**자바프로그래밍Ⅱ**

**기말프로젝트 보고서**

**<은행 계좌 관리 시스템>**

과목명 : 자바프로그래밍 Ⅱ

과목 담당교수: 김지연 교수님

제출일 : 2024년 12월 10일

|  |  |
| --- | --- |
| 학번 | 이름 |
| 202195039 | 임서윤 |

|  |  |
| --- | --- |
| * **프로젝트 명 :** | 은행 계좌 관리 시스템 |

|  |  |
| --- | --- |
| * **프로젝트 개발 목적 :** | 은행 시스템으로 계좌 개설, 입금, 출금, 잔액 조회, 거래 내역 조회 기능을 만들어 정보를 저장한다. |

|  |  |
| --- | --- |
| * **기대 효과 :** | 실제 온라인 업무를 볼 수 있는 은행들처럼 구상하기에 평소처럼 사용 가능. |

* **프로젝트 주요 기능**

|  |  |  |
| --- | --- | --- |
| 주요 기능 | 설명 | 비고 |
| 계좌 개설 | 개설을 하게 되면 실제 계좌번호들 처럼 랜덤숫자들로 이루어진 계좌번호를 발급 받을 수 있다. |  |
| 입금 | 계좌번호와 입금액을 입력받아 해당 계좌에 돈을 입금 |  |
| 출금 | 계좌번호와 출금액을 입력받아 해당 계좌에서 돈을 출금. 잔액이 부족하면 출금을 거부. |  |
| 잔액 조회 | 계좌번호를 입력받아 해당 계좌의 현재 잔액을 보여줌. |  |
| 거래 내역 조회 | 계좌번호를 입력받아 해당 계좌의 입출금 내역을 보여줌. |  |
| 종료 | 버튼을 누르면 자동종료됨. |  |

* **데이터베이스**

|  |  |  |  |
| --- | --- | --- | --- |
| 데이터베이스명 | bankaccountsystem | | |
| 테이블 명 | accounts, transactions, users | | |
|  | | | |
| 필드명 | 데이터 타입 | 용도 | 비고 |
| account\_number | Varchar(255) | 계좌 번호 (고유한 값) |  |
| user\_id | Varchar(255) | 계좌 소유자의 사용자 ID |  |
| balance | Decimal(10,2) | 계좌 잔액 |  |
| Transaction\_id | Int | 각 거래 내역을 구분하는 고유한 ID (자동 증가) |  |
| Account\_number | Varchar(255) | 거래가 발생한 계좌 번호 |  |
| Transaction\_type | Varchar(255) | 거래 종류 (예: 입금, 출금, 이체) |  |
| amount | Decimal(10,2) | 거래금액 |  |  |  |  |
| Transaction\_date | Timestamp | 거래가 발생한 날짜와 시간 |  |  |  |  |
| Id | Varchar(255) | 사용자 ID (고유한 값) |  |  |  |  |
| Password | Varchar(255) | 사용자 비밀번호 |  |  |  |  |
| name | Varchar(255) | 사용자 이름 |  |  |  |  |

* **데이터베이스 및 테이블 생성 화면**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* **제목**

|  |  |
| --- | --- |
| 클래스명 | LoginFrame |
| 클래스 설명 | 프로그램을 실행 시키면 맨 처음 로그인 창이 뜨는데 그거에 대한 코드. |
|  | |
| 메소드명 | 설명 |
| LoginFrame | 객체를 생성하고 초기화하는 역할을 함. |
| Login() | 사용자가 입력한 ID와 비밀번호를 사용하여 데이터베이스에 로그인을 시도하는 역할을 합니다. |
| createRegisterFrame() | RegisterFrame 객체를 생성하고 화면에 표시하는 역할을 합니다. |
| Main() | 프로그램의 시작점입니다. |

관련 코드 삽입

package DB;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

public class LoginFrame extends JFrame {

private JTextField idField; // 아이디 입력 필드

private JPasswordField passwordField; // 비밀번호 입력 필드

public LoginFrame() {

// 로그인 프레임 설정

setTitle("로그인"); // 프레임 제목 설정

setSize(300, 200); // 프레임 크기 설정

setLocationRelativeTo(null); // 화면 중앙에 프레임 위치

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 프레임 닫히면 프로그램 종료

JPanel loginPanel = new JPanel(); // 로그인 관련 컴포넌트를 담을 패널

loginPanel.setLayout(new GridLayout(4, 2, 10, 10)); // 4행 2열 GridLayout, 컴포넌트 간 간격 10픽셀

