

2021학년도 자바 프로젝트 완료 보고서

CAFE POS(카페 포스기)

2021년 12월 15일

컴퓨터정보과

팀명	동강
팀장	한재근(202044029)
팀원	1. 한재근(202044029) 2. 이준혁(202044021)



2021년도
컴퓨터정보과 특성화 사업
자바 프로젝트 진행 결과보고서
(분야 : 원도우 응용프로그램)

연구과제명 : 카페 포스기 원도우 응용프로그램 개발

2021. 12. 15



본 결과물은 인하공업전문대학 컴퓨터정보과 자바 프로젝트 수업의 연구 결과입니다.

제 출 문

- 분야 : 자바 윈도우 응용프로그램
- 연구과제명 : 카페 포스기 윈도우 응용프로그램
- 프로젝트 책임자 : 인하공업전문대학 컴퓨터정보과 한재근

2021년도 인하공업전문대학 컴퓨터정보과 자바 프로젝트 수업의
결과물로 이 보고서를 제출합니다.

2021. 12. 15

프로젝트 책임자 : 컴퓨터정보과(202044029) 한재근 (인)

공동 참여자 : 컴퓨터정보과(202044021) 이준혁 (인)

인하공업전문대학 컴퓨터정보과 학과장 귀하

프로젝트 요약

분야	자바 윈도우 응용프로그램 개발
프로젝트명	자바를 이용한 카페 POS 응용 프로그램
프로젝트 책임자	컴퓨터정보과(202044029) 한재근
연구 내용	
<ul style="list-style-type: none">□ 연구 주제 선정<ul style="list-style-type: none">- 버튼식 Event Handler에 대해 연구하며 어플리케이션 개발을 기초부터 응용까지 학습할 수 있는 주제를 고민하다가 POS기가 적합하다고 생각하여 주제를 정하게 되었다.□ 이론적 배경 및 선행연구<ul style="list-style-type: none">- Swing GUI Form을 활용하고 Java를 이용한 어플리케이션 개발 연구- 실제 POS기처럼 활용할 수 있는 어플리케이션의 형태로 제작하기 위해 Event Handler 학습, Java 기반 언어 학습□ 연구 방법<ul style="list-style-type: none">- 연구 주제의 필요성에 근거한 토의와 자료조사/시장조사를 통해 화면구성 틀을 제작할 수 있었다.- 이를 바탕으로 POS기를 편리하게 사용할 수 있는 어플리케이션 형태로 만들기 위해 사용자 관점에서 고민하며 프로그래밍 언어를 학습하며 하나씩 개발해 나감.<p>※ 제한점 : POS기에서 결제 방법을 현금으로 결제시 거스름돈으로 계산하여 반환이 되지만 카드 결제시에는 TextFieldArea에 text의 font를 바코드로 변환하여 바코드 형식으로 생성하려고 했지만 컴퓨터 내부에서 font를 인식하지 못하여 만들어내지 못했다.</p>□ 연구 활동 및 과정<ul style="list-style-type: none">- 기한을 정해 개발요소를 회의하고 개별 DB를 생성하여 개발 후 간트차트에 있는 일정에 맞게 미팅을 하며 서로에 대한 피드백에 대해 토의하고 진행해 나감.- DB 연동을 하면서 개별적으로 생성했던 DB를 합치기 위해 토의하고 이를 반영하기 위해 오류를 잡고 개발해나감.- 로그인, 회원가입 기능/DB 연동과 POS기능/디버깅 작업으로 나누어 연구 활동을 진행함.□ 연구 결과<ul style="list-style-type: none">- 버튼 이벤트 핸들러를 통해 버튼의 데이터 값을 JTable로 내역들을 볼 수 있다.- 거래 내역들을 리스트로 저장하여 Oracle db 연동을 구현함 .- Jtable의 Print메소드를 통해 영수증을 구현함.	

참여자 인적사항

1. 프로젝트 팀장

성명	한재근	학번	202044029
학과	컴퓨터정보과		
연락처	H.P	010-3304-0781	
	E-mail	asgardsun97@gmail.com	

2. 프로젝트 팀원

성명	이준혁	학번	202044021
학과	컴퓨터정보과		
연락처	H.P	010-2889-7096	
	E-mail	wnsgur9137@naver.com	

목 차

1. 프로젝트 목적	1
2. 프로젝트 목표와 기대 효과	1
3. 프로젝트 진행범위 및 방법	1
4. 프로젝트 주요 내용	2
5. 연구 결과물	3
6-1. 프로젝트명	3
6-2. 프로젝트 개요	3
6-3. 프로젝트 수행 세부 일정 및 내용	7
6-4. 프로젝트 결과	8
 - 부 록 -	
프로젝트 계획서	1
사용자 설명서	19

1. 프로젝트 목적

- Swing GUI Form를 사용하여 JFrame간의 화면 전환과 같은 기능을 가진 그래픽 기반의 응용 프로그램을 작성 방법을 익힌다.
- JAVA(NetBeans)와 OracleDB 연동 방법을 익힌다.
- OracleDB의 시퀀스(Sequence)와 같은 기능들을 익힌다.

2. 프로젝트 목표 및 기대효과

- 프로젝트 목표

- 카페에서 사용하는 포스기(POS)의 기능(버튼을 통한 계산)을 구현한다.
- OracleDB를 사용하여 회원가입과 로그인을 제작한다.
- GUI를 이용하여 데이터베이스에 저장된 데이터 정보들을 사용자에게 전달한다.
- 사용자가 편리하고 가독성 있게 사용하도록 제작한다.

- 기대효과

- 실질적으로 스스로 자바 애플리케이션을 이용해서 프로그램을 작성하여 객체지향적인 프로그래밍에 대해 익힐 수 있는 좋은 기회이다.
- JAVA만의 그래픽환경의 프로그램을 만드는 것과, 독특한 구현 방식으로 철저히 객체 지향적인 자바 언어를 이해하는데 큰 도움이 됨
- 자바를 이용해 더 유용한 프로그램을 작성하는데 도전할 수 있는 경험이 됨
- GUI를 사용한 응용 애플리케이션을 작성 하여 사용자에게 편리한 프로그램을 제작하는 기회가 됨

3. 프로젝트 진행범위 및 방법

- 프로젝트 진행범위

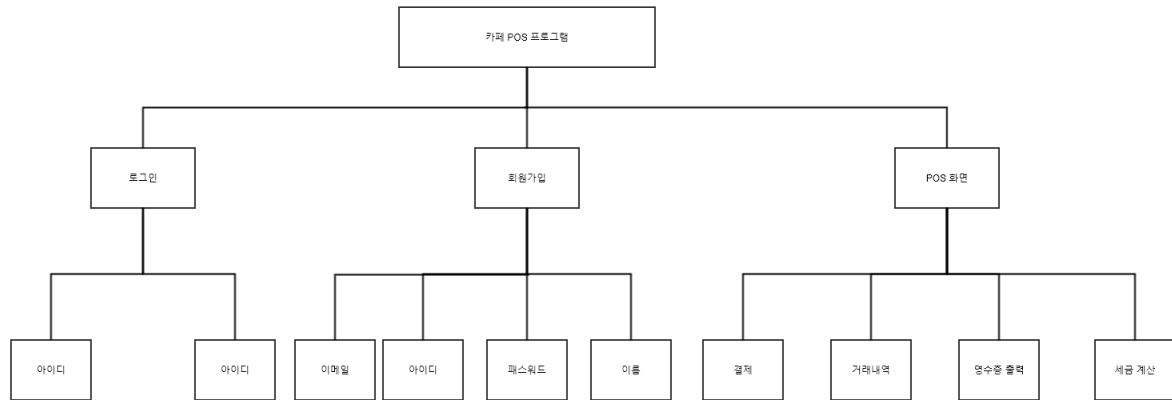
- 데이터베이스(OracleDB)를 이용한 회원가입 및 로그인 구현
- 카페 메뉴(커피 및 음료) 추가 및 계산, 결제 후 데이터베이스에 데이터 등록
- 데이터베이스에 등록된 정보(거래 내역)들을 사용자에게 전달
- 프린터 기능을 추가하여 거래 내역 프린트 기능 구현

- 프로젝트 진행방법(사용 기술)

- 그래픽 사용자 인터페이스(GUI) : Swing GUI Form을 이용한 그래픽 위주의 처리
- 데이터베이스(DataBase) : OracleDB를 이용해 정보를 저장 및 조회
- 상속 : 상위 클래스를 상속함으로써 이미 정의된 기능들을 손쉽게 사용
- 이벤트핸들러 : 해당 Component에서 발생한 Event에 대해 어떤 일을 수행할 것인지에 대한 핸들러

- 원도우리스너 : 창 닫기와 같은 이벤트(Event)들의 대한 동작들을 처리
- 넷빈즈 내의 프린트 출력 기능 : Jtable 내에 있는 Print메소드를 사용하여 결제 내역에 대한 영수증 출력

- 작업 분할 구조도(WBS)



4. 프로젝트 주요 내용

1) 화면구성

- 회원가입 : ID, PW, 이름, 이메일을 입력
- 로그인 : ID, PW로 로그인
- 포스기 화면을 제공

2) 포스기 기능

- 거래내역 : 메뉴 버튼을 클릭 시 해당 메뉴가 거래내역 Table에 추가
- 결제 방법 : 현금/카드 선택
- 결제 금액 계산 : 거래내역에 따른 금액 계산
- 세금 계산 : 결제 금액에 따른 세금계산
- 순수익 계산 : 세금을 포함한 순수익 금액 계산
- 영수증 출력 : print메소드를 이용한 영수증 출력

