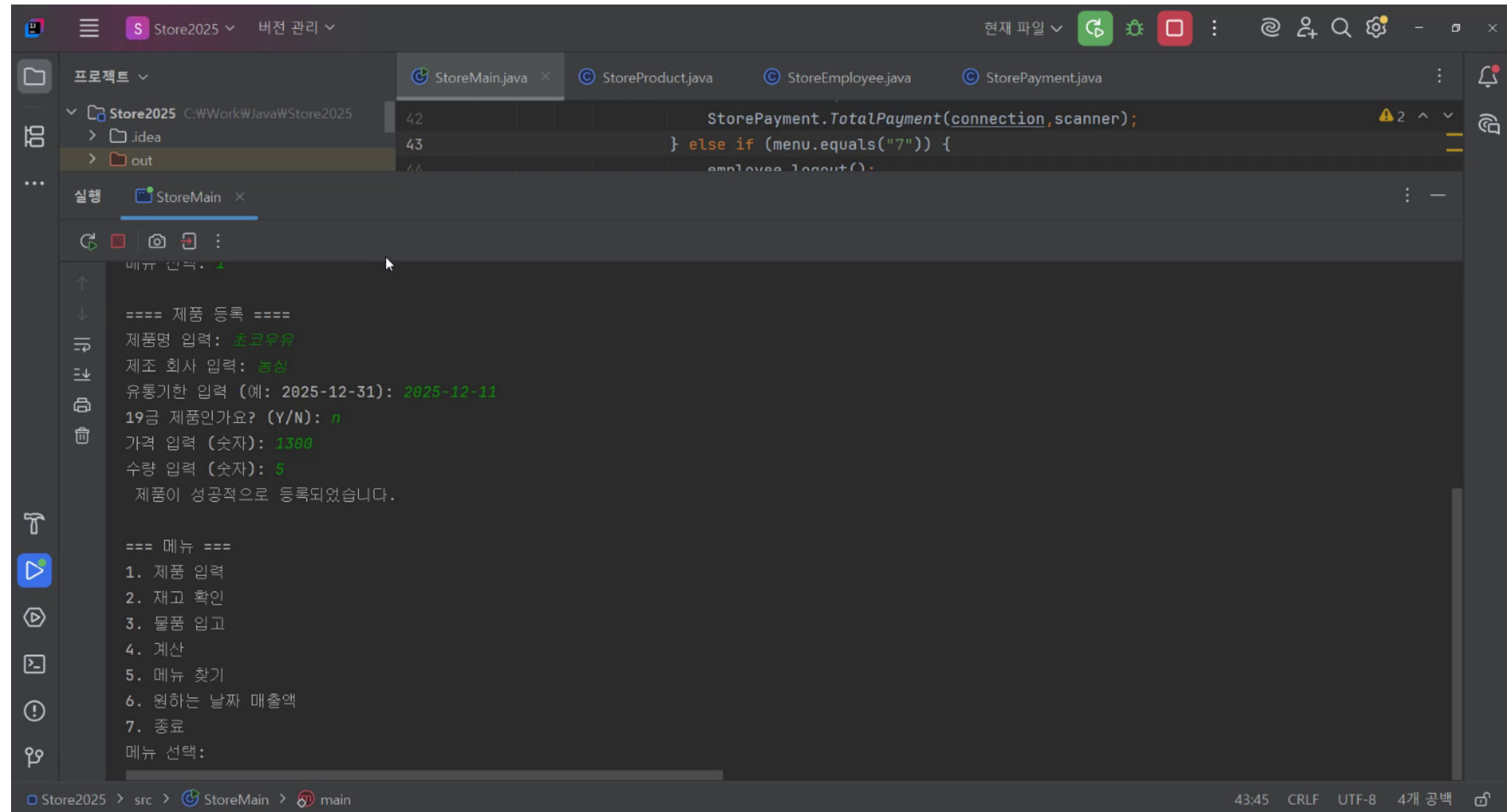
요구사항 8,9,10

1. 제품 입력 메뉴
2. 제품 입력
3. 제품 10개



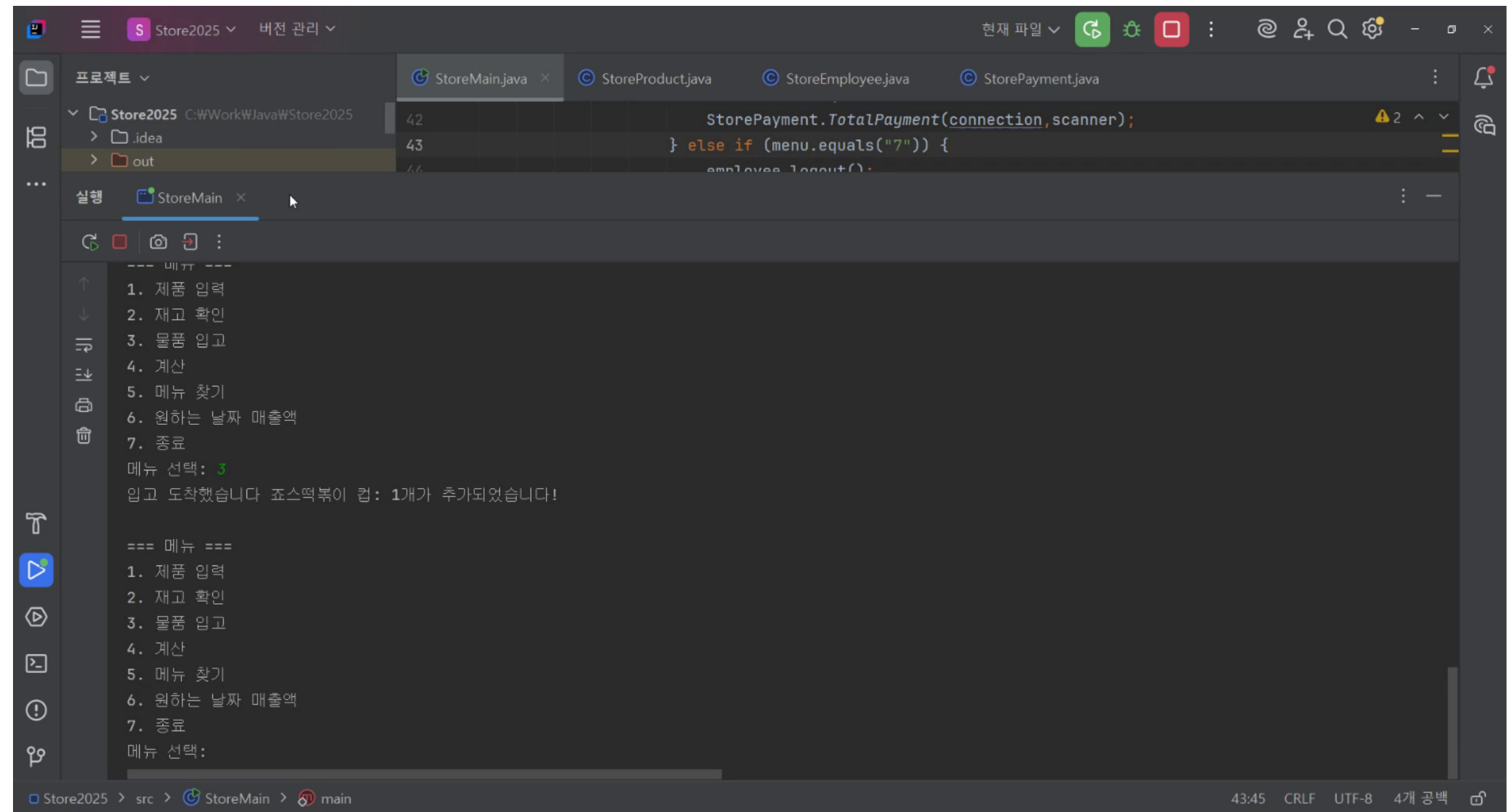
