MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 프로젝트의 개요

- 이번에 진행할 프로젝트는 쇼핑몰 등에서 사용할 수 있는 상품 관리 프로그램이다. 상품 정보 등록, 등록된 상품 현황 조회, 정보 수정 및 삭제 기능으로 구성한다

표 10-3 상품 관리 프로젝트 클래스

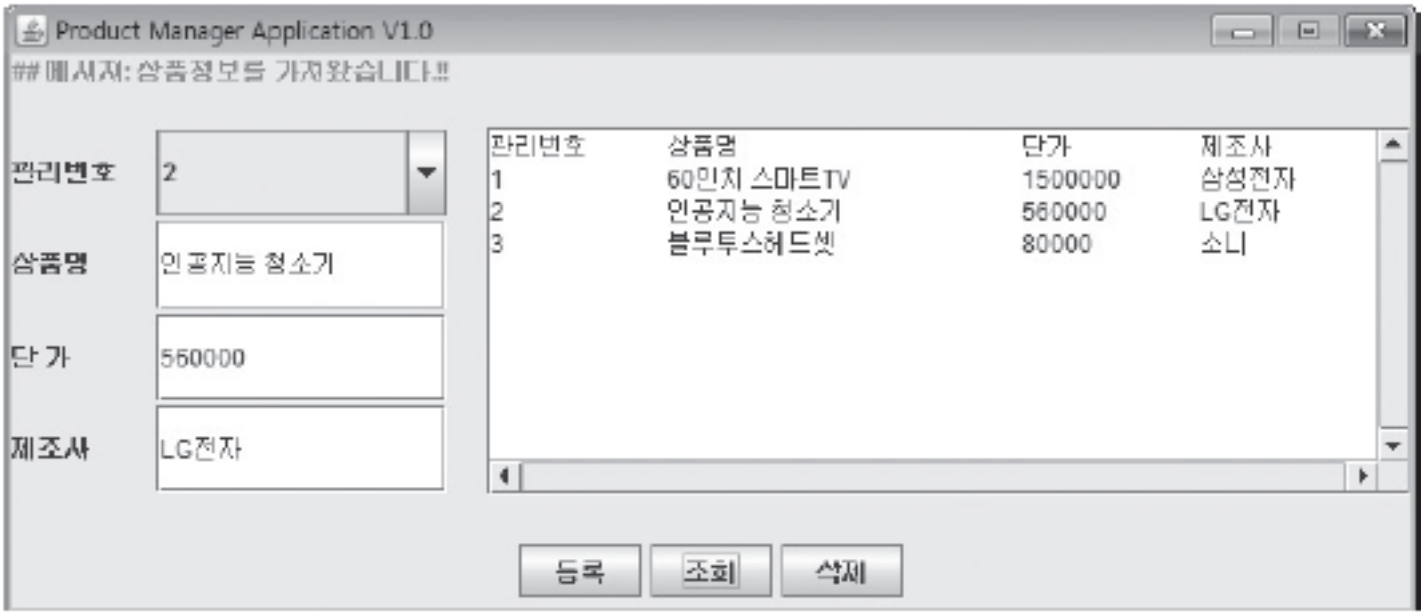
클래스	설명	비고
AppMain	프로그램의 메인 클래스로, 화면 구성과 이벤트 처리를 담당한다.	View, Controller
Product	상품 정보를 표현하는 클래스로, product 테이블과 구조가 동일하다.	Data Object
ProductDAO	데이터베이스와 연동하는 데 필요한 클래스로, 데이터베이스 연결 및 입력, 수정, 삭제 등 실제 처리 기능을 제공한다.	Data Access Object