// 컴포넌트 생성 및 배치

JLabel idLabel = new JLabel("ID:"); // 아이디 라벨

idField = new JTextField(); // 아이디 입력 필드

JLabel passwordLabel = new JLabel("비밀번호:"); // 비밀번호 라벨

passwordField = new JPasswordField(); // 비밀번호 입력 필드

JButton loginButton = new JButton("로그인"); // 로그인 버튼

JButton registerButton = new JButton("회원가입"); // 회원가입 버튼

loginPanel.add(idLabel);

loginPanel.add(idField);

loginPanel.add(passwordLabel);

loginPanel.add(passwordField);

loginPanel.add(new JLabel()); // 빈 공간 (레이아웃 맞추기 위해)

loginPanel.add(new JLabel()); // 빈 공간

loginPanel.add(loginButton);

loginPanel.add(registerButton);

add(loginPanel); // 프레임에 패널 추가

// 로그인 버튼 클릭 시 동작

loginButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

login(); // 로그인 처리 메소드 호출

} catch (SQLException ex) {

throw new RuntimeException(ex); // SQLException 발생 시 RuntimeException으로 변환

}

}

});

// 회원가입 버튼 클릭 시 동작

registerButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

createRegisterFrame(); // 회원가입 화면 생성 메소드 호출

}

});

}

// 로그인 처리

private void login() throws SQLException {

String id = idField.getText(); // 입력된 아이디 가져오기

String password = new String(passwordField.getPassword()); // 입력된 비밀번호 가져오기

// 데이터베이스 연결 및 로그인 검증

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bankaccountsystem", "root", "1234"); // MySQL 연결 (URL, 아이디, 비밀번호)

Statement stmt = conn.createStatement(); // SQL 쿼리 실행 위한 Statement 객체 생성

String sql = "SELECT \* FROM users WHERE id = '" + id + "' AND password = '" + password + "'"; // 아이디와 비밀번호로 사용자 정보 조회 쿼리

ResultSet rs = stmt.executeQuery(sql); // 쿼리 실행 결과 ResultSet으로 받음

if (rs.next()) { // 쿼리 결과가 있으면 (로그인 성공)

JOptionPane.showMessageDialog(this, rs.getString("name") + "님 환영합니다!"); // 환영 메시지 출력

// BankProgramGUI에 연결 정보 전달

BankProgramGUI.connectToDatabase("root", "1234"); // BankProgramGUI에서 사용할 DB 연결 정보 설정 (아이디, 비밀번호)

new BankProgramGUI(id).setVisible(true); // 메인 화면 호출, 사용자 ID 전달

dispose(); // 로그인 창 닫기

} else { // 쿼리 결과가 없으면 (로그인 실패)

JOptionPane.showMessageDialog(this, "로그인 실패"); // 실패 메시지 출력

}

conn.close(); // 데이터베이스 연결 종료

}

// 회원가입 화면 생성

private void createRegisterFrame() {

new RegisterFrame().setVisible(true); // RegisterFrame 객체 생성 및 화면 표시

dispose(); // 현재 로그인 창 닫기

}

public static void main(String[] args) {

// 로그인 화면 표시

new LoginFrame().setVisible(true); // LoginFrame 객체 생성 및 화면 표시

}

}

* **제목**

|  |  |
| --- | --- |
| 클래스명 | RegisterFrame |
| 클래스 설명 | 로그인전에 회원가입을 해야되므로 회원가입 창에 대한 코드. |
|  | |
| 메소드명 | 설명 |
| RegisterFrame() (생성자) | 프레임의 제목, 크기, 위치 등을 설정합니다. |
| register() | 입력 필드에서 ID, 비밀번호, 이름 값을 가져옵니다.  register() 메소드는 실제 회원가입 처리를 담당합니다. |
|  |  |
|  |  |
|  |  |
|  |  |

관련 코드 삽입

package DB;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

public class RegisterFrame extends JFrame {

private JTextField idField; // 아이디 입력 필드

private JPasswordField passwordField; // 비밀번호 입력 필드

private JTextField nameField; // 이름 입력 필드

public RegisterFrame() {

// 회원가입 프레임 설정

setTitle("회원가입"); // 프레임 제목 설정

setSize(300, 250); // 프레임 크기 설정

setLocationRelativeTo(null); // 화면 중앙에 프레임 위치

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 프레임 닫히면 프로그램 종료

JPanel registerPanel = new JPanel(); // 회원가입 관련 컴포넌트를 담을 패널

registerPanel.setLayout(new GridLayout(4, 2, 10, 10)); // 4행 2열 GridLayout, 컴포넌트 간 간격 10픽셀

