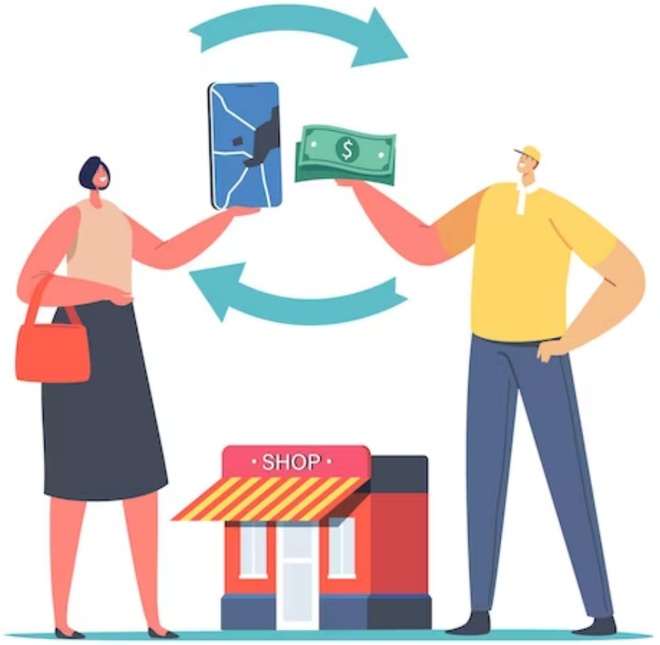
빅데이터 솔루션 S/W 개발자 양성 과정

[평가] - 쇼핑몰 관리 프로그램

이병길

목차



01. 요구사항

문제) 다음 요구사항을 잘 읽고, 조건에 맞는 프로그램을 작성하여 제출하세요

어떤 쇼핑몰의 회원 시스템을 구현하려고 한다.
만들어야 하는 쇼핑몰의 회원 시스템은 다음과 같다.

- 회원은 크게 2가지 유형으로, 판매자와 구매자가 있다.
- 모든 회원은 공통으로 [ID], [비밀번호], [이름]을 가져야 한다.
- 판매자는 공통 속성과 함께 [가게이름]을 가져야 한다.
- 구매자는 공통 속성과 함께 [배송주소]를 가져야 한다.
- 회원가입(생성), 회원탈퇴(삭제), 전체목록, 단일검색, 정렬 메뉴가 있어야 한다.
- 가입 시, ID는 다른 회원과 중복되면 안 된다.
- 검색 시, ID로만 검색하며, 회원 유형을 같이 표시해야 한다.
- 각 회원의 정보는 비밀번호를 제외한 모든 속성을 표시해야 한다.
- 배열 사용 시, 최대 회원 수는 100명으로 설정한다. (리스트는 제한없음)

각 클래스는 다음과 같은 이름을 가져야 한다.

- 회원 : Member
- 판매자 : Seller
- 구매자 : Customer
- 기능제어 : Handler
- 실행코드 : Main

02. Member Class

부모 클래스인 Member 클래스입니다.
저는 추상 클래스로 만들지 않았고 생성자에
ID,Password,name값을 받아 생성하는
생성자를 만들었습니다.

멤버 변수는 ID,passwd,name 모두 중요한
정보이므로 외부에서 바로 접근할 수 없도록
private 접근 지정자를 사용하여 선언했습니다.

getter/setter를 public 으로 만들어 멤버 변수의
값을 지정하거나 가져올 수 있습니다.

```
public  class Member {
  private String id;
  private String passwd;
  private String name;

  public Member(String id, String passwd, String name) {
    this.id = id;
    this.passwd = passwd;
    this.name = name;
  }

  public String getId() {
  public void setId(String id) {
  public String getPasswd() {
  public void setPasswd(String passwd) {
  public String getName() {
  public void setName(String name) {

}
```

03. Seller Class

Member의 자식클래스인 Seller 클래스입니다.
부모클래스의 멤버를 모두 상속 받고 Seller클래스
만의 고유한 특성인 storeName(가게 이름)을
객체를 생성할 때 생성자의 매개변수로 받습니다.

Seller의 입장에서 storeName은 중요한 데이터
이기 때문에 private으로 접근 지시자를 설정했습니다.

getter/setter를 public 으로 만들어
새롭게 추가된 멤버 변수 storeName의
값을 지정하거나 가져올 수 있습니다.

```
public class Seller extends Member {
  private String storeName;

  public Seller(String id, String passwd, String name,String storeName) {
    super(id, passwd, name);
    this.storeName = storeName;
  }

  public String getStoreName() {
  public void setStoreName(String storeName) {

}
```

04. Customer Class

Member의 자식클래스인 Customer 클래스입니다.
부모클래스의 멤버를 모두 상속 받고 Seller클래스
만의 고유한 특성인 address(배송 주소)를
필드에 선언하고
객체를 생성할 때 생성자의 매개변수로 받습니다.