6. 프로젝트 결과물

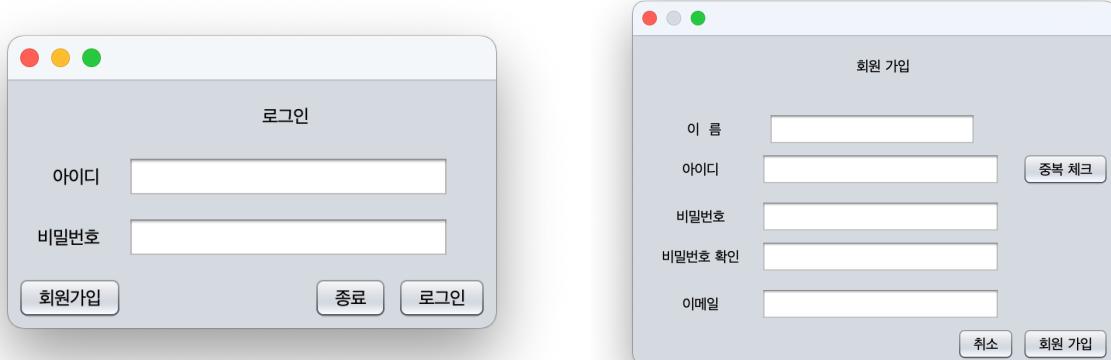
6-1. 프로젝트명

- 자바 GUI 카페 POS 응용 프로그램

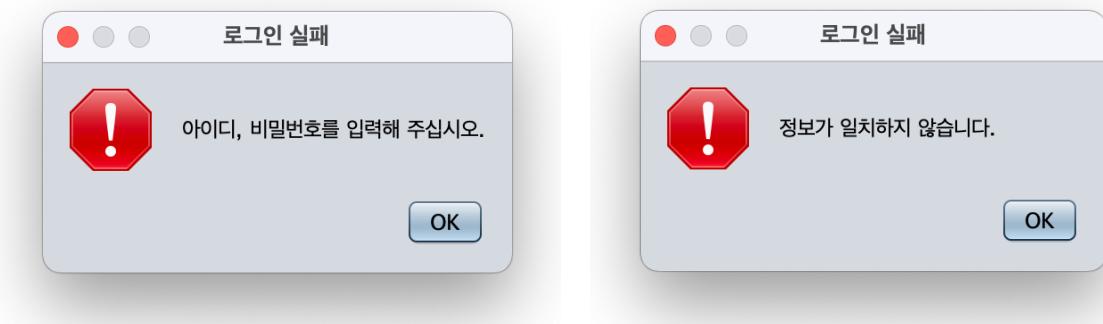
6-2. 프로젝트 개요

- 로그인 / 회원가입

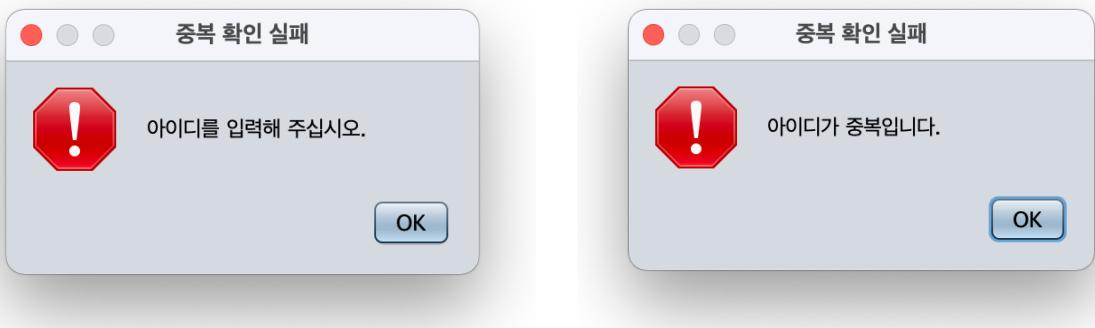
1) 화면 구성



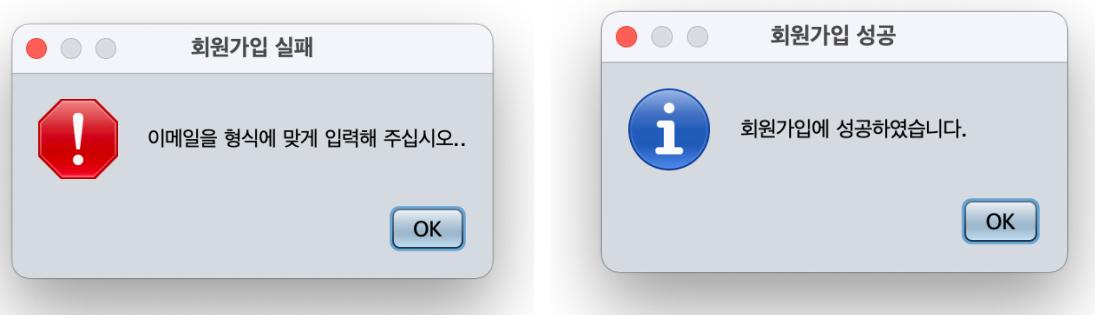
1-1) 로그인 유효성 검사 메세지



1-2) 회원가입 아이디 중복 검사 메세지



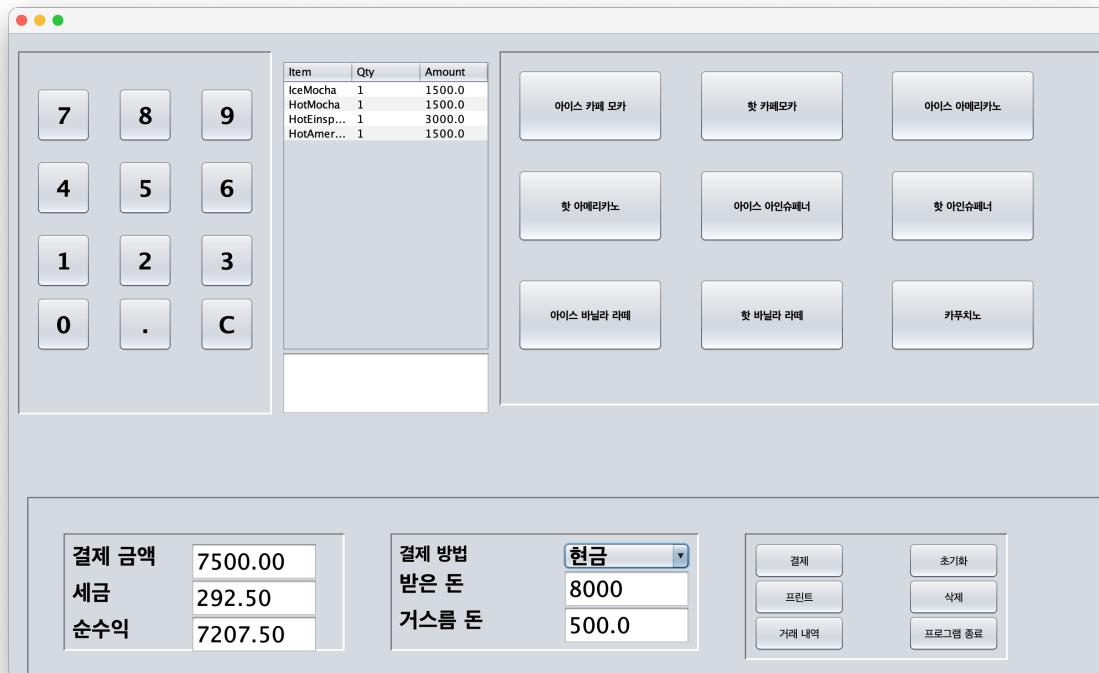
1-3) 회원가입 유효성 검사 메세지



● 포스기

1) 화면 구성

1-1) 메인 POS

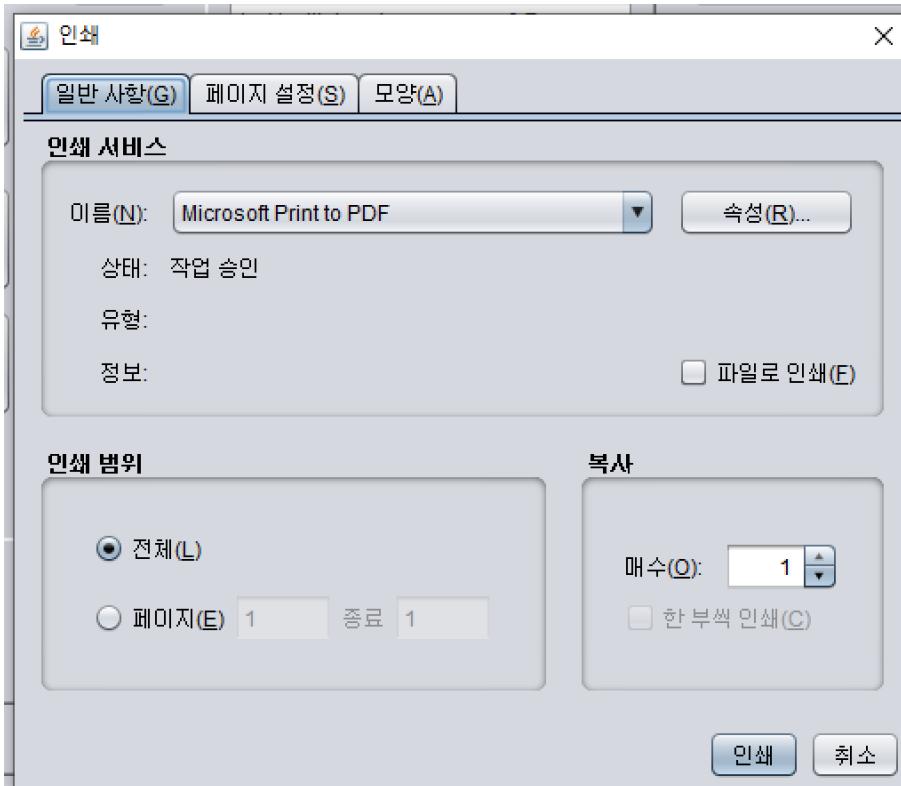


1-2) 거래 내역 리스트

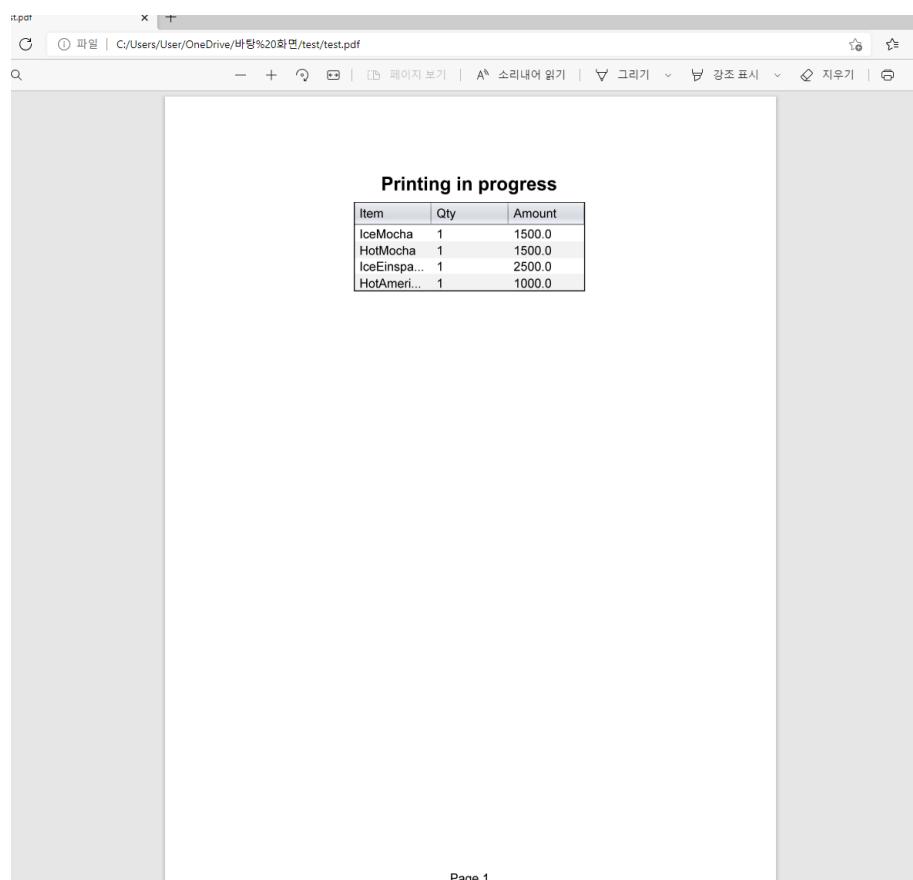
판매 현황								
번호	판매상품	수량	결제수단	가격	거스름돈	세금	순수익	시간
3	HotAmericano 외 6개	6	현금	13600.00	400.0	530.40	13069.60	2021/12/12 16:02:16
4	IceMocha 외 5개	6	카드	10300.00	700.0	401.70	9898.30	2021/12/12 16:03:44
5	IceMocha 외 11개	12	카드	21000.00	0.0	819.00	20181.00	2021/12/12 20:29:37
6	HotAmericano 외 4개	5	현금	10700.00	100.0	417.30	10282.70	2021/12/13 18:13:32
7	IceMocha 외 3개	4	카드	7500.00	500.0	292.50	7207.50	2021/12/14 20:24:38

[갱신] [종료]

1-3) 영수증 출력 창(인쇄 창)



1-4) 영수증 출력 화면(인쇄)



6-3. 프로젝트 수행 세부일정 및 내용

■ 프로젝트 수행 세부 일정

프로젝트 진행 계획(2021. 08. 31. ~ 2021. 12. 15)								
진 행 내 용	책임자	10	11	12	13	14	비고	
화면 구성 및 DB 테이블 작성	공동							
로그인 기능 구현	이준혁							
POS 기능 구현	한재근							
POS DB 연동	이준혁							
디버깅	한재근							
보고서 작성	공동							

■ 프로젝트 수행 내용

● CPM

작업	작업 설명	선행 작업	소요 기간(주)
A	로그인/회원가입 화면 개발	-	1
B	POS 화면 개발	A	-
C	DB테이블(USER, POS) 구축	A, B	1
D	로그인/회원가입 기능 구현	C	2
E	POS 계산 기능 구현	C	-
F	USER/POS DB 연동	D, E	1
G	거래내역 리스트 구현	E, F	-
H	영수증 출력(인쇄) 구현	G	-
	프로젝트 종료		

먼저 수행해야 할 작업은 과제 특성상 필수인 기능인 로그인/회원가입 화면을 최우선 작업으로 시작되어 완료되어야 한다. 로그인/회원가입 화면 개발 완료시 같은 주에 POS 화면 기본 틀을 구축한다. 화면 개발을 완료하면 OracleDB를 사용하여 데이터베이스 USER, POS 테이블을 생성한다. 그 후 로그인/회원가입 기능과 POS 계산 기능을 구현한다. 위 작업들을 모두 완료시 OracleDB와 연동시킨다. 같은 주에 Table 값에 따라 거래내역 리스트를 구현하고 동시에 영수증 출력(인쇄)를 구현하여 프로젝트를 마무리한다.

6-4. 프로젝트 결과

- 프로젝트 소스 •

DBConn.java

```
import java.sql.*;
import java.util.ArrayList;

public class StatementDAO {
    public int save(Statement data) { // 거래내역 추가
        String sql = "INSERT INTO Statements VALUES (states_seq.NEXTVAL, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        int result = 0; // 거래내역 등록 실패시 0 반환
        try {
            Statement stat = (Statement) data;
            conn = DBConn.getConnection();
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, stat.getItem());
            pstmt.setString(2, stat.getQty());
            pstmt.setString(3, stat.getPayment());
            pstmt.setString(4, stat.getAmount());
            pstmt.setString(5, stat.getChange());
            pstmt.setString(6, stat.getTax());
            pstmt.setString(7, stat.getProfit());
            pstmt.setString(8, stat.getCreated());
            rs = pstmt.executeQuery();
            System.out.println("DB 등록 성공");
            result = 1; // 거래내역 등록 성공시 1 반환
        } catch(Exception e) {
            System.out.println("save 실패");
            System.out.println(e.getMessage());
        } finally {
            try { // DB Close
                if(rs != null) rs.close();
                if(pstmt != null) pstmt.close();
                if(conn != null) conn.close();
            } catch(Exception e) {
                System.out.println("DB close 실패 (StatementDAO - save())");
                System.out.println(e.getMessage());
            }
        }
        return result;
    }
}
```

```

public ArrayList<Statement> getList() { // 거래 내역 리스트
    // ArrayList를 이용하여 거래 내역들을 저장
    ArrayList<Statement> list = new ArrayList<Statement>();
    String sql = "SELECT * FROM statements";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    try {
        conn = DBConn.getConnection();
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        while(rs.next()) { // 리스트가 존재할 경우 반복
            System.out.println("list 추가 시작"); // 안내문
            Statement stat = new Statement();
            //세터를 사용하여 각각의 정보를 입력
            stat.setNum(rs.getInt("NUM"));
            stat.setItem(rs.getString("ITEM"));
            stat.setQty(rs.getString("QTY"));
            stat.setPayment(rs.getString("PAYMENT"));
            stat.setAmount(rs.getString("AMOUNT"));
            stat.setChange(rs.getString("CHANGE"));
            stat.setTax(rs.getString("TAX"));
            stat.setProfit(rs.getString("PROFIT"));
            stat.setCreated(rs.getString("CREATED"));
            list.add(stat); // 입력한 값을 배열리스트에 추가
            System.out.println("list 추가 완료"); // 안내문
        }
    } catch(Exception e) {
        System.out.println("getList 실패");
        System.out.println(e.getMessage());
    } finally {
        try {
            if(rs != null) rs.close();
