2024학년도 자바 프로젝트 완료 보고서

노인 친화적 ATM
윈도우 응용프로그램 개발

2024년 12월 15일

컴퓨터정보과

팀명	뱅크 알고리즘
팀장	조예원(202344064)
팀원	1. 용다인(202344043) 2. 김민재(202144046)

인하공업전문대학
INHA TECHNICAL COLLEGE

2024년도

컴퓨터정보과 특성화 사업

자바 프로젝트 진행 결과보고서

(분야 : 윈도우 응용프로그램)

연구과제명 : 노인친화적 ATM 윈도우 응용프로그램 개발

2024. 12. 15

인하공업전문대학
INHA TECHNICAL COLLEGE

본 결과물은 인하공업전문대학 컴퓨터정보과 자바 프로젝트 수업의 연구 결과입니다.

제 출 문

- 분　　야 : 자바 윈도우 응용프로그램
- 연구과제명 : 노인친화적 ATM 윈도우 응용프로그램
- 프로젝트 책임자 : 인하공업전문대학 컴퓨터정보과 조예원

2024년도 인하공업전문대학 컴퓨터정보과 자바 프로젝트 수업의
결과물로 이 보고서를 제출합니다.

2024. 12. 15

프로젝트 책임자 : 컴퓨터정보과(202344064) 조 예 원 (인)

공동 참여자 : 컴퓨터정보과(202344043) 용 다 인 (인)

공동 참여자 : 컴퓨터정보과(202144046) 김 민 재 (인)

인하공업전문대학 컴퓨터정보과 학과장 귀하
프로젝트 요약문

분 야	자바 윈도우 응용프로그램 개발
프로젝트명	노인친화적 ATM 윈도우 응용프로그램
프로젝트 책임자	컴퓨터정보과(202344064) 조예원

연 구 내 용

1. UI/UX 설계
   - 간소화된 인터페이스: 큰 버튼과 명확한 라벨을 사용하여 복잡한 작업 단계를 최소화했습니다.
   - 가독성 강화: 노년층의 시력 저하를 고려하여 큰 글씨와 높은 명암 대비 색상 조합을 적용했습니다.
   - 피드백 메시지 제공: 작업 완료 및 오류 상황에 대한 명확한 피드백 메시지를 제공합니다.
2. 기능 구현
   - 기본 금융 서비스:
   - 계좌 개설: 전화번호 중복 검사 및 비밀번호 유효성 검사를 통해 계좌 생성
   - 잔액 조회: 계좌 번호 입력 후 현재 잔액 확인 기능 제공
   - 입금/출금: 간단한 과정으로 입출금을 수행하며, 작업 완료 후 거래 기록을 남김
   - 송금: 수취인 정보와 계좌 비밀번호를 검증하여 안전한 송금 제공
   - 비밀번호 변경: 기존 비밀번호 확인 및 새 비밀번호 입력 과정을 통해 보안 강화
   - 거래 내역 조회: 사용자가 특정 계좌의 거래 기록을 확인할 수 있도록 지원
3. 보안 기능
   - 비밀번호 검증: 숫자 4자리 비밀번호로 간소화하면서도 안전한 인증을 유지.
   - 자동 로그아웃 및 화면 전환: 작업 완료 후 일정 시간 뒤 메인 화면으로 전환하여 보안을 유지
4. 노인 친화적 기술 적용
   - 단순한 흐름: 복잡한 금융 용어 대신 쉬운 설명과 사용자 안내 메시지를 사용.
   - 오류 예방: 필수 입력값 누락 및 잘못된 입력에 대한 실시간 오류 메시지를 제공.

참여자 인적사항

1. 프로젝트 팀장

성 명		조예원	학 번	202344064
학 과		컴퓨터정보과		
연락처	H.P	010-9120-9878		
	E-mail	q5556@naver.com		

2. 프로젝트 팀원

성 명		용다인	학 번	202344043
학 과		컴퓨터정보과		
연락처	H.P	010-8899-7343		
	E-mail	cccohdlakk339@naver.com		

성 명		김민재	학 번	202144046
학 과		컴퓨터정보과		
연락처	H.P	010-8512-0389		
	E-mail	minjae549@naver.com		

목 차

1. 프로젝트 목적

노년층을 대상으로 한 디지털 금융 서비스 제공을 목적으로, 사용 편의성과 안전성을 강화한 노인친화적 ATM 프로그램을 개발합니다.

본 프로젝트는 디지털 금융 환경에서 노년층이 겪는 불편함을 해결하고, 금융 접근성을 높이는 것을 주요 목표로 합니다.

2. 프로젝트 목표 및 기대효과

프로젝트 목표

- 노년층이 쉽게 사용할 수 있는 간소화된 사용자 인터페이스(UI) 제공.
- 입금, 출금, 송금, 계좌 조회 등 주요 금융 거래 기능의 안전한 구현.
- 비밀번호 보안 및 오류 방지 기능을 통해 금융 서비스의 신뢰성 확보.
- 사용자의 실수를 최소화하고 금융 사고를 예방할 수 있는 실시간 안내 시스템 제공.

기대 효과

- 디지털 금융 서비스를 처음 접하는 노년층도 손쉽게 이용할 수 있어 디지털 소외를 해소.
- 금융 사고를 방지하고, 사용자에게 신뢰감을 제공하여 사용자 만족도 향상.
- 노년층 사용자의 독립적인 금융 관리 능력을 지원.
- 간소화된 UI를 기반으로 타 연령층 사용자에게도 직관적이고 편리한 금융 서비스를 제공.

3. 프로젝트 진행범위 및 방법

진행 범위

기능 구현:

- 계좌 개설, 계좌 조회, 입금, 출금, 송금, 거래 내역 조회, 비밀번호 변경.