요구사항 13,14,15,16,17,18,19

1. 제품 확인 메뉴
2. *통해 보여줌
3. 제품 입고 메뉴
4. 랜덤 물품 도착



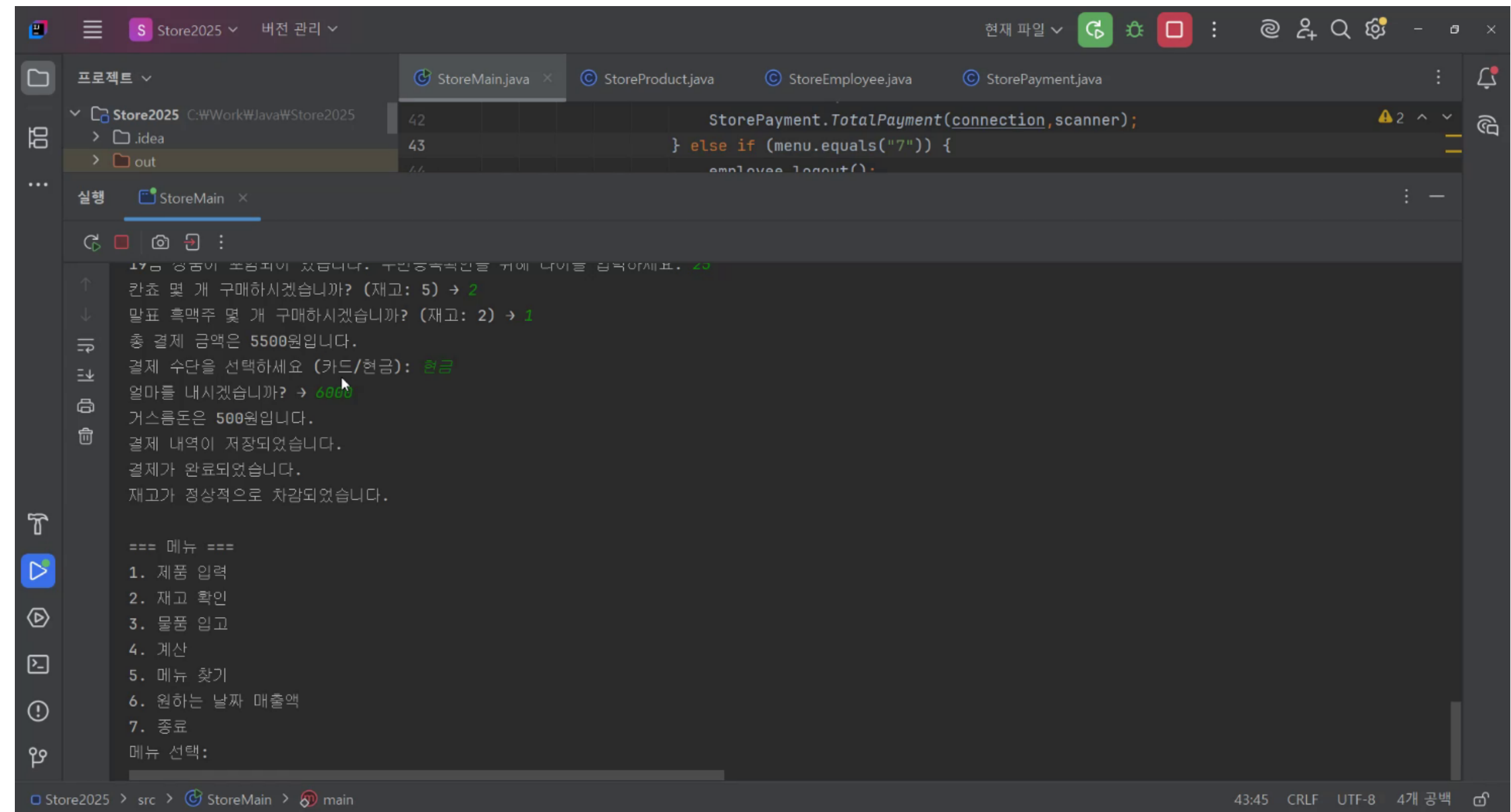
요구사항 19,20,21,22,23,25

1. 계산 메뉴 구성
2. 제품을 구매하는 것처럼 구성
3. 19금 물품 확인
4. 현금 제출 거스름돈 계산
5. 재고, 잔고 업데이트
6. 유통기한 지난 제품 확인