            if(pstmt != null) pstmt.close();
            if(conn != null) conn.close();
        } catch(Exception e) {
            System.out.println("DB close 실패 (StatementDAO - getList())");
            System.out.println(e.getMessage());
        }
    }
    return list;
}
}

```

User.java

```
public class User { // POS 사용자
    private Integer userKey;      // 사용자 번호
    private String userName;     // 사용자 이름
    private String userID;       // 사용자 아이디
    private String userPWD;      // 사용자 비밀번호
    private String userEmail;    // 사용자 이메일
    private String userCreated;  // 사용자 생성 시간

    // 생성자
    public User() {}

    public User(Integer userKey, String userName, String userID, String userPWD, String
    userEmail, String userCreated) {
        this.userKey = userKey;
        this.userName = userName;
        this.userID = userID;
        this.userPWD = userPWD;
        this.userEmail = userEmail;
        this.userCreated = userCreated;
    }

    // 게터 세터
    public Integer getUserKey() { return userKey; }
    public void setUserKey(Integer userKey) { this.userKey = userKey; }

    public String getUserName() { return userName; }
    public void setUserName(String userName) { this.userName = userName; }

    public String getUserId() { return userID; }
    public void setUserId(String userID) { this.userID = userID; }

    public String getUserPWD() { return userPWD; }
    public void setUserPWD(String userPWD) { this.userPWD = userPWD; }

    public String getUserEmail() { return userEmail; }
    public void setUserEmail(String userEmail) { this.userEmail = userEmail; }

    public String getUserCreated() { return userCreated; }
    public void setUserCreated(String userCreated) { this.userCreated = userCreated; }

}
```

UserDAO.java

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class UserDAO {

    public int join(User data) { // 회원가입
        String sql = "INSERT INTO Users VALUES (users_seq.NEXTVAL, ?, ?, ?, ?, ?)";
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        int result = -1; // 회원가입 실패시 -1 반환
        try {
            User user = (User) data; // 입력받은 유저 정보 (이름, 아이디, 비밀번호, 이메일, 생성일자)
            conn = DBConn.getConnection();
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, user.getUserName());
            pstmt.setString(2, user.getUserID());
            pstmt.setString(3, user.getUserPWD());
            pstmt.setString(4, user.getUserEmail());
            pstmt.setString(5, user.getUserCreated());

            rs = pstmt.executeQuery();
            System.out.println("회원가입 성공");
            result = 1; // 회원가입 성공시 1 반환
        } catch(Exception e) {
            System.out.println("join 실패");
            System.out.println(e.getMessage());
        } finally {
            try { // DB Close
                if(rs != null) rs.close();
                if(pstmt != null) pstmt.close();
                if(conn != null) conn.close();
            } catch(Exception e) {
                System.out.println("DB close 실패 (LoginForm - btnJoinAction())");
                System.out.println(e.getMessage());
            }
        }
        return result;
    }
}
```

```

public int idCheck(String userID) { // 회원가입 - 아이디 중복 확인
    String sql = "SELECT * FROM Users WHERE userID = ?";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    int result = -1; // DB 처리 실패시 -1 반환
    try {
        conn = DBConn.getConnection();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, userID);
        rs = pstmt.executeQuery();
        if(rs.next()) {
            result = 0; // 중복일 경우 0 반환
        } else {
            result = 1; // 사용 가능할 경우 1 반환
        }
    } catch(Exception e) {
        System.out.println("IDCheck 실패");
        System.out.println(e.getMessage());
    } finally {
        try { // DB Close
            if(rs != null) rs.close();
            if(pstmt != null) pstmt.close();
            if(conn != null) conn.close();
        } catch(Exception e) {
            System.out.println("DB close 실패 (LoginForm - btnIDCheckAction())");
            System.out.println(e.getMessage());
        }
    }
    return result;
}

public int login(String userID, String userPWD) { // 로그인
    String sql = "SELECT * FROM Users WHERE userID = ? AND userPWD = ?";
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    int result = -1; // DB 오류시 -1 반환
    try {
        conn = DBConn.getConnection();
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, userID);
        pstmt.setString(2, userPWD);
    }
}

```