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

프로젝트의 개요

- 화면 왼쪽에는 상품 정보를 입력하거나 수정할 수 있는 양식이, 화면 오른쪽에는 등록한 상품을 모두 보여 주는 목록을 배치

그림 10-27 프로그램 동작 화면



MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 프로젝트의 개요

표 10-4 프로그램 주요 기능

주요 기능	설명	비고
상품 등록	상품 정보 입력 후 [등록] 버튼을 눌러 등록한다.	등록 후에는 오른쪽 목록이 갱신되어 새로 등록한 내용을 확인할 수 있다.
상품 수정	관리번호 콤보박스에서 원하는 상품 번호를 선택하고 [조회] 버튼을 눌러 정보를 로딩한다. 내용을 수정한 후에는 [등록] 버튼을 눌러 업데이트한다.	전체를 선택할 때는 전체 목록을 다시 출력한다.
상품 삭제	관리번호 콤보박스에서 원하는 상품 정보를 불러온 후 [삭제] 버튼을 눌러 삭제한다.	삭제 후 전체 목록을 다시 출력한다.
전체 보기	관리번호 콤보박스에서 전체를 선택하고, [조회] 버튼을 누르면 전체 목록을 다시 출력한다.	프로그램의 시작은 전체 목록 보기에서 시작한다.

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

테이블 생성

- 테이블은 product 이름으로 생성하며, MySQL Workbench를 이용하여 SQL 문을 입력하거나 테이블 생성 메뉴를 이용하여 입력 방식으로 생성할 수 있다.

```
CREATE TABLE product (
    prcode int(11) NOT NULL AUTO_INCREMENT,
    prname varchar(45) NOT NULL,
    price int(11) NOT NULL,
    manufacture varchar(20) NOT NULL,
    PRIMARY KEY(prcode)
) ENGINE = InnoDB AUTO_INCREMENT = 15 DEFAULT CHARSET = utf8;
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- **테이블 생성**

표 10-5 product 테이블 컬럼의 용도

컬럼	데이터 타입	용도
prcode	int(11)	상품 관리번호[주 키]
prname	varchar(45)	상품 이름
price	int(11)	상품 가격
manufacture	varchar(20)	제조사

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 클래스 구현

- Product 클래스 구현 : 상품 정보 테이블의 데이터를 표현하는 Data Object 클래스로 테이블 컬럼 구조와 동일하며, 필드와 getter/setter로만 구성

예제 10-1 상품 정보 클래스 구현하기 Product.java

```
01 package javabook.ch10;
02
03 // 상품 정보 테이블 데이터 표현을 위한 클래스
04 public class Product {
05
06     // 컬럼 정보에 따른 필드 선언
07     private int prcode;
08     private String prname;
09     private int price;
10     private String manufacture;
11
12     // getter/setter 메서드
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
13      public int getPrcode() {
14          return prcode;
15      }
16      public void setPrcode(int prcode) {
17          this.prcode = prcode;
18      }
19      public String getPrname() {
20          return prname;
21      }
22      public void setPrname(String prname) {
23          this.prname = prname;
24      }
25      public int getPrice() {
26          return price;
27      }
28      public void setPrice(int price) {
29          this.price = price;
30      }
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
31      public String getManufacture() {
32          return manufacture;
33      }
34      public void setManufacture(String manufacture) {
35          this.manufacture = manufacture;
36      }
37  }
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

- 상품 정보 데이터베이스와 연동하여 데이터를 처리하는 데 필요한 ProductDAO 클래스를 구현해 본다.

표 10-6 ProductDAO 클래스 주요 메서드

메서드	설명
void connectDB( )	DB 연결 메서드
void closeDB( )	DB 연결 종료 메서드
ArrayList<Product> getAll( )	전체 Product로 구성된 ArrayList를 리턴
Product getProduct(int prcode)	파라미터의 prcode에 해당하는 상품을 리턴
boolean newProduct(Product product)	파라미터의 Product 객체의 내용을 DB에 저장
boolean delProduct(int prcode)	파라미터의 prcode에 해당하는 상품을 삭제
boolean updateProduct(Product product)	파라미터의 Product 객체의 내용으로 업데이트
Vector<String> getItems( )	콤보박스용 관리번호 목록을 리턴

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

- 클래스 선언부

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

▶ *getAll( ) 메서드 구현하기

getAll( )은 전체 상품 목록을 가져오는 메서드이다.

