



자바를 활용한 신발 쇼핑몰 구현

목차

1 Introduction

- 개요
- 요구사항
- 목적

2 Application

- 시스템 구성도
- 세부구현

3 Result

- Menu Map
- 시연영상

4 Conclusion

- 느낀점
- 발전방향



1. Introduction

요구사항

- 고객 관리 및 상품 관리 프로그램 -> 신발 쇼핑몰
- 데이터 관리 구조, 책임 중심 설계 기법 사용 -> 시스템 설계
- 파일 저장 포맷 설계 및 구현 -> CSV
- 자바의 컬렉션을 사용하여 자료 관리 및 검색 구현 -> ArrayList / HashMap

개요

Java를 사용해 관리 및 이용이 편리한 신발 쇼핑몰을 구현하고자 한다.

csv file과 console 간의 입출력을 통해 쇼핑몰을 이용할 수 있도록 제작한다.

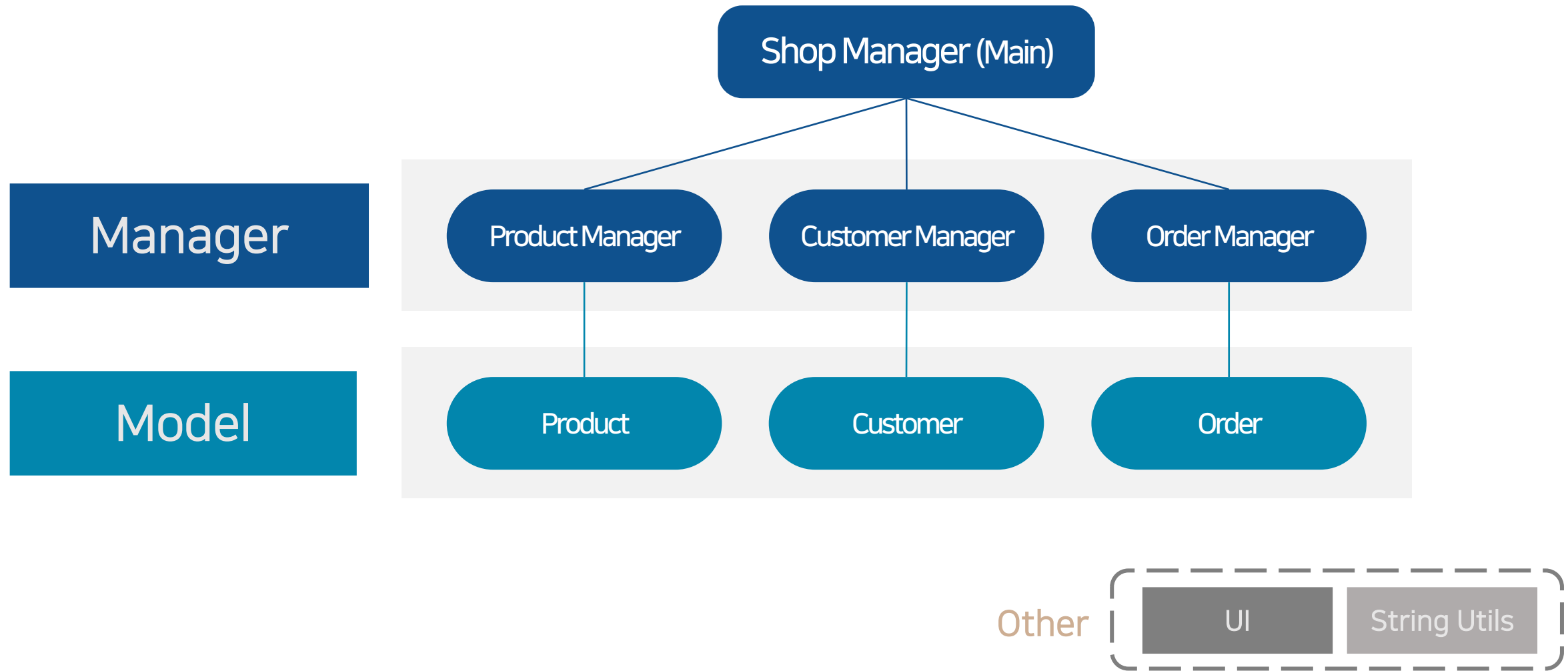
목적

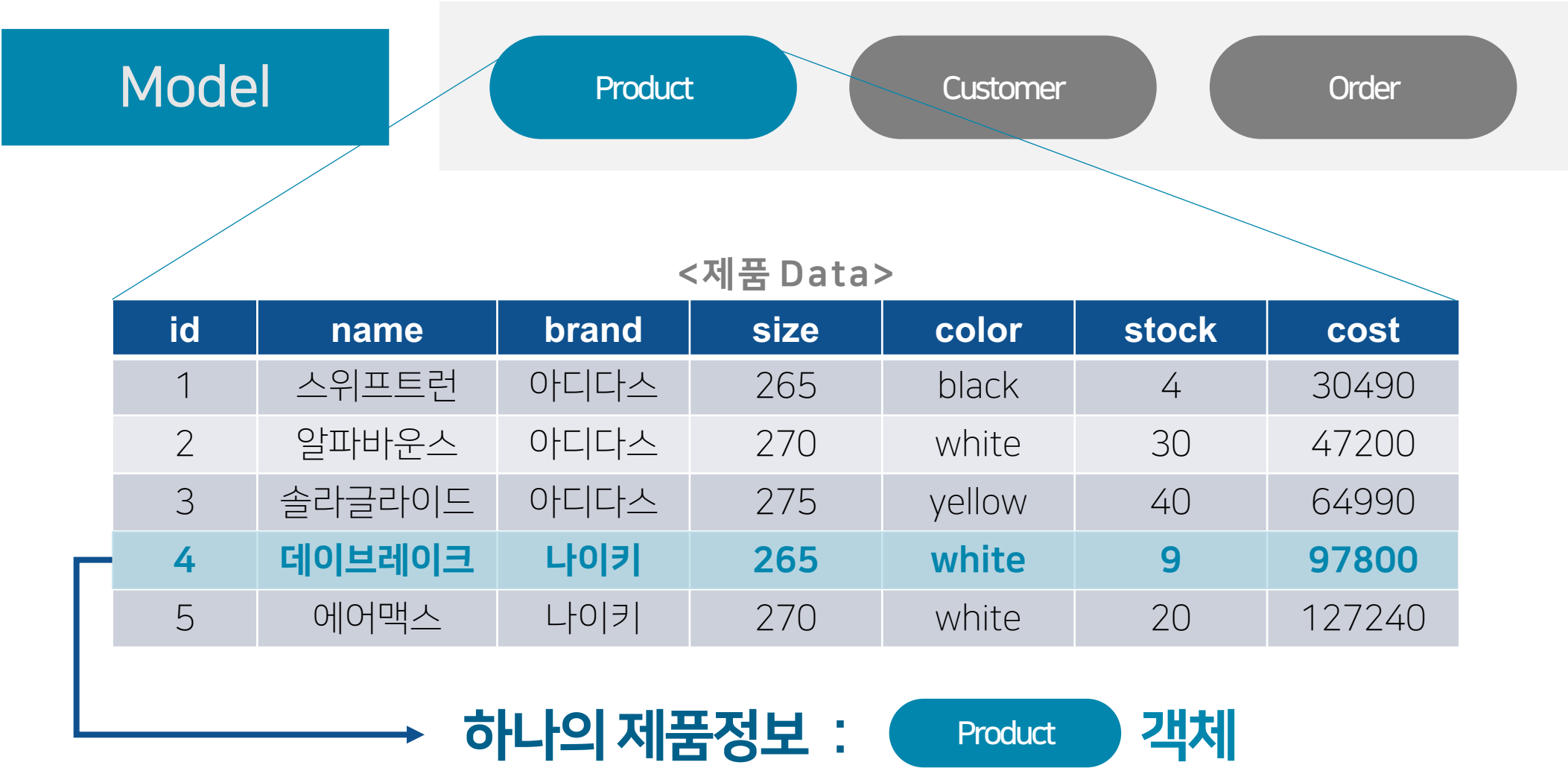
회원가입, 로그인 기능으로 고객과 관리자를 구분하여 서로 다른 화면과 기능을 제공한다.