```
        rs = pstmt.executeQuery();
        if(rs.next()) {
            result = 1; // 로그인 성공시 1 반환
        } else {
            result = 0; // 로그인 실패(정보 불일치)시 0 반환
        }
    } catch(Exception e) {
        System.out.println("Login 실패");
        System.out.println(e.getMessage());
    } finally {
        try {
            if(rs != null) rs.close();
            if(pstmt != null) pstmt.close();
            if(conn != null) conn.close();
        } catch(Exception e) {
            System.out.println("DB Close 실패 (LoginForm - btnLoginAction)");
            System.out.println(e.getMessage());
        }
    }
}

return result;
}
}
```

Statement.java

```
public class Statement { // 거래 내역
    private int num;          // 거래 내역 번호
    private String item;      // 거래 상품
    private String qty;       // 상품 수량
    private String payment;   // 결제 방식
    private String amount;    // 가격
    private String change;    // 거스름 돈
    private String tax;       // 세금
    private String profit;   // 순수익
    private String created;  // 거래 시간

    // 생성자
    public Statement() {}

    public Statement(String item, String qty, String payment, String amount, String change,
String tax, String profit, String created) {
        this.item = item;
        this.qty = qty;
        this.payment = payment;
        this.amount = amount;
        this.change = change;
        this.tax = tax;
        this.profit = profit;
        this.created = created;
    }

    //게터 세터
    public int getNum() { return num; }
    public void setNum(int num) { this.num = num; }

    public String getItem() { return item; }
    public void setItem(String item) { this.item = item; }

    public String getPayment() { return payment; }
    public void setPayment(String payment) { this.payment = payment; }

    public String getQty() { return qty; }
    public void setQty(String qty) {this.qty = qty; }

    public String getAmount() { return amount; }
    public void setAmount(String amount) { this.amount = amount; }

    public String getChange() { return change; }
    public void setChange(String change) { this.change = change; }
```

```

public String getTax() { return tax; }
public void setTax(String tax) { this.tax = tax; }

public String getProfit() { return profit; }
public void setProfit(String profit) { this.profit = profit; }

public String getCreated() { return created; }
public void setCreated(String created) {this.created = created; }

}

```

StatementDAO.java

```

import java.sql.*;
import java.util.ArrayList;

public class StatementDAO {
    public int save(Statement data) { // 거래내역 추가
        String sql = "INSERT INTO Statements VALUES (states_seq.NEXTVAL, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        int result = 0; // 거래내역 등록 실패시 0 반환
        try {
            Statement stat = (Statement) data;
            conn = DBConn.getConnection();
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, stat.getItem());
            pstmt.setString(2, stat.getQty());
            pstmt.setString(3, stat.getPayment());
            pstmt.setString(4, stat.getAmount());
            pstmt.setString(5, stat.getChange());
            pstmt.setString(6, stat.getTax());
            pstmt.setString(7, stat.getProfit());
            pstmt.setString(8, stat.getCreated());

            rs = pstmt.executeQuery();
            System.out.println("DB 등록 성공");
            result = 1; // 거래내역 등록 성공시 1 반환
        } catch(Exception e) {
            System.out.println("save 실패");
            System.out.println(e.getMessage());
        }
    }
}

```

```

} finally {
    try { // DB Close
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    } catch(Exception e) {
        System.out.println("DB close 실패 (StatementDAO - save())");
        System.out.println(e.getMessage());
    }
}
return result;
}

public ArrayList<Statement> getList() { // 거래 내역 리스트
    // ArrayList를 이용하여 거래 내역들을 저장
    ArrayList<Statement> list = new ArrayList<Statement>();
    String sql = "SELECT * FROM statements";

    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    try {
        conn = DBConn.getConnection();
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        while(rs.next()) { // 리스트가 존재할 경우 반복
            System.out.println("list 추가 시작"); // 안내문
            Statement stat = new Statement();
            // 세터를 사용하여 각각의 정보를 입력
            stat.setNum(rs.getInt("NUM"));
            stat.setItem(rs.getString("ITEM"));
            stat.setQty(rs.getString("QTY"));
            stat.setPayment(rs.getString("PAYMENT"));
            stat.setAmount(rs.getString("AMOUNT"));
            stat.setChange(rs.getString("CHANGE"));
            stat.setTax(rs.getString("TAX"));
            stat.setProfit(rs.getString("PROFIT"));
            stat.setCreated(rs.getString("CREATED"));
            list.add(stat); // 입력한 값을 배열리스트에 추가
            System.out.println("list 추가 완료"); // 안내문
        }
    } catch(Exception e) {
        System.out.println("getList 실패");
        System.out.println(e.getMessage());
    } finally {
        try {

```

```

        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
    } catch(Exception e) {
        System.out.println("DB close 실패 (StatementDAO - getList())");
        System.out.println(e.getMessage());
    }
}
return list;
}
}
}

```

LoginForm.java

```

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.JOptionPane;
import java.awt.*;
import java.awt.event.*;

public class LoginForm extends javax.swing.JFrame {

    int loginResult = -1;

    public LoginForm() {
        this.addWindowListener(new WindowListener() { // WindowListener
            @Override
            public void windowOpened(WindowEvent e) {
                throw new UnsupportedOperationException("Not supported yet.");
            }

            @Override
            public void windowClosing(WindowEvent e) {
                throw new UnsupportedOperationException("Not supported yet.");
            }

            @Override
            public void windowClosed(WindowEvent e) { // 로그인 창이 꺼진 뒤
                if(loginResult == -1) { // 로그인이 되지 않은 경우
                    System.exit(0); // 프로그램 종료
                } else { // 로그인 성공시
                    new JavaPOS().setVisible(true); // 메인 포스기 창 표시
                }
            }
        });
    }
}

```

```

@Override
public void windowIconified(WindowEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public void windowDeiconified(WindowEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public void windowActivated(WindowEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public void windowDeactivated(WindowEvent e) {
    throw new UnsupportedOperationException("Not supported yet.");
}

});

initComponents();
}

```

@SuppressWarnings("unchecked")

Generated Code

```

private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // 로그인 버튼

    // ID, PWD 텍스트 필드에 값을 넣지 않은 경우
    if(txtID.getText().length() < 1 ||
       txtPWD.getText().length() < 1) {
        System.out.println("아이디, 비밀번호를 입력해 주십시오.");
        JOptionPane.showMessageDialog(null, "아이디, 비밀번호를 입력해 주십시오.", "로그인 실패",
                                   JOptionPane.ERROR_MESSAGE);
        return;
    }

    String userID = txtID.getText();
    String userPWD = txtPWD.getText();

    UserDAO userDAO = new UserDAO();
    int result = userDAO.login(userID, userPWD);
    if(result != 1) { // 로그인 실패일 경우
        System.out.println("로그인 실패");
        JOptionPane.showMessageDialog(null, "정보가 일치하지 않습니다.", "로그인 실패",
                                   JOptionPane.ERROR_MESSAGE);
    }
}

```

```

        JOptionPane.ERROR_MESSAGE);

    } else { // 로그인 성공일 경우
        JOptionPane.showMessageDialog(null, "환영합니다.", "로그인 성공",
                JOptionPane.INFORMATION_MESSAGE);
        System.out.println("로그인 성공");
        loginResult = 1;
        dispose();
    }

}

private void btnJoinFormActionActionPerformed(java.awt.event.ActionEvent evt) {
    // 회원가입 창 표시
    joinForm.setLocation(350, 350);
    joinForm.setSize(400, 300);
    joinForm.show();
}

private void btnLoginExitActionPerformed(java.awt.event.ActionEvent evt) {
    // 로그인 취소(프로그램 종료) 버튼
    dispose();
}

// 회원가입 -----
// 유저 키는 오라클 스퀄스 사용
// CREATE SEQUENCE users_seq START WITH 1 INCREMENT BY 1 MAXVALUE 200 NOCYCLE
NOCACHE;

// -1일경우 중복 확인 안함, 0일경우 중복, 1일경우 사용 가능한 아이디
int idCheck = -1;

int nullCheck() { // 회원가입 프레임의 텍스트 필드 값이 입력되었는지 확인
    int result = 1;
    if (txtJoinName.getText().length() < 1 ||
        txtJoinID.getText().length() < 1 ||
        txtJoinPWD.getText().length() < 1 ||
        txtJoinPWDCheck.getText().length() < 1 ||
        txtJoinEmail.getText().length() < 1) {

        result = -1;
    }
    return result;
}

```

```

private void btnJoinActionActionPerformed(java.awt.event.ActionEvent evt) {
    // 회원가입 버튼
    if(idCheck == -1) { // 중복체크를 하지 않은 경우
        System.out.println("중복체크 해 주십시오.");
        JOptionPane.showMessageDialog(null, "중복 확인 해 주십시오..", "회원가입 실패",
                JOptionPane.ERROR_MESSAGE);
    } else if(idCheck == 0) { // 아이디가 중복인 경우
        System.out.println("아이디 중복입니다.");
        JOptionPane.showMessageDialog(null, "아이디 중복입니다.", "회원가입 실패",
                JOptionPane.ERROR_MESSAGE);
    }
}

int emptyCheck = nullCheck(); // 회원가입에 필요한 정보들을 모두 입력했는지 확인
if(emptyCheck != 1) { // 정보를 입력하지 않은 경우
    System.out.println("빈칸을 채워주십시오.");
    JOptionPane.showMessageDialog(null, "빈칸을 채워주십시오.", "회원가입 실패",
            JOptionPane.ERROR_MESSAGE);
    return;
}

String userName = txtJoinName.getText();
String userID = txtJoinID.getText();
String userPWD = txtJoinPWD.getText();
String userPWDCheck = txtJoinPWDCheck.getText();
String userEmail = txtJoinEmail.getText();

// 유저 생성 날짜/시간
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date date = new Date();
String userCreated = dateFormat.format(date);

// 비밀번호와 비밀번호 확인 텍스트 필드가 서로 일치하지 않은 경우
if(!userPWD.equals(userPWDCheck)) {
    System.out.println("비밀번호가 일치하지 않습니다.");
    JOptionPane.showMessageDialog(null, "비밀번호가 일치하지 않습니다.", "회원가입 실패",
            JOptionPane.ERROR_MESSAGE);
    return;
}

// 입력한 이메일이 형식에 맞지 않을 경우
if(userEmail.length() < 1 || !userEmail.contains("@") || !userEmail.contains(".")) {
    System.out.println("이메일 형식에 맞게 입력해주십시오.");
    JOptionPane.showMessageDialog(null, "이메일을 형식에 맞게 입력해 주십시오..",
            "회원가입 실패", JOptionPane.ERROR_MESSAGE);
    return;
}

```