[getAll( ) 메서드의 앞부분]

```
public ArrayList<Product> getAll() {
    connectDB();
    sql = "select * from product";

    // 전체 검색 데이터를 전달하는 ArrayList
    ArrayList<Product> datas = new ArrayList<Product>();

    // 관리번호 콤보박스 데이터를 위한 벡터 초기화
    items = new Vector<String>();
    items.add("전체");
.....
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

▶ *getAll( ) 메서드 구현하기

getAll( )은 전체 상품 목록을 가져오는 메서드이다.

[while( ) 부분]

```
while(rs.next()) {
    Product p = new Product();
    p.setPrcode(rs.getInt("prcode"));
    p.setPrname(rs.getString("prname"));
    p.setPrice(rs.getInt("price"));
    p.setManufacture(rs.getString("manufacture"));
    datas.add(p);
    items.add(String.valueOf(rs.getInt("prcode")));
}
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

▶ *getProduct( ) 메서드 구현하기

prcode가 WHERE 조건절에 들어간다.

```
sql = "select * from product where prcode = ?";
Product p = null;
try {
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, prcode);
    rs = pstmt.executeQuery();
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

▶ *getProduct( ) 메서드 구현하기

prcode가 WHERE 조건절에 들어간다.

[rs.next( )로 첫 번째 데이터만 처리]

```
        rs.next();
        p = new Product();
        p.setPrcode(rs.getInt("prcode"));
        p.setPrname(rs.getString("prname"));
        p.setPrice(rs.getInt("price"));
        p.setManufacture(rs.getString("manufacture"));
    }
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

▶ newProduct( ) 메서드 구현하기

newProduct( )는 새로운 상품을 등록하는 메서드이다.

[insert 문 작성을 할 때]

```
sql = "insert into product(prname, price, manufacture) values(?, ?, ?)";
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

▶ delProduct( ) 메서드 구현하기

delProduct( )는 특정 상품을 삭제하는 메서드로, 파라미터로 받은 prcode를 WHERE 조건절에 두어 처리한다.

▶ updateProduct( ) 메서드 구현하기

updateProduct( )는 newProduct( )와 구조적으로는 비슷하며, WHERE를 사용하여 특정 데이터만 새로운 내용으로 업데이트한다. SQL 문에서 모든 컬럼을 업데이트하는 형식으로 처리했으며, WHERE 조건절에 prcode가 들어가는 점에 주의한다.

```
sql = "update product set prname = ?, price = ?, manufacture = ? where prcode = ?";
try {
    pstmt = conn.prepareStatement(sql);
    pstmt.setString(1, product.getPrname());
    pstmt.setInt(2, product.getPrice());
    pstmt.setString(3, product.getManufacture());
    pstmt.setInt(4, product.getPrcode());
    pstmt.executeUpdate();
}
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 데이터베이스 연동 클래스 구현 : 상품 정보 데이터베이스 처리 클래스 구현

▶ getItems( ) 메서드 구현하기

getItems( )는 사용자 편의를 위해 관리번호 콤보박스에 들어갈 목록을 제공하는 메서드이다. 벡터 데이터는 getAll( )에서 추가하며, 여기서는 단순히 items 변수를 리턴한다.

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- 앞에서 살펴본 내용으로 완성한 ProductDAO 클래스의 전체 코드

