

Java Project

Market 프로그램

개발자:김도영

개발기간:2024.04.15-2024.04.22

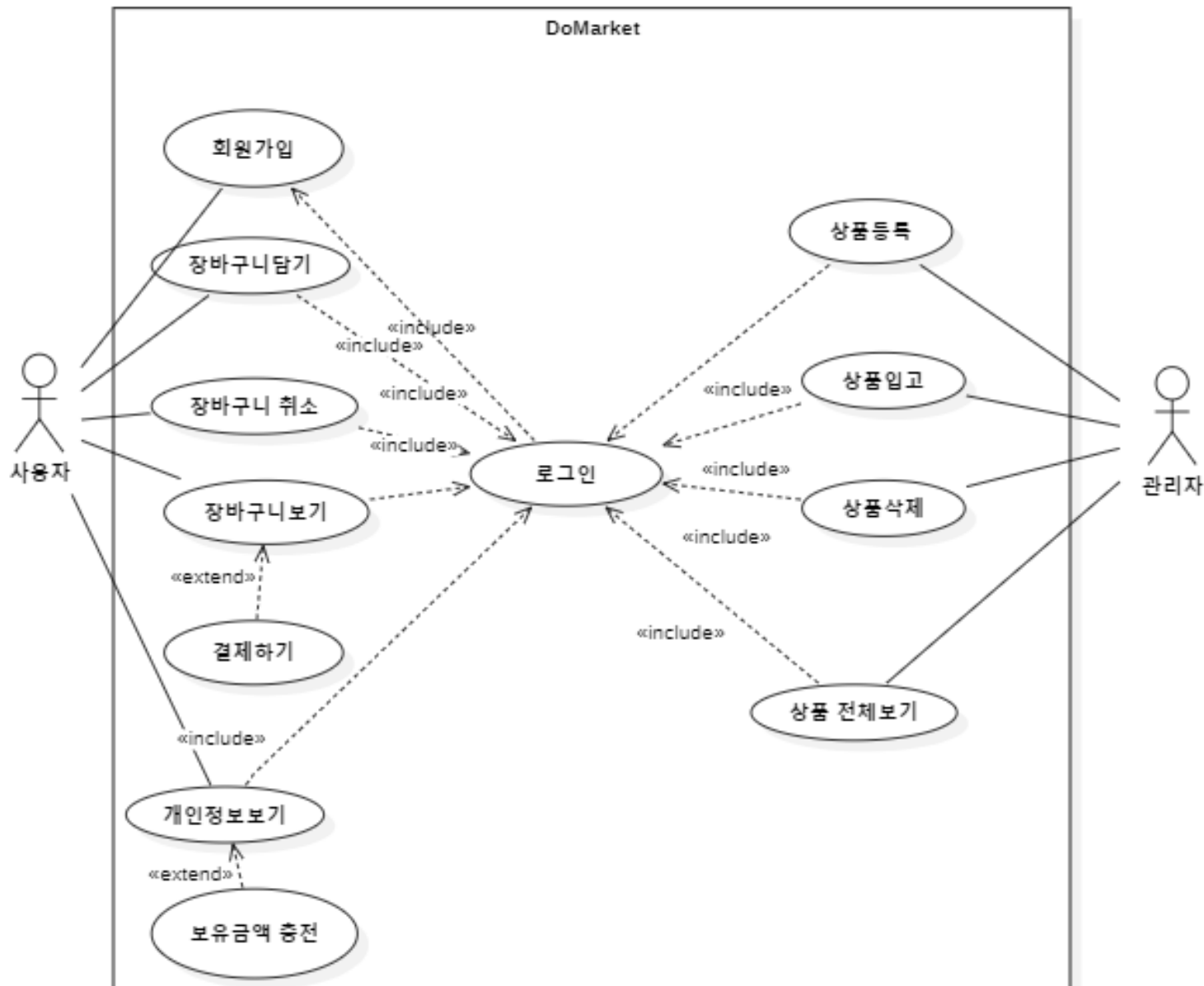
목차

- 001 — 요구사항분석
- 002 — 유스케이스
- 003 — Class Diagram
- 004 — ERD
- 005 — 코드 증빙 및 해설
- 006 — 시연영상