```

User user = new User(null, userName, userID, userPWD, userEmail, userCreated);
UserDAO userDAO = new UserDAO();

int result = userDAO.join(user);
if(result != 1) { // 회원가입 실패인 경우
    System.out.println("회원가입 실패");
    JOptionPane.showMessageDialog(null, "회원가입 실패.", "회원가입 실패",
        JOptionPane.ERROR_MESSAGE);
} else { // 회원가입 성공인 경우
    System.out.println("회원가입 성공");
    JOptionPane.showMessageDialog(null, "회원가입에 성공하였습니다.", "회원가입 성공",
        JOptionPane.INFORMATION_MESSAGE);
    txtClear();
    joinForm.dispose(); // 회원가입 창 종료
}

}

private void btnJoinExitActionPerformed(java.awt.event.ActionEvent evt) {
    // 회원가입 취소 버튼
    txtClear();
    joinForm.dispose(); // 회원가입 창 종료
}

private void txtClear() { // 회원가입 창의 텍스트필드들에 입력된 값을 없앰
    txtJoinName.setText("");
    txtJoinID.setText("");
    txtJoinPWD.setText("");
    txtJoinPWDCheck.setText("");
    txtJoinEmail.setText("");
}

private void btnIDCheckActionPerformed(java.awt.event.ActionEvent evt) {
    // 회원가입 아이디 중복 체크 버튼

    if(txtJoinID.getText().length() < 1) {
        System.out.println("아이디를 입력해주십시오.");
        JOptionPane.showMessageDialog(null, "아이디를 입력해 주십시오.", "중복 확인 실패",
            JOptionPane.ERROR_MESSAGE);
        return;
    }
}

```

```

String userID = txtJoinID.getText();

UserDAO userDAO = new UserDAO();
int result = userDAO.idCheck(userID); // 아이디 중복 체크
if(result != 1) { // 아이디가 중복일 경우(동일한 아이디가 DB에 이미 있는 경우)
    System.out.println("중복");
    JOptionPane.showMessageDialog(null, "아이디가 중복입니다.", "중복 확인 실패",
        JOptionPane.ERROR_MESSAGE);
    idCheck = 0; // idCheck 전역변수 0 (아이디 중복)
} else { // 아이디 중복이 아닌 경우
    System.out.println("아이디 사용 가능");
    JOptionPane.showMessageDialog(null, "사용가능한 아이디입니다.", "중복 확인 성공",
        JOptionPane.INFORMATION_MESSAGE);
    idCheck = 1; // idCheck 전역변수 1 (사용 가능)
}
}

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new LoginForm().setVisible(true);
        }
    });
}
}

```

JavaPOS.java

```
import java.text.DateFormat;
import javax.swing.table.DefaultTableModel;
import java.text.MessageFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JTable;

public class JavaPOS extends javax.swing.JFrame {

    public JavaPOS() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    //=====Function=====
    //함수 시작
    //품목 비용 계산
    public void ItemCost(){
        double sum = 0; //변수 sum을 double로 선언
        //for문을 사용하여 jTable2의 열의 갯수만큼 반복하며 ++해준다.
        for (int i = 0; i < jTable2.getRowCount(); i++){
            sum = sum + Double.parseDouble(jTable2.getValueAt(i,2).toString());
            //변수 sum은 기존 있었던 sum에 jTable2의 Value(i,2)위치에 있는 값을 가져와서 String으로
변환하고
            //문자열 형태를 double형을 바꾼다.
        }

        //sum의 변수를 문자열 형태로 변환하여 결제금액 필드에 set시켜준다.
        jtxtSubTotal.setText(Double.toString(sum));
        double cTotal = Double.parseDouble(jtxtSubTotal.getText());
        //double cTotal 변수에 결제금액의 text값을 가져와서 문자열 형태를 double형으로 바꾼다.

        double cTax = (cTotal * 3.9)/100; //결제금액의 x 3.9/100값을 사용해서 세금값을 이용함.
        String iTaxTotal = String.format("%.2f",cTax);
        //문자열 iTaxTotal 변수에 String.format을 사용하여 실수형 숫자형식으로 표현
        jtxtTax.setText(iTaxTotal); //세금 textField에 iTaxTotal을 set해준다.

        String iSubTotal = String.format("%.2f" ,cTotal);
        //문자열 iSubTotal 변수에String.format을 이용하여 실수형 숫자형식으로 표현
        jtxtSubTotal.setText(iSubTotal); //결제금액textField에 iSubTotal을 set해준다.

        String iTotal = String.format("%.2f", cTotal - cTax);
        //문자열 iTotal 변수에 String format을 이용하여 cTotal - cTax값을 실수형 숫자형식으로 표현한다.
        jtxtTotal.setText(iTotal); //순수익 textField에 iTotal을set해준다.

    }
}
```

```

//=====Function Changer=====
//=====함수 변환=====
public void Change(){
    double sum = 0;      //변수 sum 선언
    double tax = 3.9;    //변수 tax 선언
    double cash = Double.parseDouble(jtxtDisplay.getText());
    //변수 cash에 받은돈의 텍스트필드의 text를 가져와서 실수형으로 변환

    for (int i = 0; i < jTable2.getRowCount(); i++){
        sum = sum + Double.parseDouble(jTable2.getValueAt(i,2).toString());
    } //변수sum은 기존 있었던 sum에 jTable2의 Value(i,2)위치에 있는 값을 가져와서
    // String으로 변환하고 문자열 형태를 double형으로 바꾼다.

    double cTax = (sum * 3.9)/100;      //변수 cTax에 sum*3.9/100 나눈값을 대입한다.
    double cChange = (cash - (sum));
    //변수 cChange에 cash-sum의 값을 대입한다. -> 받은돈 - 결제금액 계산

    String ChangeGiven = String.format("%.1f",cChange);
    //문자열 ChangeGiven 변수에 String.format을 이용하여 실수형 숫자형식으로 표현
    jtxtChange.setText(ChangeGiven);   //거스름돈 텍스트필드에 ChangeGiven을 set해준다.
}

private void IceMochaActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 1500; //아이스 모카의 값을 1500으로 선언.
    //DefaultTableModel 클래스를 사용하는 이유
    //jTable에서는 추가, 삭제기능을 사용하지 못하지만 model을 사용하여
    //addRow(),removeRow()가 사용 가능하다

    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"IceMocha","1",PriceOfItem} );
    //DefaultTableModel 클래스를 사용하여 이를 jTable2에 새로운 행을 추가하고
    // Object형으로 아이스모카의 데이터값을 저장해준다.
    ItemCost(); //ItemCosh()함수 실행
}

//숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
//Enternumber를 String으로 선언하고
//버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
private void jbtn7ActionPerformed(java.awt.event.ActionEvent evt)
{
    String EnterNumber = jtxtDisplay.getText();
    if(EnterNumber == ""){
        jtxtDisplay.setText(jbtn7.getText());
    }else{
        EnterNumber = jtxtDisplay.getText() + jbtn7.getText();
        jtxtDisplay.setText(EnterNumber);
    }
}

//숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
//Enternumber를 String으로 선언하고
//버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
private void jbtn8ActionPerformed(java.awt.event.ActionEvent evt)
{
    String EnterNumber = jtxtDisplay.getText();
    if(EnterNumber == ""){
        jtxtDisplay.setText(jbtn8.getText());
    }
}

```

```

        }else{
            Enternumber = jtxtDisplay.getText() + jbtn8.getText();
            jtxtDisplay.setText(Enternumber);
        }
    }

    //숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
    //Enternumber를 String으로 선언하고
    //버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
    private void jbtn9ActionPerformed(java.awt.event.ActionEvent evt)
    {
        String Enternumber = jtxtDisplay.getText();
        if(Enternumber == ""){
            jtxtDisplay.setText(jbtn9.getText());
        }else{
            Enternumber = jtxtDisplay.getText() + jbtn9.getText();
            jtxtDisplay.setText(Enternumber);
        }
    }

    //숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed0~9
    //Enternumber를 String으로 선언하고
    //버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
    private void jbtn4ActionPerformed(java.awt.event.ActionEvent evt)
    {
        String Enternumber = jtxtDisplay.getText();
        if(Enternumber == ""){
            jtxtDisplay.setText(jbtn4.getText());
        }else{
            Enternumber = jtxtDisplay.getText() + jbtn4.getText();
            jtxtDisplay.setText(Enternumber);
        }
    }

    //숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
    //Enternumber를 String으로 선언하고
    //버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
    private void jbtn5ActionPerformed(java.awt.event.ActionEvent evt)
    {
        String Enternumber = jtxtDisplay.getText();
        if(Enternumber == ""){
            jtxtDisplay.setText(jbtn5.getText());
        }else{
            Enternumber = jtxtDisplay.getText() + jbtn5.getText();
            jtxtDisplay.setText(Enternumber);
        }
    }

    //숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
    //Enternumber를 String으로 선언하고
    //버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
    private void jbtn6ActionPerformed(java.awt.event.ActionEvent evt)
    {
        String Enternumber = jtxtDisplay.getText();
        if(Enternumber == ""){
            jtxtDisplay.setText(jbtn6.getText());
        }else{

```

```

        EnterNumber = jtxtDisplay.getText() + jbtn6.getText();
        jtxtDisplay.setText(EnterNumber);
    }      // TODO add your handling code here:
}

//숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
//EnterNumber를 String으로 선언하고
//버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
private void jbtn1ActionPerformed(java.awt.event.ActionEvent evt)
{
    String EnterNumber = jtxtDisplay.getText();
    if(EnterNumber == ""){
        jtxtDisplay.setText(jbtn1.getText());
    }else{
        EnterNumber = jtxtDisplay.getText() + jbtn1.getText();
        jtxtDisplay.setText(EnterNumber);
    }
}

//숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
//EnterNumber를 String으로 선언하고
//버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
private void jbtn2ActionPerformed(java.awt.event.ActionEvent evt)
{
    String EnterNumber = jtxtDisplay.getText();
    if(EnterNumber == ""){
        jtxtDisplay.setText(jbtn2.getText());
    }else{
        EnterNumber = jtxtDisplay.getText() + jbtn2.getText();
        jtxtDisplay.setText(EnterNumber);
    }
}

//숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
//EnterNumber를 String으로 선언하고
//버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
private void jbtn3ActionPerformed(java.awt.event.ActionEvent evt)
{
    String EnterNumber = jtxtDisplay.getText();
    if(EnterNumber == ""){
        jtxtDisplay.setText(jbtn7.getText());
    }else{
        EnterNumber = jtxtDisplay.getText() + jbtn7.getText();
        jtxtDisplay.setText(EnterNumber);
    }
}

//숫자 키패드를 입력하는 이벤트 핸들러 jbtnActionPerformed 0~9
//EnterNumber를 String으로 선언하고
//버튼을 눌렀을때 -> 버튼에 해당하는 숫자 text값을 getText해서 setText해준다.
private void jbtn0ActionPerformed(java.awt.event.ActionEvent evt)
{
    String EnterNumber = jtxtDisplay.getText();
    if(EnterNumber == ""){
        jtxtDisplay.setText(jbtn0.getText());
    }
}

```