예제 10-2 상품 정보 데이터베이스 처리 클래스 구현하기 — ProductDAO.java

```
01  package javabook.ch10;
02
03  import java.sql.*;
04  import java.util.*;
05
06  public class ProductDAO {
07      String jdbcDriver = "com.mysql.jdbc.Driver";
08      String jdbcUrl = "jdbc:mysql://localhost/javadb";
09      Connection conn;
10
11      PreparedStatement pstmt;
12      ResultSet rs;
13
14      Vector<String> items = null;
15      String sql;
16
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
17      // 콤보박스의 상품 관리 번호 목록을 위한 벡터 리턴
18      public Vector<String> getItems() {
19          return items;
20      }
21
22      // 전체 상품 목록을 가져오는 메서드
23      public ArrayList<Product> getAll() {
24          connectDB();
25          sql = "select * from product";
26
27          // 전체 검색 데이터를 전달하는 ArrayList
28          ArrayList<Product> datas = new ArrayList<Product>();
29
30          // 관리번호 콤보박스 데이터를 위한 벡터 초기화
31          items = new Vector<String>();
32          items.add("전체");
33
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
34          try {
35              pstmt = conn.prepareStatement(sql);
36              rs = pstmt.executeQuery();
37
38              // 검색된 데이터 수만큼 루프를 돌며 Product 객체를 만들고 이를 다시 ArrayList에 추가
39              while(rs.next()) {
40                  Product p = new Product();
41                  p.setPrcode(rs.getInt("prcode"));
42                  p.setPrname(rs.getString("prname"));
43                  p.setPrice(rs.getInt("price"));
44                  p.setManufacture(rs.getString("manufacture"));
45                  datas.add(p);
46                  items.add(String.valueOf(rs.getInt("prcode")));
47              }
48          } catch (SQLException e) {
49              e.printStackTrace();
50              return null;
51          } finally {
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
52              closeDB();
53          }
54          return datas;
55      }
56
57      // 선택한 상품 코드에 해당하는 상품 정보를 가져오는 메서드
58      public Product getProduct(int prcode) {
59          connectDB();
60          sql = "select * from product where prcode = ?";
61          Product p = null;
62          try {
63              pstmt = conn.prepareStatement(sql);
64              pstmt.setInt(1, prcode);
65              rs = pstmt.executeQuery();
66              rs.next();
67              p = new Product();
68              p.setPrcode(rs.getInt("prcode"));
69              p.setPrname(rs.getString("prname"));
70              p.setPrice(rs.getInt("price"));
71              p.setManufacture(rs.getString("manufacture"));
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
72          } catch (SQLException e) {
73              e.printStackTrace();
74              return null;
75          } finally {
76              closeDB();
77          }
78          return p;
79      }
80
81      // 새로운 상품을 등록하는 메서드
82      public boolean newProduct(Product product) {
83          connectDB();
84
85          // prcode는 자동 증가 컬럼이므로 직접 입력하지 않음
86          sql = "insert into product(prname, price, manufacture) values(?, ?, ?)";
87          try {
88              pstmt = conn.prepareStatement(sql);
