김명호/김승민/김은비/김재준/정소연

(A-3)

나자바봐라

목차

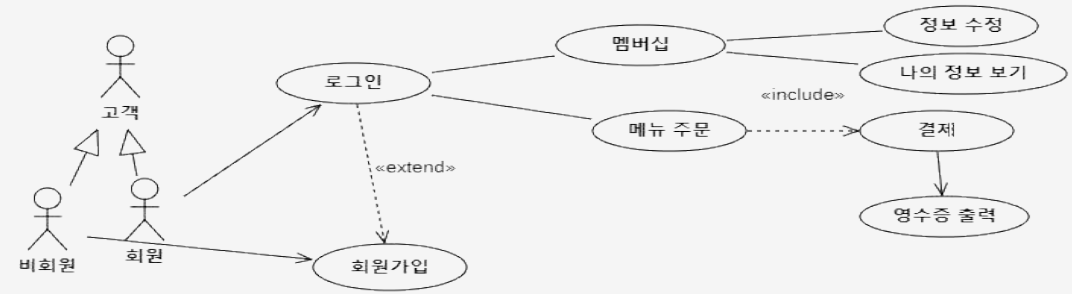
01

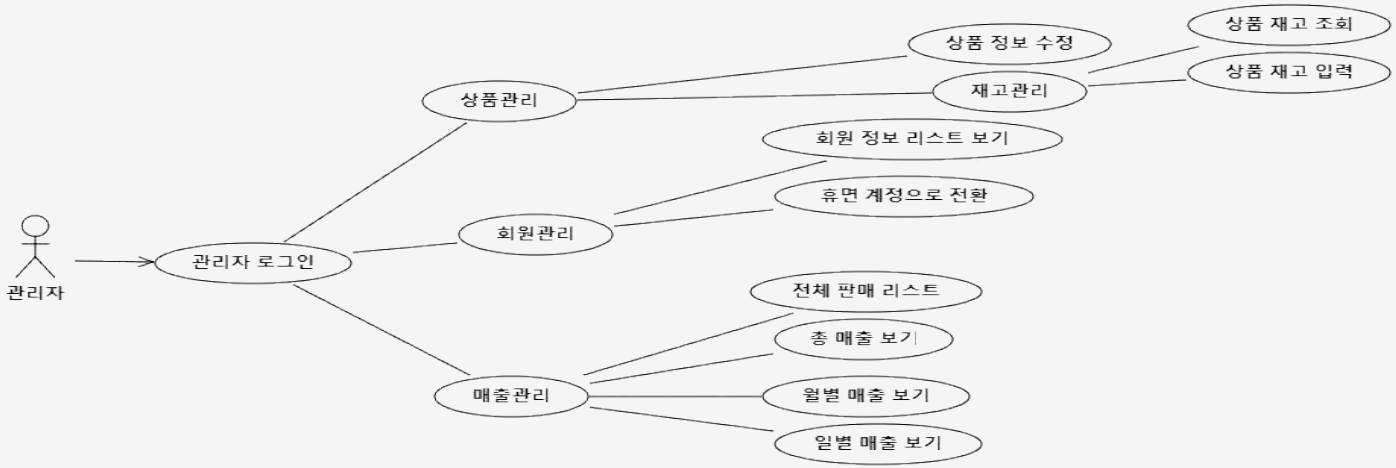
프로그램 흐름도

“

아이스크림 매장 관리 및 주문 프로그램

프로그램 흐름도


고객
비회원
회원
로그인
멤버십
정보 수정
나의 정보 보기
«include»
메뉴 주문
결제
영수증 출력
«extend»
회원가입

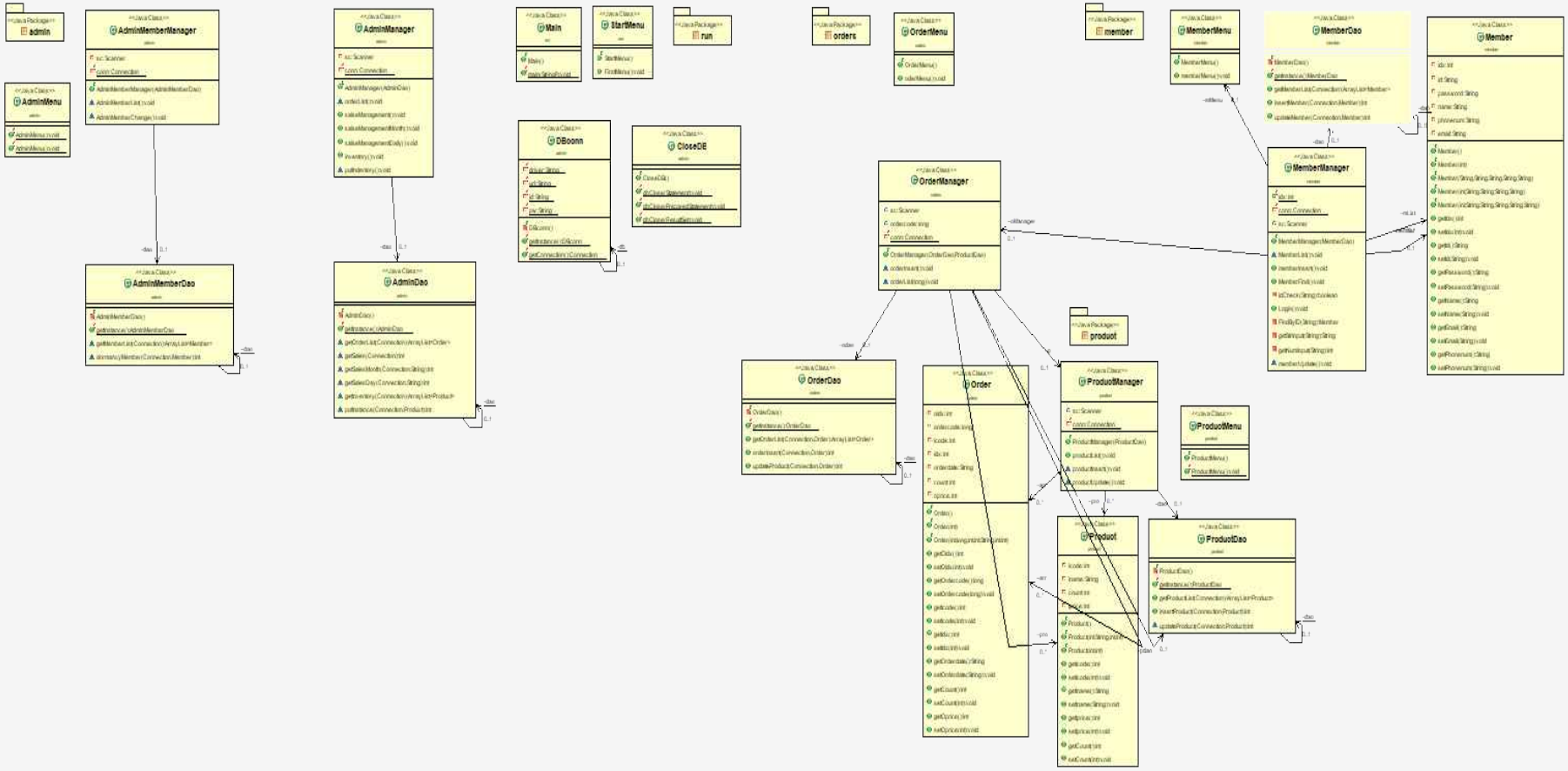

관리자
관리자 로그인
상품관리
상품 정보 수정
재고관리
상품 재고 조회
상품 재고 입력
회원관리
회원 정보 리스트 보기
휴면 계정으로 전환
매출관리
전체 판매 리스트
총 매출 보기
월별 매출 보기
일별 매출 보기