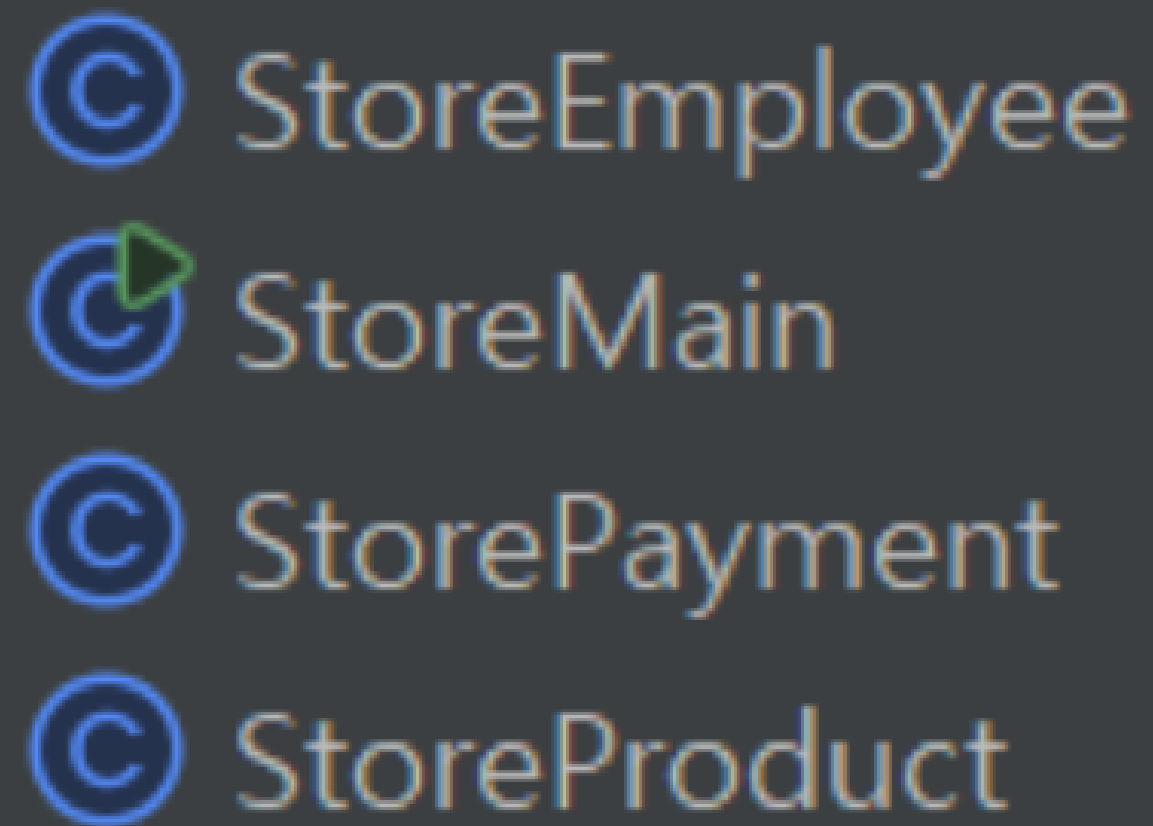


요구사항 26, 27, 28

1. 제품 찾는 메뉴
2. 원하는 날짜 매출 정보
3. 종료 후 인사말



편의점 관리 시스템 클래스 구조 설명



© StoreEmployee
© StoreMain
© StorePayment
© StoreProduct

1. 직원 관련 기능 클래스
2. 메인 실행부 클래스
3. 결제 기능 클래스
4. 상품 관련 기능 클래스

메서드 기능 구현

제품 입력

- 제품명, 제조사, 유통기한, 가격, 수량, 19금 여부
- product에 입력한 제품 데이터가 insert됨
- 19금 항목은 Y/N 체크
- 최소 10개 입력