89              pstmt.setString(1, product.getPrname());
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
90              pstmt.setInt(2, product.getPrice());
91              pstmt.setString(3, product.getManufacture());
92              pstmt.executeUpdate();
93          } catch (SQLException e) {
94              e.printStackTrace();
95              return false;
96          } finally {
97              closeDB();
98          }
99          return true;
100     }
101
102     // 선택한 상품을 삭제하는 메서드
103     public boolean delProduct(int prcode) {
104         connectDB();
105         sql = "delete from product where prcode = ?";
106         try {
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
107             pstmt = conn.prepareStatement(sql);
108             pstmt.setInt(1, prcode);
109             pstmt.executeUpdate();
110         } catch (SQLException e) {
111             e.printStackTrace();
112             return false;
113         } finally {
114             closeDB();
115         }
116         return true;
117     }
118     // 수정한 정보로 상품 정보를 업데이트하는 메서드
119     public boolean updateProduct(Product product) {
120         connectDB();
121         sql = "update product set prname = ?, price = ?, manufacture = ? where prcode = ?";
122         try {
123             pstmt = conn.prepareStatement(sql);
124             pstmt.setString(1, product.getPrname());
125             pstmt.setInt(2, product.getPrice());
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
126                 pstmt.setString(3, product.getManufacture());
127                 pstmt.setInt(4, product.getPrcode());
128                 pstmt.executeUpdate();
129             } catch (SQLException e) {
130                 e.printStackTrace();
131                 return false;
132             } finally {
133                 closeDB();
134             }
135             return true;
136         }
137 
138         // DB 연결 메서드
139         public void connectDB() {
140             try {
141                 Class.forName(jdbcDriver);
142                 conn = DriverManager.getConnection(jdbcUrl, "javabook", "1234");
143             } catch (Exception e) {
144                 e.printStackTrace();
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
145             }
146         }
147
148         // DB 연결 종료 메서드
149         public void closeDB() {
150             try {
151                 pstmt.close();
152                 conn.close();
153             } catch (SQLException e) {
154                 e.printStackTrace();
155             }
156         }
157     }
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

GUI 처리 클래스 구현 : AppMain 클래스 개요

- GUI는 AppMain 클래스에서 구현한다.

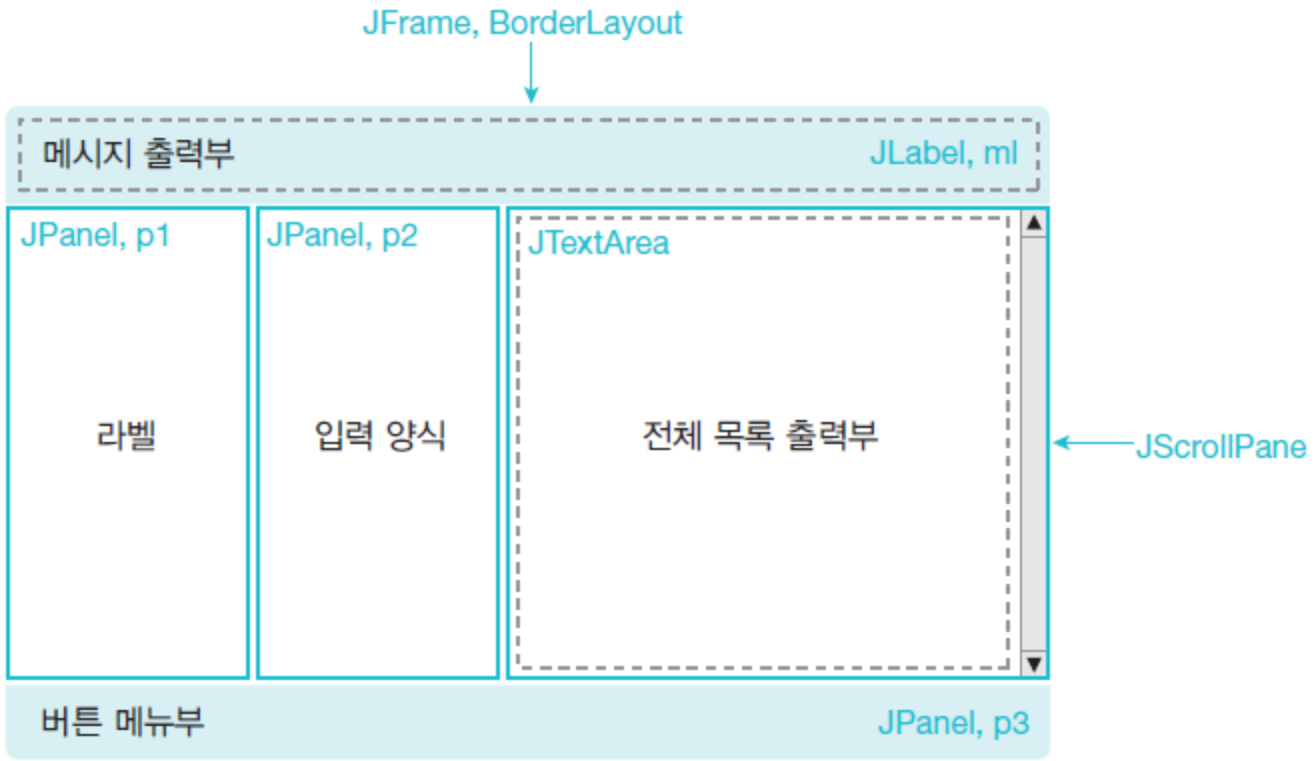
[기본 구성]

- 화면 구성을 위한 컴포넌트 선언 및 초기화
- 생성자에서 화면 크기 등 기본값 설정
- startUI( ) 메서드에서 화면 구성
- actionPerformed( ) 메서드에서 이벤트 처리
- refreshData( ) 메서드에서 화면 데이터 로딩 및 재로딩 처리
- clearField( ) 메서드에서 입력 양식 초기화

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- **GUI 처리 클래스 구현 : AppMain 클래스 개요**

그림 10-28 전체 화면 컴포넌트와 레이아웃



MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- **GUI 처리 클래스 구현 : 레이아웃을 이용한 패널 배치**

[배치의 최종적인 BorderLayout 코드]