이에 따라 관리자는 쇼핑몰 이용, 관리 기능(고객, 상품)을 수행할 수 있고

고객은 구매, 주문확인, 정보수정 및 탈퇴 기능을 사용하여 쇼핑몰을 이용할 수 있다.

2. Application





Manager

ProductManager

CustomerManager

OrderManager

```
products = new ArrayList<Product>();  
productsHash = new HashMap<Integer, Product>();
```

데이터 저장 방법 : ArrayList와 HashMap에 value로 Product 객체를 넣는다.

Manager

ProductManager

CustomerManager

OrderManager

2가지 자료구조를 선택한 이유?

[ArrayList]

- 장점 : 저장한 순서대로 정렬되게 출력 가능
- 단점 : 검색, 수정, 삭제가 느림

[HashMap]

- 장점 : 검색, 수정, 삭제가 빠름
- 단점 : 정렬 출력x

-> 서로의 장단점을 상호보완하여 빠른 기능 수행이 가능!

Manager

ProductManager

CustomerManager

OrderManager

ex) 삭제기능

```
Product p = productsHash.get(id);  
products.remove(p);  
productsHash.remove(id);
```

- HashMap을 통해 해당 id의 Product객체 가져오기
- ArrayList에서 remove메소드로 같은 객체를 바로 삭제
- HashMap에서도 삭제

-> ArrayList에서의 객체 삭제를 위해 for문을 사용하지 않아도 됨

Manager

ProductManager

CustomerManager

OrderManager

Product Manager 주요 기능

제품 등록

제품 삭제

제품 구매

제품 정보 검색(by all, id, brand, name)

-> 제품구매와 관리를 위한 기능이 수행됨

Manager

ProductManager

CustomerManager

OrderManager

Customer Manager 주요 기능

사용자 등록

사용자 삭제(탈퇴)

정보 수정

사용자 정보 검색(by id, all)