계산 시스템

- 고객이 제품을 선택 → 가격 합산 → 결제 수단 선택
- 카드/현금에 따라 처리 방식
- 결제 내역은 PAYMENT 테이블에 자동 저장
- 19금 제품 포함 시, 성인 인증 확인 절차 있음
- 결제 완료 후 상품 재고 차감, 잔고 갱신 처리
- 유통기한 지난 제품 구매 시, 경고 메시지 출력 기능 구현

제품 입력 - 사용자 입력 단계



```
public static StoreProduct InsertProduct(Connection connection, Scanner scanner) {  
    System.out.println("\n=== 제품 등록 ===");  
  
    System.out.print("제품명 입력: ");  
    String name = scanner.nextLine();  
  
    System.out.print("제조 회사 입력: ");  
    String manufacturer = scanner.nextLine();  
  
    System.out.print("유통기한 입력 (예: 2025-12-31): ");  
    String expDateStr = scanner.nextLine();  
    LocalDate expirationDate;  
  
    System.out.print("19금 제품인가요? (Y/N): ");  
    String adultInput = scanner.nextLine().trim().toUpperCase();  
    boolean isAdultOnly = adultInput.equals("Y");  
  
    System.out.print("가격 입력 (숫자): ");  
    int price = scanner.nextInt();  
  
    System.out.print("수량 입력 (숫자): ");  
    int quantity = scanner.nextInt();  
    scanner.nextLine();  
}
```

1. 제품 입력 기능은 데이터베이스와 연결하기 위해 Connection과 사용자 입력을 받기 위한 Scanner를 사용합니다.
2. 사용자에게 제품명, 제조사, 유통기한, 가격, 수량, 19금 여부를 지역 변수에 입력받습니다.

제품 입력 - 유효성 검사 단계



```
try {  
    expirationDate = LocalDate.parse(expDateStr);  
} catch (Exception e) {  
    System.out.println("유통기한 형식 오류입니다.");  
    return null;  
}  
  
if (expirationDate.isBefore(LocalDate.now())) {  
    System.out.println("유통기한이 지난 제품은 등록할 수 없습니다.");  
    return null;  
}
```

1. 유통기한은 문자열로 입력받기 때문에 날짜 형식이 맞는지 검사 후 오늘보다 날짜 이전이면 등록을 막는 로직 구현

제품 입력 - 데이터베이스 저장



```
final String insert_sql = new StringBuilder()
    .append("INSERT INTO PRODUCT (")
    .append("productName, manufacturer, expirationDate, isAdultOnly, ")
    .append("price, quantity) ")
    .append("VALUES (?, ?, ?, ?, ?, ?)")
    .toString();

try (PreparedStatement preparedStatement = connection.prepareStatement(insert_sql))
    preparedStatement.setString(parameterIndex: 1, name);
    preparedStatement.setString(parameterIndex: 2, manufacturer);
    preparedStatement.setDate(parameterIndex: 3, Date.valueOf(expirationDate));
    preparedStatement.setString(parameterIndex: 4, isAdultOnly ? "Y" : "N");
    preparedStatement.setInt(parameterIndex: 5, price);
    preparedStatement.setInt(parameterIndex: 6, quantity);

    int result = preparedStatement.executeUpdate();
    if (result > 0) {
        System.out.println(" 제품이 성공적으로 등록되었습니다.");
    } else {
        System.out.println(" 제품 등록 실패");
    }
}
```

1. SQL 구문은 가독성을 높이기 위해서 StringBuilder를 사용하여 여러 줄로 나누어 작성했습니다.
2. 입력받은 값들은 PreparedStatement의 ? 자리에 순서대로 바인딩됩니다.
3. executeUpdate()를 실행하면 실제로 DB에 저장되며 try - catch로 오류 처리했습니다.