```
add(ml, BorderLayout.PAGE_START);
add(p1, BorderLayout.LINE_START);
add(p2, BorderLayout.CENTER);
add(scroll, BorderLayout.LINE_END);
add(p3, BorderLayout.PAGE_END);
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- **GUI 처리 클래스 구현 : 관리번호를 선택하는 콤보박스 구성**

[콤보박스 생성 코드]

```
cb = new JComboBox();
    ta = new JTextArea(10, 40);
    JScrollPane scroll = new JScrollPane(ta, JScrollPane.VERTICAL_SCROLLBAR_
ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

GUI 처리 클래스 구현 : refreshData( ) 메서드 구현

- refreshData( )는 콤보 박스 데이터 및 전체 목록 데이터를 그때그때 최신 데이터로 업데이트하는 메서드

예제 10-3 refreshData( ) 메서드 구현하기

```
01      public void refreshData() {
02          ta.setText("");
03          clearField();
04          editmode = false;
05
06          ta.append("관리번호\t상품명\t\t단가\t제조사\n");
07          datas = dao.getAll();
08
09          // 데이터를 변경하면 콤보박스 데이터 갱신
10          cb.setModel(new DefaultComboBoxModel(dao.getItems()));
11
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
12          if(datas != null) {
13              // ArrayList의 전체 데이터를 형식에 맞춰 출력
14              for(Product p : datas) {
15                  StringBuffer sb = new StringBuffer();
16                  sb.append(p.getPrcode() + "\t");
17                  sb.append(p.getPrname() + "\t\t");
18                  sb.append(p.getPrice() + "\t");
19                  sb.append(p.getManufacture() + "\n");
20                  ta.append(sb.toString());
21              }
22          }
23          else {
24              ta.append("등록된 상품이 없습니다. !!\n상품을 등록해 주세요 !!");
25          }
26      }
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

■ 이벤트 처리

- 프로그램 시작
- [등록] 버튼 클릭
- [조회] 버튼 클릭
- [삭제] 버튼 클릭

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

프로그램 시작

- 다른 모든 프로그램과 마찬가지로 시작은 main( )에서 처리한다. 먼저 AppMain 객체를 생성하고 startUI( ) 메서드를 호출하는 것으로 시작 이벤트의 동작이 마무리된다.

[등록] 버튼 클릭

- 화면에 데이터를 입력하고 [등록] 버튼을 누르면 발생하는 이벤트로, actionPerformed( ) 메서드에서는 getSource( )가 [등록] 버튼인 b1의 경우에 해당한다.

```
if(obj == b1) {
    product = new Product();
    product.setPrname(t1.getText());
    product.setPrice(Integer.parseInt(t2.getText()));
    product.setManufacture(t3.getText());

    // 수정일 때
    if(editmode == true) {
        product.setPrcode(Integer.parseInt((String)cb.getSelectedItem()));
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
            if(dao.updateProduct(product)) {
                ml.setText(msg + "상품을 수정했습니다!!");
                clearField();
                editmode = false;
            }
            else
                ml.setText(msg + "상품 수정이 실패했습니다!!");
        }
        // 등록일 때
        else {
            if(dao.newProduct(product)) {
                ml.setText(msg + "상품을 등록했습니다!!");
            }
            else
                ml.setText(msg + "상품 등록이 실패했습니다!!");
        }
        // 화면 데이터 갱신
        refreshData();
    }
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

[등록] 버튼 클릭

- 등록·수정 모두 입력 양식에 있는 데이터가 필요하기 때문에 Product 객체를 생성하고 각 필드에 데이터를 입력한다.