02

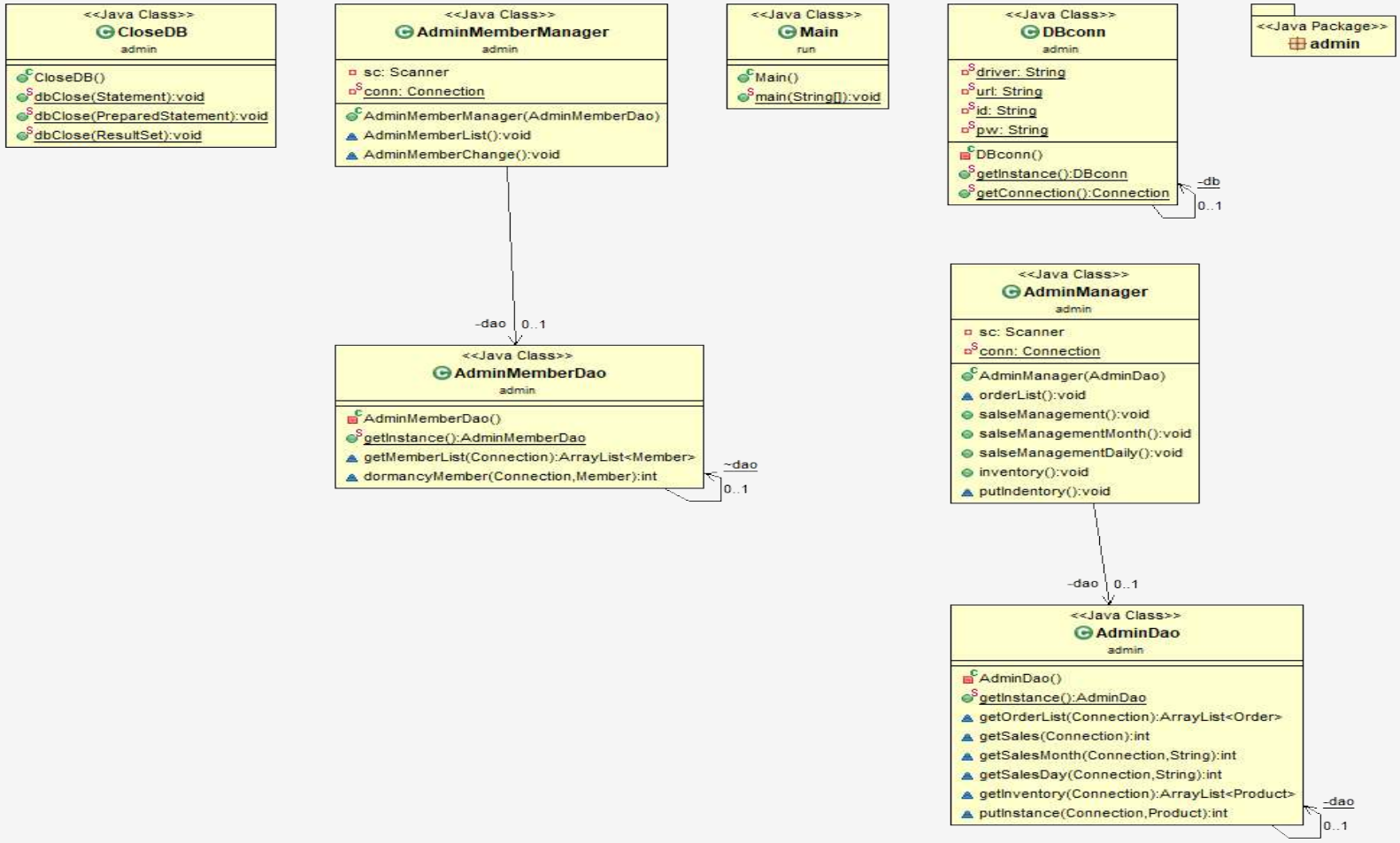
설계

클래스 다이어그램 및 ERD

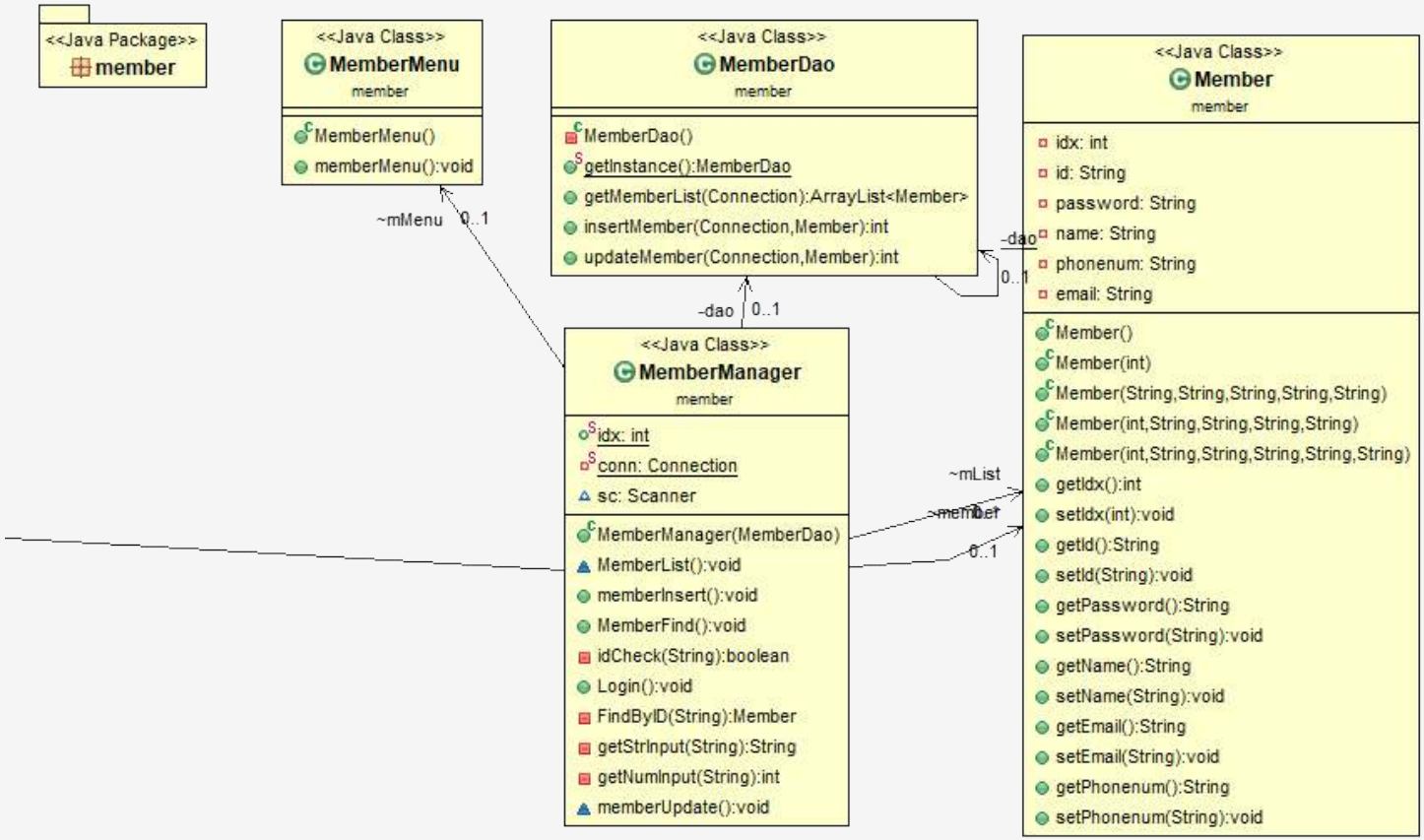
설계_클래스 다이어그램


admin
AdminMenu
AdminMemberManager
AdminMemberDao
AdminManager
AdminDao
Main
StartMenu
run
DBconn
CloseDB
orders
OrderMenu
OrderManager
OrderDao
Order
member
MemberMenu
MemberDao
MemberManager
Member
produot
ProductManager
ProductMenu
Produot
ProductDao