제품 입력 - 결과값



==== 제품 등록 ====

제품명 입력: **햇반**

제조 회사 입력: **농심**

유통기한 입력 (예: 2025-12-31): **2025-07-31**

19금 제품인가요? (Y/N): **n**

가격 입력 (숫자): **2200**

수량 입력 (숫자): **5**

제품이 성공적으로 등록되었습니다.



PRODUC...	PRODUCTNAME	MANUFACTURER	EXPIRA1
4	신라면	농심	25/12/31
5	칸초	롯데제과	25/11/30
6	햇식스	롯데칠성음료	25/10/15
7	진로 소주	하이트진로	25/07/06
8	CU 바나나우유	빙그레	25/08/20
9	매운김밥	자체제작	25/07/05
10	콘치즈삼각김밥	자체제작	25/07/04
11	말표 흑맥주	서울장수	25/12/01
12	햇바 불닭맛	CJ	25/08/10
13	조스떡볶이 컵	조스푸드	25/10/25
41	햇반	농심	25/07/31

--

계산 - 상품 목록 호출



```
public static void cashProduct(Connection connection, Scanner scanner) { 1개 사용 위치
    final String show_sql = ""
    SELECT productId, productName, manufacturer, expirationDate,
           isAdultOnly, price, quantity
    FROM Product
    "";
    List<StoreProduct> productList = new ArrayList<>();

    System.out.println("=====상품 목록=====");

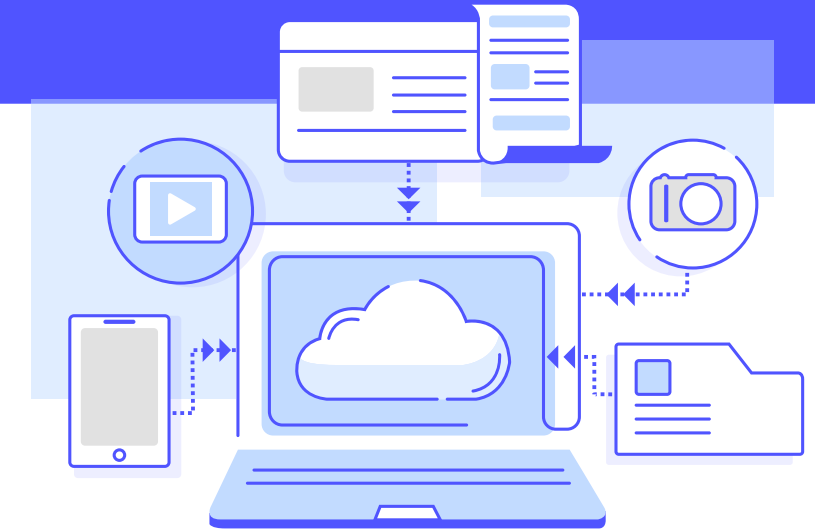
    try (PreparedStatement preparedStatement = connection.prepareStatement(show_sql);
         ResultSet rs = preparedStatement.executeQuery()) {

        while (rs.next()) {
            StoreProduct product = new StoreProduct(
                rs.getString(columnLabel: "productName"),
                rs.getString(columnLabel: "manufacturer"),
                rs.getDate(columnLabel: "expirationDate").toLocalDate(),
                rs.getString(columnLabel: "isAdultOnly").equals("Y"),
                rs.getInt(columnLabel: "price"),
                rs.getInt(columnLabel: "quantity")
            );
            product.setProductId(rs.getInt(columnLabel: "productId"));
            productList.add(product);

            System.out.printf("%s : %d개\n", product.getProductName(), product.getQuantity());
        }
    }
}
```

1. 상품 목록을 불러오기위해 connection 객체를 통해 연결 후 scanner 객체를 통해 입력을 받을 수 있도록 합니다.
2. 조회된 상품 정보를 저장하기 위해 리스트를 새로 생성합니다
3. SQL문을 실행해 상품정보를 가져와서 리스트에 추가합니다.
4. 각 상품명을 출력하면서 재고도 함께 보여줍니다 EX) 신라면 : 8개