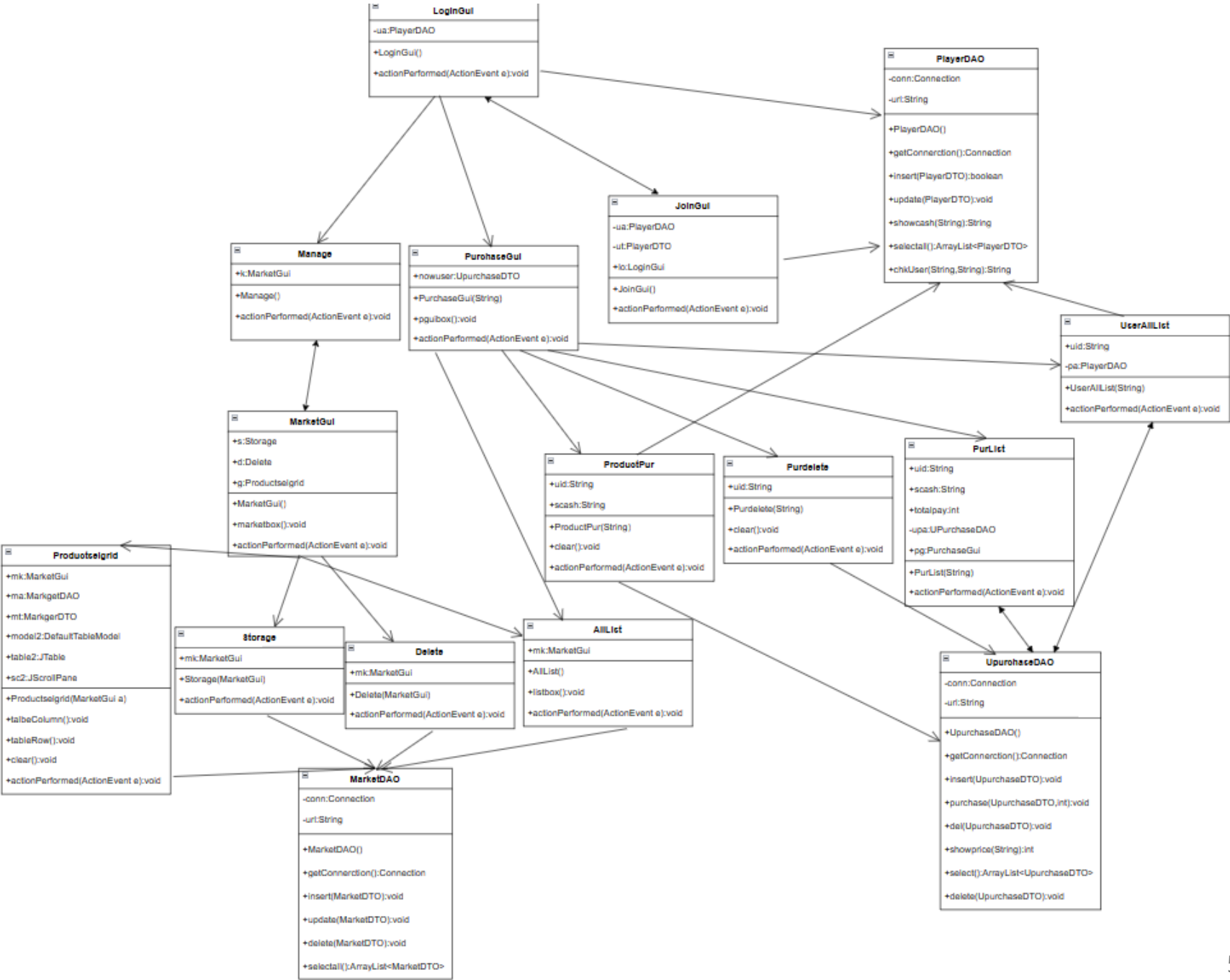
1. 요구사항 분석

| 구분 | 서비스 | 기능 | 기능설명 |
|-----|------|---------|---|
| 관리자 | 상품관리 | 상품등록 | 상품 등록 기능 (상품명,상품정보,수량,가격 입력) |
| | | 상품입고 | 상품 입고 기능 (상품명,수량 입력) |
| | | 상품삭제 | 상품 삭제 기능 (상품명 입력) |
| | | 상품전체보기 | 관리자가 등록한 상품 전체 보기 기능 |
| 사용자 | 상품구매 | 장바구니 담기 | 관리자가 등록한 상품 장바구니 등록 기능 (상품명,수량 입력) |
| | | 장바구니취소 | 장바구니에 담긴 상품취소 기능 (상품명, 수량 입력) |
| | | 장바구니 보기 | 장바구니 목록 확인 및 결제 기능 (결제하기 버튼 클릭) (장바구니목록,총결제금액 ,보유금액 확인가능) |
| | | 개인정보 보기 | 자신의 개인정보 확인 및 보유금액 충전기능 (충전할 금액 입력) |
| 공동 | 로그인 | 관리자 로그인 | 지정된 관리자 비밀번호 입력시 로그인 기능 |
| | | 사용자 로그인 | 회원가입 후 로그인 기능 |