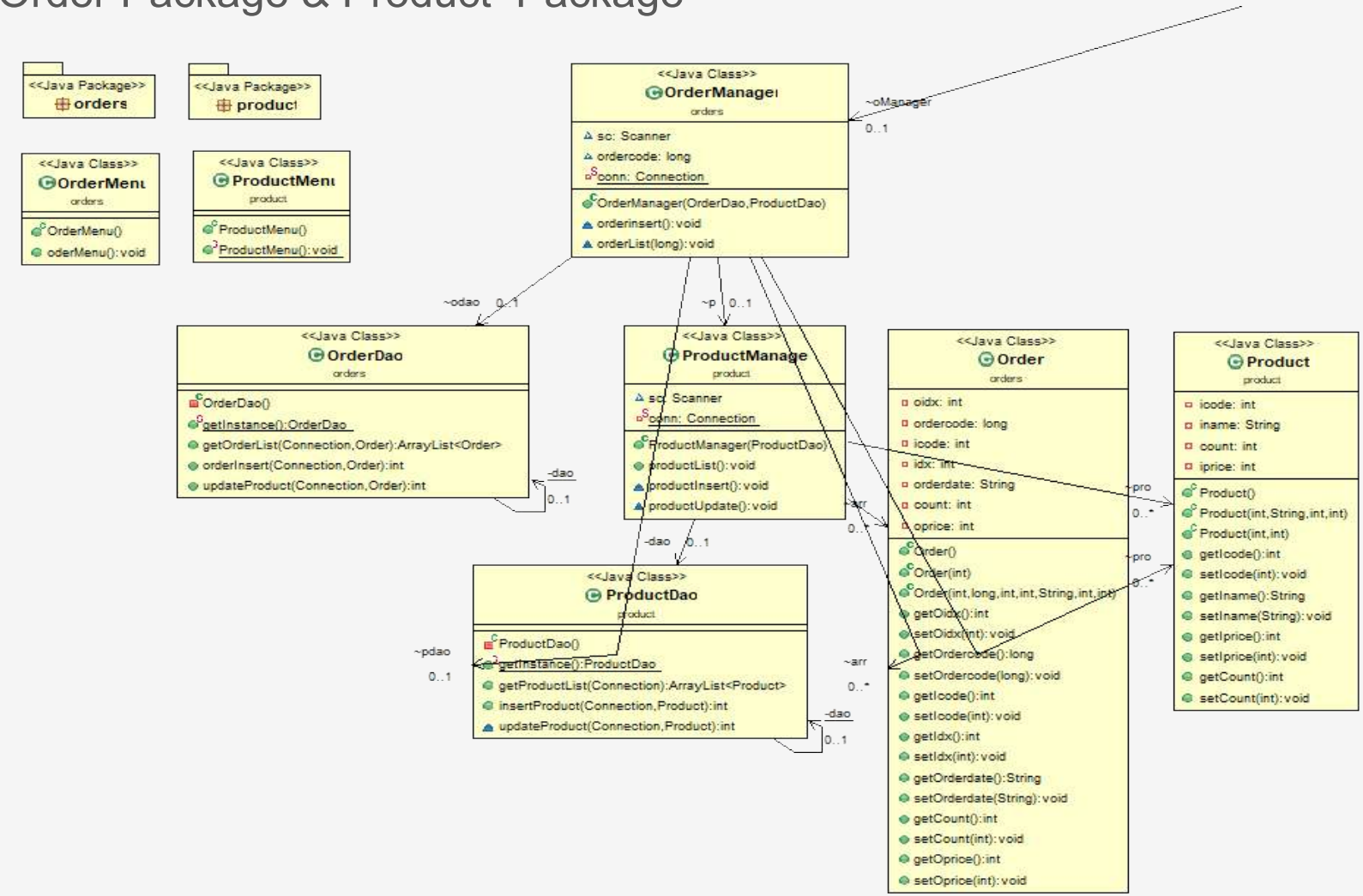
설계_클래스 다이어그램
Admin Package



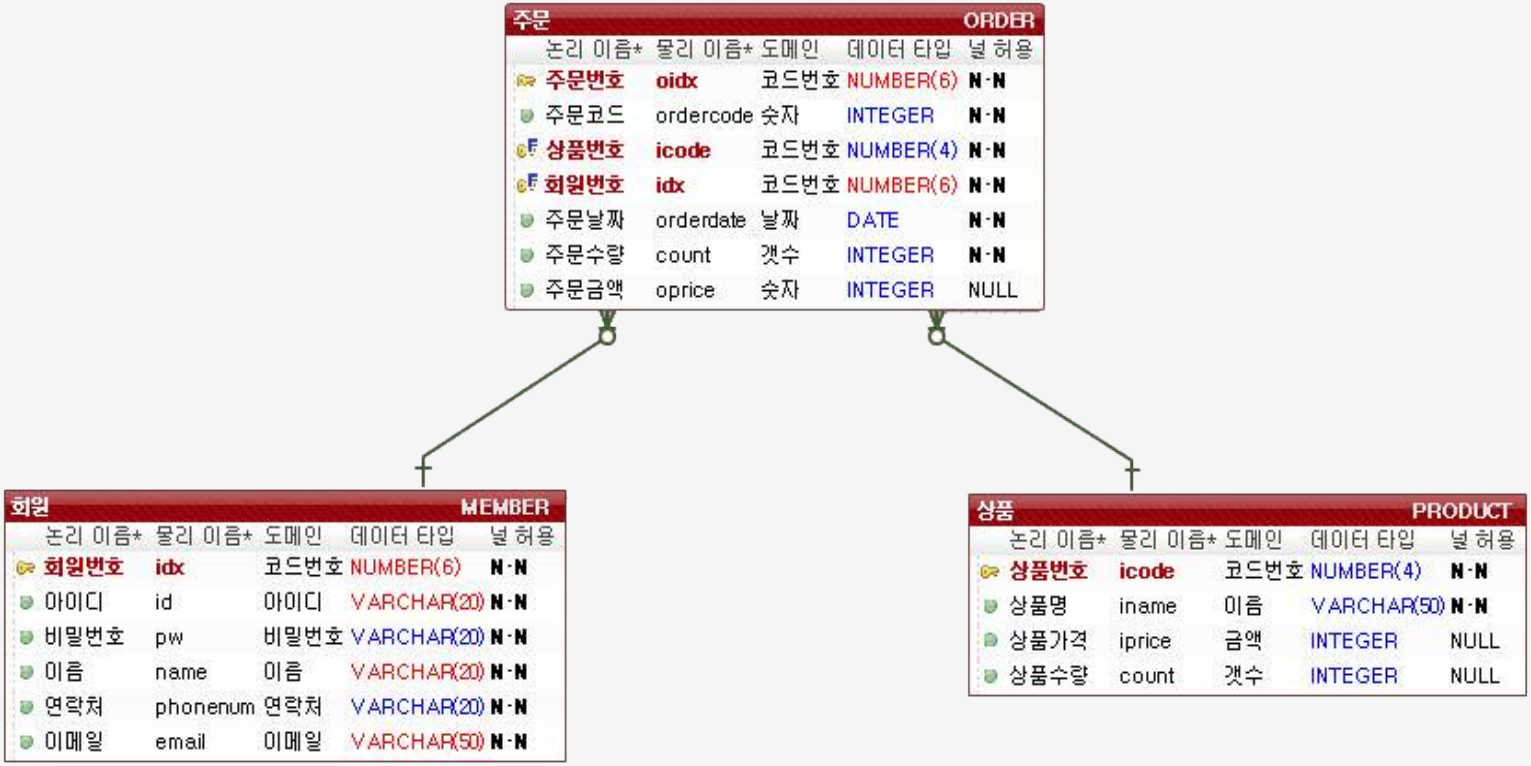
설계_클래스 다이어그램
Member Package



설계_클래스 다이어그램
Order Package & Product Package



설계_ERD


주문 ORDER
논리 이름* 물리 이름* 도메인 데이터 타입 널 허용
주문번호 oidx 코드번호 NUMBER(6) N·N
주문코드 ordercode 숫자 INTEGER N·N
상품번호 icode 코드번호 NUMBER(4) N·N
회원번호 idx 코드번호 NUMBER(6) N·N
주문날짜 orderdate 날짜 DATE N·N
주문수량 count 갯수 INTEGER N·N
주문금액 oprice 숫자 INTEGER NULL
회원 MEMBER
논리 이름* 물리 이름* 도메인 데이터 타입 널 허용
회원번호 idx 코드번호 NUMBER(6) N·N
아이디 id 아이디 VARCHAR(20) N·N
비밀번호 pw 비밀번호 VARCHAR(20) N·N
이름 name 이름 VARCHAR(20) N·N
연락처 phonenum 연락처 VARCHAR(20) N·N
이메일 email 이메일 VARCHAR(50) N·N
상품 PRODUCT
논리 이름* 물리 이름* 도메인 데이터 타입 널 허용
상품번호 icode 코드번호 NUMBER(4) N·N
상품명 iname 이름 VARCHAR(50) N·N
상품가격 iprice 금액 INTEGER NULL
상품수량 count 갯수 INTEGER NULL

03

기능별 소개

클래스 및 메서드 기능 설명
코드 작성 의도 설명

- JDBC

DB Connection

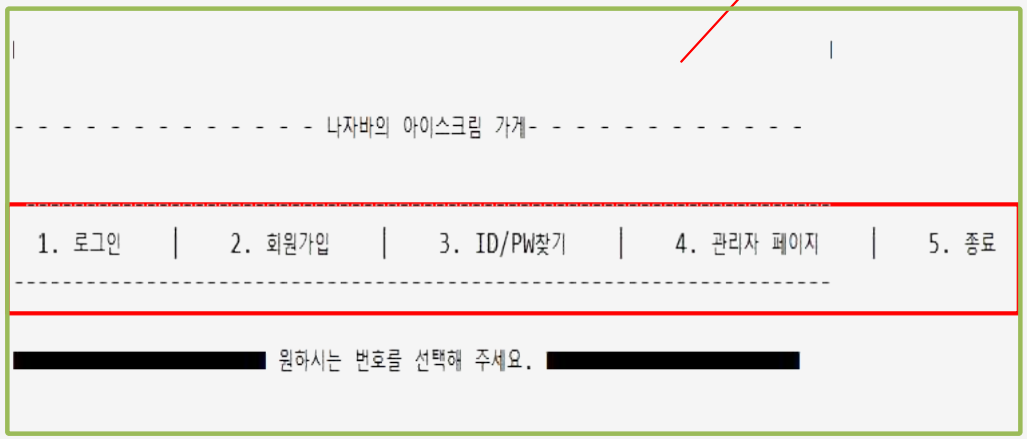
```java
public class DBconn {
    private static String driver = "oracle.jdbc.driver.OracleDriver";
    private static String url = "jdbc:oracle:thin:@localhost:1521:xe";
    private static String id = "hr";
    private static String pw = "tiger";
    private static DBconn db = new DBconn();

    private DBconn() {}

    public static DBconn getInstance() {
        return db;
    }

    public static Connection getConnection() {
        Connection conn = null;

        try {
            Class.forName(driver);
            conn = DriverManager.getConnection(url, id, pw);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return conn;
    }
}
```