계산 - 상품 선택 받기



```
System.out.println("어떤 것을 구매하시겠습니까?");
System.out.println("원하는 상품을 입력하세요! (여러개는 , 쉼표로 구분)");

String input = scanner.nextLine();
String[] selectedNames = input.split(regex: ",");

List<StoreProduct> selectedProducts = new ArrayList<>();

for (String rawName : selectedNames) {
    String trimmedInput = rawName.trim().replaceAll(regex: "\\s+", replacement: "");
    for (StoreProduct product : productList) {
        String dbNameNoSpace = product.getProductName().replaceAll(regex: "\\s+", replacement: "");
        if (trimmedInput.equalsIgnoreCase(dbNameNoSpace)) {
            selectedProducts.add(product);
            break;
        }
    }
}
```

1. 사용자에게 상품명을 입력.(쉼표로 여러 개 가능)
2. 일치하는 상품을 찾아 selectList에 추가
3. 존재하지 않는 상품일 경우 예외처리

계산 - 유통기한 19금 상품 여부 확인



```
if (!expired.isEmpty()) {
    System.out.println("유통기한이 지난 상품은 구매할 수 없습니다.");
    for (StoreProduct p : expired) {
        System.out.println("- " + p.getProductName());
    }
    selectedProducts.removeAll(expired);
}

boolean hasAdultItem = selectedProducts.stream().anyMatch(StoreProduct::isAdultOnly);
if (hasAdultItem) {
    System.out.print("19금 상품이 포함되어 있습니다. 주민등록확인을 위해 나이를 입력하세요: ");
    int age = Integer.parseInt(scanner.nextLine());
    if (age < 20) {
        System.out.println("미성년자는 19금 상품을 구매할 수 없습니다.");
        selectedProducts.removeIf(StoreProduct::isAdultOnly);
    }
}
```

1. 오늘 날짜와 비교하여 유통기한 지난 상품을 찾음
2. 선택된 상품 중 isAdultOnly가 true인 항목이 있는지 확인
3. 있다면 나이를 입력받아 20세 이상인지 판단

계산 - 구매 수량 입력



```
for (StoreProduct product : selectedProducts) {  
    int stock = product.getQuantity();  
    int count = 0;  
  
    while (true) {  
        System.out.printf("%s 몇 개 구매하시겠습니까? (재고: %d) → ", product.getProductName(), stock);  
        String qtyInput = scanner.nextLine();  
        try {  
            count = Integer.parseInt(qtyInput);  
            if (count <= 0) {  
                System.out.println("1개 이상 입력해야 합니다.");  
            } else if (count > stock) {  
                System.out.println("재고보다 많은 수량을 입력할 수 없습니다.");  
            } else {  
                break;  
            }  
        } catch (NumberFormatException e) {  
            System.out.println("숫자로 입력해주세요.");  
        }  
    }  
  
    productNames.add(product.getProductName());  
    quantities.add(count);  
    totalPrice += product.getPrice() * count;  
}  
System.out.printf("총 결제 금액은 %d원입니다.\n", totalPrice);
```

1. 남은 상품 중 각 상품마다 몇 개 살 건지 물어봄
2. 재고보다 많은 수량은 입력할 수 없음
3. 총 결제 금액 계산

계산 - 결제 수단 선택



```
System.out.print("결제 수단을 선택하세요 (카드/현금): ");
String paymentMethod = scanner.nextLine().trim();

if (!paymentMethod.equals("카드") && !paymentMethod.equals("현금")) {
    System.out.println("올바른 결제 수단이 아닙니다. 결제 취소합니다.");
    return;
}
if (paymentMethod.equals("현금")) {
    System.out.print("얼마를 내시겠습니까? → ");
    int cash = Integer.parseInt(scanner.nextLine());
    if (cash < totalPrice) {
        System.out.println("지불 금액이 부족합니다. 결제를 취소합니다.");
        return;
    }
    int change = cash - totalPrice;
    System.out.printf("거스름돈은 %d원입니다.\n", change);
}
```