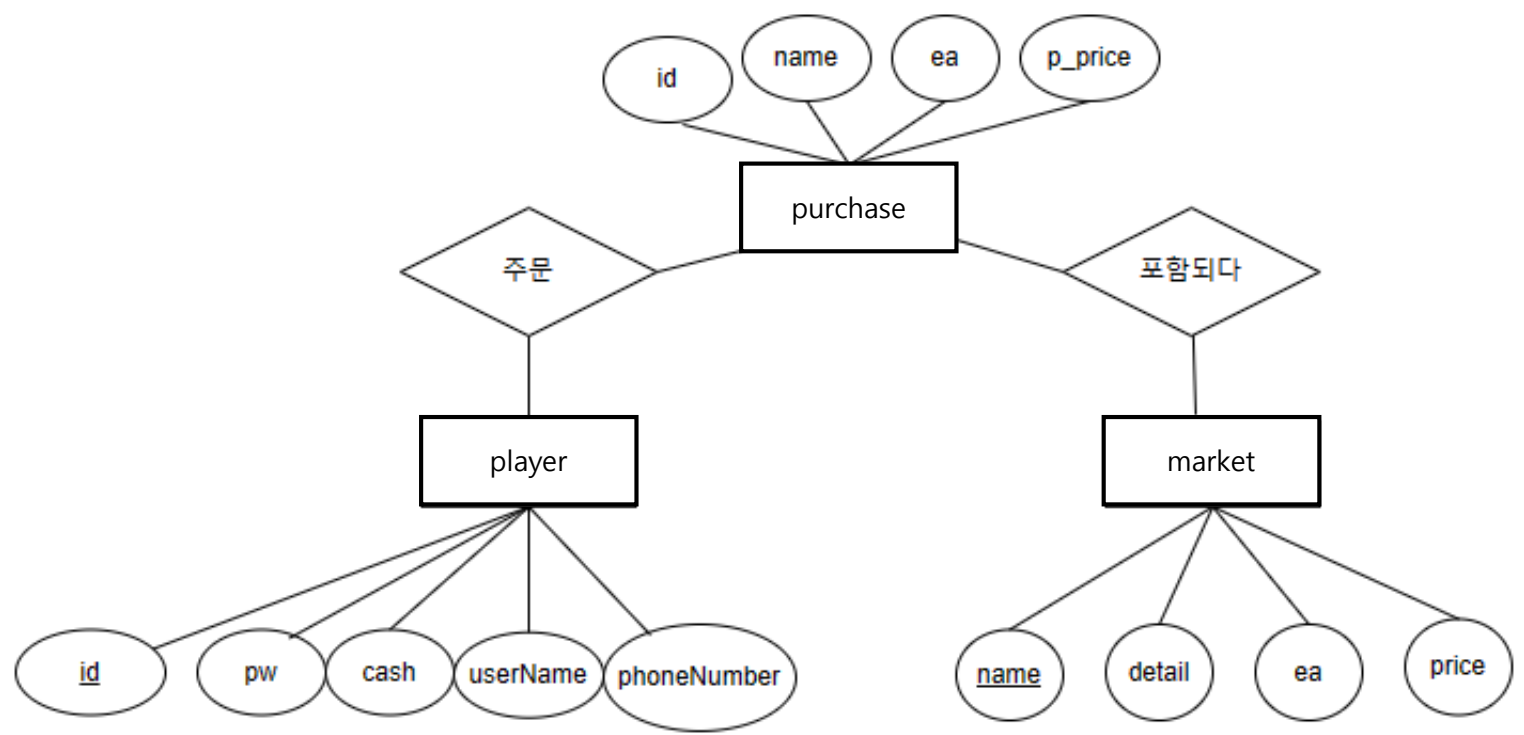
2. 유스케이스



3. Class Diagram

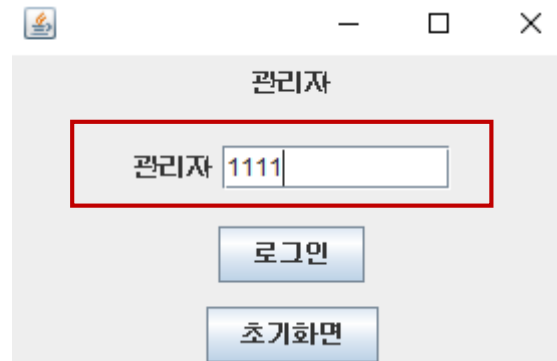
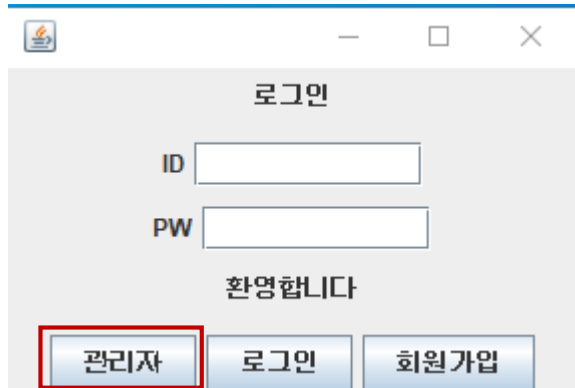


4. ERD



5. 코드 증빙 및 해설-관리자 로그인

ActionListener인터페이스의
actionPerformed메소드를 통하여 버튼을 클릭하였을 시
actionPerformed메소드를 실행시켜
해당 객체를 보여지게 함



LoginGui.java

```
public class LoginGui extends JFrame implements ActionListener {
```

```
    login.addActionListener(this);  
    join.addActionListener(this);  
    manage.addActionListener(this);
```

```
    @Override  
    public void actionPerformed(ActionEvent e) {  
        //관리자 버튼  
        if (e.getSource().equals(manage)) {  
            this.setVisible(false);  
            new Manage();  
        }  
    }
```

Manage.java

```
@Override  
public void actionPerformed(ActionEvent e) {  
    String getpw = pwt.getText();  
    if (e.getSource().equals(login)) {  
        if (getpw.equals("1111")) {  
            this.setVisible(false);  
            k.marketbox();  
        }  
    }  
}
```

MarketGui.java

```
public void marketbox() {  
    this.setVisible(true);  
}
```

5. 코드 증빙 및 해설-상품관리 화면

- 1.필드 변수에 객체를 생성
- 2.현재클래스의주소를 주입
3. ActionListener인터페이스의actionPerformed 메소드를 통하여 버튼을 클릭하였을시해당 객체를 보여지게 함



MarketGui.java

```
public class MarketGui extends JFrame implements ActionListener {  
    Storage s = new Storage(this);  
    Delete d = new Delete(this);  
    Productselgrid g = new Productselgrid(this);
```

```
@Override  
public void actionPerformed(ActionEvent e) {  
    if (e.getSource().equals(b1)) {  
        this.setVisible(false);  
        g.setVisible(true);  
    }  
    if (e.getSource().equals(b2)) {  
        this.setVisible(false);  
        s.setVisible(true);  
    }  
    if (e.getSource().equals(b3)) {  
        this.setVisible(false);  
        d.setVisible(true);  
    }  
    if (e.getSource().equals(b4)) {  
        new AllList();  
    }  
    if (e.getSource().equals(b6)) {  
        this.setVisible(false);  
        new LoginGui();  
    }  
}
```


5. 코드 증빙 및 해설-상품등록

1. 텍스트 필드의 값을 MarketDTO에 저장
2. 필드변수 MarketDAO의 메소드 insert 호출
3. Insert 메소드의 매개변수로 DTO주소 주입
4. Insert 메소드를 통하여 market테이블에 정보 저장

상품등록화면

상품명: 밤양갱 상품정보: 달달 수량: 20 가격: 600

상품등록 메뉴 지우개

| | NAME | DETAIL | EA | PRICE |
|---|------|--------|-----|-------|
| 1 | 감자칩 | 맛있는감자칩 | 100 | 1500 |

| | NAME | DETAIL | EA | PRICE |
|---|------|--------|-----|-------|
| 1 | 밤양갱 | 달달 | 20 | 600 |
| 2 | 감자칩 | 맛있는감자칩 | 100 | 1500 |

```
Productselgrid.java
public class Productselgrid extends JFrame implements ActionListener {
    DefaultTableModel model2 = new DefaultTableModel();
    JTable table2 = new JTable(model2);
    JScrollPane sc2 = new JScrollPane(table2);
    JButton en = new JButton("상품등록");
    MarketDAO ma = new MarketDAO();
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
    MarketDTO mt = new MarketDTO();
    if (e.getSource().equals(en)) {
        String getname = gname.getText();
        String getdetail = gdt.getText();
        int k = Integer.parseInt(gcat.getText());
        int p = Integer.parseInt(gpricet.getText());
        mt.setName(getname);
        mt.setDetail(getdetail);
        mt.setEa(k);
        mt.setPrice(p);
        ma.insert(mt);
        wel.setText("등록완료");
        tableRow();
    }
}
```

```
MarketDAO.java
public void insert(MarketDTO mt) {
    String sql = "insert into market values (?, ?, ?, ?)";
}
```

5. 코드 증빙 및 해설-상품등록

5. MarketDAO의 selectAll 메서드를 통하여
오라클 market테이블의 정보가 저장된 의 값을 자료형ArrayList 인 변수 g에 주입
- 6.Jtable 의 행을 만들기 위해 필드변수 MarketDAO의 메소드 selectAll 호출
- 7.호출을 통해 리턴값 g 와 확장된 for문을 이용하여 jTable 행에 정보 저장 하여 상품등록을 확인할수도록함

```
MarketDAO.java  
  
public ArrayList<MarketDTO> selectAll() {  
    ArrayList<MarketDTO> g = new ArrayList<>();  
    if (getConnection() != null) {  
        try {  
            Statement st = null;  
            ResultSet rs = null;  
            String sql = "select * from market";  
            st = conn.createStatement();  
            rs = st.executeQuery(sql);  
            while (rs.next()) {  
                String name = rs.getString("name");  
                String detail = rs.getString("detail");  
                int ea = rs.getInt("ea");  
                int price = rs.getInt("price");  
                MarketDTO newg = new MarketDTO();  
                newg.setName(name);  
                newg.setDetail(detail);  
                newg.setEa(ea);  
                newg.setPrice(price);  
                g.add(newg);  
            }  
        } catch (Exception e) {  
            // TODO: handle exception  
        }  
    }  
    return g;  
}
```



```
Productselgrid.java  
  
public void talbeColumn() {  
    model2.addColumn("상품명");  
    model2.addColumn("상품정보");  
    model2.addColumn("수량");  
    model2.addColumn("가격");  
}  
  
public void tableRow() {  
    ArrayList<MarketDTO> g2 = ma.selectAll();  
    for (MarketDTO temp2 : g2) {  
        String getName2 = temp2.getName();  
        String getdetail2 = temp2.getDetail();  
        int getea2 = temp2.getEa();  
        int getprice2 = temp2.getPrice();  
        model2.addRow(new Object[] { getName2, getdetail2, getea2, getprice2 });  
    }  
    m[4].add(sc2);  
}
```