```
product.setPrcode(Integer.parseInt((String)cb.getSelectedItem()));
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

[조회] 버튼 클릭

- [조회] 버튼은 b2이고 콤보박스에 선택된 prcode에 해당하는 데이터를 가져와 양식에 출력한다. 이때 '전체'를 선택하면 화면을 지우고 새로운 데이터를 로딩한다

```
// 조회 버튼일 때
else if(obj == b2) {
    String s = (String)cb.getSelectedItem();
    if(!s.equals("전체")) {
        product = dao.getProduct(Integer.parseInt(s));
        if(product != null) {
            ml.setText(msg + "상품정보를 가져왔습니다!!");
            t1.setText(product.getPrname());
            t2.setText(String.valueOf(product.getPrice()));
            t3.setText(product.getManufacture());
            // cb.setSelectedIndex(anIndex);
            editmode = true;
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

```
            else {
                ml.setText(msg + "상품이 검색되지 않았습니다!!");
            }
        }
    }
```

- 가격을 양식, 즉 JTextField에 출력할 때는 숫자값을 문자열로 처리해야 하므로 String.valueOf( )를 사용한다

```
t2.setText(String.valueOf(product.getPrice()));
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- **[삭제] 버튼 클릭**

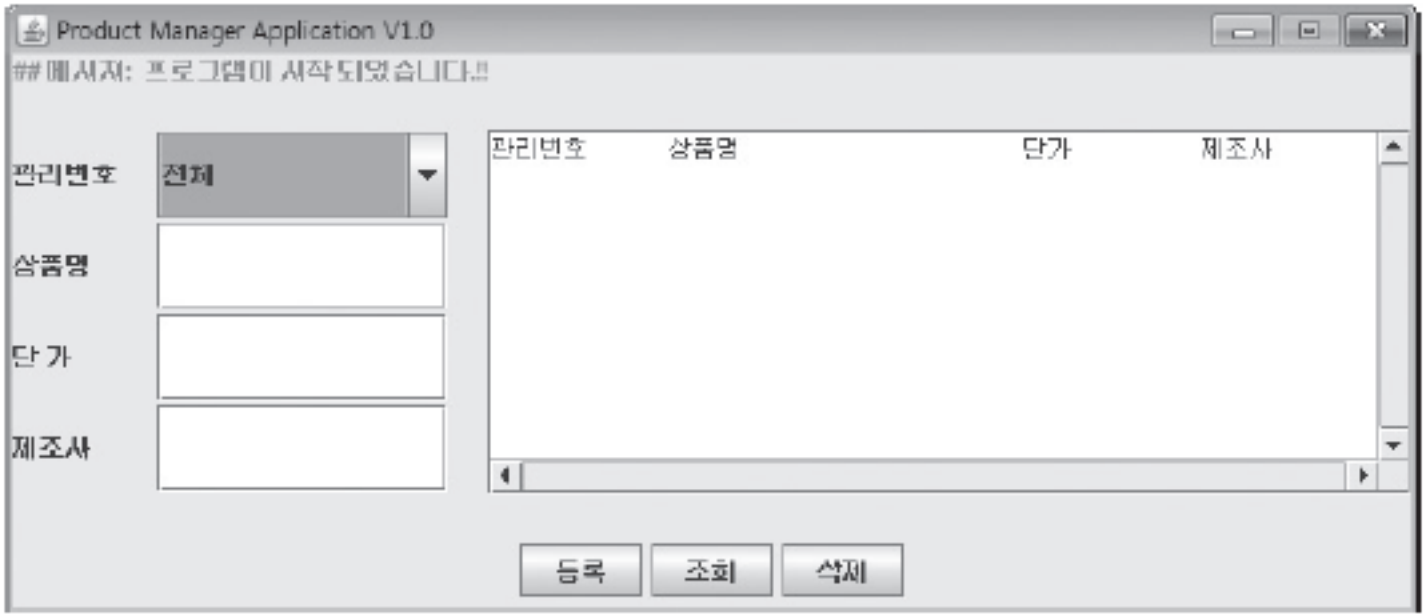
```
// 삭제 버튼일 때
else if(obj == b3) {
    String s = (String)cb.getSelectedItem();
    if(s.equals("전체")) {
        ml.setText(msg + "전체 삭제는 되지 않습니다!!");
    }
    else {
        if(dao.delProduct(Integer.parseInt(s))) {
            ml.setText(msg + "상품이 삭제되었습니다.!!");
        }
        else {
            ml.setText(msg + "상품이 삭제되지 않았습니다.!!");


        }
    }
    // 화면 데이터 갱신
    refreshData();
}
```

MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- 프로그램 실행 및 테스트

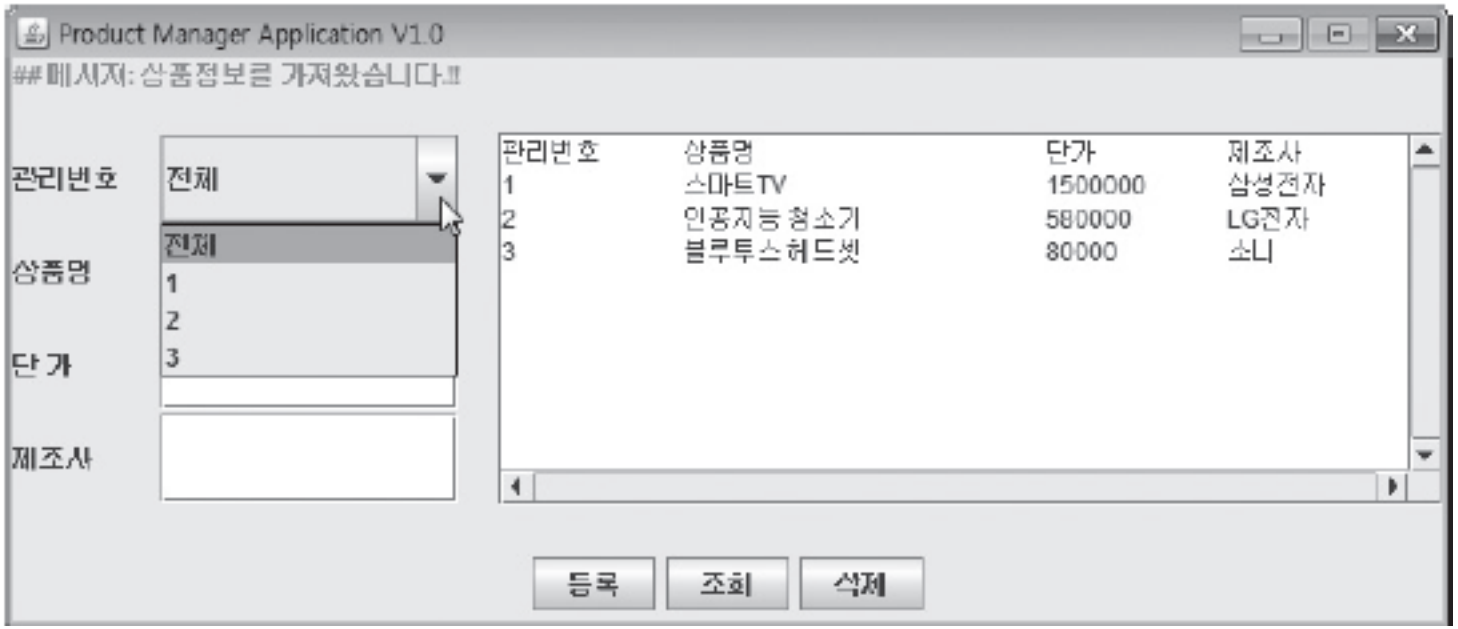
그림 10-29 최초 실행 화면



MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- 프로그램 실행 및 테스트

그림 10-30 등록 후 관리번호 선택



MVC 기반의 JDBC를 이용한 상품 관리 프로그램 작성

- 프로그램 실행 및 테스트

그림 10-31 두 번 상품 조회