UI/UX 설계:

- 노년층의 사용성을 고려한 큰 글씨, 직관적인 버튼, 명확한 피드백 메시지.

보안 강화:

- 비밀번호 검증(숫자 4자리)과 사용자 인증을 통해 금융 사고 방지.

사용자 피드백 기반 개선:

- 테스트 후, 사용자의 요구에 따라 기능 및 인터페이스 조정.

진행 방법

요구사항 분석:

- 노년층의 디지털 금융 사용 현황 및 주요 불편 사항 파악.

시스템 설계:

- 데이터베이스 및 프로그램 흐름 설계.

기능 구현:

- Java를 활용하여 계좌 관리, 거래 처리 기능 개발.

UI/UX 설계 및 테스트:

- 사용성 테스트를 통해 인터페이스 개선.

배포 및 유지보수:

- 사용자 피드백 반영 및 프로그램 업데이트.

4. 프로젝트 주요 내용

1) 사용자 인터페이스(UI)

큰 버튼과 큰 글씨로 노년층이 쉽게 조작할 수 있는 화면 제공.

단순한 단계로 금융 거래 절차를 설계하여 오류를 줄임.

2) 주요 기능

계좌 개설: 전화번호 중복 확인 및 비밀번호 설정 기능.

입출금 및 송금: 거래 완료 시 잔액 갱신 및 실시간 확인.

거래 내역 조회: 사용자가 계좌의 상세 거래 내역 확인 가능.

계좌 조회: 이름과 전화번호 입력으로 빠르고 정확한 계좌 및 잔액 확인.

비밀번호 변경: 안전한 금융 거래를 위한 비밀번호 갱신 기능 제공.

3) 보안 및 안정성

비밀번호 유효성 검사 및 사용자 인증을 통해 안전한 거래 환경 제공.

일정 시간 후 화면 전환 및 자동 로그아웃을 통해 사용자 정보를 보호.

4) 기술 스택

프로그래밍 언어: Java (Swing 기반 GUI).

데이터베이스: MySQL.

배포 환경: Java 기반 독립 실행 프로그램.

5. 프로젝트 결과물

5-1. 프로젝트명

노인친화적 ATM 윈도우 응용프로그램

5-2. 프로젝트 개요

본 프로젝트는 노년층을 대상으로 한 노인친화적 ATM 프로그램 개발을 목표로 합니다. 디지털 금융 환경에서 발생하는 노년층의 불편함과 어려움을 해소하기 위해 사용 편의성과 보안성을 강화한 프로그램을 설계 및 구현하였습니다.

이 프로그램은 계좌 개설, 잔액 조회, 입금, 출금, 송금, 거래 내역 조회, 비밀번호 변경 등 주요 금융 서비스를 제공하며, 사용자가 직관적으로 조작할 수 있는 UI/UX를 기반으로 제작되었습니다.

노년층 사용자의 디지털 금융 접근성을 높이고, 금융 사고를 예방하며, 안정적이고 신뢰할 수 있는 금융 환경을 제공하는 것이 본 프로젝트의 핵심입니다.

5-3. 프로젝트 수행 세부 일정 및 내용

■ 프로젝트 수행 세부 일정

프로젝트 진행 계획(2024. 11. 07. - 2024. 12. 11)							
진 행 내 용	책임자	1	2	3	4	5	비고
초기화면 디자인	김민재	V					기본 UI/UX 설계 완료
DB 연동	김민재	V					MySQL DB 연결
계좌 개설	조예원		V				비밀번호 유효성 검사
계좌 조회 구현	용다인		V				
입금/출금 기능 구현	조예원			V			
송금 기능 구현	조예원				V		수취인 정보 유효성 검사
거래 내역 조회 기능 구현	용다인			V			
비밀번호 변경 기능 구현	용다인				V		기존 비밀번호 확인 절차
자동 화면 전환 및 로그아웃 구현	김민재					V	
사용자 테스트	김민재					V	UI 사용성 피드백

■ 프로젝트 수행 내용

1. 기획 및 요구사항 분석
   - 프로젝트 목표 설정: 노년층의 디지털 금융 접근성을 높이고 안전한 금융 거래 환경을 제공.
   - 사용자 요구사항 분석: 노년층 사용자가 겪는 어려움을 조사하여 간소화된 UI와 실시간 피드백 시스템 필요성 도출.
   - 기능 정의:
     • 계좌 개설, 조회, 입금/출금, 송금, 거래 내역 조회, 비밀번호 변경 등 기본 금융 기능.
     • 보안 강화(4자리 숫자 비밀번호, 자동 화면 전환, 로그아웃).

2. 시스템 설계
   - UI/UX 설계:
     • 큰 글씨와 버튼, 직관적인 화면 구성으로 사용자 편의성 강화.
     • 작업 완료 후 명확한 피드백 메시지 표시.
   - 데이터베이스 설계:
     • 계좌(Account), 사용자(User), 거래 내역(Transaction)을 관리하는 MySQL 데이터베이스 구축.
   - 주요 테이블: User, Account, Transaction.
   - 기능 설계:
     • 기능별 흐름도 작성: 계좌 생성, 거래 프로세스, 송금 확인 등.

3. 구현
   - 프로그래밍 언어 및 도구:
     • Java Swing을 사용하여 GUI 구성.
     • MySQL과 연동하여 데이터 관리.
   - 기능 구현:
     • 계좌 개설: 전화번호 중복 검사, 비밀번호 유효성 검사(숫자 4자리) 적용.
     • 계좌 조회: 이름과 전화번호 입력 후 계좌 및 잔액 출력.
     • 입금/출금: 거래 후 잔액 실시간 갱신 및 거래 기록 저장.
     • 송금: 송금 프로세스 중 수취인 정보 유효성 검증.
     • 거래 내역 조회: 날짜별 필터링 및 거래 상세 정보 확인.
     • 비밀번호 변경: 기존 비밀번호 확인 후 새 비밀번호 설정.
     • 자동 화면 전환 및 로그아웃: 거래 완료 후 일정 시간 후 자동 로그아웃.