Seller의 입장에서 address는 중요한 데이터
이기 때문에 private으로 접근 지시자를 설정했습니다.

getter/setter를 public 으로 만들어
새롭게 추가된 멤버 변수 address의
값을 지정하거나 가져올 수 있습니다.

```
public class Customer extends Member{
  // 배송 주소
    private String address;

  public Customer(String id, String passwd, String name,String address) {
    super(id, passwd, name);
    this.address=address;
  }

  public String getAddress() {
  public void setAddress(String address) {

}
```

05. Handler와 Main의 기능별 동작

1. Main의 구조

- 먼저, main함수가 있는 Main클래스의 구조를 먼저 살펴보겠습니다.

```
public class Main {
  public static void main(String[] args) {
    Handler handler = new Handler();          // 핸들러 객체 생성
    String id = null;                         // id 값을 받을 변수
    String keyword = null;                    // 검색 키워드를 받을 변수
    Member[] searchedmember = null;           // 검색 후 결과 Member 배열 받을 변수
    Scanner sc = new Scanner(System.in);      // 입력을 받기 위한 Scanner
    int menu = 0;                             // 메뉴 변수
    int res= 0;                               // 회원가입이나 탈퇴의 결과상태를 알려주는 int 변수
```

  - main() 함수내에서 사용되는 지역변수들입니다.

05. Handler와 Main의 기능별 동작

1. Main의 구조

- main() 함수 내에서 사용되는 static함수들입니다.
- selectMenu()함수는 Scanner를 매개변수로 받아서 주메뉴의 순환을 담당합니다.
- selectMemberType()함수도 Scanner를 매개변수로 받아서 회원가입 시 구매자와 판매자의 여부를 확인할 수 있도록 해주는 함수입니다.
- 구매자인 1을 선택하면 1을 리턴하고 판매자인 2를 선택하면 2를 리턴하는 함수입니다.

```java
// 주 메뉴 선택 함수
private static int selectMenu(Scanner sc) {
    System.out.println("1. 회원가입");
    System.out.println("2. 전체 목록");
    System.out.println("3. 회원검색");
    System.out.println("4. 회원탈퇴");
    System.out.println("5. 회원 정렬");
    System.out.println("0. 종료");
    System.out.print("메뉴 선택 >> ");
    int num = Integer.parseInt(sc.nextLine());
    return num;
}

// 회원가입시 멤버 유형 선택 함수
private static int selectMemberType(Scanner sc) {
    System.out.print("회원 유형을 선택하세요 (1.구매자 2.판매자) >> ");
    return Integer.parseInt(sc.nextLine());
}
```

04. Handler와 Main의 기능별 동작

1. Main의 구조

- 주 흐름은 while 문에서 메뉴를 0(종료) 입력받을 때까지 순환하는 구조입니다.

- 1. 회원가입 2.전체 목록 3. id로 회원검색, 4.회원 탈퇴, 5.정렬, 0.프로그램 종료 순으로 메뉴를 구성 했습니다.

- 자세한 코드는 Handler와 함께 다루면서 설명하겠습니다.

```
while(true) {
  menu = selectMenu(sc);                    // 주메뉴 선택 함수로 리턴받은 값이 메뉴
  switch (menu) {
  case 1:
    menu = selectMemberType(sc);
    res = handler.join(menu,sc);
    if(res != 0) {
      System.out.println("회원가입이 정상적으로 처리되었습니다.");
    }else {
      System.out.println("회원가입에 실패하셨습니다.");
    }
    break;
  case 2:
    handler.selectAll();
    break;
  case 3:
    System.out.println("\t\t 아이디로 검색합니다!!");
    System.out.print("검색할 id를 입력하세요 >> ");
    keyword = sc.nextLine();
    searchedmember = handler.searchMember(keyword);
    handler.selectAll(searchedmember);
    break;
  case 4:
    System.out.print("아이디를 정확히 입력하여 탈퇴합니다");
    id = sc.nextLine();
    res = handler.out(id,sc);
    if(res == 1) {
      System.out.println("\t\t 회원 탈퇴 처리하였습니다.");
    }else if(res == 0){
      System.out.println("\t\t 회원 탈퇴를 취소하였습니다.");
    }else {
      System.out.println("\t\t 회원 탈퇴 실패하였습니다.");
    }
    break;
  case 5:
    handler.sort();
```

04. Handler와 Main의 기능별 동작

2. 회원가입 설명- 아이디 중복 체크

```
case 1:
  menu = selectMemberType(sc);
  res = handler.join(menu,sc);
  if(res != 0) {
    System.out.println("회원가입이 정상적으로 처리되었습니다.");
  }else {
    System.out.println("회원가입에 실패하셨습니다.");
  }
  break;
```