DB Connection Close

```java
public class CloseDB {

    public static void dbClose(Statement statement){
        if (statement != null){
            try {
                statement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public static void dbClose(PreparedStatement preparedStatement){
        if (preparedStatement != null){
            try {
                preparedStatement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public static void dbClose(ResultSet resultSet){
        if (resultSet != null){
            try {
                resultSet.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

- 메인 페이지

관리자 페이지 메뉴




```java
try {
    choice= Integer.parseInt(sc.next());
    if(choice<1 || choice>5 ) {
        throw new Exception();
    }

    switch(choice) {
    case 1 :
        System.out.println();
        System.out.println(" ▶ ▶ 로그인을 시작합니다.\n ");

        mManager.Login();
        break;
    case 2 :
        System.out.println();
        System.out.println(" ▶ ▶ 회원가입을 시작합니다.\n");
        mManager.memberInsert();
        break;
    case 3 :
        System.out.println(" ▶ ▶ ID/PW찾기를 시작합니다.\n");
        mManager.MemberFind();
        break;
    case 4 :
        admMenu.AdminMenu();
        break;
    case 5 :
        System.out.println(" ▶ ▶ 시스템을 종료합니다.");
        System.out.println("감사합니다. ");
        System.exit(4);
    }
} catch (Exception e) {
    System.out.println("※ 잘못입력하셨습니다. 1,2,3,4번 중 하나를 선택해주세요. \n");
}
```

- 회원가입

```java
// 가입할때 아이디 중복 체크
private boolean idCheck(String id) {
    boolean check = true;
    Member member = FindByID(id);
    if(member == null) {
        check = false;
        return check;
    }
    return check;
}
```

```
▶ ▶ 회원가입을 시작합니다.

ID :
seungmin2
PW :
1234
PW CONFIRM :
1234
NAEM :
김승민
PHONE :
010-0000-0000
EMAIL :
email@email.com
seungmin2님 가입을 축하드립니다.
```

```java
// 회원가입
public void memberInsert() {

    try {
        mList = dao.getMemberList(conn);
        while(true) {

            String id = getStrInput("ID : ");
            if(idCheck(id)) {
                System.out.println("※ 중복된 id입니다.\n");
                continue;
            }
            String pw = getStrInput("PW : ");
            String pw2 = getStrInput("PW CONFIRM : ");
            if(!(pw.equals(pw2))) {
                System.out.println("※ 비밀번호를 잘못입력하셨습니다. 다시입력하세요.\n");
                continue;
            }
            String name = getStrInput("NAEM : ");
            String phone = getStrInput("PHONE : ");
            String email = getStrInput("EMAIL : ");

            if (pw.equals(pw2)) {
                Member mem = new Member(id, pw, name, phone, email);
                dao.insertMember(conn, mem);
                System.out.println(id + "님 가입을 축하드립니다.\n");
                break;
            } else {
                System.out.println("※ 비밀번호를 확인해주세요.\n");
            }
        }
    } catch (Exception e) {
        System.out.println("※ 잘못입력하셨습니다. ");
    }

}
```

- 로그인

```java
// 로그인 구현 기능
public void Login() {
    try {
        String id = getStrInput("id : ");
        String password = getStrInput("pw : ");

        mList = dao.getMemberList(conn);

        Member member = FindByID(id);

        if(member == null) {
            System.out.println("※ 등록되지 않은 ID입니다.");

        } else if(member.getPassword().equals(password)) {
            System.out.println("☞ [" + member.getId() + "]님께서 로그인 하셨습니다.\n");
            idx = member.getIdx();
            mMenu.memberMenu();

        } else {
            System.out.println("※ 비밀번호가 틀렸습니다.");
        }

    } catch (Exception e) {
        System.out.println("※ 잘못입력하셨습니다. ");

    }
}
```

```java
// 해당 아이디를 전체회원리스트에서 비교,확인 하는 메소드
private Member FindByID(String id) {
    for(Member memberDTO : mList) {
        if(memberDTO.getId().equals(id)) {
            return memberDTO;
        }
    }
    return null;
}
```

```
▶ ▶ 로그인을 시작합니다.

id :
seungmin2
pw :
1234
☞ [seungmin2]님께서 로그인 하셨습니다.

[1]회원정보수정 [2]회원정보보기 [3]주문 [4]로그아웃
```

- 로그인_마이페이지

회원 정보 수정

MemberDao

```java
// 로그인한 회원의 정보 수정
public int updateMember(Connection conn, Member member) {
    int result = 0;
    PreparedStatement pstmt = null;

    try {

        String sql = "update member set pw=?, name=?, phonenum=?, email=? "
                    + "where idx=?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, member.getPassword());
        pstmt.setString(2, member.getName());
        pstmt.setString(3, member.getPhonenum());
        pstmt.setString(4, member.getEmail());
        pstmt.setInt(5, member.getIdx());

        result = pstmt.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        CloseDB.dbClose(pstmt);
    }
    return result;
}
```

```java
// 로그인한 회원의 회원정보 수정
void memberUpdate() {

    try {
        while(true) {
            System.out.println(" ▶ ▶ 회원정보를 수정합니다.\n");

            String pw = getStrInput("수정하실 패스워드 : ");
            String name = getStrInput("수정하실 이름 : ");
            String phone = getStrInput("수정하실 핸드폰번호 : ");
            String email = getStrInput("수정하실 메일 : ");


            System.out.println(" ▶ ▶ 입력한 사항이 모두 맞습니까? 예(1) 아니오(2)\n");
            int input = Integer.parseInt(sc.nextLine());
            System.out.println();
            if(input == 1) {
                System.out.println("☞ 수정이 완료되었습니다.\n");
                Member member = new Member(idx, pw, name, phone, email);
                int result = dao.updateMember(conn, member);
                break;
            } else if(input == 2) {
                System.out.println("※ 메인으로 이동\n");
                break;
            } else {
                System.out.println("※ 잘못 누르셨습니다. 초기 메뉴로 이동합니다.\n");
                break;
            }
        }
    } catch (Exception e) {
        System.out.println("※ 잘못입력하셨습니다. ");

    }
}
```

- 로그인_마이페이지

나의 회원 정보

MemberDao

```java
//모든 회원의 정보 읽기
public ArrayList<Member> getMemberList(Connection con){

    ArrayList<Member> list = null;

    Statement stmt = null;
    ResultSet rs = null;
    String sql = "select * from member";

    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(sql);

        list = new ArrayList<>();

        while(rs.next()) {
            list.add(new Member(rs.getInt(1), rs.getString(2),
                    rs.getString(3),rs.getString(4), rs.getString(5),
                    rs.getString(6)));
        }

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        CloseDB.dbClose(rs);
        CloseDB.dbClose(stmt);
    }
    return list;
}
```

```java
//나의 정보 보기
void MemberList() {

    try {

        mList = dao.getMemberList(conn);

        System.out.println("---------------------- 나의 정보 보기 ----------------------");
        System.out.println("------------------------------------------------------------");
        for (int i = 0; i < mList.size(); i++) {
            if(idx==mList.get(i).getIdx()) {
                System.out.println("나의 ID : " + mList.get(i).getId());
                System.out.println("나의 PW : " + mList.get(i).getPassword());
                System.out.println("나의 이름 : " + mList.get(i).getName());
                System.out.println("나의 핸드폰 : " + mList.get(i).getPhonenum());
                System.out.println("나의 이메일 : " + mList.get(i).getEmail());
                System.out.println("------------------------------------------------------
            }
        }

    } catch (Exception e) {
        System.out.println("※ 잘못입력하셨습니다. ");
    }
}
```

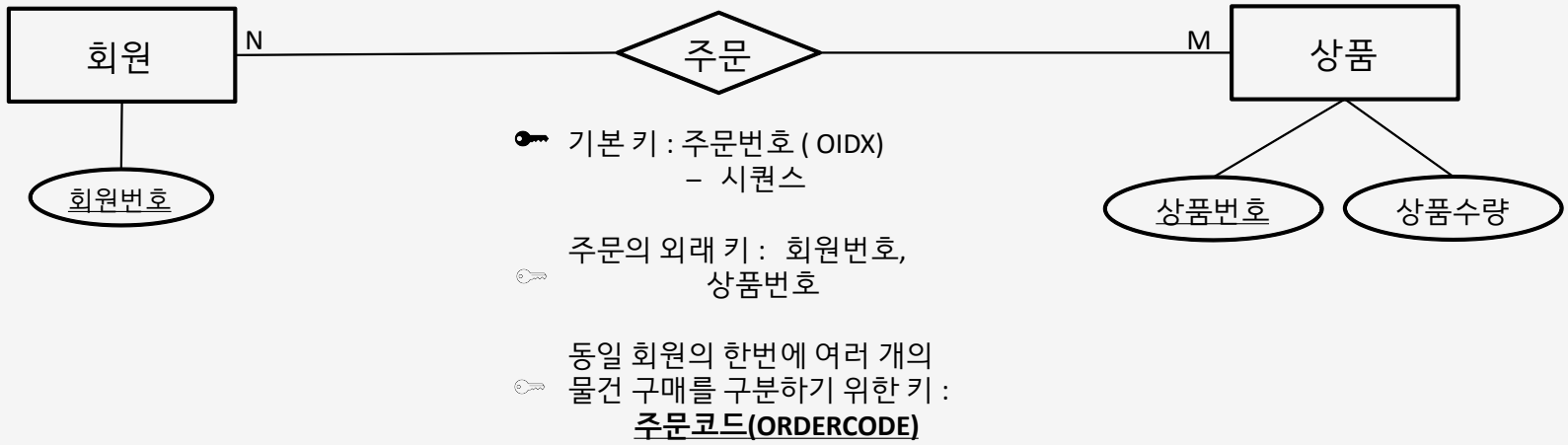
- ID/PW 찾기

```
원하시는 번호를 선택해 주세요.
3
 ▶ ▶ ID/PW찾기를 시작합니다.