4. 테스트 및 검증
   - 단위 테스트: 각 기능(계좌 개설, 잔액 조회 등)에 대한 독립적인 테스트 수행.
   - 통합 테스트: 전체 시스템을 통합하여 데이터 흐름 및 기능 간 상호작용 확인.

5. 최종 결과

노인 친화적 ATM 프로그램 완성:

- 사용자가 손쉽게 금융 서비스를 이용할 수 있는 시스템 제공.
- 안전하고 직관적인 금융 거래 환경 구현.

5-4. 프로젝트 결과

• 데이터베이스

데이터베이스 생성 쿼리문

```
CREATE DATABASE BankSys
CHARACTER SET utf8mb4
COLLATE utf8mb4_general_ci;
USE BankSys;

-- Identity 테이블
CREATE TABLE Identity (
user_id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL UNIQUE,
phone_number VARCHAR(15),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Transaction 테이블
CREATE TABLE Transaction (
transaction_id INT AUTO_INCREMENT PRIMARY KEY,
account_id INT NOT NULL,
amount DECIMAL(15, 2) NOT NULL,
transaction_type ENUM('입금', '출금', '이체') NOT NULL,
transaction_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY(account_id) REFERENCES Account(account_id)
);

-- Account 테이블
CREATE TABLE Account (
account_id INT AUTO_INCREMENT PRIMARY KEY,
user_id INT NOT NULL,
account_number VARCHAR(20) NOT NULL UNIQUE,
balance DECIMAL(15, 2) DEFAULT 0.00,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
password VARCHAR(255) NOT NULL,
FOREIGN KEY(user_id) REFERENCES Identity(user_id)
);

-- User 테이블
CREATE TABLE User (
user_id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(100) NOT NULL,
email VARCHAR(100) NOT NULL UNIQUE,
phone_number VARCHAR(15),
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

- **프로젝트 소스**

1. MainFrame.java

프로그램의 메인 화면으로, 각 기능에 대한 버튼을 통해 다른 화면으로 이동.

주요 버튼:

- 입금 (btnDeposit)
- 출금 (btnWithdrawal1)
- 송금 (btnRemittance)
- 조회 및 개설 (btnCheck)

소스코드

```
public class MainFrame extends javax.swing.JFrame {
    public MainFrame() {

        initComponents();
    }

    private void btnDepositActionPerformed(java.awt.event.ActionEvent evt) {
        deposit depositFrame = new deposit();
        depositFrame.setVisible(true);
        this.dispose();
    }
    private void btnRemittanceActionPerformed(java.awt.event.ActionEvent evt) {
        remittance remittanceFrame = new remittance();
        remittanceFrame.setVisible(true);
        this.dispose();
    }
    private void btnCheckActionPerformed(java.awt.event.ActionEvent evt) {
        CheckFrame checkFrame = new CheckFrame();
        checkFrame.setVisible(true);
        this.dispose();
    }
    private void btnWithdrawal1ActionPerformed(java.awt.event.ActionEvent evt) {
        withdrawal withdrawalFrame = new withdrawal();
    withdrawalFrame.setVisible(true);
    this.dispose();
    }
    public static void main(String args[]) {

        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
                java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
```

```
                java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {

                new MainFrame().setVisible(true);

            }
        });
    }
}
```

2. CheckFrame.java
조회와 계좌 개설 기능을 선택할 수 있는 화면.

각 버튼 클릭 시 해당 프레임으로 전환:
- btnBalanceInquiry: 잔액 조회 화면으로 이동.
- btnRegister: 계좌 개설 화면으로 이동.

소스코드

```
import java.awt.Dimension;

public class CheckFrame extends javax.swing.JFrame {
    public CheckFrame() {
        initComponents();
    }

    private void btnBalanceInquiryActionPerformed(java.awt.event.ActionEvent evt) {
        BalanceInquiry BalanceInquiryFrame = new BalanceInquiry();
        BalanceInquiryFrame.setSize(400,450);
        BalanceInquiryFrame.setMinimumSize(new Dimension(400, 450));
        BalanceInquiryFrame.setMaximumSize(new Dimension(400, 450));
        BalanceInquiryFrame.setPreferredSize(new Dimension(400, 450));
        BalanceInquiryFrame.setVisible(true);
        this.dispose();
    }
    private void btnRegisterActionPerformed(java.awt.event.ActionEvent evt) {
        register RegisterFrame = new register();
        RegisterFrame.setVisible(true);
        this.dispose();
    }
    private void btnStatementActionPerformed(java.awt.event.ActionEvent evt) {
        Statement StatementFrame = new Statement();
        StatementFrame.setVisible(true);
        this.dispose();
    }
    private void btnChangePasswordActionPerformed(java.awt.event.ActionEvent evt) {
        ChangePassword ChangePasswordFrame = new ChangePassword();
        ChangePasswordFrame.setVisible(true);
        this.dispose();
    }
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        MainFrame mainframe = new MainFrame();
        mainframe.setVisible(true);
        this.dispose();
    }