// 컴포넌트 생성 및 배치

JLabel idLabel = new JLabel("ID:"); // 아이디 라벨

idField = new JTextField(); // 아이디 입력 필드

JLabel passwordLabel = new JLabel("비밀번호:"); // 비밀번호 라벨

passwordField = new JPasswordField(); // 비밀번호 입력 필드

JLabel nameLabel = new JLabel("이름:"); // 이름 라벨

nameField = new JTextField(); // 이름 입력 필드

JButton registerButton = new JButton("가입"); // 회원가입 버튼

registerPanel.add(idLabel);

registerPanel.add(idField);

registerPanel.add(passwordLabel);

registerPanel.add(passwordField);

registerPanel.add(nameLabel);

registerPanel.add(nameField);

registerPanel.add(new JLabel()); // 빈 공간 (레이아웃 맞추기 위해)

registerPanel.add(registerButton);

add(registerPanel); // 프레임에 패널 추가

// 회원가입 버튼 클릭 시 동작

registerButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

register(); // 회원가입 처리 메소드 호출

} catch (SQLException ex) {

throw new RuntimeException(ex); // SQLException 발생 시 RuntimeException으로 변환

}

}

});

}

// 회원가입 처리

private void register() throws SQLException {

String id = idField.getText(); // 입력된 아이디 가져오기

String password = new String(passwordField.getPassword()); // 입력된 비밀번호 가져오기

String name = nameField.getText(); // 입력된 이름 가져오기

// 데이터베이스 연결 및 회원가입 처리

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bankaccountsystem", "root", "1234"); // MySQL 연결 (URL, 아이디, 비밀번호) - 실제 값으로 변경 필요

String sql = "INSERT INTO users (id, password, name) VALUES (?, ?, ?)"; // 사용자 정보 삽입 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) { // PreparedStatement 사용 (SQL Injection 방지)

pstmt.setString(1, id); // 아이디 설정

pstmt.setString(2, password); // 비밀번호 설정

pstmt.setString(3, name); // 이름 설정

pstmt.executeUpdate(); // 쿼리 실행

}

JOptionPane.showMessageDialog(this, "회원가입이 완료되었습니다."); // 회원가입 완료 메시지

new LoginFrame().setVisible(true); // 로그인 화면 표시

dispose(); // 현재 회원가입 창 닫기

}

}

* **제목**

|  |  |
| --- | --- |
| 클래스명 | BankProgramGUI |
| 클래스 설명 | 로그인을 완료하면 이제 실제 은행 계좌 관리하는 것 처럼 화면이 나오는 코드. |
|  | |
| 메소드명 | 설명 |
| BankProgramGUI | 로그인한 사용자 ID를 받아 GUI를 초기화합니다. |
| createAccount() | 계좌 개설 기능을 수행하는 메소드입니다. |
| deposit() | 입금 기능을 수행하는 메소드입니다. |
| withdraw() | 출금 기능을 수행하는 메소드입니다. |
| transfer() | 계좌 간 이체 기능을 수행하는 메소드입니다. |
| getTransactionHistory() | 거래 내역 조회 기능을 수행하는 메서드입니다. |
| getAccountInfo() | 계좌 정보 조회 기능을 수행하는 메서드입니다. |
| recordTransaction | 거래 내역을 데이터베이스에 저장하는 메서드입니다. |
| connectToDatabase | 데이터베이스에 연결하는 메서드입니다. |

관련 코드 삽입

package DB;

import java.awt.GridLayout;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Timestamp;

import java.util.Random;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

public class BankProgramGUI extends JFrame {

static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver"; // MySQL JDBC 드라이버 클래스 이름

static final String DB\_URL = "jdbc:mysql://localhost:3306/bankaccountsystem"; // MySQL 데이터베이스 URL

static Connection conn = null; // 데이터베이스 연결 객체

static Statement stmt = null; // SQL 쿼리 실행 객체

private JTextField idField; // 사용자 ID 표시 필드

private JPasswordField passwordField; // 비밀번호 입력 필드 (현재 사용 안 함)

private String userId; // 로그인한 사용자 ID

public BankProgramGUI(String userId) {

this.userId = userId; // 로그인 화면에서 전달받은 사용자 ID 저장

// 프레임 설정

setTitle("Bank Program"); // 프레임 제목 설정

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); // 프레임 닫히면 프로그램 종료

setSize(400, 300); // 프레임 크기 설정

setLocationRelativeTo(null); // 화면 중앙에 프레임 위치

// 메인 패널에 버튼 추가

JPanel mainPanel = new JPanel(); // 버튼들을 담을 패널

mainPanel.setLayout(new GridLayout(7, 1, 10, 10)); // 7행 1열 GridLayout, 컴포넌트 간 간격 10픽셀