상품등록화면


상품명 상품정보 수량 가격

등록완료

| 상품명 | 상품정보 | 수량 | 가격 |
|-----|--------|-----|------|
| 밤양갱 | 달달 | 20 | 600 |
| 감자칩 | 맛있는감자칩 | 100 | 1500 |

5. 코드 증빙 및 해설-상품입고

1. 텍스트필드의 값을 MarketDTO에 저장
2. 필드변수 MarketDAO의 메소드 update 호출
3. update 메소드의 매개변수로 DTO주소 주입
4. update 메소드를 통하여 market테이블의 해당 상품명의 수량 추가



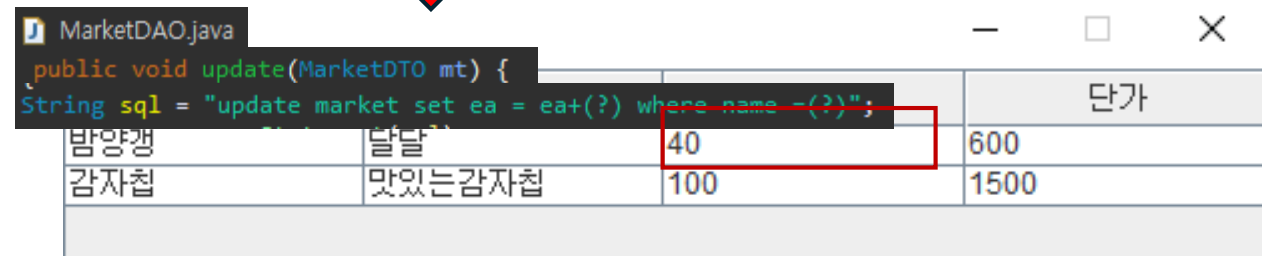
상품입고 메뉴

상품명 밤양갱

추가 수량 20

입고완료

```
Storage.java
@Override
public void actionPerformed(ActionEvent e) {
    MarketDAO ma = new MarketDAO();
    MarketDTO mt = new MarketDTO();
    if (e.getSource().equals(button)) {
        String getname = gname.getText();
        int k = Integer.parseInt(geat.getText());
        mt.setName(getname);
        mt.setEa(k);
        ma.update(mt);
        wel.setText("입고완료");
        JOptionPane.showMessageDialog(null, "입고되었습니다", "상품입고", JOptionPane.PLAIN_MESSAGE);
    }
}
```

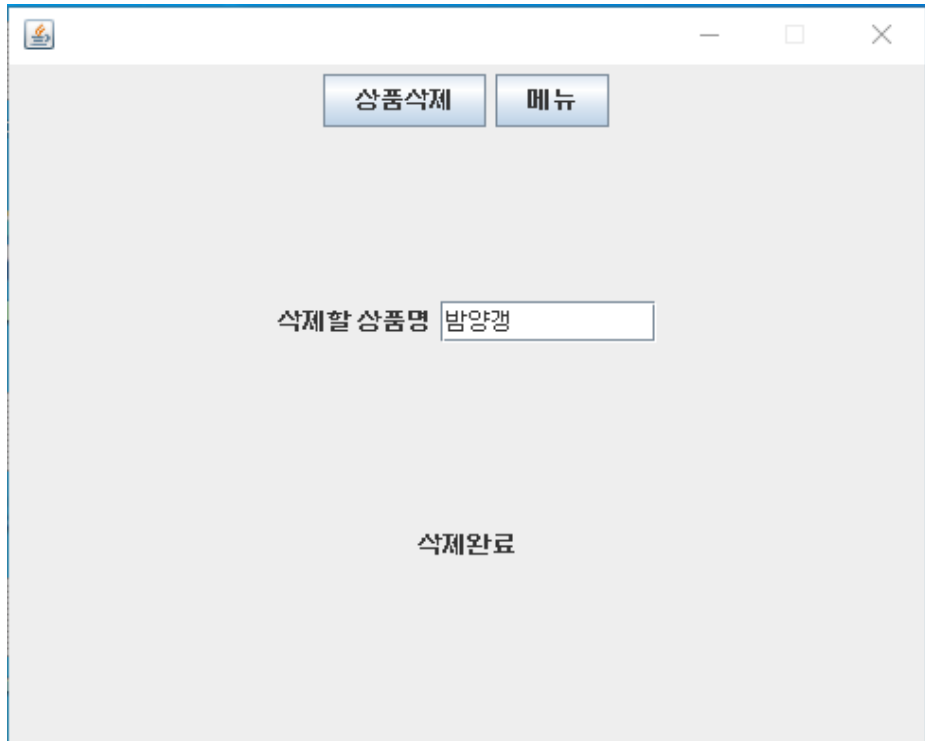


```
MarketDAO.java
public void update(MarketDTO mt) {
    String sql = "update market set ea = ea+(?) where name = (?)";
    ...
}
```

| | | | 단가 |
|-----|--------|-----|------|
| 밤양갱 | 달달 | 40 | 600 |
| 감자칩 | 맛있는감자칩 | 100 | 1500 |

5. 코드 증빙 및 해설-상품삭제

1. 텍스트필드의 값을 MarketDTO에 저장
2. 지역변수 MarketDAO의 메소드 delete 호출
3. delete 메소드의 매개변수로 DTO주소 주입
4. delete 메소드를 통하여 market테이블의 해당 상품명 튜플 삭제



```
Delete.java
@Override
public void actionPerformed(ActionEvent e) {
    MarketDAO ma = new MarketDAO();
    MarketDTO mt = new MarketDTO();
    if (e.getSource().equals(button)) {
        String dname = dnamet.getText();
        mt.setName(dname);
        ma.delete(mt);
        wel.setText("삭제완료");
        JOptionPane.showMessageDialog(null, "삭제되었습니다", "상품삭제", JOptionPane.PLAIN_MESSAGE);
    }
}
```

MarketDAO.java

```
public void delete(MarketDTO mt) {  
    String sql = "delete from market where name =(?)";
```

| 상품명 | 수량 | 단가 | |
|-----|--------|-----|------|
| 감사칩 | 맛있는감사칩 | 100 | 1500 |

5. 코드 증빙 및 해설-회원가입

1. 텍스트 필드의 값을 PlayerDTO에 저장
2. 필드 변수 PlayerDAO의 메소드 insert 호출
3. Insert 메소드의 매개변수로 DTO주소 주입
4. Insert 메소드를 통하여 player테이블에 정보 저장