    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
```

```
                java.util.logging.Logger.getLogger(CheckFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(CheckFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
                java.util.logging.Logger.getLogger(CheckFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(CheckFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new CheckFrame().setVisible(true);
            }
        });
    }
}
```

3. register.java
계좌 개설 화면.

주요 기능:
- 전화번호 중복 확인.
- 4자리 숫자로 구성된 비밀번호 설정.
- 비밀번호 확인값 일치 여부 검증.
- DB에 사용자 데이터를 추가하고 중복 여부 확인.

소스코드

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.regex.Pattern;
import javax.swing.Timer;
public class register extends javax.swing.JFrame {
    DB_MAN DBM = new DB_MAN();

    public register() {
        initComponents();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String name = jTextField1.getText();
        String email = jTextField2.getText();
        String phone = jTextField3.getText();
        String password = txtPass.getText(); // 비밀번호 값 가져오기
        String passwordCheck = txtPassCheck.getText(); // 비밀번호 확인 값 가져오기
        // 유효성 검사
        if (name.isEmpty() || email.isEmpty() || phone.isEmpty()) {
            jLabel6.setText("모든 정보를 입력해주세요.");
            return;
        }
        if (!Pattern.matches("^[\\w.%+-]+@[\\w.-]+\\.[a-zA-Z]{2,6}$", email)) {
            jLabel6.setText("이메일 형식이 올바르지 않습니다.");
            return;
        }
        if (!phone.matches("\\d{10,11}")) {
            jLabel6.setText("전화번호는 10~11자리 숫자만 입력 가능합니다.");
            return;
        }

        if (!password.matches("\\d{4}")) { // 비밀번호가 숫자 4자리인지 확인
            jLabel6.setText("비밀번호는 숫자 4자리여야 합니다.");
            return;
        }
        if (!password.equals(passwordCheck)) { // 비밀번호와 확인 값이 같은지 확인
            jLabel6.setText("비밀번호가 일치하지 않습니다.");
            return;
        }

        // DB 연결 및 데이터 삽입
        try {
            DBM.dbOpen();
```

```
            // 전화번호 중복 검사
            String checkPhoneQuery = "SELECT COUNT(*) AS count FROM User WHERE phone_number = ?";
            PreparedStatement pstmt = DBM.DB_con.prepareStatement(checkPhoneQuery);
            pstmt.setString(1, phone);
            ResultSet rs = pstmt.executeQuery();
            if (rs.next() && rs.getInt("count") > 0) {
                jLabel6.setText("이미 등록된 전화번호입니다.");
                DBM.dbClose();
                return;
            }
            // 사용자 삽입
            String insertUserQuery = String.format(
                "INSERT INTO User (name, email, phone_number, created_at) VALUES ('%s', '%s', '%s', NOW())",
                name, email, phone
            );
            DBM.DB_stmt.executeUpdate(insertUserQuery);
            // 사용자 ID 가져오기
            DBM.DB_rs = DBM.DB_stmt.executeQuery("SELECT LAST_INSERT_ID() AS user_id");
            DBM.DB_rs.next();
            int userId = DBM.DB_rs.getInt("user_id");
            // 계좌 생성
            String accountNumber = String.valueOf((int) (Math.random() * 900000000) + 100000000);
            String insertAccountQuery = String.format(
                  "INSERT INTO Account (user_id, account_number, password,  created_at) VALUES (%d, '%s', '%s',
NOW())",
                userId, accountNumber, password
            );
            DBM.DB_stmt.executeUpdate(insertAccountQuery);
            jLabel6.setText("계좌가 성공적으로 생성되었습니다. 계좌번호: " + accountNumber);
            DBM.dbClose();


            // 7초 후 ReturnFrame으로 이동
            Timer timer = new Timer(7000, e -> {
                ReturnFrame returnFrame = new ReturnFrame();
                returnFrame.setVisible(true);
                this.dispose(); // 현재 창 닫기
            });
            timer.setRepeats(false); // 타이머를 한 번만 실행
            timer.start();
        } catch (Exception e) {
            jLabel6.setText("오류 발생: " + e.getMessage());
        }
    }
    private void btnCancleActionPerformed(java.awt.event.ActionEvent evt) {
        ReturnFrame ReturnFrame = new ReturnFrame();
        ReturnFrame.setVisible(true);
        this.dispose();
    }
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
```

```
                java.util.logging.Logger.getLogger(register.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(register.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
                java.util.logging.Logger.getLogger(register.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(register.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new register().setVisible(true);
            }
        });
    }
}
```

4. BalanceInquiry.java

계좌 조회 화면.

이름과 전화번호 입력 후 DB에서 해당 정보의 계좌와 잔액 정보를 가져와 표시.

조회 결과는 resultTextArea에 출력.