로그인

회원가입

-> 사용자(고객/관리자) 관리를 위한 기능이 수행됨

Manager

ProductManager

CustomerManager

OrderManager

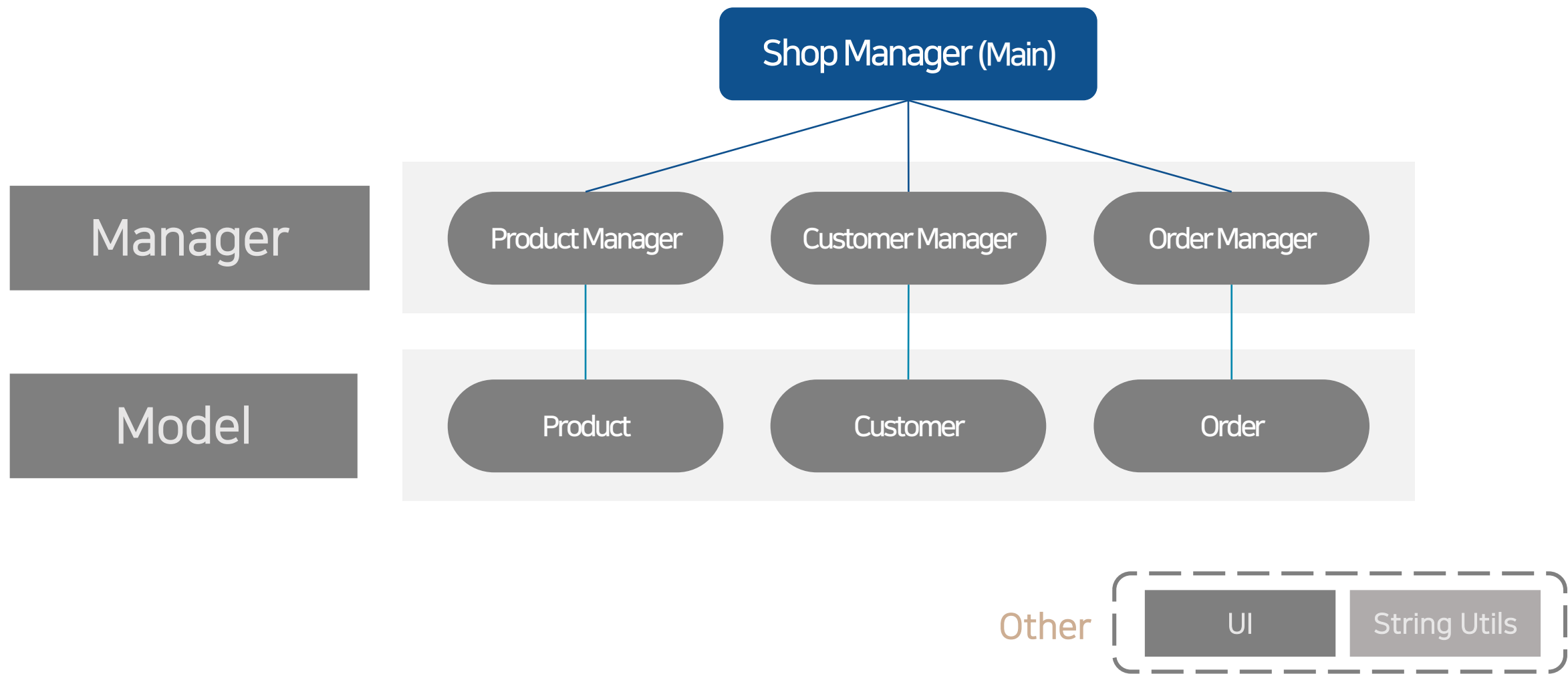
Order Manager 주요 기능

주문 등록

주문 삭제(회원탈퇴 시)

주문 정보 검색(by uid, all)

-> 주문관리를 위한 기능이 수행됨



Shop Manager 주요 기능

1. 프로그램 시작 시 data 읽기 및 저장
2. 메뉴의 이동
3. 메뉴 선택에 따른 Manager(Product/Customer/Order) 통합기능 수행

-> 쇼핑몰 이용 및 관리를 위한 통합기능이 수행됨

세부구현 3가지

1. do while을 사용한 Menu 이동
2. 숫자형 변수에는 숫자만 입력
3. 현재 메뉴 위치 표시

Console의 특성 고려

1. do while을 사용한 Menu 이동

[SHOPPING MENU]

1. 전체 제품
2. 상세 검색
3. 주문 확인
4. 정보 수정
5. 탈퇴
0. 이전 메뉴로

```
private int displayShoppingMenu(){ 하나의메뉴마다하나의메소드를생성
    int input;
    Scanner sc = new Scanner(System.in);
    // 사용자 메뉴
    do { do while을 사용. 0을 입력시 이전 메뉴로 돌아갈 수 있도록 함
        menu.displayShoppingMenu(currentCustomer.getIsSuperUser());
        input = sc.nextInt();

        switch (input) { Enhanced switch문을 사용해 메뉴 이동
            case 1 -> {
                productManager.show(); 입력 1: 전체 제품 show & 구매 기능
                displayOrderDecision();
            }
            case 2 -> handleSearchMenu(); 입력 2: 상세 검색 메뉴..
            case 3 -> orderManager.showOrderByCustomer(customerManager, productManager, currentCustomer);
            case 4 -> customerManager.edit(currentCustomer);
            case 5 -> {
                customerManager.remove(currentCustomer.getId(), orderManager);
                return input;
            }
        }

    } while (input != 0);

    return input;
}
```

2. 숫자형 변수에는 숫자만 입력

이름: 하재민

닉네임: 찜찜

주소: 동작구 상도동

나이: 스물여섯

* 잘못된 입력입니다. 숫자를 정확히 입력해주세요.