// 버튼 생성

JButton createAccountButton = new JButton("계좌 개설");

JButton depositButton = new JButton("입금");

JButton withdrawButton = new JButton("출금");

JButton transferButton = new JButton("계좌 간 이체");

JButton transactionHistoryButton = new JButton("거래 내역 조회");

JButton accountInfoButton = new JButton("계좌 정보 조회");

JButton exitButton = new JButton("종료");

// 패널에 버튼 추가

mainPanel.add(createAccountButton);

mainPanel.add(depositButton);

mainPanel.add(withdrawButton);

mainPanel.add(transferButton);

mainPanel.add(transactionHistoryButton);

mainPanel.add(accountInfoButton);

mainPanel.add(exitButton);

add(mainPanel); // 프레임에 패널 추가

// 각 버튼에 액션 리스너 추가 (람다식 사용)

createAccountButton.addActionListener(e -> createAccount());

depositButton.addActionListener(e -> deposit());

withdrawButton.addActionListener(e -> withdraw());

transferButton.addActionListener(e -> transfer());

transactionHistoryButton.addActionListener(e -> getTransactionHistory());

accountInfoButton.addActionListener(e -> getAccountInfo());

exitButton.addActionListener(e -> System.exit(0));

// idField 초기화 (로그인한 사용자 ID 표시)

idField = new JTextField(userId);

idField.setEditable(false); // 편집 불가능하도록 설정

}

// 계좌 개설 기능

private void createAccount() {

try {

String accountNumber = String.valueOf(new Random().nextLong()).substring(1, 11); // 랜덤 계좌 번호 생성

String sql = "INSERT INTO accounts (account\_number, user\_id, balance) VALUES (?, ?, ?)"; // 계좌 정보 삽입 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) { // PreparedStatement 사용 (SQL Injection 방지)

pstmt.setString(1, accountNumber); // 계좌 번호 설정

pstmt.setString(2, userId); // 사용자 ID 설정

pstmt.setDouble(3, 0); // 초기 잔액 0 설정

pstmt.executeUpdate(); // 쿼리 실행

}

JOptionPane.showMessageDialog(this, "계좌가 개설되었습니다. 계좌 번호: " + accountNumber); // 계좌 개설 성공 메시지

} catch (SQLException e) {

e.printStackTrace(); // 예외 발생 시 스택 트레이스 출력

JOptionPane.showMessageDialog(this, "계좌 개설 실패"); // 계좌 개설 실패 메시지

}

}

// 입금 기능

private void deposit() {

try {

String accountNumber = JOptionPane.showInputDialog(this, "계좌 번호를 입력하세요:"); // 계좌 번호 입력

String amountStr = JOptionPane.showInputDialog(this, "입금액을 입력하세요:"); // 입금액 입력

double amount = Double.parseDouble(amountStr); // 문자열을 숫자로 변환

String sql = "UPDATE accounts SET balance = balance + ? WHERE account\_number = ? AND user\_id = ?"; // 잔액 업데이트 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setDouble(1, amount); // 입금액 설정

pstmt.setString(2, accountNumber); // 계좌 번호 설정

pstmt.setString(3, userId); // 사용자 ID 설정

pstmt.executeUpdate(); // 쿼리 실행

}

recordTransaction(accountNumber, "입금", amount); // 거래 내역 기록

JOptionPane.showMessageDialog(this, "입금이 완료되었습니다."); // 입금 완료 메시지

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "입금액이 잘못되었습니다."); // 잘못된 입금액 입력 시 메시지

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "입금 실패"); // 입금 실패 메시지

}

}

// 출금 기능

private void withdraw() {

try {

String accountNumber = JOptionPane.showInputDialog(this, "계좌 번호를 입력하세요:"); // 계좌 번호 입력