소스코드

```
import javax.swing.*;
import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
public class BalanceInquiry extends JFrame {
    private final DB_MAN dbManager = new DB_MAN();  // DBMAN 객체 생성
    public BalanceInquiry() {
        initComponents();
        setSize(450, 350);
    }

    private void checkBalanceButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String name = nameField.getText().trim();  // 이름 입력받기
        String phoneNumber = phoneField.getText().trim();  // 전화번호 입력받기
        // 이름과 전화번호 입력 확인
        if (name.isEmpty() || phoneNumber.isEmpty()) {
            resultTextArea.setText("이름과 전화번호를 모두 입력해주세요.");
            return;
        }
        try {
            dbManager.dbOpen(); // DB 연결
            // 이름과 전화번호로 user_id 조회
            String getAccountQuery = "SELECT user_id FROM User WHERE name = ? AND phone_number = ?";
            PreparedStatement stmt = dbManager.DB_con.prepareStatement(getAccountQuery);
            stmt.setString(1, name);  // 이름 바인딩
            stmt.setString(2, phoneNumber);  // 전화번호 바인딩
            ResultSet rs = stmt.executeQuery();
            // 결과 확인
            if (rs.next()) {
                int userId = rs.getInt("user_id");  // user_id 조회
                // 해당 user_id로 계좌 정보와 잔고 조회
                String balanceQuery = "SELECT account_number, balance FROM Account WHERE user_id = ?";
                PreparedStatement balanceStmt = dbManager.DB_con.prepareStatement(balanceQuery);
                balanceStmt.setInt(1, userId);  // user_id로 바인딩
                ResultSet balanceRs = balanceStmt.executeQuery();
                if (balanceRs.next()) {
                    String accountNumber = balanceRs.getString("account_number");  // 계좌 번호 조회
                    double balance = balanceRs.getDouble("balance");  // 잔고 조회
                    // 소수점 없는 정수로 변환하여 출력
                       resultTextArea.setText("계좌 번호: " + accountNumber + "\n잔고: " + String.format("%,.0f",
balance) + "원");

                    // 7초 후 ReturnFrame으로 이동
                    Timer timer = new Timer(7000, e -> {
                        ReturnFrame returnFrame = new ReturnFrame();
                        returnFrame.setVisible(true);
                        this.dispose(); // 현재 창 닫기
                    });
```

```
                    timer.setRepeats(false); // 타이머를 한 번만 실행
                    timer.start();
                } else {
                    resultTextArea.setText("잔고를 조회할 수 없습니다.");
                }
            } else {
                resultTextArea.setText("입력한 이름과 전화번호에 일치하는 계좌가 없습니다.");
            }
        } catch (SQLException e) {
            resultTextArea.setText("SQL 오류: " + e.getMessage());
        } catch (IOException e) {
            resultTextArea.setText("IO 오류: " + e.getMessage());
        } finally {
            try {
                dbManager.dbClose();  // DB 연결 종료
            } catch (IOException e) {
                resultTextArea.setText("DB 연결 종료 오류: " + e.getMessage());
            }
        }
    }

    private void btnCancleActionPerformed(java.awt.event.ActionEvent evt) {
        ReturnFrame ReturnFrame = new ReturnFrame();
        ReturnFrame.setVisible(true);
        this.dispose();
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                new BalanceInquiry().setVisible(true);
            }
        });
    }
}
```

5. deposit.java
입금 화면.
입력받은 계좌 번호와 금액을 기반으로 DB에서 잔액을 업데이트.
거래 후 Transaction 테이블에 기록 추가.

소스코드

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.Timer;
public class deposit extends javax.swing.JFrame {
    DB_MAN DBM = new DB_MAN();
    public deposit() {
        initComponents();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String name = nameField.getText();
        String amount = amountField.getText();
        String account = accountField.getText();
        if (name.isEmpty() || account.isEmpty() || amount.isEmpty()) {
            info.setText("모든 정보를 입력해주세요.");
        } else if (!account.matches("\\d+")) {
            info.setText("계좌번호는 숫자만 입력 가능합니다.");
        }else if (!amount.matches("\\d+")) {
            info.setText("입금 금액은 숫자만 입력 가능합니다.");
        }
        int nAmount = Integer.parseInt(amount);
        try {
            DBM.dbOpen();
            // 계좌 확인
            String query = String.format("SELECT balance FROM Account WHERE account_number = '%s'",
                account
            );
            DBM.DB_rs = DBM.DB_stmt.executeQuery(query);
            if (DBM.DB_rs.next()) {
                int currentBalance = DBM.DB_rs.getInt("balance");
                // 계좌 잔액 업데이트
                String updateQuery = String.format(
                    "UPDATE Account SET balance = balance + %d WHERE account_number = '%s'", nAmount, account
                );
                DBM.DB_stmt.executeUpdate(updateQuery);
                info.setText("입금이 완료되었습니다. 현재 잔액: " + (currentBalance + nAmount) + "원");

                // 입금 완료 후 Transaction 기록 추가
                  String insertTransactionQuery = "INSERT INTO Transaction (account_id, amount, transaction_type,
transaction_date) " + "VALUES ((SELECT account_id FROM Account WHERE account_number = ?), ?, '입금', NOW())";
                PreparedStatement transactionStmt = DBM.DB_con.prepareStatement(insertTransactionQuery);
                transactionStmt.setString(1, account);
                transactionStmt.setInt(2, nAmount);
                transactionStmt.executeUpdate();

                // 7초 후 ReturnFrame으로 이동
                Timer timer = new Timer(7000, e -> {
```

```
                ReturnFrame returnFrame = new ReturnFrame();
                returnFrame.setVisible(true);
                this.dispose(); // 현재 창 닫기
            });
            timer.setRepeats(false); // 타이머를 한 번만 실행
            timer.start();
        } else {
            info.setText("계좌 정보를 찾을 수 없습니다.");
        }
        DBM.dbClose();
    } catch (Exception e) {
        info.setText("오류 발생: " + e.getMessage());
    }


}
private void btnCancleActionPerformed(java.awt.event.ActionEvent evt) {
    ReturnFrame ReturnFrame = new ReturnFrame();
    ReturnFrame.setVisible(true);
    this.dispose();
}
public static void main(String args[]) {
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(deposit.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(deposit.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(deposit.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(deposit.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);
    }