```

    }else{
        Enternumber = jtxtDisplay.getText() + jbtn0.getText();
        jtxtDisplay.setText(Enternumber);
    }
}

//C 버튼을 눌렀을 때 해당 Textfield공백으로 처리
private void jbtnCActionPerformed(java.awt.event.ActionEvent evt)
{
    jtxtDisplay.setText(""); // jtxtDisplay(받은돈을 나타내는 Textfield)를 공백으로 처리해줌
    jtxtChange.setText(""); //jtxtChange필드(거스름돈을 나타내는 Textfield)를 공백으로 처리해줌
}

private void jbtnDotActionPerformed(java.awt.event.ActionEvent evt)
{
    //버튼 클릭 시 jtxtDisplay 텍스트 필드에 소수점을 처리해준다.
    if(! jtxtDisplay.getText().contains(".")){
        jtxtDisplay.setText(jbtnDot.getText() + jtxtDisplay.getText());
    }
}

//DefaultTableModel 클래스를 사용하는 이유 : jTable에서는 추가, 삭제기능을 사용하지 못하지만 model을 사용하여
//addRow(),removeRow()가 사용 가능하다
private void HotMochaActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 1500; //핫 모카의 값을 1500으로 설정.
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel(); //모델 객체 선언 부분
    model.addRow(new Object [] {"HotMocha","1",PriceOfItem} );
    //DefaultTableModel 클래스를 사용하여 이를 jTable2에 새로운 행을 추가하고
    //Object형으로 핫모카의 데이터값을 저장해준다.
    ItemCost(); //ItemCost함수실행
}

private void IceAmericanoActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 1500; //아이스 아메리카노의 값을 1500으로 설정
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"IceAmericano","1",PriceOfItem} );
    //DefaultTableModel 클래스를 사용하여 이를 jTable2에 새로운 행을 추가하고
    //Object형으로 아이스 아메리카노의 데이터값을 저장해준다.
    ItemCost();
}

private void HotAmericanoActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 1500; //뜨거운 아메리카노의 값을 1500으로 설정
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"HotAmericano","1",PriceOfItem} );
    //DefaultTableModel 클래스를 사용하여 이를 jTable2에 새로운 행을 추가하고
    //Object형으로 핫아메리카노의 데이터값을 저장해준다.
    ItemCost();
}

```

```

//버튼의 메뉴들은 위와같이 반복해서 선언해준다.
private void IceEinspannerActionPerformed(java.awt.event.ActionEvent evt) {
    double PriceOfItem = 3000;      //아이스 아인슈페너의 값을 3000으로 설정
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"IceEinspanner","1",PriceOfItem} );
    ItemCost();
}

private void HotEinspannerActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 3000;      //핫 아인슈페너의 값을 3000으로 설정
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"HotEinspanner","1",PriceOfItem} );
    ItemCost();
}

private void IceVanillalatteActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 1400;      //아이스 바닐라라떼의 값을 1400으로 설정.
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"IceVanillalatte","1",PriceOfItem} );
    ItemCost();
}

private void HotVanillalatteActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 1400;      //뜨거운바닐라라떼의 값을 1400으로 설정;
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"HotVanillalatte","1",PriceOfItem} );
    ItemCost();
}

private void CappuccinoActionPerformed(java.awt.event.ActionEvent evt)
{
    double PriceOfItem = 1800;      //카푸치노의 값을 1800으로 설정
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.addRow(new Object [] {"Cappuccino","1",PriceOfItem} );
    ItemCost();
}

//결제 버튼 클릭시
private void jbtnPayActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:

    //if문을 사용하여 받은돈의 텍스트 길이가 1이 더크다면 결제 실패 메시지 창 생성
    if(jtxtDisplay.getText().length() < 1) {
        JOptionPane.showMessageDialog(null, "결제 금액을 입력해 주십시오.",
                                   "결제 실패",JOptionPane.ERROR_MESSAGE);
        return;
    }
    //if문을 사용하여 결제방법의 메뉴를 선택하고 그에 따른 Change함수 실행
    if(jcboPayment.getSelectedItem().equals("현금")){
        Change();
    }else if(jcboPayment.getSelectedItem().equals("카드")){
        Change();
    }
}

```

```

// DB 연결/저장을 위한 코드
int count = -1;
for (int i = 0; i < jTable2.getRowCount(); i++){
    jTable2.getValueAt(i,2).toString();
    count++;
}

//jTable2의 메뉴를String item으로 선언
String item = jTable2.getValueAt(0,0).toString() + " 외 " + count + "개";
//결제방법에 대한 항목을 String payment로 선언
String payment = jcboPayment.getSelectedItem().toString();
//결제금액을 jtxtSubTotal에서 값을 가져와 String amount로 선언
String amount = jtxtSubTotal.getText();
//거스름돈을 jtxtChange에서 값을 가져와 String change로 선언
String change = jtxtChange.getText();
String tax = jtxtTax.getText(); //세금 금액을 jtxtTax에서 값을 가져와 String tax로 선언
String profit = jtxtTotal.getText(); //순수익을 jtxtTotal에서 값을 가져와 String profit으로 선언

//데이터 저장 날짜 생성
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date date = new Date();
String created = dateFormat.format(date);

Statement stat = new Statement(item, Integer.toString(count+1), payment, amount, change,
                                tax, profit, created);
//stat 객체에 위에서 선언한 변수들을 담아준다.
StatementDAO statDAO = new StatementDAO(); //DB에 필요한 DAO생성

int result = statDAO.save(stat); //statDao에 stat를 담고 db에 저장한다.
if(result != 1) {
    JOptionPane.showMessageDialog(null, "DB 추가 실패",
                               "DB 실패", JOptionPane.ERROR_MESSAGE);
}

}

//초기화 버튼을 눌렀을때 jTable2의 값들이 초기화 되며 입력되는 모든 jTextField값들이 공백으로 설정
private void jButtonResetActionPerformed(java.awt.event.ActionEvent evt)
{
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
    model.setRowCount(0);
    jtxtChange.setText("");
    jtxtTax.setText("");
    jtxtTotal.setText("");
    jtxtSubTotal.setText("");
    jtxtDisplay.setText("");
    jtxtBarcode.setText("");
}

//영수증 출력을 위한 jTable.print 메소드 사용하여 jTable의 해당 값을 인쇄 하는 코드
private void jButtonPrintActionPerformed(java.awt.event.ActionEvent evt)
{
    MessageFormat header = new MessageFormat("Printing in progress");
    MessageFormat footer = new MessageFormat("Page {0, number, integer}");
}

```

```

try{
    jTable2.print(JTable.PrintMode.NORMAL, header, footer);
} catch(java.awt.print.PrinterException e){
    System.err.format("프린터기를 찾을 수 없습니다.",e.getMessage());
}
}

//버튼을 클릭했을때 model 객체의removeRow를 사용하여 jTable에 있는 항목을 선택하고 삭제
private void jbtnRemoveActionPerformed(java.awt.event.ActionEvent evt)
{
    DefaultTableModel model = (DefaultTableModel) jTable2.getModel(); //model 객체 선언

    //RemoveItem을 변수로 선언하고 jTable2의 항목을 클릭시 해당 열의 값을 가져옴.
    int RemoveItem = jTable2.getSelectedRow();
    if(RemoveItem >= 0){ //RemoveItem이 0보다 클시 해당 열을 삭제
        model.removeRow(RemoveItem);
    }

    ItemCost(); //ItemCost 함수 실행

    if(jcboPayment.getSelectedItem().equals("현금")){
        Change(); //Change 함수 실행
    } else{
        jtxtChange.setText(""); //거스름돈의 필드를 공백으로 설정
        jtxtDisplay.setText(""); //받은돈의 필드를 공백으로 설정
    }
}
private JFrame frame;

//프로그램 종료 버튼 실행
private void jbtnExitActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    frame = new JFrame("Exit"); //frame 객체 선언
    if(JOptionPane.showConfirmDialog(frame,"프로그램을 종료하시겠습니까?", "Point of Sale",JOptionPane.YES_NO_OPTION)==JOptionPane.YES_NO_OPTION) {
        System.exit(0); //프로그램 종료 창 생성
    }
}

private void jbtnListActionPerformed(java.awt.event.ActionEvent evt)
{
    ListForm listForm = new ListForm();
    listForm.show();
}

//로그인 폼을 실행하는 메서드
public void login() {
    LoginForm loginForm = new LoginForm();

    loginForm.show();
}

public static void main(String args[])
{

```

```

/* Set the Nimbus look and feel */
Look and feel setting code (optional)
/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new LoginForm().setVisible(true); //프로젝트 실행시 로그인폼 창 생성
        //new JavaPOS().setVisible(true);
    }
});
```

}

ListForm.java

```

import java.util.ArrayList;

public class ListForm extends javax.swing.JFrame {
    public ListForm() {
        initComponents();
        getList(); // ListForm이 실행될 경우(표시될 경우) 바로 DB 리스트 출력
    }

    @SuppressWarnings("unchecked")
    Generated Code
    private void btnExitActionPerformed(java.awt.event.ActionEvent evt) {
        dispose(); // 리스트 창 종료
    }

    private void btnGetListActionPerformed(java.awt.event.ActionEvent evt) {
        getList(); // DB데이터를 표시(갱신으로 사용)
    }

    public void getList() {
        String strOutput = "번호\t판매상품\t수량\t결제수단\t가격\t거스름돈\t세금\t손수익\t시간\n";
        try {
            Statement stat = new Statement();
            StatementDAO statDAO = new StatementDAO();
            ArrayList<Statement> list = statDAO.getList();

            for(int i=0; i < list.size(); i++) { // 리스트로 출력
                strOutput += list.get(i).getNum() + "\t"; // 거래내역 키
                strOutput += list.get(i).getItem() + "\t"; // 판매 상품
                strOutput += list.get(i).getQty() + "\t"; // 판매 수량
                strOutput += list.get(i).getPayment() + "\t"; // 결제수단
                strOutput += list.get(i).getAmount() + "\t"; // 가격
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