회원 이름 :
나자바봐라
회원 이메일 :
qwe@qwe.com
------------------------------------------------------------
☞ [나자바봐라]님의 ID : seungmin2
☞ [나자바봐라]님의 PW : 12345
------------------------------------------------------------
```

```java
// 회원아이디/비번찾기
    public void MemberFind() {

        try {

            mList = dao.getMemberList(conn);
            String name = getStrInput("회원 이름 : ");
            String email = getStrInput("회원 이메일 : ");

            int cnt = 0;
            for (int i = 0; i < mList.size(); i++) {
                if(name.equals(mList.get(i).getName()) && email.equals(mList.get(i).getEmail())) {
                    cnt++;
                    System.out.println("------------------------------------------------------------");
                    System.out.println("☞ [" + name + "]님의 ID : " + mList.get(i).getId());
                    System.out.println("☞ [" + name + "]님의 PW : " + mList.get(i).getPassword());
                    System.out.println("------------------------------------------------------------");
                    break;
                }
            } if(cnt == 0) {
                System.out.println("회원정보가 틀렸습니다.");
            }

        } catch (Exception e) {
            System.out.println("※ 잘못입력하셨습니다. ");
        }
    }
```

주문하기 – 흐름 다시 살펴보기

E-R 다이어그램



문제점

- 한 건의 주문마다 주문번호는 시퀀스로 하나씩 올라가고 한 회원이 여러 개의 물건을 구매하면 한번의 주문을 구별 할 수 없다. → 회원번호로 구별하자 !
- 회원번호로 한 개의 주문을 구별시에 한 회원의 로그아웃하고 다시 로그인 한 경우, 이전에 결제한 주문까지 같이 출력된다. → 동일회원의 한번의 주문에 대한 키 값을 부여 !

```java
public class OrderMenu {

    public void orderMenu() {
        Scanner sc= new Scanner(System.in);
        OrderManager oManager = new OrderManager(OrderDao.getInstance(), ProductDao.getInstance());


        System.out.println("▶ ▶ 주문을 시작합니다.\n ");
        while (true) {
            System.out.println(" [1]주문하기    [0]돌아가기 ");
            int input = sc.nextInt();
            System.out.println();
            switch (input) {
            case 1:
                oManager.orderInsert();
                break;
            case 0:
                return;
            default :
                System.out.println("※ 잘못입력하셨습니다.");
                continue;
            }
        }
    }
}
```

```
▶ ▶ 주문을 시작합니다.

 [1]주문하기    [0]돌아가기
1

                    MENU
-------------------------------------------------------------------
상품번호     상품이름              상품가격    상품 갯수
-------------------------------------------------------------------
1          바닐라 아이스크림       2000     100
2          초코 아이스크림         2500     100
3          딸기 아이스크림         2500     100
4          바나나 아이스크림       2700     100
5          커피 아이스크림         2700     100
6          민트 아이스크림         2800     100
7          요거트 아이스크림       3000     100

▶ ▶ 주문하실 메뉴 번호를 선택해주세요.
1
▶ ▶주문 수량을 선택해주세요.
5
 ☞ 장바구니 담기 완료 !
```

ProductDao

```java
// 상품테이블에서 모든 상품 정보 읽어오기
   public ArrayList<Product> getProductList(Connection conn) {

      ArrayList<Product> list = null;
      Statement stmt = null;
      ResultSet rs = null;

      try {
         stmt = conn.createStatement();
         String sql = "select * from product order by icode";

         rs = stmt.executeQuery(sql);

         list = new ArrayList<>();

         while (rs.next()) {
            Product d = new Product(rs.getInt(1),  rs.getString(2), rs.getInt(3), rs.getInt(4));
            list.add(d);
         }
      } catch (SQLException e) {
         e.printStackTrace();
      } finally {
          CloseDB.dbClose(rs);
          CloseDB.dbClose(stmt);
      }
      return list;
   }
```

ProductManager

```java
//상품 테이블에서 읽어온 정보로 메뉴판 보여주기
public void productList() {

    try {

        pro = dao.getProductList(conn);

        System.out.println("                    MENU                    ");
        System.out.println("-------------------------------------------------------------------");
        System.out.println("상품번호 \t 상품이름 \t\t 상품가격 \t 상품 갯수");
        System.out.println("-------------------------------------------------------------------");


        for (Product p : pro) {
            System.out.printf("%d \t %s \t %d \t %d \n", p.getIcode(), p.getIname(),
                                                        p.getIprice(), p.getCount());
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

주문하기

```java
// 주문 테이블에 주문항목 추가하기
void orderInsert() {

    arr.clear(); //새로운 주문시에 ArrayList에 담긴 내역 초기화 (장바구니초기화)

    try {

        conn.setAutoCommit(false); //트랜잭션 시작
        pro = pdao.getProductList(conn);

        while (true) {
            order = new Order(MemberManager.idx);
            //로그인한 회원의 회원코드를 주문테이블에 넣기위해 사용.
            pManager.productList();
            System.out.println();
            System.out.println("▶ ▶ 주문하실 메뉴 번호를 선택해주세요.");
            int a = sc.nextInt();
            order.setIcode(a);
            System.out.println("▶ ▶주문 수량을 선택해주세요.");
            int b = sc.nextInt();
            order.setCount(b);

            for (int i = 0; i < pro.size(); i++) {
                if(pro.get(i).getIcode() == a) {
                    order.setOprice(pro.get(i).getIprice()*b);
                    break;
                }
            }

            arr.add(order); // 주문한 내역을 ArrayList에 담기

            System.out.println("  ☞ 장바구니 담기 완료 ! \n");

            System.out.println(" [1]  계속 주문하기 ,  [2] 결제하기  , [3] 돌아가기 ");
            String input = sc.next();

            if (input.equals("1")) {
                System.out.println();
                continue;
            } else if (input.equals("2")) {
                System.out.println();
                break;
            }else if (input.equals("3")) {
                arr.clear(); //돌아갈때 담았던 ArrayList 초기화
                System.out.println("이전으로 돌아갑니다.");
                return;

            } else {
                System.out.println("※ 잘못입력하셨습니다.");
                return;
            }
        }
```

ArrayList<Product> pro;

ArrayList<Order> arr;

회원이 입력한 메뉴번호와 같은 번호의 상품을 ArrayList에서 찾아,
해당 상품의 상품금액에 회원이 주문한 수량만큼 곱해 그 값을
주문객체의 주문금액에 넣는다.

주문하기

```
        } else {
            System.out.println("※ 잘못입력하셨습니다.");
            return;
        }
    }

    ordercode = System.nanoTime(); //회원이 한번에 여러개를 주문할때 필요한 키값.
    //구동중인 JVM에서 임의로 고정된 구간으로부터 현재 나노세컨즈nanoseconds 값을 반환

    for (int i = 0; i < arr.size(); i++) { //장바구니의 길이만큼 같은 키값을 부여한다.

        arr.get(i).setOrdercode(ordercode);
        int result = odao.insertOrder(conn, arr.get(i));
    }

    for (int i = 0; i < arr.size(); i++) { //장바구니의 길이만큼 주문할때마다 상품의 재고 감소시키기

        int result = odao.updateProduct(conn, arr.get(i));
    }

    conn.commit(); //트랜잭션 완료

} catch (SQLException e) {
    try {
        conn.rollback(); // 트랜잭션 수행중 에러시 이전 단계로 롤백
    } catch (SQLException e1) {
        e1.printStackTrace();
    }
    e.printStackTrace();
}


orderList(ordercode);
```

→ 결제 완료 후 영수증 출력

OrderDao

```
//주문테이블에 주문 내역 추가하기
public int insertOrder(Connection conn, Order order) {
    Product p = new Product();
    int result = 0;
    PreparedStatement pstmt = null;
    try {

        String sql = "insert into iorder values (iorder_oidx_seq.nextval"
                                                + ", ?, ?, ?, sysdate, ?, ?)";
        pstmt = conn.prepareStatement(sql);
        pstmt.setLong(1, order.getOrdercode());
        pstmt.setInt(2, order.getIcode());
        pstmt.setInt(3, order.getIdx());
        pstmt.setInt(4, order.getCount());
        pstmt.setInt(5, order.getOprice());
        result = pstmt.executeUpdate();

    } catch (SQLException e) {
        System.out.println("※ 잘못된 입력입니다. ");

    } finally {
        CloseDB.dbClose(pstmt);

    }
    return result;
}
```