String amountStr = JOptionPane.showInputDialog(this, "출금액을 입력하세요:"); // 출금액 입력

double amount = Double.parseDouble(amountStr); // 문자열을 숫자로 변환

String sql = "SELECT balance FROM accounts WHERE account\_number = ? AND user\_id = ?"; // 잔액 조회 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, accountNumber); // 계좌 번호 설정

pstmt.setString(2, userId); // 사용자 ID 설정

try (ResultSet rs = pstmt.executeQuery()) { // 쿼리 실행 결과

if (rs.next()) { // 계좌가 존재하면

double currentBalance = rs.getDouble("balance"); // 현재 잔액 가져오기

if (currentBalance >= amount) { // 잔액이 충분하면

sql = "UPDATE accounts SET balance = balance - ? WHERE account\_number = ? AND user\_id = ?"; // 잔액 업데이트 쿼리

try (PreparedStatement pstmt2 = conn.prepareStatement(sql)) {

pstmt2.setDouble(1, amount); // 출금액 설정

pstmt2.setString(2, accountNumber); // 계좌 번호 설정

pstmt2.setString(3, userId); // 사용자 ID 설정

pstmt2.executeUpdate(); // 쿼리 실행

}

recordTransaction(accountNumber, "출금", amount); // 거래 내역 기록

JOptionPane.showMessageDialog(this, "출금이 완료되었습니다."); // 출금 완료 메시지

} else {

JOptionPane.showMessageDialog(this, "잔액이 부족합니다."); // 잔액 부족 메시지

}

} else {

JOptionPane.showMessageDialog(this, "계좌를 찾을 수 없습니다."); // 계좌 없음 메시지

}

}

}

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "출금액이 잘못되었습니다."); // 잘못된 출금액 입력 시 메시지

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "출금 실패"); // 출금 실패 메시지

}

}

// 계좌 간 이체 기능

private void transfer() {

try {

String fromAccount = JOptionPane.showInputDialog(this, "출금할 계좌 번호를 입력하세요:"); // 출금 계좌 번호 입력

String toAccount = JOptionPane.showInputDialog(this, "입금할 계좌 번호를 입력하세요:"); // 입금 계좌 번호 입력

String amountStr = JOptionPane.showInputDialog(this, "이체액을 입력하세요:"); // 이체액 입력

double amount = Double.parseDouble(amountStr); // 문자열을 숫자로 변환

String sql = "SELECT balance FROM accounts WHERE account\_number = ? AND user\_id = ?"; // 잔액 조회 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, fromAccount); // 출금 계좌 번호 설정

pstmt.setString(2, userId); // 사용자 ID 설정

try (ResultSet rs = pstmt.executeQuery()) { // 쿼리 실행 결과

if (rs.next()) { // 출금 계좌가 존재하면

double currentBalance = rs.getDouble("balance"); // 현재 잔액 가져오기

if (currentBalance >= amount) { // 잔액이 충분하면

sql = "UPDATE accounts SET balance = balance - ? WHERE account\_number = ? AND user\_id = ?"; // 출금 계좌 잔액 업데이트 쿼리

try (PreparedStatement pstmt2 = conn.prepareStatement(sql)) {

pstmt2.setDouble(1, amount); // 이체액 설정

pstmt2.setString(2, fromAccount); // 출금 계좌 번호 설정

pstmt2.setString(3, userId); // 사용자 ID 설정

pstmt2.executeUpdate(); // 쿼리 실행

}

sql = "UPDATE accounts SET balance = balance + ? WHERE account\_number = ?"; // 입금 계좌 잔액 업데이트 쿼리

try (PreparedStatement pstmt3 = conn.prepareStatement(sql)) {

pstmt3.setDouble(1, amount); // 이체액 설정

pstmt3.setString(2, toAccount); // 입금 계좌 번호 설정

pstmt3.executeUpdate(); // 쿼리 실행

}

recordTransaction(fromAccount, "이체 출금", amount); // 출금 계좌 거래 내역 기록

recordTransaction(toAccount, "이체 입금", amount); // 입금 계좌 거래 내역 기록

JOptionPane.showMessageDialog(this, "이체가 완료되었습니다."); // 이체 완료 메시지

} else {

JOptionPane.showMessageDialog(this, "잔액이 부족합니다."); // 잔액 부족 메시지

}

} else {

JOptionPane.showMessageDialog(this, "출금 계좌를 찾을 수 없습니다."); // 출금 계좌 없음 메시지

}

}

}

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(this, "이체액이 잘못되었습니다."); // 잘못된 이체액 입력 시 메시지