LoginGui.java

```
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource().equals(join)) { // 회원가입 버튼
        this.setVisible(false);
        new JoinGui();
    }
}
```

JoinGui.java

```
public class JoinGui extends JFrame implements ActionListener {
    JButton jb = new JButton("회원가입");
    private PlayerDTO ut = new PlayerDTO();
    private PlayerDAO ua = new PlayerDAO();
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
    String getId = idt.getText();
    String getPw = pwt.getText();
    String getUsername = unt.getText();
    String pnun = pnt.getText();
    if (e.getSource().equals(jb)) { // 회원가입 버튼
        ut.setId(getId);
        ut.setPw(getPw);
        ut.setUsername(getUsername);
        ut.setPhonenumber(pnun);
        if (ua.insert(ut)) {

```

PlayerDAO.java

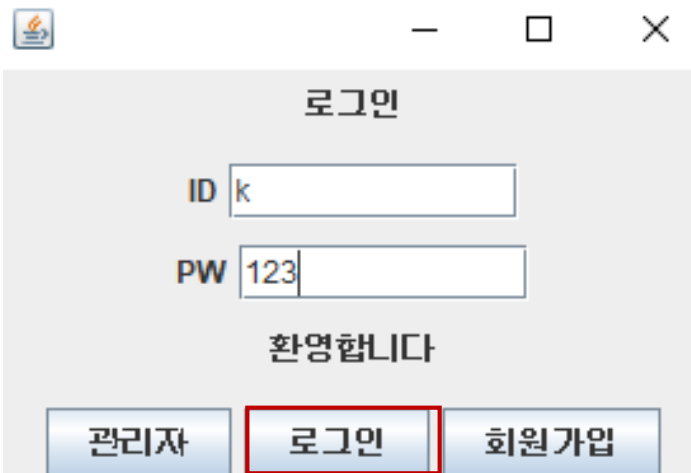
```
public boolean insert(PlayerDTO ut) {
    String sql = "insert into player values (?, ?, ?, ?, ?)";
}
```

Player table

| | ID | PW | USERNAME | PHONENUMBER | CASH |
|---|----|-----|----------|---------------|------|
| 1 | k | 123 | 김도영 | 010-7688-8774 | 0 |

5. 코드 증빙 및 해설-로그인

1. 텍스트필드의 값을 변수 getid, pw에 저장
2. 필드변수 PlayerDAO의 메소드 chkUser호출
3. chkUser 메소드의 매개변수로 변수 대입
4. chkUser 메소드를 통하여 player테이블의 id, pw와 동일한지 확인후 로그인
5. 현재 아이디를 purchaseGui의 매개변수로 대입하여 market 화면으로 들어가게 함



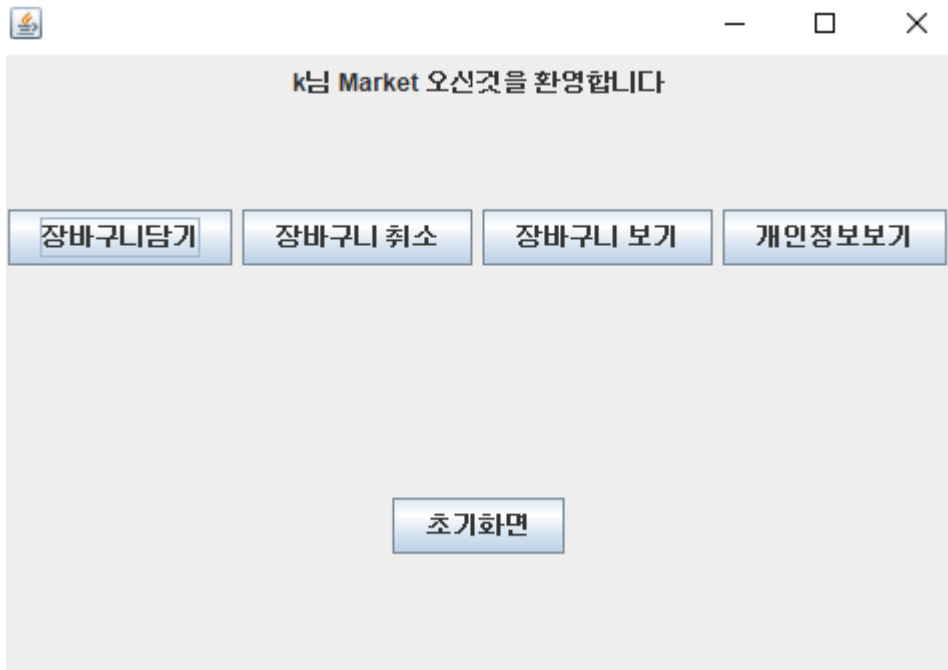
```
LoginGui.java
else if (e.getSource().equals(login)) { // 로그인버튼
    wel.setText(getid + "/" + getpw + "님 로그인");
    if (ua.chkUser(getid, getpw) != null) {
        this.setVisible(false);
        new PurchaseGui(getid);
    } else {
        System.out.println("실패");
    }
}
```



```
PlayerDAO.java
public String chkUser(String id, String pw) {
    if (getConnection() != null) {
        try {
            Statement st = null;
            ResultSet rs = null;
            String sql = "select * from player";
            st = conn.createStatement();
            rs = st.executeQuery(sql);
            while (rs.next()) {
                String cid = rs.getString("id");
                String cpw = rs.getString("pw");
                if (cid.equals(id) && cpw.equals(pw)) {
                    System.out.println(id + "님 로그인 하셨습니다");
                    return id;
                }
            }
        } catch (Exception e) {
            System.out.println("일련 오류");
        }
    }
    return null;
}
```

5. 코드 증빙 및 해설-장바구니화면

- 1.필드변수에 현재아이디를 저장하기위한 nowuser변수 선언
- 2.매개변수를 통하여 현재아이디를 변수 nowuser에 저장
- 3.ActionListener 인터페이스의 actionPerformed메소드를 통하여 버튼을 클릭하였을시 객체를 생성하여 해당 객체를 보여지게함