OrderDao

```
//상품테이블의 재고수량을 구매수량만큼 차감하기
public int updateProduct(Connection conn, Order order) {

    int result = 0;
    PreparedStatement pstmt = null;

    try {

        String sql = "update product set count = count-? where icode=?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, order.getCount());
        pstmt.setInt(2, order.getIcode());

        result = pstmt.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        CloseDB.dbClose(pstmt);
    }
    return result;
}
```

주문하기

```java
//주문한 내역의 영수증
void orderList(long ordercode) {

    Order order = new Order();
    try {

        arr = odao.getOrderList(conn, order);

        int sum =0;
        System.out.println("  ▶ 결제 완료 \n");
        System.out.println("■■■■■■■■■■■■■■■■■■■■영수증■■■■■■■■■■■■■■■■■■■■");
        System.out.println("--------------------------------------------------------------------------------");
        for (int i = 0; i < arr.size(); i++) {
            if(ordercode ==arr.get(i).getOrdercode()) {

                System.out.printf(" 상품번호 : %d번 | 구매 수량 : %d개 | 구매가격 : %d원 | 주문날짜 :%s \n" ,
                                arr.get(i).getIcode(),arr.get(i).getCount(),
                                arr.get(i).getOprice(),arr.get(i).getOrderdate());

                sum += arr.get(i).getOprice();

            }

        }
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("      ☞ 총 구매 내역 :  \t\t   "+sum +" 원");
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println();
        System.out.println();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
//주문테이블에서 주문내역 읽어오기
public ArrayList<Order> getOrderList(Connection conn, Order order) {

    ArrayList<Order> list = null;
    Statement stmt = null;
    ResultSet rs = null;

    try {

        String sql = "select * from iorder";

        stmt = conn.createStatement();
        rs = stmt.executeQuery(sql);
        list = new ArrayList<>();

        while (rs.next()) {
            Order d = new Order(rs.getInt(1), rs.getLong(2), rs.getInt(3),
                    rs.getInt(4),rs.getString(5), rs.getInt(6), rs.getInt(7));
            list.add(d);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        CloseDB.dbClose(rs);
        CloseDB.dbClose(stmt);
    }
    return list;
}
```

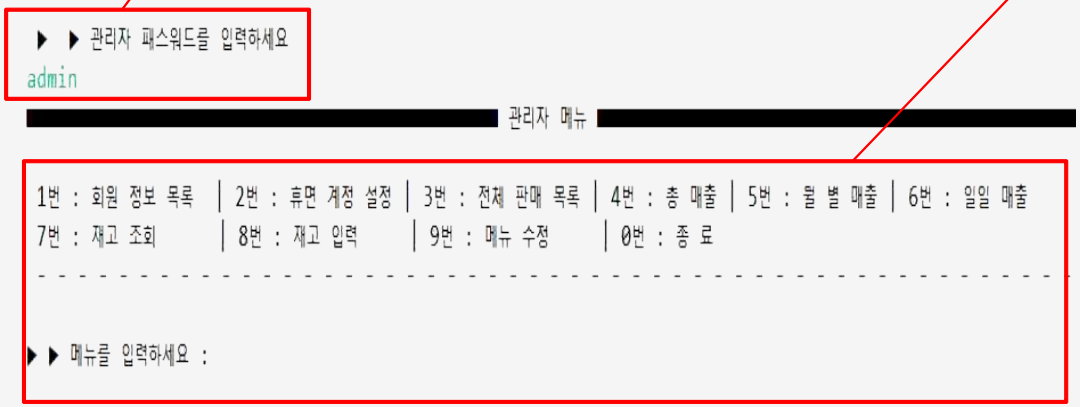
OrderDao

```
 ▶ 결제 완료

■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■영수증■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
--------------------------------------------------------------------------------
상품번호 : 1번 | 구매 수량 : 5개 | 구매가격 : 10000원 | 주문날짜 :2021-06-26 15:43:30
상품번호 : 2번 | 구매 수량 : 5개 | 구매가격 : 12500원 | 주문날짜 :2021-06-26 15:43:31
--------------------------------------------------------------------------------
     ☞ 총 구매 내역 :                  22500 원
--------------------------------------------------------------------------------


[1]주문하기   [0]돌아가기
```

주문하기

- 관리자 페이지

```
System.out.println(" ▶  ▶ 관리자 패스워드를 입력하세요 ");
String pw = sc.nextLine();

if(!pw.equals("admin")) {
    System.out.println("※  패스워드를 잘못 입력하셨습니다.");
    return;
}
```




```
try {
    if(choice<0 || choice>9  ) {

        throw new Exception(" ※ 잘못입력하셨습니다. 0~9번 중 하나를 선택해주세요.\n ");
    }
    switch(choice) {
    case 1 : // 회원 정보 목록
        manager.AdminMemberList();
        break;
    case 2 : // 휴면 계정 설정
        manager.AdminMemberChange();
        manager.AdminMemberList();
        break;
    case 3 : // 전체 판매 목록
        svcmanager.orderList();
        break;
    case 4 : // 총 매출
        svcmanager.salseManagement();
        break;
    case 5 : // 월 별 매출
        svcmanager.salseManagementMonth();
        break;
    case 6 : // 일일 매출
        svcmanager.salseManagementDaily();
        break;
    case 7 : // 재고 조회
        svcmanager.inventory();
        break;
    case 8 : // 재고 입력
        svcmanager.putIndentory();
        break;
    case 9 : // 메뉴 수정
        pMenu.ProductMenu();
        break;
    case 0 :
    System.out.println(" ※ 이전메뉴로 돌아갑니다.");
        return;
    }
```

- 관리자 페이지

회원 정보 목록

```
▶ ▶ 메뉴를 입력하세요 : 1
================================================================================
회원 정보 리스트
--------------------------------------------------------------------------------
회원번호    아이디    비밀번호    이름    연락처    이메일
--------------------------------------------------------------------------------
1        seungmin2        12345          나자바봐라  010-1234-5678    qwe@qwe.com
--------------------------------------------------------------------------------
```

```java
// 모든 회원 정보 리스트
void AdminMemberList() {

    try {

        List<Member> list = dao.getMemberList(conn);
        System.out.println("================================================================================");
        System.out.println("회원 정보 리스트");
        System.out.println("--------------------------------------------------------------------------------");
        System.out.println("회원번호 \t 아이디 \t 비밀번호 \t  이름 \t 연락처 \t 이메일");
        System.out.println("--------------------------------------------------------------------------------");

        for (Member Member : list) {
            System.out.printf("%d\t %s\t  %s \t %s\t %s\t  %s\t \n",
                    Member.getIdx(), Member.getId(), Member.getPassword(), Member.getName(), Member.getPhonenum(), Member.getEmail());
        }

        System.out.println("--------------------------------------------------------------------------------");


    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

AdminMemberDao

```java
//모든 회원의 정보 읽기
public ArrayList<Member> getMemberList(Connection con){

    ArrayList<Member> list = null;

    Statement stmt = null;
    ResultSet rs = null;
    String sql = "select * from member";

    try {
        stmt = con.createStatement();
        rs = stmt.executeQuery(sql);

        list = new ArrayList<>();

        while(rs.next()) {
            list.add(new Member(rs.getInt(1), rs.getString(2), rs.getString(3), rs.getString(4), rs.getString(5), rs.getString(6)));
        }

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        CloseDB.dbClose(rs);
        CloseDB.dbClose(stmt);
    }

    return list;
}
```

- 관리자 페이지

휴면 계정 설정

```
// 휴면계정으로 변경할 데이터의 회원번호 입력
// 해당 회원번호의 데이터 수정
   void AdminMemberChange() {

       try {

         AdminMemberList();
         System.out.println("휴면계정 설정을 원하시는 회원의 회원번호를 입력해주세요.");
         int idx = sc.nextInt();


         Member member = new Member(idx);

         int result = dao.dormancyMember(conn, member);
         System.out.println("해당 회원이 휴면계정으로 전환됩니다.");

      } catch (Exception e) {
         e.printStackTrace();
      }
   }
```

AdminMemberDao

```
// 관리자로부터 회원번호(idx) 입력 받아서 해당 회원 휴면계정으로 전환
// 휴면계정 : 회원번호(idx)제외하고 모든 값을 0000으로 변경 ( 주문내역 삭제하지 않기위해)
int dormancyMember(Connection conn, Member member) {

    int result = 0;

    PreparedStatement pstmt = null;


    try {
        String Sql = "UPDATE MEMBER SET ID = '0000', PW = '0000', NAME = '휴면',"
                + " PHONENUM = '0000',  EMAIL = '0000' WHERE idx = ?";

        pstmt = conn.prepareStatement(Sql);
        pstmt.setInt(1, member.getIdx());

        result = pstmt.executeUpdate();