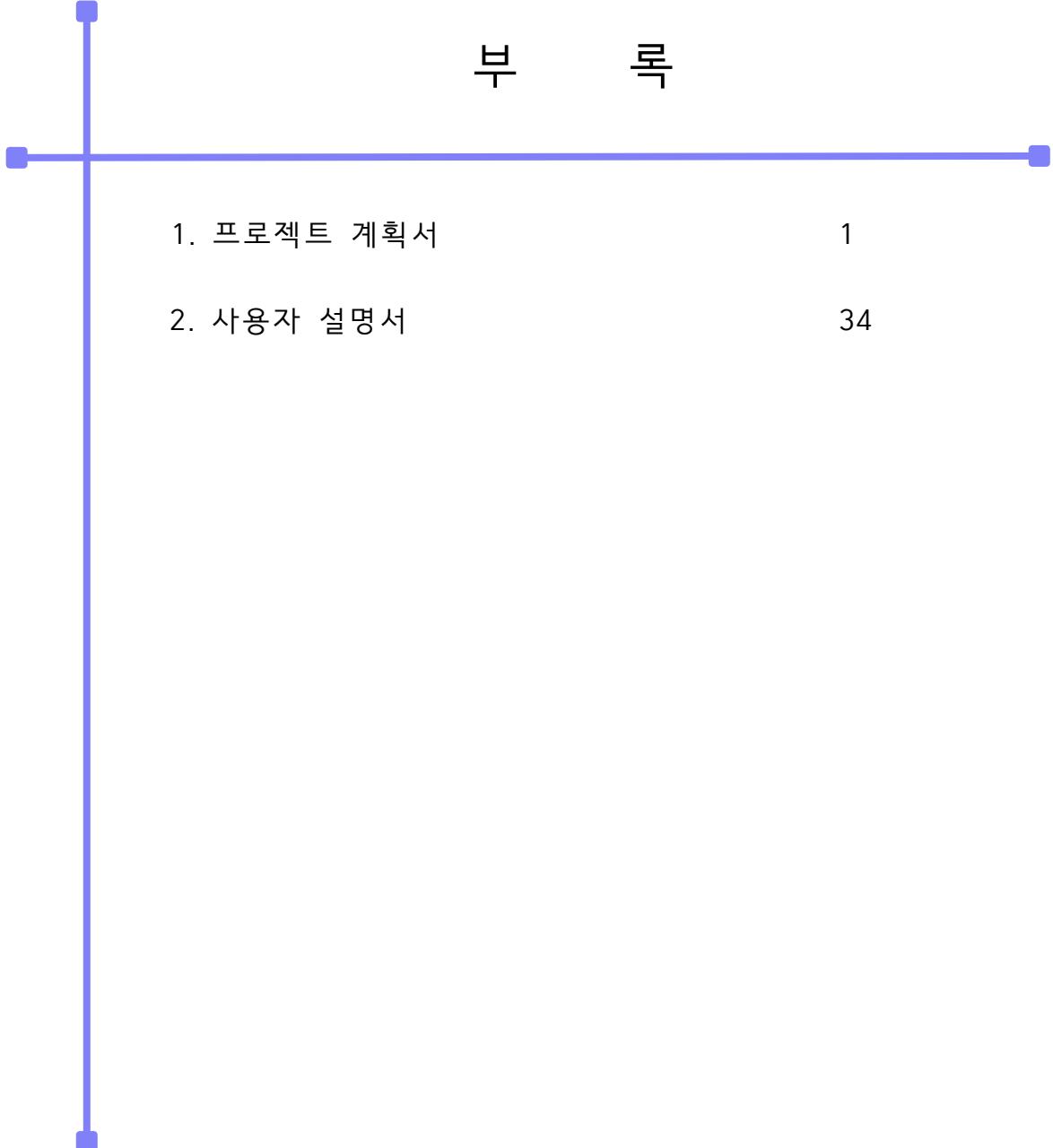
```

        strOutput += list.get(i).getChange() + "\t";    // 거스름돈
        strOutput += list.get(i).getTax() + "\t";        // 세금
        strOutput += list.get(i).getProfit() + "\t";     // 손수익
        strOutput += list.get(i).getCreated() + "\n";    // 거래 시간
    }
    jTextArea1.append(strOutput);
} catch(Exception e) {
    System.out.println("SQLException : " + e.getMessage());
}
}

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new ListForm().setVisible(true);
        }
    });
}

```

- 사용자 메뉴얼 (부록)



부 록

- | | |
|-------------|----|
| 1. 프로젝트 계획서 | 1 |
| 2. 사용자 설명서 | 34 |

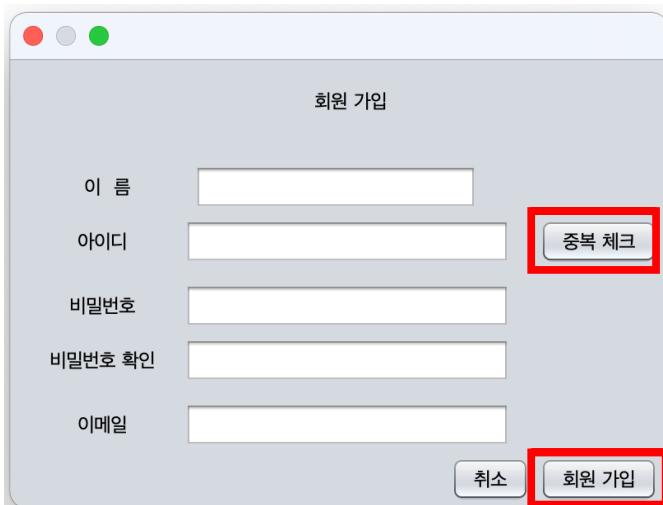
POS 사용자 설명서

1. 로그인 및 회원가입

POS를 사용하기 위해 로그인합니다. 아이디가 없다면 회원가입을 우선 진행합니다.



회원가입 버튼을 누릅니다.



이름, 아이디, 비밀번호, 이메일을 입력합니다.

아이디 중복체크를 통해 사용할 수 있는 아이디인지 확인합니다.

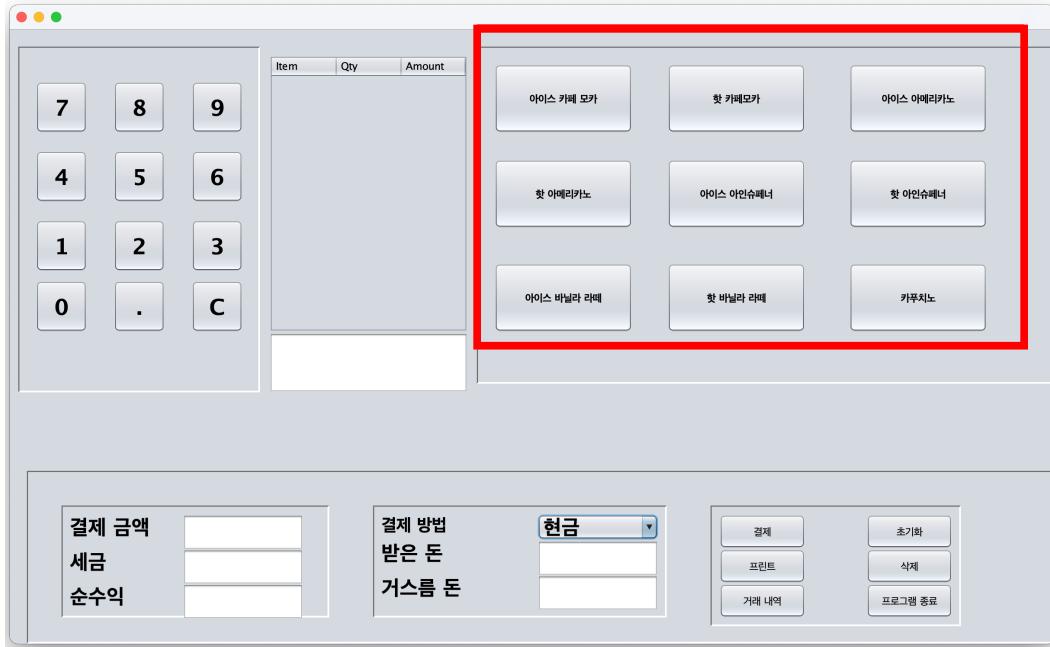
모두 입력했으면 회원가입 버튼을 클릭합니다.



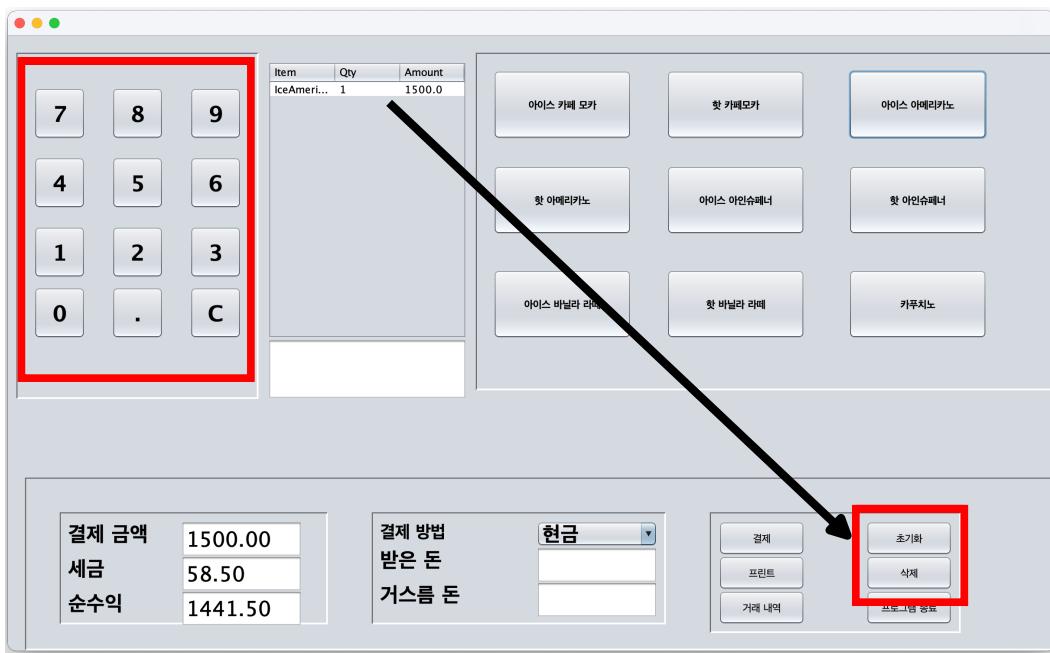
아이디, 비밀번호를 입력 후 로그인 버튼을 클릭하여 POS 화면으로 들어갑니다.

2. POS 주문/결제

POS를 사용하여 카페음료를 주문받고 결제하는 내용입니다.



주문받은 음료(메뉴) 버튼을 클릭합니다. 예시로 아이스 아메리카노를 클릭해 보겠습니다.