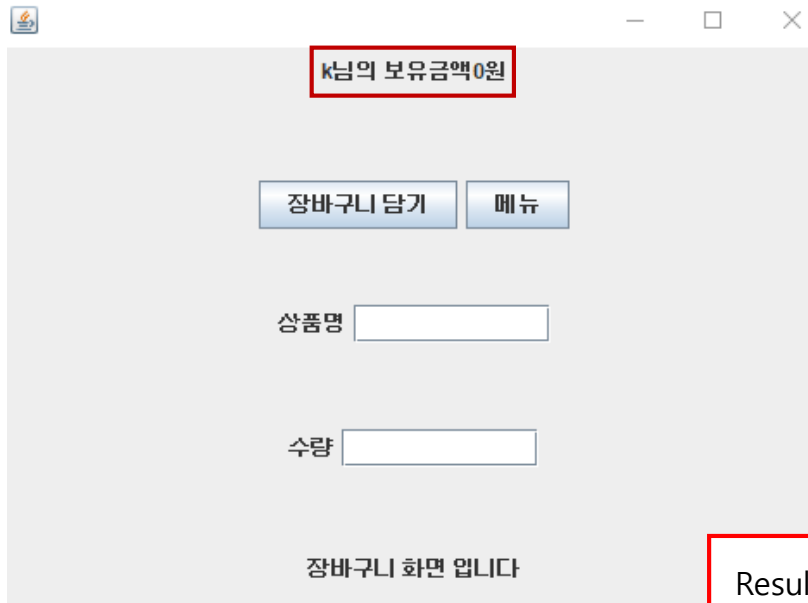
```
PurchaseGui.java
UpurchaseDTO nowuser;

public PurchaseGui(String uid) {
    UpurchaseDTO a = new UpurchaseDTO();
    title.setText(uid+"님 Market 오신것을 환영합니다");
    a.setId(uid);
    nowuser = a;
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource().equals(b1)) { //장바구니담기
        new ProductPur(nowuser.getId());
        new Alllist();
        this.setVisible(false);
    }
    if (e.getSource().equals(b2)) { //장바구니취소
        this.setVisible(false);
        new Purdelete(nowuser.getId());
    }
    if (e.getSource().equals(b3)) { //장바구니보기 및 결제
        this.setVisible(false);
        new PurList(nowuser.getId());
    }
    if (e.getSource().equals(b4)) { //개인정보보기 및 충전
        this.setVisible(false);
        new UserAlllist(nowuser.getId());
    }
    if (e.getSource().equals(b5)) { //초기화면
        this.setVisible(false);
        new LoginGui();
    }
}
```

5. 코드 증빙 및 해설-장바구니담기(보유금액)

1. 현재 id 저장 변수 uid 와 보유금액 저장 변수 scash를 필드변수에 선언
2. 매개변수를 통하여 현재 아이디를 uid에 저장
3. PlayerDAO객체를 생성하여 DAO의 메소드 showcash 호출
4. 메서드를 호출하여 현재 보유금액이 저장된 리턴값 scash를 변수 scash에 대입후 현재 보유금액을 라벨에 저장



```
ProductPur.java
public class ProductPur extends JFrame implements ActionListener {
    String uid = null; // 현재 id 저장 변수
    String scash = null; // 보유금액 저장 변수
    public ProductPur(String id) { // 생성자
        uid = id;
        PlayerDAO pad = new PlayerDAO();
        scash = pad.showcash(uid);
        cucash.setText(uid + "님의 보유금액" + scash + "원"); // 해당아이디 보유금액
    }
}
```

```
PlayerDAO.java
public String showcash(String id) {
    ResultSet rs = null;
    PreparedStatement ps = null;
    String scash = null;
    if (getConnection() != null) {
        try {
            String sql = "select cash from player where id = (?)";
            ps = conn.prepareStatement(sql);
            ps.setString(1, id);
            ps.executeUpdate();
            rs = ps.executeQuery(sql);
            while (rs.next()) {
                scash = rs.getString("cash" + "");
            }
        } catch (Exception e) {
            // TODO: handle exception
        }
    }
    return scash;
}
```

ResultSet의 메서드 next를 참조하여
오라클의 보유금액이 저장된 값을
한 행단위로 불러와 결과 값을
변수 scash에 대입하여 현재 보유금액을
저장후 그 값을 리턴

5. 코드 증빙 및 해설-장바구니담기

1. 텍스트 필드의 값을 UpurchaseDTO에 저장
2. UpurchaseDAO객체 생성후 DAO의메소드 insert 호출
3. Insert 메소드의 매개변수로 DTO주소 주입
4. Insert 메소드를 통하여 결제금액을 계산후 해당 금액을 purchase테이블에 정보 저장
5. eaupdate 메소드의 매개변수로 DTO주소 주입
6. eaupdate 메소드를 통하여 market테이블의 해당 상품명의 수량 감소

```
ProductPur.java
@Override
public void actionPerformed(ActionEvent e) {
    UpurchaseDAO pa = new UpurchaseDAO();
    UpurchaseDTO pt = new UpurchaseDTO();
    if (e.getSource().equals(button)) {
        if (Integer.parseInt(scash) > 0) {
            String getname = gname.getText();
            int getea = Integer.parseInt(geat.getText());
            pt.setName(getname);
            pt.setEa(getea);
            pt.setId(uid);
            pa.insert(pt);
            pa.eaupdate(pt);
        }
    }
}
```

장바구니 담기

장바구니 담기 메뉴

상품명 감자칩

수량 3

장바구니 화면입니다

Purchase table

| ID | NAME | EA | P_PRICE |
|----|------|-----|---------|
| 1 | k | 감자칩 | 3 4500 |

market table

| NAME | DETAIL | EA | PRICE |
|------|--------|--------|----------|
| 1 | 감자칩 | 맛있는감자칩 | 100 1500 |

market table

| NAME | DETAIL | EA | PRICE |
|------|--------|--------|---------|
| 1 | 감자칩 | 맛있는감자칩 | 97 1500 |

```
UpurchaseDAO.java
public void insert(UpurchaseDTO p) { // 장바구니 담기
    String sql = "select price from market where name =(?)"; // 해당 상품 단가 가져오기
    int pr = 0;
    while (rs.next()) {
        pr = rs.getInt("price");
    }
    int pp = pr * p.getEa(); // 단가 * 입력한 수량 = 결제 금액

    String sql2 = "insert into purchase values(?,?,?,?)"; // 장바구니 테이블 insert
    ps = conn.prepareStatement(sql2);
    ps.setString(1, p.getId());
    ps.setString(2, p.getName());
    ps.setInt(3, p.getEa());
    ps.setInt(4, pp);

    UpurchaseDAO.java
    public void eaupdate(UpurchaseDTO p) { // 수량 감소
        String sql3 = "update market set ea = ea-(?) where name =(?)";
    }
}
```

5. 코드 증빙 및 해설-장바구니담기

k님의 보유금액 10000원

장바구니 담기 메뉴

상품명 감자칩

수량 3

장바구니 화면 입니다



| 장바구니 | | | |
|------|--------|--------|-------|
| id | 구매한 상품 | 구매한 수량 | 결제 금액 |
| k | 감자칩 | 3 | 4500 |