나이: |

입력값이 숫자가 아님을 확인하는 메소드

```
public static boolean containsOnlyNumbers(String input) {  
    Pattern pattern = Pattern.compile(regex: "[0-9]+$");  
    if(!pattern.matcher(input).matches()){  
        System.out.println("* 잘못된 입력입니다. 숫자를 정확히 입력해주세요.");  
        return false;  
    }  
    return true;  
}
```

메소드 사용 예시 - 회원가입

```
String tempAge = "";  
do{  
    tempAge = StringUtils.printAndGetInput( printMessage: "나이: ");  
}while(!StringUtils.containsOnlyNumbers(tempAge));  
int age = Integer.parseInt(tempAge);
```

do while을 사용해 숫자가 아니라면 재입력 -> 입력 실수 방지

3. 현재 메뉴 위치 표시

[MANAGER MENU] > [1. SHOPPING MENU] > [2. SEARCH MENU]

- 1. 제품 번호
- 2. 브랜드
- 3. 제품명
- 0. 이전 메뉴로

현재 메뉴의 위치 -> 직관적 이동

상세 검색 Menu - 고객 or 관리자에 따라 현재 위치를 다르게 표시

```
public void displaySearchMenu(boolean isSupper){
    System.out.println(str_bar);
    if (isSupper)
        System.out.println("[MANAGER MENU] > [1. SHOPPING MENU] > [2. SEARCH MENU]");
    else
        System.out.println("[SHOPPING MENU] > [2. SEARCH MENU]");
    System.out.println("1. 제품 번호");
    System.out.println("2. 브랜드");
    System.out.println("3. 제품명");
    System.out.println("0. 이전 메뉴로");
    System.out.println(str_bar);
}
```