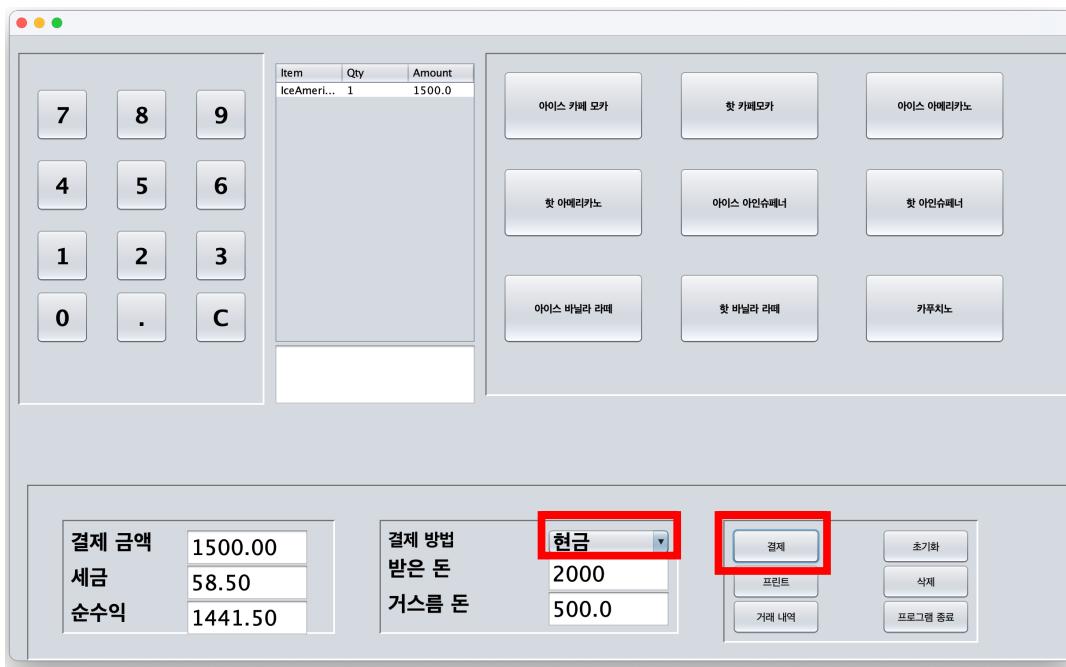


아이스 아메리카노가 선택되어 결제 금액, 세금, 순수익이 표시된 상태입니다.

키패드를 이용해 손님에게 받은 금액을 입력합니다.

잘못 클릭한 경우 리스트에서 삭제할 항목을 선택하고 삭제 버튼을 클릭합니다.

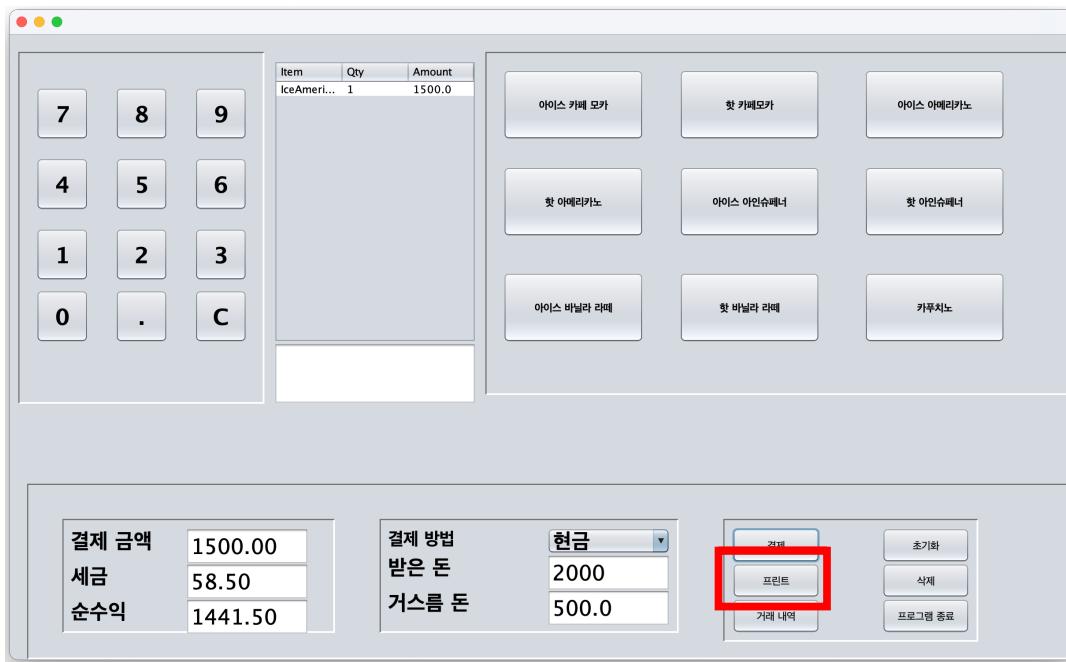
모두 삭제할 경우 초기화 버튼을 클릭합니다.



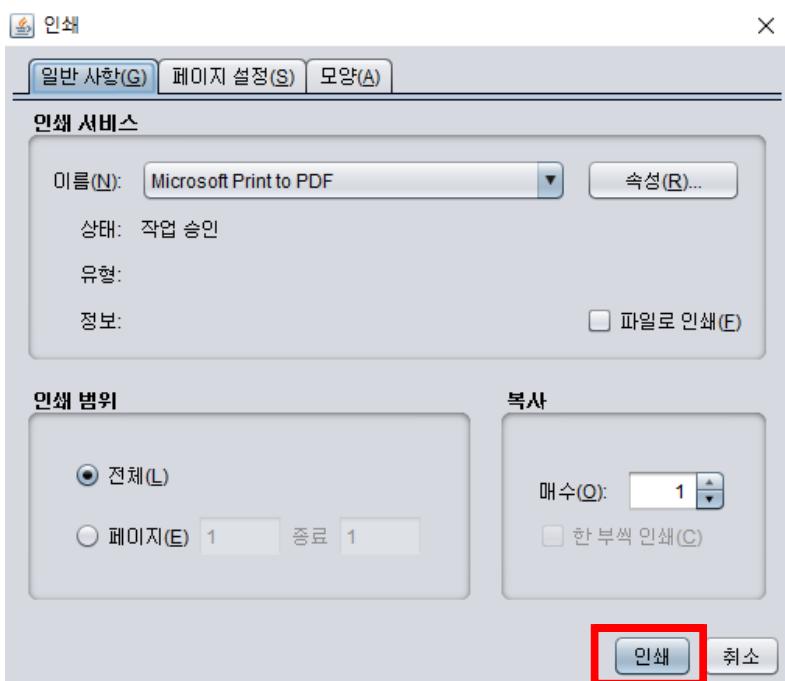
카페드를 이용해 손님에게 받은 돈 2000원을 입력한 상태입니다.

결제를 진행하기 위해 결제 방법을 선택하고 결제 버튼을 클릭합니다.

3. 영수증 출력



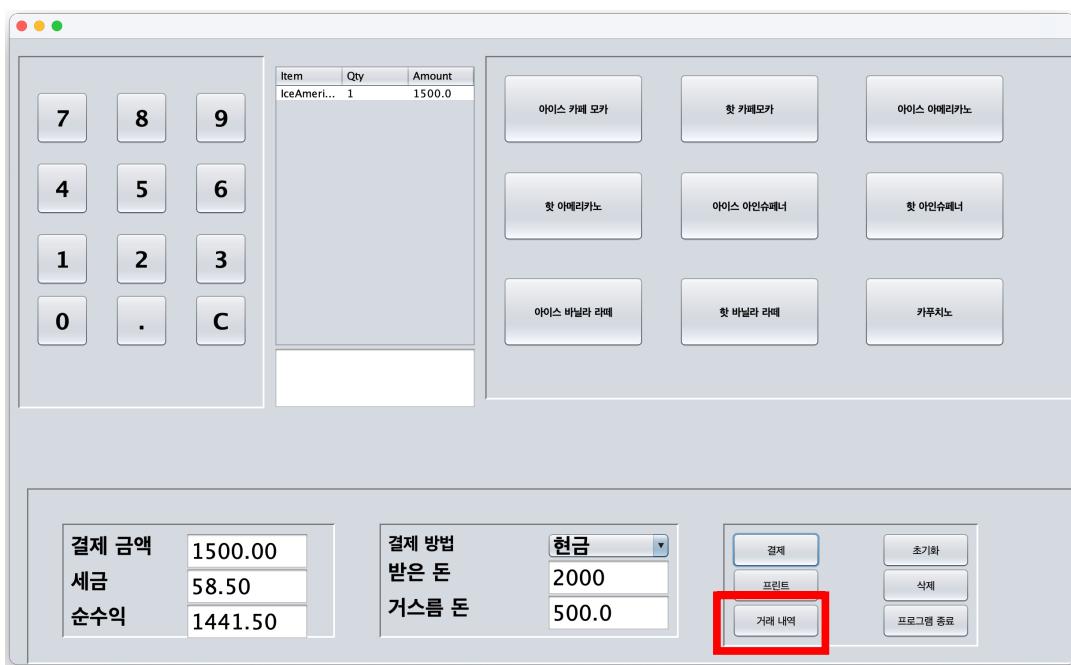
영수증이 필요할시 프린트 버튼을 클릭합니다.



다음 화면에서 프린터 이름, 인쇄 범위, 매수를 선택하여 인쇄 버튼을 클릭합니다.
페이지 설정과 모양은 상단 탭에 들어가 설정할 수 있습니다.

4. 거래 내역 확인

여태까지의 거래 내역을 보는 방법입니다.



거래 내역을 클릭합니다.



자동으로 거래 내역 리스트가 출력됩니다.

갱신이 필요할 경우 갱신 버튼을 클릭합니다.

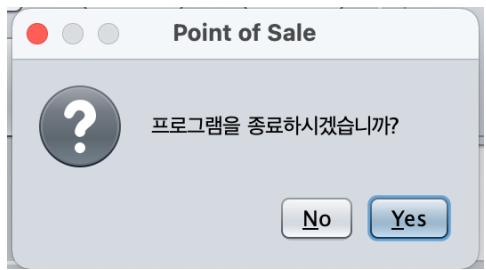
거래 내역은 프로그램을 종료하고 다시 실행시켜도 그대로 남아있습니다.

5. POS 프로그램 종료

POS 프로그램을 종료하는 방법입니다.



프로그램 종료 버튼을 클릭합니다.



YES 버튼을 클릭하여 프로그램을 종료합니다.

NO 버튼을 클릭 할 경우 프로그램은 종료되지 않습니다.