- main을 통해 handler의 join(menu,sc)을 호출합니다. 여기서 menu는 판매자와 구매자를 구분하기 위한 메뉴값(1,2)입니다.
- 정상가입이면 1, 정상적으로 가입되지 않았다면 0을 반환하기 때문에 조건에 따라 결과를 다르게 출력합니다.
- 먼저, id를 입력 받는데 while문으로 Id값이 중복값이라면 id_check()라는 함수로 계속해서 중복인지 아닌지를 체크합니다.

```
// 회원 가입
public int join(int menu, Scanner sc) {
  int res = 0;
  boolean isInvalidID = false;        // while문을 타기 위해 초기값 false
  String id = null;

  System.out.println();
  while (!isInvalidID) {
    System.out.print("아이디를 입력하세요 >> ");
    id = sc.nextLine();
    isInvalidID = id_check(id); // 중복체크
  }
```

```
// 아이디 중복 체크
public boolean id_check(String id) {
  for (Member m : members) {
    if (m != null && m.getId().equals(id)) {
      System.out.println("중복된 아이디 입니다. 다시입력하세요");
      return false;
    }
  }
  return true;
}
```

04. Handler와 Main의 기능별 동작

2. 회원가입 설명- 나머지 정보 입력받기

- 먼저, 비밀번호를 입력받아서 join의 지역변수 passwd에 저장합니다.

- 이름을 입력받아서 join의 지역변수 name에 저장합니다.

- 그리고 구매자와 판매자를 조건문으로 구분하여 구매자면 배송 주소를 판매자면 가게 이름을 입력받습니다.

- Handler의 멤버배열members에서 비어있는 배열 영역을 찾아서 지금까지 입력받은 정보들로 새로운 객체를 생성(new)해서 담아줍니다.
- 담았으면 마지막으로 res에 1값을 주고 리턴합니다

```
System.out.print("비밀번호를 입력하세요 >> ");
String passwd = sc.nextLine();

System.out.print("이름을 입력하세요 >> ");
String name = sc.nextLine();

if (menu == 1) {
  System.out.print("배송 주소를 입력해주세요 >> ");
  String address = sc.nextLine();
  for (int i = 0; i < members.length; i++) {
    if (members[i]==null) {
      members[i] = new Customer(id, passwd, name, address);
      res +=1;
      break;
    }
  }
} else if (menu == 2) {
  System.out.print("가게 이름을 입력해주세요 >>");
  String storeName = sc.nextLine();
  for (int i = 0; i < members.length; i++) {
    if (members[i] == null) {
      members[i] = new Seller(id, passwd, name, storeName);
      res +=1;
      break;
    }
  }
}
return res;
}
```

04. Handler와 Main의 기능별 동작

3. 전체 목록 출력하기 - 구매자 판매자 배열 만들기

```
case 2:
  handler.selectAll();
  break;
```

- main은 간단하게 handler에서 selectAll() 함수를 호출합니다.
- 먼저, 전체 회원이 담겨있는 배열 members에서 구매자와 판매자 배열을 만들어줍니다.
- 이 때 나누는 기준은 객체의 클래스 이름으로 getClass로 이름을 받아서 Customer있으면 구매자 나머지는 Seller라는 것을 알 수 있어서 구분합니다.
- 구분하여 배열의 크기를 잡은 다음 배열을 만들어줍니다. customers, sellers

```
public void selectAll() {
  System.out.println("\t\t 전체 회원 정보 ");
  int cIndex = 0;      // 구매자 index
  int sIndex = 0;      // 판매자 index

  for (int i = 0; i < members.length; i++) {
    if (members[i] != null) {
      if (members[i].getClass().toString().split("\\.")[1].equals("Customer")) {
        cIndex++;
      } else {                                                    // Seller
        sIndex++;
      }
    } // end of members null
  }
  Customer[] customers = new Customer[cIndex];
  Seller[] sellers = new Seller[sIndex];
  cIndex = 0;
  sIndex = 0;
```

04. Handler와 Main의 기능별 동작

3. 전체 목록 출력하기 - 구매자 판매자 배열 만들기

- Seller와 Customer는 Member의 자식클래스이므로 다운캐스팅을 통하여 members의 배열중 조건에 만족하는 배열을 sellers, customers 배열에 담아줍니다.