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new deposit().setVisible(true);
        }
    });
}
}
```

6. withdrawal.java

출금 화면.

입력받은 계좌 번호와 금액, 비밀번호를 기반으로 출금 처리.

거래 후 Transaction 테이블에 기록 추가.

출금 후 잔액 부족 여부 확인.

소스코드

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.Timer;

public class withdrawal extends javax.swing.JFrame {
    DB_MAN DBM = new DB_MAN();
    public withdrawal() {
        initComponents();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String name = nameField.getText();  // 사용자 이름
        String account = accountField.getText();  // 계좌 번호
        String amount = amountField.getText();  // 출금 금액
        String accountPassword = lblPass.getText();  // 계좌 비밀번호
        if (name.isEmpty() || account.isEmpty() || amount.isEmpty() || accountPassword.isEmpty()) {
            info.setText("모든 정보를 입력해주세요.");
            return;
        }
        if (!amount.matches("\\d+")) {
            info.setText("출금 금액은 숫자로 입력해주세요.");
            return;
        }
        try {
            DBM.dbOpen();
             String query = "SELECT balance FROM Account WHERE account_number = ? AND password = ? AND user_id =
(SELECT user_id FROM User WHERE name = ?)";
            PreparedStatement pstmt = DBM.DB_con.prepareStatement(query);
            pstmt.setString(1, account);
            pstmt.setString(2, accountPassword);
            pstmt.setString(3, name);
            ResultSet rs = pstmt.executeQuery();
            if (!rs.next()) {
                info.setText("계좌 번호, 비밀번호 또는 이름이 일치하지 않습니다.");
                DBM.dbClose();
                return;
            }
            int currentBalance = rs.getInt("balance");
            int withdrawAmount = Integer.parseInt(amount);
            if (currentBalance < withdrawAmount) {
                info.setText("잔액이 부족합니다.");
                DBM.dbClose();
                return;
            }
            // 출금 처리
            String updateQuery = "UPDATE Account SET balance = balance - ? WHERE account_number = ?";
            pstmt = DBM.DB_con.prepareStatement(updateQuery);
```

```
            pstmt.setInt(1, withdrawAmount);
            pstmt.setString(2, account);
            pstmt.executeUpdate();
            info.setText("출금이 완료되었습니다. 현재 잔액: " + (currentBalance - withdrawAmount) + "원");

               String insertTransactionQuery = "INSERT INTO Transaction (account_id, amount, transaction_type,
transaction_date) " + "VALUES ((SELECT account_id FROM Account WHERE account_number = ?), ?, '출금', NOW())";
            PreparedStatement transactionStmt = DBM.DB_con.prepareStatement(insertTransactionQuery);
            transactionStmt.setString(1, account);
            transactionStmt.setInt(2, withdrawAmount);
            transactionStmt.executeUpdate();
            DBM.dbClose();

            Timer timer = new Timer(7000, e -> {
                ReturnFrame returnFrame = new ReturnFrame();
                returnFrame.setVisible(true);
                this.dispose(); // 현재 창 닫기
            });
            timer.setRepeats(false); // 타이머를 한 번만 실행
            timer.start();
        } catch (Exception e) {
            info.setText("오류 발생: " + e.getMessage());
        }
    }
    private void btnCancle1ActionPerformed(java.awt.event.ActionEvent evt) {
        ReturnFrame ReturnFrame = new ReturnFrame();
        ReturnFrame.setVisible(true);
        this.dispose();
    }
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
                java.util.logging.Logger.getLogger(withdrawal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(withdrawal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
                java.util.logging.Logger.getLogger(withdrawal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(withdrawal.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new withdrawal().setVisible(true);
            }
        });
    }
}
```

7. remittance.java

송금 화면.

보내는 계좌, 받는 계좌, 금액, 비밀번호를 입력받아 송금 처리.

송금 후 두 계좌의 잔액 업데이트 및 거래 내역 기록.

소스코드

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import javax.swing.Timer;
public class remittance extends javax.swing.JFrame {
    DB_MAN DBM = new DB_MAN();
    boolean isrecipientName = false;
    public remittance() {
        initComponents();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String senderAccount = jTextField2.getText();  // 보내는 계좌
        String recipientAccount = jTextField4.getText(); // 받는 계좌
        String amountStr = jTextField5.getText(); // 송금 금액
        String accountPassword = lblPass.getText(); // 계좌 비밀번호
        // 유효성 검사
        if (senderAccount.isEmpty() || recipientAccount.isEmpty() || amountStr.isEmpty()) {
            info.setText("모든 필드를 채워주세요.");
            return;
        }
        if (!senderAccount.matches("\\d+") || !recipientAccount.matches("\\d+") || !amountStr.matches("\\d+")) {
            info.setText("계좌와 금액은 숫자로만 입력해주세요.");
            return;
        }
        try {
            DBM.dbOpen();
            String senderQuery = "SELECT balance FROM Account WHERE account_number = ? AND password = ?";
            PreparedStatement pstmt = DBM.DB_con.prepareStatement(senderQuery);
            pstmt.setString(1, senderAccount);
            pstmt.setString(2, accountPassword);
            ResultSet rs = pstmt.executeQuery();
            if (!rs.next()) {
                info.setText("계좌 번호 또는 비밀번호가 일치하지 않습니다.");
                DBM.dbClose();
                return;
            }
            int balance = rs.getInt("balance");
            if (balance < Integer.parseInt(amountStr)) {
                info.setText("잔액이 부족합니다.");
                DBM.dbClose();
                return;
            }
             String recipientQuery = String.format("SELECT * FROM Account WHERE account_number = '%s'",
recipientAccount);
            DBM.DB_rs = DBM.DB_stmt.executeQuery(recipientQuery);
            if (!DBM.DB_rs.next()) {
                info.setText("받는 계좌를 찾을 수 없습니다.");
```

```
                DBM.dbClose();
                return;
            }
                DBM.DB_stmt.executeUpdate(String.format("UPDATE Account SET balance = balance - %s WHERE account_number = '%s'", amountStr, senderAccount));
                DBM.DB_stmt.executeUpdate(String.format("UPDATE Account SET balance = balance + %s WHERE account_number = '%s'", amountStr, recipientAccount));
            info.setText("송금이 완료되었습니다.");