5. 코드 증빙 및 해설-개인정보보기 및 충전

1. 텍스트 필드의 값을 형변환을 통하여 자료형 int 변수 money에 저장
2. 변수 money를 PlayerDTO에 저장
3. 필드변수 PlayerDAO의 메소드 update호출
4. update 메소드의 매개변수로 DTO주소 주입
5. update 메소드를 통하여 player테이블의 보유금액 을 추가

개인정보

| id | 이름 | 전화번호 | 보유금액 |
|----|-----|---------------|------|
| k | 김도영 | 010-7688-8774 | 0 |

충전

메뉴

Player table

| ID | PW | USERN... | PHONENUMBER | CASH |
|-----|----|----------|---------------|------|
| 1 k | 1 | 김도영 | 010-7688-8774 | 0 |

↓

Player table

| ID | PW | USERNAME | PHONENUMBER | CASH |
|-----|----|----------|---------------|-------|
| 1 k | 1 | 김도영 | 010-7688-8774 | 10000 |

```
UserAllList.java
public class UserAllList extends JFrame implements ActionListener {
    PlayerDAO pa = new PlayerDAO();

    @Override
    public void actionPerformed(ActionEvent e) {
        PlayerDTO pt = new PlayerDTO();
        if (e.getSource().equals(charge)) {
            wel.setText("충전되었습니다.");
            int money = Integer.parseInt(ch.getText());
            pt.setCash(money);
            pt.setId(uid);
            pa.update(pt);
        }
    }
}
```

```
UpurchaseDAO.java
public void update(PlayerDTO pt) {
    ResultSet rs = null;
    PreparedStatement ps = null;
    if (getConnection() != null) {
        try {
            String sql2 = "update player set cash = cash+(?) where id =(?)";
            ps = conn.prepareStatement(sql2);
            ps.setInt(1, pt.getCash());
            ps.setString(2, pt.getId());
            ps.executeUpdate();
        } catch (Exception e) {
            // TODO: handle exception
        }
    }
}
```

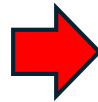
5. 코드 증빙 및 해설-개인정보보기 및 충전

개인정보

| id | 이름 | 전화번호 | 보유금액 |
|----|-----|---------------|------|
| k | 김도영 | 010-7688-8774 | 0 |

10000 충전

메뉴

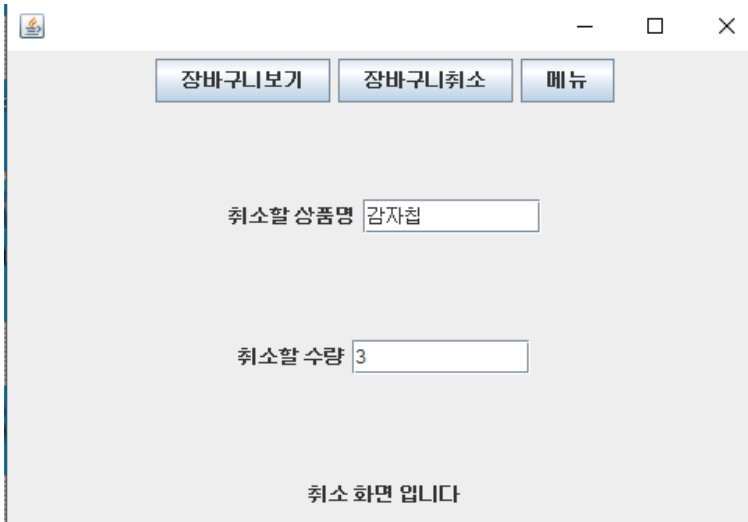


개인정보

| id | 이름 | 전화번호 | 보유금액 |
|----|-----|---------------|-------|
| k | 김도영 | 010-7688-8774 | 10000 |

5. 코드 증빙 및 해설-장바구니 취소

1. 텍스트 필드의 값을 UpurchaseDTO에 저장(ea 는 int 형으로 넘겨줘야되기때문에 형변환후 DTO저장)
2. 지역변수 UpurchaseDAO의 메소드 delete 호출
3. Delete 메소드의 매개변수로 DTO주소 주입
4. Delete 메소드를 통하여 market테이블의 해당 상품명 수량 복구 , purchase 테이블의 해당 상품명들의 튜플 삭제



market table

| | NAME | DETAIL | EA | PRICE |
|---|------|--------|----|-------|
| 1 | 감자칩 | 맛있는감자칩 | 97 | 1500 |

↓

market table

| | NAME | DETAIL | EA | PRICE |
|---|------|--------|-----|-------|
| 1 | 감자칩 | 맛있는감자칩 | 100 | 1500 |

↓

Purchase table

| | ID | NAME | EA | P_PRICE |
|---|----|------|----|---------|
| 1 | k | 감자칩 | 3 | 4500 |

↓

Purchase table

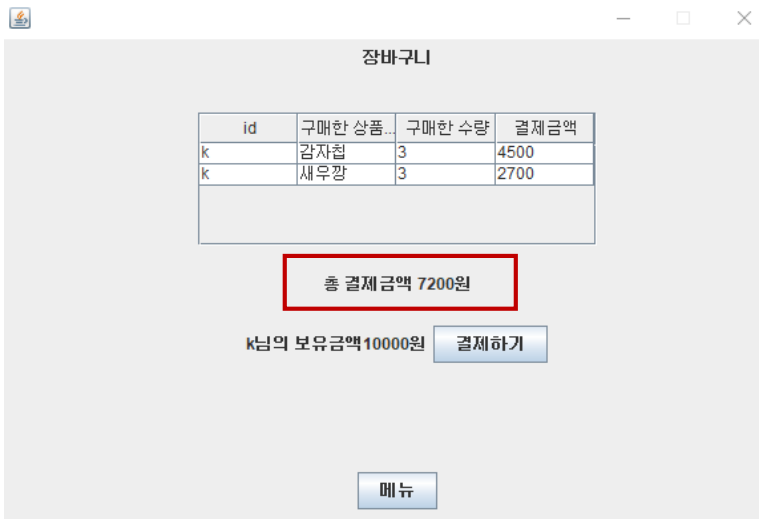
| | ID | NAME | EA | P_PRICE |
|--|----|------|----|---------|
| | | | | |

```
Purdelete.java
@Override
public void actionPerformed(ActionEvent e) {
    UpurchaseDAO pa = new UpurchaseDAO();
    UpurchaseDTO pt = new UpurchaseDTO();
    if (e.getSource().equals(button)) {
        String getname = dname.getText();
        int getea = Integer.parseInt(deat.getText());
        pt.setName(getname);
        pt.setEa(getea);
        pa.delete(pt);
        wel.setText("취소 완료");
        JOptionPane.showMessageDialog(null, "취소 되었습니다", "취소", JOptionPane.PLAIN_MESSAGE);
    }
}
```

```
UpurchaseDAO.java
public void delete(UpurchaseDTO p) { // 장바구니 취소 기능
    if (getConnection() != null) {
        try {
            PreparedStatement ps = null;
            String sql = "update market set ea = ea + (?) where name = (?)";
            ps = conn.prepareStatement(sql);
            ps.setInt(1, p.getEa());
            ps.setString(2, p.getName());
            ps.executeUpdate();
            String sql2 = "delete from purchase where name = (?) and ea = (?)";
            ps = conn.prepareStatement(sql2);
            ps.setString(1, p.getName());
            ps.setInt(2, p.getEa());
            ps.executeUpdate();
        } catch (Exception e) {
            // TODO: handle exception
        }
    }
}
```