- 구매자가 1명 이상이면 반복문을 통해서 출력을 합니다.

```
for (int i = 0; i < members.length; i++) {
  if (members[i] != null) {
    if (members[i].getClass().toString().split("\\.")[1].equals("Customer")) {
      customers[cIndex] = (Customer) members[i];
      cIndex++;
    } else {
      sellers[sIndex] = (Seller) members[i];
      sIndex++;
    }
  } // end of members null
}

System.out.println(cIndex==0 ? "":"[구매자 정보]");
for (int i = 0; i < customers.length; i++) {
  System.out.println("ID : " + customers[i].getId());
  System.out.println("Passwd : " + customers[i].getPasswd().replaceAll(".", "*"));
  System.out.println("Name : " + customers[i].getName());
  System.out.println("배송주소 : " + customers[i].getAddress());
  System.out.println();
}

System.out.println(sIndex==0 ? "":"[판매자 정보]");
for (int i = 0; i < sellers.length; i++) {
  System.out.println("ID : " + sellers[i].getId());
  System.out.println("Passwd : " + sellers[i].getPasswd().replaceAll(".", "*"));
  System.out.println("Name : " + sellers[i].getName());
  System.out.println("배송주소 : " + sellers[i].getStoreName());
  System.out.println();
}
}
```

04. Handler와 Main의 기능별 동작

4. 회원 검색

```
case 3:
  System.out.println("\t\t 아이디로 검색합니다!!");
  System.out.print("검색할 id를 입력하세요 >> ");
  keyword = sc.nextLine();
  searchedmember = handler.searchMember(keyword);
  handler.selectAll(searchedmember);
  break;
```

- 3번 메뉴를 선택하면 회원 검색 메뉴를 선택한 것인데 먼저 검색할 키워드를 입력받습니다.
- 입력받은 키워드를 인자값을 받는 searchMember라는 함수를 handler에서 호출을 하고 searchedMember라는 Member 배열 변수가 받습니다.
- 이 배열의 내용을 handler의 selectAll()의 인자로 받아서 모두 출력하는 형태입니다.

```
// 검색
public Member[] searchMember(String keyword) {
  int index = 0;
  for(int i=0;i<members.length;i++) {
    if(members[i] != null) {
      if(members[i].getId().contains(keyword))
      index ++;
    }
  }

  Member[] searchedArray = new Member[index];
  index = 0;

  for(int i=0;i<members.length;i++) {
    if(members[i] != null ) {
      if(members[i].getId().contains(keyword)) {
        searchedArray[index] = members[i];
        index++;
      }
    }
  }

  return searchedArray;
}
```

- String의Contains()함수로 keyword를 포함하는 ID를 가진 모든 멤버를 searchedArray변수에 담아 반환합니다.

04. Handler와 Main의 기능별 동작

5. 회원 탈퇴

```
case 4:
  System.out.print("아이디를 정확히 입력하여 탈퇴합니다");
  id = sc.nextLine();
  res = handler.out(id,sc);
  if(res == 1) {
    System.out.println("\t\t 회원 탈퇴 처리하였습니다.");
  }else if(res == 0){
    System.out.println("\t\t 회원 탈퇴를 취소하였습니다.");
  }else {
    System.out.println("\t\t 회원 탈퇴 실패하였습니다.");
  }
  break;
```

- 4번 메뉴를 선택하면 회원 탈퇴 메뉴입니다.
- 탈퇴 메서드를 res에 담아서 비교하는데
- 1이면 탈퇴 성공, 2이면 탈퇴 취소, 그 외의 값은 탈퇴 실패입니다.
- 기입한 id가 members배열안에 있는 member의 ID와 정확히 일치하는 것이 있다면 삭제를 할 수 있습니다.

```
// 탈퇴
public int out(String id, Scanner sc) {
  int res = 3;
  for(Member m : members) {
    if(m != null && m.getId().equals(id)) {
      System.out.println();
      System.out.print("정말 탈퇴 하시겠습니까? (Y or N) >>");
      char ch = sc.next().charAt(0);                    // 첫 글자가 y
      sc.nextLine();                                    // 버퍼 비우기
      if(ch==89 || ch==121) {                           // y, Y, Yes
        m = null;
        res = 1;
        break;
      }else if(ch == 78 || ch == 110) {                 // n, N, No
        res = 0;
        break;
      }else {
        break;
      }
    }
  }
  return res;
}
```

- 이 때, 진짜 탈퇴할 것인지 재 확인을 하는데 y,Y,yes 을 누르면 삭제를 하고 1을 반환,
- n, N, No를 입력하면 삭제를 취소하고 0을 반환합니다.

04. Handler와 Main의 기능별 동작

6. 회원 목록 정렬

```
case 5:
  handler.sort();
  break;
```

- 5번 메뉴를 선택하여 정렬을 합니다.
- 정렬은 compareTo로 Member객체의 ID값을 오름차순으로 정렬하였습니다.
- 마지막으로 selectAll()로 정렬된 결과를 출력합니다.

```
// 정렬
public void sort() {
  Arrays.sort(members, new Comparator<Member>() {
    @Override
    public int compare(Member o1, Member o2) {
      if(o1 != null && o2 != null) {
        return o2.getId().compareTo(o1.getId());      // 오름차순
      }
      return 0;
    }
  });
  selectAll();
}
```