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "이체 실패"); // 이체 실패 메시지

}

}

// 거래 내역 조회 기능

private void getTransactionHistory() {

try {

String sql = "SELECT \* FROM transactions WHERE user\_id = ?"; // 거래 내역 조회 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, userId); // 사용자 ID 설정

try (ResultSet rs = pstmt.executeQuery()) { // 쿼리 실행 결과

StringBuilder transactionHistory = new StringBuilder(); // 거래 내역 저장할 StringBuilder

while (rs.next()) { // 각 거래 내역을 StringBuilder에 추가

transactionHistory.append("계좌 번호: " + rs.getString("account\_number") + ", ");

transactionHistory.append("거래 종류: " + rs.getString("transaction\_type") + ", ");

transactionHistory.append("금액: " + rs.getDouble("amount") + ", ");

transactionHistory.append("거래 시간: " + rs.getTimestamp("transaction\_date") + "\n");

}

JOptionPane.showMessageDialog(this, transactionHistory.toString()); // 거래 내역 출력

}

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "거래 내역 조회 실패"); // 거래 내역 조회 실패 메시지

}

}

// 계좌 정보 조회 기능

private void getAccountInfo() {

try {

String sql = "SELECT \* FROM accounts WHERE user\_id = ?"; // 계좌 정보 조회 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, userId); // 사용자 ID 설정

try (ResultSet rs = pstmt.executeQuery()) { // 쿼리 실행 결과

StringBuilder accountInfo = new StringBuilder(); // 계좌 정보 저장할 StringBuilder

while (rs.next()) { // 각 계좌 정보를 StringBuilder에 추가

accountInfo.append("계좌 번호: " + rs.getString("account\_number") + ", ");

accountInfo.append("잔액: " + rs.getDouble("balance") + "\n");

}

JOptionPane.showMessageDialog(this, accountInfo.toString()); // 계좌 정보 출력

}

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "계좌 정보 조회 실패"); // 계좌 정보 조회 실패 메시지

}

}

// 거래 내역 저장

private void recordTransaction(String accountNumber, String transactionType, double amount) {

try {

String sql = "INSERT INTO transactions (account\_number, transaction\_type, amount, user\_id, transaction\_date) VALUES (?, ?, ?, ?, ?)"; // 거래 내역 삽입 쿼리

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, accountNumber); // 계좌 번호 설정

pstmt.setString(2, transactionType); // 거래 종류 설정 (입금, 출금, 이체 등)

pstmt.setDouble(3, amount); // 거래 금액 설정

pstmt.setString(4, userId); // 사용자 ID 설정

pstmt.setTimestamp(5, new Timestamp(System.currentTimeMillis())); // 현재 시간 설정

pstmt.executeUpdate(); // 쿼리 실행

}

} catch (SQLException e) {

e.printStackTrace();

}

}

// 데이터베이스 연결 메서드

public static void connectToDatabase(String username, String password) {

try {

Class.forName(JDBC\_DRIVER); // JDBC 드라이버 로드

conn = DriverManager.getConnection(DB\_URL, username, password); // 데이터베이스 연결

stmt = conn.createStatement(); // Statement 객체 생성

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

// LoginFrame에서 호출되므로 main 메서드는 필요하지 않습니다.

}

}

|  |  |
| --- | --- |
| 화면명 | 로그인 화면(접속 성공 및 실패) |
| 설명 | 로그인 화면과 로그인이 성공하면 “(이름)님 환영합니다!”가 뜨고, 실패하게 된다면 “로그인 실패”뜸. |
| 화면 캡쳐 | |
|  | |

|  |  |
| --- | --- |
| 화면명 | 회원가입 창 |
| 설명 | 로그인 전에 회원가입을 할려면 ID, 비밀번호, 이름을 입력해야지만 회원가입이 가능해진다. |
| 화면 캡쳐 | |
|  | |

|  |  |
| --- | --- |
| 화면명 | 로그인 후 화면 |
| 설명 | 로그인 이후에 뜨는 화면인데 이 화면에서 은행 계좌 관리를 할 수 있다. |
| 화면 캡쳐 | |
| 계좌 개설    입금        출금          계좌 간 이채        실패 시    성공 시    거래 내역 조회    계좌 정보 조회 | |

* **프로젝트 완성 후 느낀점**

|  |  |
| --- | --- |
| 이름 | 느낀점 |
| 처음으로 혼자 해본 프로젝트인데 코딩이 아직 익숙하지 않아 처음에는 힘들었지만 모르는게 있을 때는 지피티나 친구들의 도움으로 잘 해결할 수 있었다. 프로젝트를 함으로써 조금이라도 자바에 대한 공부를 알차게 한 것 같다. | |