    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        CloseDB.dbClose(pstmt);
    }

    return result;
}
```

- 관리자 페이지

전체 판매 목록

AdminDao

```
//주문 테이블의 모든 주문내역 보기
ArrayList<Order> getOrderList(Connection conn){
    ArrayList<Order> list= null;

    Statement stmt = null;
    ResultSet rs = null;

    try {
        stmt = conn.createStatement();
        String sql = "select * from iorder order by ordercode, oidx";

        rs = stmt.executeQuery(sql);

        list = new ArrayList<>();

        while (rs.next()) {
            list.add(new Order(rs.getInt(1), rs.getLong(2),rs.getInt(3),rs.getInt(4),rs.getString(5),rs.getInt(6),rs.getInt(7) ));
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        CloseDB.dbClose(stmt);

    }
    return list;
}
```

```
// 전체 주문 리스트 출력 메소드
void orderList() {

    try {
        List<Order> list = dao.getOrderList(conn);

        System.out.println("주문 정보 리스트 ");
        System.out.println("-------------------------------------------------------------------------------------------------------------");
        System.out.println("주문번호 \t\t 주문코드 \t \t 회원번호  \t 상품번호 \t\t 주문날짜 \t\t 상품수량 \t 주문금액 ");
        System.out.println("-------------------------------------------------------------------------------------------------------------");

        for(Order order : list) {
            System.out.printf("%d \t %d \t %d \t\t %d \t %s \t %d \t %d \n",
                    order.getOidx(), order.getOrdercode(), order.getIdx(),order.getIcode(),order.getOrderdate(),order.getCount(),order.getOprice());

        }
        System.out.println("-------------------------------------------------------------------------------------------------------------");


    } catch (Exception e) {
        e.printStackTrace();
    }

}
```

```
▶ ▶ 메뉴를 입력하세요 : 3
주문 정보 리스트
-------------------------------------------------------------------------------------------------------------
주문번호        주문코드        회원번호        상품번호        주문날짜        상품수량    주문금액
-------------------------------------------------------------------------------------------------------------
1       1208104599978000        1       1       2021-06-26 15:43:30     5       10000
2       1208104599978000        1       2       2021-06-26 15:43:31     5       12500
-------------------------------------------------------------------------------------------------------------
```

- 관리자 페이지

총 매출

```java
//총 매출 출력
public void salseManagement() {

    try {

        int sum = dao.getSales(conn);

        System.out.println("총 매출은 "+ sum +"원 입니다. ");

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

AdminDao

```java
// 총 매출보기
int getSales (Connection conn) {

    int result =0;

    Statement stmt = null;
    ResultSet rs = null;

    try {
        stmt = conn.createStatement();
        String sql = "select sum(oprice) from iorder";

        rs = stmt.executeQuery(sql);

        while (rs.next()) {
            result =rs.getInt(1);
        }


    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        CloseDB.dbClose(stmt);
    }

    return result;
}
```

```
관리자 메뉴

1번 : 회원 정보 목록 | 2번 : 휴면 계정 설정 | 3번 : 전체 판매 목록 | 4번 : 총 매출 | 5번 : 월 별 매출 | 6번 : 일일 매출
7번 : 재고 조회      | 8번 : 재고 입력      | 9번 : 메뉴 수정      | 0번 : 종 료
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

▶▶ 메뉴를 입력하세요 : 4
총 매출은 22500원 입니다.
```

- 관리자 페이지

월 별 매출

```java
//달별 매출 출력
public void salseManagementMonth() {

    try {

        System.out.println("검색하실 월을 입력해주세요.");
        System.out.println("2021년 5월 매출을 보시려면 21/05 형식으로 입력해주세요. ");
        String dno = sc.nextLine();

         boolean check = Pattern.matches("\\d{2}/\\d{2}", dno);
         if(check == true) {
            int sum = dao.getSalesMonth(conn, dno);
            System.out.println("총 매출은 "+ sum + "원 입니다. ");
         } else {
            System.out.println("입력 값이 올바르지 않습니다.");
            return;
         }


    } catch (Exception e) {
        System.out.println("잘못입력하셨습니다.");
        e.printStackTrace();
    }
}
```

AdminDao

```java
//달별 매출 보기
int getSalesMonth(Connection conn, String dno) {

    int result =0;

    Statement stmt = null;
    ResultSet rs = null;

    try {
        stmt = conn.createStatement();

        String sql = "select sum(oprice) from iorder where substr(orderdate,1,5) = '"+dno+"'";

        rs = stmt.executeQuery(sql);

        while (rs.next()) {
            result =rs.getInt(1);
        }


    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        CloseDB.dbClose(stmt);
    }

    return result;
}
```

```
▶ ▶ 메뉴를 입력하세요 : 5
검색하실 월을 입력해주세요.
2021년 5월 매출을 보시려면 21/05 형식으로 입력해주세요.
21/06
총 매출은 22500원 입니다.
```

- 관리자 페이지

일일 매출

```java
//일별 매출 출력
public void salseManagementDaily() {

    try {

        System.out.println("검색하실 월과 일자를 입력해주세요. ");
        System.out.println("6월 1일이면 06/01 형식으로 입력해주세요. ");
        String dday = sc.nextLine();

        boolean check = Pattern.matches("\\d{2}/\\d{2}", dday);

         if(check == true) {
            int sum = dao.getSalesDay(conn, dday);

            System.out.println("총 매출은 "+ sum+"원 입니다. ");
         } else {
            System.out.println("입력 값이 올바르지 않습니다.");
            return;
         }


    } catch (Exception e) {
        e.printStackTrace();
        System.out.println("잘못입력하셨습니다.");
    }
}
```

AdminDao

```java
//일별 매출 보기
int getSalesDay(Connection conn, String dday) {

    int result =0;

    Statement stmt = null;
    ResultSet rs = null;

    try {
        stmt = conn.createStatement();

        String sql = "select sum(oprice) from iorder where substr(orderdate,4,5) = '" +dday+"'";


        rs = stmt.executeQuery(sql);


        while (rs.next()) {
            result =rs.getInt(1);
        }


    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        CloseDB.dbClose(stmt);

    }

    return result;
}
```

```
▶ ▶ 메뉴를 입력하세요 : 6
검색하실 월과 일자를 입력해주세요.
6월 1일이면 06/01 형식으로 입력해주세요.
06/26
총 매출은 22500원 입니다.
```

- 관리자 페이지

재고 조회

```java
// 재고보기
public void inventory() {

    try {
        List<Product> list = dao.getInventory(conn);

        System.out.println("상품 재고 리스트 ");
        System.out.println("-------------------------------------------------");
        System.out.println("상품번호 \t 상품명 \t\t 상품가격 \t 상품수량");
        System.out.println("-------------------------------------------------");

        for(Product product : list) {
            System.out.printf("%d \t %s \t %d \t\t %d \n",
                    product.getIcode(),product.getIname(),product.getIprice(),product.getCount());

        }
    } catch (Exception e) {
        System.out.println("잘못 입력하셨습니다.");
        e.printStackTrace();
    }
}
```

AdminDao

```java
//상품테이블에서 상품과 재고 보기
ArrayList<Product> getInventory(Connection conn){
    ArrayList<Product> list= null;

    Statement stmt = null;
    ResultSet rs = null;

    try {
        stmt = conn.createStatement();
        String sql = "select * from product order by icode";

        rs = stmt.executeQuery(sql);

        list = new ArrayList<>();

        while (rs.next()) {
            list.add(new Product(rs.getInt(1), rs.getString(2),rs.getInt(3),rs.getInt(4) ));
        }

    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        CloseDB.dbClose(stmt);
    }

    return list;
}
```

```
상품 재고 리스트
-------------------------------------------------
상품번호    상품명              상품가격    상품수량
-------------------------------------------------
1          바닐라 아이스크림     2000          95
2          초코 아이스크림       2500          95
3          딸기 아이스크림       2500          100
4          바나나 아이스크림     2700          100
5          커피 아이스크림       2700          100
6          민트 아이스크림       2800          100
7          요거트 아이스크림     3000          100
```

- 관리자 페이지

재고 입력

```java
//재고 넣기
void putIndentory() {

    inventory();

    try {

        System.out.println("재고 수량을 입력합니다. 상품 번호를 입력해주세요.");
        int icode = sc.nextInt();
        System.out.println("추가하실 재고 수량을 입력해주세요.");
        int count = sc.nextInt();

        Product product = new Product (icode , count);

        int result = dao.putInstance(conn,product);

        if(result >0) {
            System.out.println(" 수정되었습니다.");
            System.out.println();
            inventory();
        }else {
            System.out.println("수정실패 ");
        }

     }catch (NumberFormatException e){
        System.out.println("숫자로만 입력해주세요.");
    } catch (Exception e) {
        System.out.println("잘못 입력하셨습니다.");
    }


}
```