              String senderTransactionQuery = "INSERT INTO Transaction (account_id, amount, transaction_type, transaction_date) " + "VALUES ((SELECT account_id FROM Account WHERE account_number = ?), ?, '출금', NOW())";
            PreparedStatement senderStmt = DBM.DB_con.prepareStatement(senderTransactionQuery);
            senderStmt.setString(1, senderAccount);
            senderStmt.setInt(2, -Integer.parseInt(amountStr));  // 송금은 음수로 기록
            senderStmt.executeUpdate();

             String recipientTransactionQuery = "INSERT INTO Transaction (account_id, amount, transaction_type, transaction_date) " + "VALUES ((SELECT account_id FROM Account WHERE account_number = ?), ?, '입금', NOW())";
            PreparedStatement recipientStmt = DBM.DB_con.prepareStatement(recipientTransactionQuery);
            recipientStmt.setString(1, recipientAccount);
            recipientStmt.setInt(2, Integer.parseInt(amountStr));  // 수신은 양수로 기록
            recipientStmt.executeUpdate();
            DBM.dbClose();

            Timer timer = new Timer(7000, e -> {
                ReturnFrame returnFrame = new ReturnFrame();
                returnFrame.setVisible(true);
                this.dispose(); // 현재 창 닫기
            });
            timer.setRepeats(false); // 타이머를 한 번만 실행
            timer.start();
        } catch (Exception e) {
            info.setText("오류 발생: " + e.getMessage());
        }
    }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        jTextField1.setText("");
        jTextField2.setText("");
        jTextField4.setText("");
        jTextField5.setText("");
        jLabel7.setText("");  // Clear recipient name label
        info.setText("");
    }
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        info.setText("송금 완료되었습니다.");
        jDialog1.setVisible(false);
    }
    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        info.setText("송금 취소되었습니다.");
        jDialog1.setVisible(false);
    }
    private void jTextField4FocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_jTextField4FocusLost
        String recipientAccount = jTextField4.getText();
        if (recipientAccount.isEmpty()) {
            jLabel7.setText(""); // 계좌 입력이 없을 경우 초기화
            return;
        }
        try {
```