5. 코드 증빙 및 해설-장바구니 보기 및 결제(총 결제금액)

1. 총 결제금액 저장변수 totalpay 를 필드변수에 선언
2. 매개변수를 통하여 현재 아이디를 uid에 저장
3. 필드변수 UpurchaseDAO의 메소드 showprice 호출
4. 메서드를 호출하여 총결제금액이 저장된 리턴값 shprice 를 변수 totalpay 대입 후 총결제금액을 라벨에 저장



```
PurList.java
public class PurList extends JFrame implements ActionListener {
    String uid = null; // 현재 id 저장 변수
    String scash = null; // 보유금액 저장 변수
    int totalpay; // 총 결제 금액 저장 변수
    UpurchaseDAO upa = new UpurchaseDAO();

    public PurList(String id) {
        uid = id;
        PlayerDAO pa = new PlayerDAO();
        scash = pa.showcash(uid); // 해당 id 보유금액
        System.out.println("보유금액" + scash); //
        totalpay = upa.showprice(uid); // 해당 id 결제 금액
        pay.setText("총 결제금액 " + totalpay + "원");
        cucash.setText(uid + "님의 보유금액" + scash + "원");
    }
}
```



```
UpurchaseDAO.java
public int showprice(String id) { // 총결제금액 계산하는 메서드
    ResultSet rs = null;
    PreparedStatement ps = null;
    int shprice = 0;
    if (getConnection() != null) {
        try {
            String sql = "select p_price from purchase where id =(?)";
            ps = conn.prepareStatement(sql);
            ps.setString(1, id);
            ps.executeUpdate();
            rs = ps.executeQuery();
            while (rs.next()) {
                shprice = shprice + rs.getInt("p_price"); // 총결제금액으로 계산
            }
        } catch (Exception e) {
            // TODO: handle exception
        }
    }
    return shprice;
}
```

ResultSet의 메서드 next 를 참조하여 오라클의 결제금액이 저장된 값을 한 행단위로 불러와 결과 값을 변수 shprice 에 누적으로 대입하여 총결제금액을 저장후 그값을 리턴

5. 코드 증빙 및 해설-장바구니 보기 및 결제

1. UpurchaseDTO객체를 생성후 DTO에 현재 아이디 저장된 변수 uid를 저장
- 2.필드변수 UpurchaseDAO의메소드 purchase,del호출
- 3.purchase 메소드의 매개변수로 totalpay와 DTO주소 주입
- 4.purchase메소드를 통하여 보유금액 - 총결제금액 을 통해 보유금액차감
- 5.del 메소드의 매개변수로 DTO주소 주입
- 6.del 메소드를 통하여 purchase테이블의 해당 id의 튜플 삭제

```
PurList.java
String uid = null; // 현재 id 저장 변수
String scash = null; // 보유금액 저장 변수
int totalpay; // 총 결제 금액 저장 변수
UpurchaseDAO upa = new UpurchaseDAO();
```

```
@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource().equals(pur)) {
        if (Integer.parseInt(scash) - totalpay >= 0) {
            UpurchaseDTO upt = new UpurchaseDTO();
            upt.setId(uid);
            upa.purchase(upt, totalpay);
            upa.del(upt);
            wei.setText("구매 완료");
        }
    }
}
```

장바구니

| id | 구매한 상품... | 구매한 수량 | 결제금액 |
|----|-----------|--------|------|
| k | 감자칩 | 3 | 4500 |
| k | 새우깡 | 3 | 2700 |

총 결제금액 7200원

k님의 보유금액 10000원 결제하기

메뉴

| player table | | | | | |
|--------------|----|----------|-------------|---------------|-------|
| ID | PW | USERNAME | PHONENUMBER | CASH | |
| 1 | k | 1 | 김도영 | 010-7688-8774 | 10000 |

player table

| ID | PW | USERNAME | PHONENUMBER | CASH | |
|----|----|----------|-------------|---------------|------|
| 1 | k | 1 | 김도영 | 010-7688-8774 | 2800 |

| purchase table | | | | |
|----------------|----|------|----|---------|
| | ID | NAME | EA | P_PRICE |
| 1 | k | 감자칩 | 3 | 4500 |
| 2 | k | 새우깡 | 3 | 2700 |

purchase table

| ID | NAME | EA | P_PRICE |
|----|------|----|---------|
|----|------|----|---------|

```
UpurchaseDAO.java
public void purchase(UpurchaseDTO p, int totalpay) { // 상품구매시 보유금액-총결제금액
    String sql = "update player set cash = cash-(?) where id =(?)";
    ps = conn.prepareStatement(sql);
    ps.setInt(1, totalpay);
    ps.setString(2, p.getId());
}
```

```
UpurchaseDAO.java
public void del(UpurchaseDTO p) { // 상품구매시 장바구니목록 삭제
    String sql = "delete from purchase where id =(?)";
}
```

5. 코드 증빙 및 해설-장바구니 보기 및 결제

장바구니

| id | 구매한 상품... | 구매한 수량 | 결제금액 |
|----|-----------|--------|------|
| k | 감자칩 | 3 | 4500 |
| k | 새우깡 | 3 | 2700 |

총 결제금액 7200원

k님의 보유금액 10000원

결제하기

메뉴



장바구니

| id | 구매한 상품... | 구매한 수량 | 결제금액 |
|----|-----------|--------|------|
|----|-----------|--------|------|

총 결제금액 0원

k님의 보유금액 2800원

결제하기

6. 시연영상

https://drive.google.com/file/d/1DU63-4ZsVouOuZpokxUldG-yDIB3PQwX/view?usp=drive_link

Thank you

Market 프로그램