1. 사용자에게 결제 수단(카드/현금) 선택 받음
2. 현금이면 지불 금액도 입력받아 거스름돈 계산
3. 카드/현금 외 입력 시 결제 취소 처리

계산 - 잔액 업데이트 및 재고 차감



```
final String updateBalanceSql = "UPDATE STORE_STATUS SET balance = balance - ? WHERE statusId = 1";
try (PreparedStatement preparedStatement = connection.prepareStatement(updateBalanceSql)) {
    preparedStatement.setInt( parameterIndex: 1, totalPrice);
    preparedStatement.executeUpdate();
}

StorePayment payment = new StorePayment(productNames, quantities, totalPrice, paymentMethod);
payment.insertPayment(connection);

System.out.println("결제가 완료되었습니다.");

final String updateStockSql = "UPDATE Product SET quantity = quantity - ? WHERE productName = ?";
try (PreparedStatement preparedStatement = connection.prepareStatement(updateStockSql)) {
    for (int i = 0; i < productNames.size(); i++) {
        preparedStatement.setInt( parameterIndex: 1, quantities.get(i));
        preparedStatement.setString( parameterIndex: 2, productNames.get(i));
        preparedStatement.executeUpdate();
    }
    System.out.println("재고가 정상적으로 차감되었습니다.");
} catch (SQLException e) {
    System.out.println("재고 차감 중 오류 발생");
    e.printStackTrace();
}
```

1. 구매한 결제 금액만큼 잔고 증가
2. 결제한 상품 각각에 대해 구매 수량만큼 재고 감소

계산 - 콘솔 실행 결과

어떤 것을 구매하시겠습니까?
 원하는 상품을 입력하세요! (여러개는 , 쉼표로 구분)
 햇반, 말표 흑맥주
 19금 상품이 포함되어 있습니다. 주민등록확인을 위해 나이를 입력하세요: 25
 햇반 몇 개 구매하시겠습니까? (재고: 5) → 2
 말표 흑맥주 몇 개 구매하시겠습니까? (재고: 3) → 2
 총 결제 금액은 9400원입니다.
 결제 수단을 선택하세요 (카드/현금): 현금
 얼마를 내시겠습니까? → 10000
 거스름돈은 600원입니다.
 결제 내역이 저장되었습니다.
 결제가 완료되었습니다.
 재고가 정상적으로 차감되었습니다.



	TOTALPRICE	PAYMENTMETHOD	PAYMENTID	PAYMENTDA...
1	8000	카드	1	25/07/04
2	1000	현금	2	25/07/04
3	3500	현금	3	25/07/04
4	2500	카드	4	25/07/04
5	2000	현금	5	25/07/04
6	22400	카드	6	25/07/04
7	2500	카드	7	25/07/07
8	1800	카드	8	25/07/07
9	2000	카드	9	25/07/07
10	9400	현금	10	25/07/08

	PRODUCTID	PRODUCTNAME	QUANTITY
1	5	칸초	5
2	6	핫식스	3
3	7	진로 소주	6
4	8	CU 바나나우유	9
5	9	매운김밥	5
6	10	콘치즈삼각김밥	5
7	11	말표 흑맥주	1
8	12	햇바 불닭맛	5
9	13	조스떡볶이 컵	2
10	4	신라면	8
11	41	햇반	3

기능 확장을 위한 개발 아이디어



1. 유통기한 임박 상품 자동경고 - 3일 이하 남은 제품은 별도 표시
2. 반품 및 환불 처리 기능 - 결재한 상품을 일정 시간 내에 반품하고 환불 금액을 잔고에서 차감
3. 고객 영수증 출력 기능 - 구매 내역을 형식화해서 콘솔에 출력

느낀 점

이번 프로젝트를 진행하면서,
강사님이 알려주신 코드를 하나하나 따라하며 구조를 이해하고
직접 머릿속으로 흐름을 떠올리면서 구현해보는 경험이 새로웠습니
다.

처음엔 막막했지만,
천천히 단계를 나눠서 접근하니 생각보다 잘 동작하는 게 신기했고,
이 과정을 통해 실제로 실력이 많이 향상되었다고 느꼈습니다.