3. Results

1. 쇼핑 물

1.1 전체 제품

1.2 상세 검색

1.3 주문 확인

1.4 정보 수정

1.5 탈퇴

1.2.1 제품 번호

1.2.2 브랜드

1.2.3 제품명

2. 고객 관리

2.1 등록

2.2 탈퇴

2.3 회원 정보

3. 제품 관리

3.1 등록

3.2 수정

3.3 삭제

3.4 전체 제품

4. 주문 관리

4.1 전체 주문

4.2 고객별 주문

Manager Menu Map

1. 쇼핑물

1.1 전체 제품

1.2 상세 검색

1.3 주문 확인

1.4 정보 수정

1.5 탈퇴

1.2.1 제품 번호

1.2.2 브랜드

1.2.3 제품명

2. 고객 관리

2.1 등록

2.2 탈퇴

2.3 회원 정보

3. 제품 관리

3.1 등록

3.2 수정

3.3 삭제

3.4 전체 제품

4. 주문 관리

4.1 전체 주문

4.2 고객별 주문

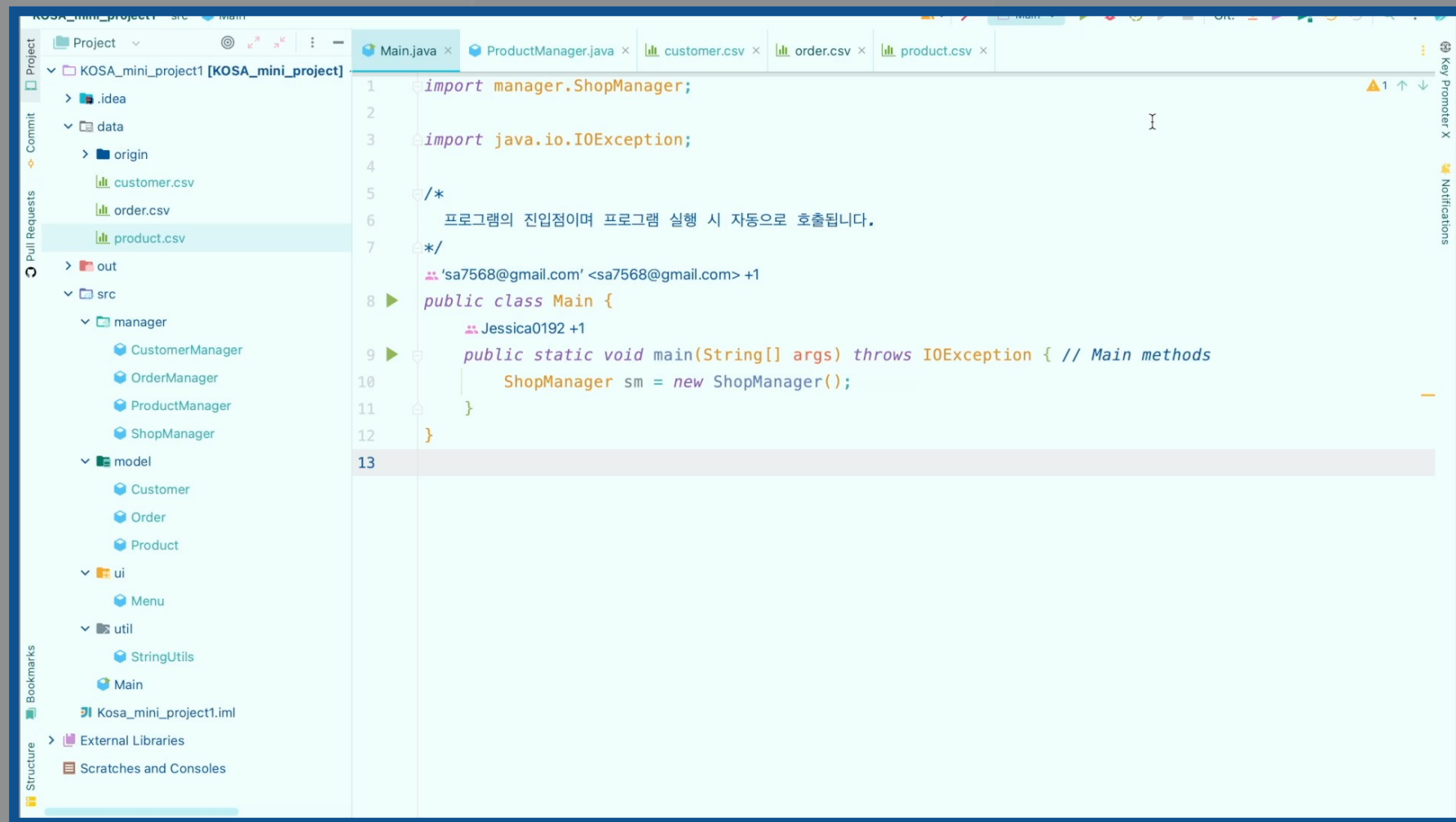
Customer Menu Map

시연 영상

1. 관리자로 회원가입

2. 에어조던1 구매

3. Data 확인



The screenshot shows an IDE window for a project named 'KOSA_mini_project1'. The left sidebar displays the project structure, including folders for 'data', 'out', 'src', 'model', 'ui', and 'util'. The 'src' folder is expanded, showing subfolders like 'manager' and 'model'. The 'manager' folder contains 'CustomerManager', 'OrderManager', 'ProductManager', and 'ShopManager'. The 'model' folder contains 'Customer', 'Order', and 'Product'. The 'ui' folder contains 'Menu'. The 'util' folder contains 'StringUtils' and 'Main'. The 'Main' class is highlighted in the 'util' folder. The main editor area shows the code for 'Main.java'. The code includes imports for 'manager.ShopManager' and 'java.io.IOException'. It contains a comment in Korean: '프로그램의 진입점이며 프로그램 실행 시 자동으로 호출됩니다.' (This is the entry point of the program and is called automatically when the program is executed). Below the comment is a Java class definition for 'Main' with a 'main' method that creates a 'ShopManager' object.

```
1 import manager.ShopManager;
2
3 import java.io.IOException;
4
5 /*
6  프로그램의 진입점이며 프로그램 실행 시 자동으로 호출됩니다.
7  */
8 public class Main {
9     Jessica0192 +1
10     public static void main(String[] args) throws IOException { // Main methods
11         ShopManager sm = new ShopManager();
12     }
13 }
```


4. Conclusion

느낀점

1. 기능의 세부적 구현보다는 전체적인 설계가 더욱 중요

- 자료구조의 선택에 따라 성능이 달라짐
- Class와 Method를 어떤 목적으로 나누는지에 따라 구현의 난이도, 가시성이 달라짐
- 보안을 고려한 접근제어자를 적절히 사용 해야함

2. 팀원과의 협업을 통해 git 버전관리, 분업, 소통의 중요성을 알게됨

발전방향

1. 장바구니 기능 추가
2. Edge case testing을 통한 안정성 향상



Thank You