AdminDao

```java
//상품의 재고 수량 더하기
int putInstance(Connection conn, Product product){
    int result = 0;

    PreparedStatement pstmt = null;


    try {
        String sql = "update product set count=count+? where icode = ?";

        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, product.getCount());
        pstmt.setInt(2, product.getIcode());


        result = pstmt.executeUpdate();

    } catch (SQLException e) {
        e.printStackTrace();
    }finally {
        CloseDB.dbClose(pstmt);

    }

    return result;

}
```

- 관리자 페이지_메뉴 수정

상품 추가

ProductDao

```java
// 아이스크림 신메뉴 등록
public int insertProduct(Connection conn, Product product) {

   int result = 0;
   PreparedStatement pstmt = null;
   try {

      String sql = "insert into product (icode, iname, iprice, count) values (?, ?, ?, ?)";
      pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, product.getIcode());
      pstmt.setString(2, product.getIname());
      pstmt.setInt(3, product.getIprice());
      pstmt.setInt(4, product.getCount());

      result = pstmt.executeUpdate();


   } catch (SQLException e) {
      e.printStackTrace();
   } finally {
       CloseDB.dbClose(pstmt);
   }
   return result;
}
```

```java
//상품테이블에 신메뉴 추가하기
void productInsert() {

    try {

        while(true) {
            Product p = new Product();
            while(true) {
                System.out.println("아이스크림 메뉴번호를 입력해주세요.");
                int icode = Integer.parseInt(sc.nextLine());
                int cnt = 0;
                for (int i = 0; i < pro.size(); i++) {
                    if(icode == pro.get(i).getIcode()) {
                        cnt++;
                        System.out.println("메뉴번호가 중복입니다. 다시입력하세요.");
                        break;
                    }
                }
                if(cnt != 1) {
                    p.setIcode(icode);
                    break;
                }
            }
            System.out.println("▶ ▶ 아이스크림 상품명을 입력해주세요. ");
            p.setIname(sc.nextLine());
            System.out.println("▶ ▶ 아이스크림 가격을 입력해주세요.");
            p.setIprice(Integer.parseInt(sc.nextLine()));
            System.out.println("▶ ▶ 아이스크림 수량을 입력해주세요.");
            p.setCount(Integer.parseInt(sc.nextLine()));


            System.out.println("입력하시겠습니까?[1] 예 ,[2] 아니오");
            int input = Integer.parseInt(sc.nextLine());
            if(input==1) {
                pro.add(p);
                int result = dao.insertProduct(conn, p);
                System.out.println("☞ 입력되셨습니다. \n");
                break;
            } else if(input==2) {
                System.out.println("※ 다시입력해주세요.");
            } else {
                System.out.println("※ 잘못입력하셨습니다. 초기화면으로 이동합니다.");
                break;
            }

        }
```

- 관리자 페이지_메뉴 수정

상품 수정

ProductDao

```java
// 아이스크림 메뉴 수정
int updateProduct(Connection conn, Product product) {

   int result = 0;
   PreparedStatement pstmt = null;

   try {

      String sql = "update product set icode=?,iname=?, iprice=?, count=?"
                   + " where icode ='" + product.getIcode() + "'" ;
      pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, product.getIcode());
      pstmt.setString(2, product.getIname());
      pstmt.setInt(3, product.getIprice());
      pstmt.setInt(4, product.getCount());


      result = pstmt.executeUpdate();

   } catch (SQLException e) {
      e.printStackTrace();
   } finally {
       CloseDB.dbClose(pstmt);
   }
   return result;
}

}
```

```java
//상품테이블에 기존의 상품 수정
void productUpdate() {
    try {
        while(true) {
            System.out.println("▶ ▶ 수정할 아이스크림 정보를 입력해주세요.");
            System.out.println("▶ 상품번호, 상품이름, 상품가격, 상품 갯수 순으로 입력해주세요.");
            System.out.println("예 ) 1, 바닐라 아이스크림, 2000, 5 ( 쉼표 포함 )");
            String editData = sc.nextLine().trim();
            String[] eData = editData.split(",");
            Product p;
            for(int i =0; i<eData.length;i++) {
                eData[i] = eData[i].trim();
            }
            if(eData.length == 4) {
                for (int i = 0; i < pro.size(); i++) {
                    if((Integer.parseInt(eData[0])) == pro.get(i).getIcode()) {
                        p = new Product(Integer.parseInt(eData[0]), eData[1],
                                Integer.parseInt(eData[2]), Integer.parseInt(eData[3]));
                        int result = dao.updateProduct(conn, p);
                        System.out.println("    ☞ 해당 상품 정보가 변경 되었습니다.\n");
                        return;
                    }
                }
                System.out.println("※ 해당 하는 상품이 없습니다.");
                return;
            } else {
                System.out.println("※ 입력 형식을 올바르게 입력해주세요. ");
                continue;
            }
        }
    } catch (Exception e) {
        System.out.println(" ※ 잘못입력하셨습니다. 이전페이지로 돌아갑니다. ");
    }
}
```

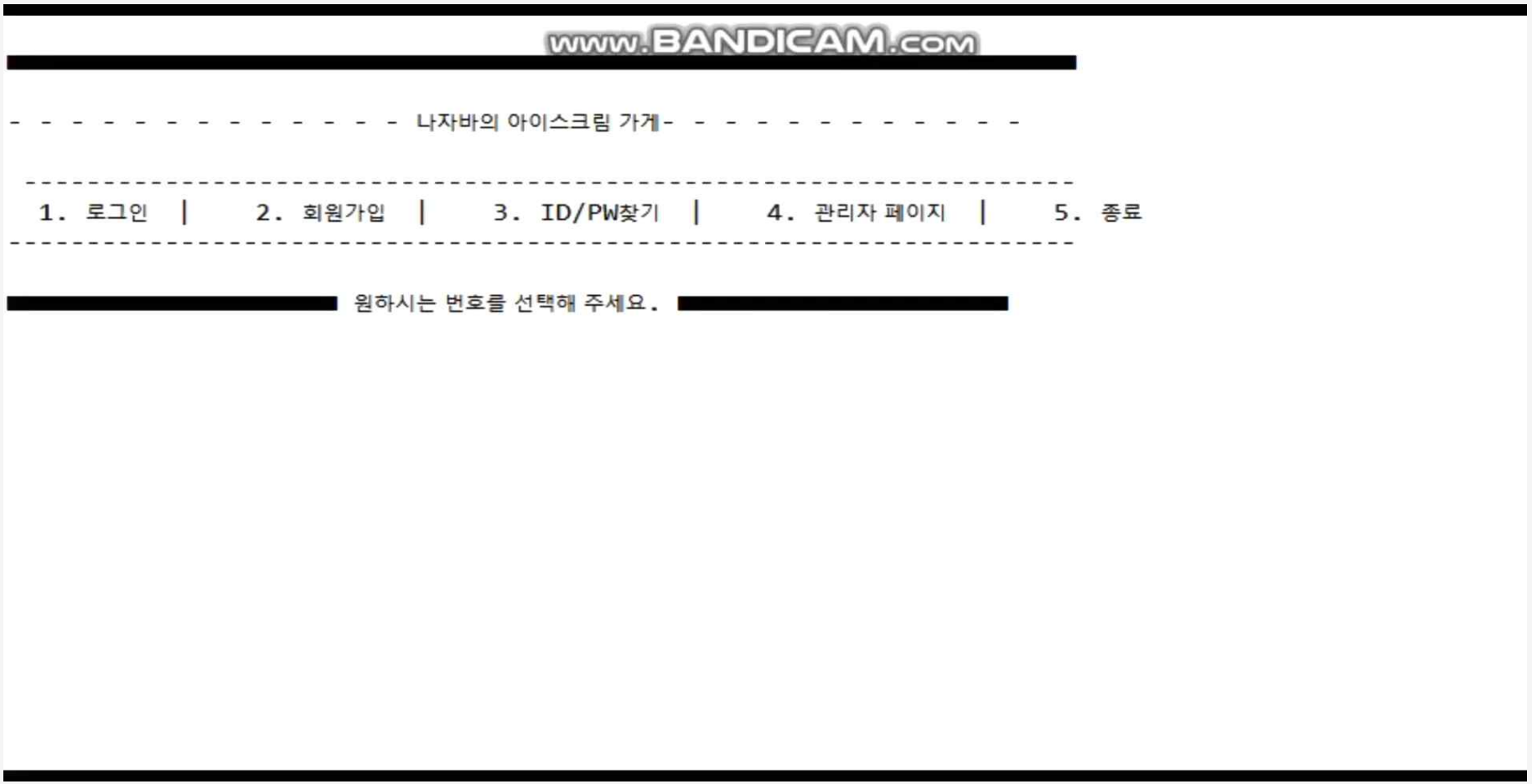
04

소감

05

프로그램 구현 영상

- 프로그램 구현 영상


www.BANDICAM.com
- - - - - - - - - - - - - - 나자바의 아이스크림 가게- - - - - - - - - - - - -
1. 로그인 | 2. 회원가입 | 3. ID/PW찾기 | 4. 관리자 페이지 | 5. 종료
원하시는 번호를 선택해 주세요.

발표를 들어주셔서

감사합니다 :)