```
            DBM.dbOpen();
            String query = String.format(
                "SELECT name FROM User u JOIN Account a ON u.user_id = a.user_id WHERE a.account_number = '%s'",
                recipientAccount
            );
            DBM.DB_rs = DBM.DB_stmt.executeQuery(query);
            if (DBM.DB_rs.next()) {
                String recipientName = DBM.DB_rs.getString("name");
                jLabel7.setText(recipientName);
            } else {
                info.setText("받는 계좌를 찾을 수 없습니다.");
            }
            DBM.dbClose();
        } catch (Exception e) {
            jLabel7.setText("오류 발생: " + e.getMessage());
        }
    }
    private void btnCancleActionPerformed(java.awt.event.ActionEvent evt) {
        ReturnFrame ReturnFrame = new ReturnFrame();
        ReturnFrame.setVisible(true);
        this.dispose();
    }
    private void btnCancle1ActionPerformed(java.awt.event.ActionEvent evt) {
        ReturnFrame ReturnFrame = new ReturnFrame();
        ReturnFrame.setVisible(true);
        this.dispose();
    }
    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
                java.util.logging.Logger.getLogger(remittance.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
                java.util.logging.Logger.getLogger(remittance.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
                java.util.logging.Logger.getLogger(remittance.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
                java.util.logging.Logger.getLogger(remittance.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new remittance().setVisible(true);
            }
        });
    }
}
```

8. Statement.java
통장 정리 화면.
입력받은 계좌 번호로 Transaction 테이블에서 거래 내역 조회.
결과는 resultTextArea에 표시.

소스코드

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class Statement extends javax.swing.JFrame {
    DB_MAN DBM = new DB_MAN();
    public Statement() {
        initComponents();
    }

    private void checkButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String account = accountField.getText();  // 계좌 번호 입력 받음
        if (account.isEmpty()) {
            resultTextArea.setText("계좌 번호를 입력해주세요.");
            return;
        }
        // 계좌 번호 형식 검증 (숫자만 입력 허용)
        if (!account.matches("\\d+")) {
            resultTextArea.setText("계좌 번호는 숫자만 입력 가능합니다.");
            return;
        }
        try {
            DBM.dbOpen();
            // 1. 계좌 번호로 account_id 조회
            String getAccountIdQuery = "SELECT account_id FROM Account WHERE account_number = ?";
            PreparedStatement stmt = DBM.DB_con.prepareStatement(getAccountIdQuery);
            stmt.setString(1, account);  // 계좌 번호를 바인딩
            DBM.DB_rs = stmt.executeQuery();
            // 계좌 번호에 해당하는 account_id가 있는지 확인
            if (DBM.DB_rs.next()) {
                int accountId = DBM.DB_rs.getInt("account_id");  // account_id를 가져옴
                // 2. account_id로 거래 내역 조회
                 String transactionQuery = "SELECT transaction_id, transaction_type, amount, transaction_date FROM
Transaction WHERE account_id = ? ORDER BY transaction_date DESC";
                PreparedStatement transactionStmt = DBM.DB_con.prepareStatement(transactionQuery);
                transactionStmt.setInt(1, accountId);  // account_id로 바인딩
                DBM.DB_rs = transactionStmt.executeQuery();
                // 거래 내역이 있을 경우
                if (DBM.DB_rs.next()) {
                    StringBuilder transactions = new StringBuilder();
                    do {
                        String transactionId = DBM.DB_rs.getString("transaction_id");
                        String type = DBM.DB_rs.getString("transaction_type");
                        String amount = DBM.DB_rs.getString("amount");
                        String date = DBM.DB_rs.getString("transaction_date");
                        transactions.append("거래 ID: ").append(transactionId)
                                    .append("\n유형: ").append(type)
                                    .append("\n금액: ").append(amount)
```

```
                                .append("\n날짜: ").append(date)
                                .append("\n------------------\n");  // 거래 내역 간 간격 추가
                    } while (DBM.DB_rs.next());
                    resultTextArea.setText(transactions.toString());  // 거래 내역 출력
                } else {
                    resultTextArea.setText("거래 내역이 없습니다.");
                }
            } else {
                resultTextArea.setText("입력한 계좌 번호에 해당하는 계좌가 없습니다.");
            }
            DBM.dbClose();

            // 7초 후 ReturnFrame으로 이동
            Timer timer = new Timer(7000, e -> {
                ReturnFrame returnFrame = new ReturnFrame();
                returnFrame.setVisible(true);
                this.dispose(); // 현재 창 닫기
            });
            timer.setRepeats(false); // 타이머를 한 번만 실행
            timer.start();
        } catch (SQLException e) {
            resultTextArea.setText("데이터베이스 오류: " + e.getMessage());
        } catch (Exception e) {
            resultTextArea.setText("오류 발생: " + e.getMessage());
        }
    }
    private void btnCancleActionPerformed(java.awt.event.ActionEvent evt) {
        ReturnFrame ReturnFrame = new ReturnFrame();
        ReturnFrame.setVisible(true);
        this.dispose();
    }
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Statement().setVisible(true);  // Statement 프레임을 표시
            }
        });
    }

    boolean isClosed() {
        throw new UnsupportedOperationException("Not supported yet.");
    }
    void close() {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

9. ChangePassword.java
비밀번호 변경 화면.
계좌 번호와 기존 비밀번호로 사용자 인증.
새로운 비밀번호를 설정하고 확인 값이 일치해야 변경 가능.

소스코드

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Arrays;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ChangePassword extends javax.swing.JFrame {
    DB_MAN DBM = new DB_MAN();
    public ChangePassword() {
    initComponents();  // GUI 컴포넌트 초기화

    // 본인 인증 버튼에 ActionListener 추가
    jButton1.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            accountFieldActionPerformed(e);  // 본인 인증 메서드 호출
        }
    });
}

    private void accountFieldActionPerformed(java.awt.event.ActionEvent evt) {
        String account = accountField.getText();  // 계좌 번호
        char[] currentPassword = currentPasswordField.getPassword();  // 현재 비밀번호
        try {
            DBM.dbOpen(); // DB 연결 열기
            // Account 테이블에서 account_number와 password를 확인하는 쿼리
            String query = "SELECT * FROM Account WHERE account_number = ? AND password = ?";
            PreparedStatement stmt = DBM.DB_con.prepareStatement(query);
            stmt.setString(1, account);  // 계좌 번호를 바인딩
            stmt.setString(2, new String(currentPassword));  // 비밀번호를 바인딩
            // 쿼리 실행
            DBM.DB_rs = stmt.executeQuery();
            // 결과 확인
            if (DBM.DB_rs.next()) {
                resultTextArea.setText("본인 인증 되었습니다.");
                changePasswordButton.setEnabled(true); // 비밀번호 변경 버튼 활성화
            } else {
                resultTextArea.setText("사용자 정보를 찾을 수 없습니다.");
                changePasswordButton.setEnabled(false); // 비밀번호 변경 버튼 비활성화
            }
            DBM.dbClose(); // DB 연결 닫기
        } catch (SQLException e) {
            resultTextArea.setText("DB 오류: " + e.getMessage());
        } catch (IOException ex) {
            Logger.getLogger(ChangePassword.class.getName()).log(Level.SEVERE, null, ex);
```

```
    }
}
private void changePasswordButtonActionPerformed(java.awt.event.ActionEvent evt) {
    char[] newPassword = newPasswordField.getPassword();
    char[] confirmPassword = confirmPasswordField.getPassword();
    if (newPassword.length != 4 || confirmPassword.length != 4) {
        resultTextArea.setText("비밀번호는 4자리여야 합니다.");

        return;
    }
    if (Arrays.equals(newPassword, confirmPassword)) {
        try {
            DBM.dbOpen();
            String updateQuery = "UPDATE Account SET password = ? WHERE account_number = ?";
            PreparedStatement stmt = DBM.DB_con.prepareStatement(updateQuery);
            stmt.setString(1, new String(newPassword));
            stmt.setString(2, accountField.getText());
            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                resultTextArea.setText("비밀번호 변경이 완료되었습니다.");
            } else {
                resultTextArea.setText("비밀번호 변경에 실패했습니다.");
            }
            DBM.dbClose();
            Timer timer = new Timer(7000, e -> {
                ReturnFrame returnFrame = new ReturnFrame();
                returnFrame.setVisible(true);
                this.dispose();
            });
            timer.setRepeats(false);
            timer.start();
        } catch (SQLException e) {
            resultTextArea.setText("DB 오류: " + e.getMessage());
        } catch (IOException ex) {
            Logger.getLogger(ChangePassword.class.getName()).log(Level.SEVERE, null, ex);
        }
    } else {
        resultTextArea.setText("새 비밀번호와 확인 비밀번호가 일치하지 않습니다.");
    }
}
private void btnCancleActionPerformed(java.awt.event.ActionEvent evt) {
    ReturnFrame ReturnFrame = new ReturnFrame();
    ReturnFrame.setVisible(true);
    this.dispose();
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String account = accountField.getText().trim();
    char[] currentPassword = currentPasswordField.getPassword();
    if (account.isEmpty()) {
        resultTextArea.setText("계좌 번호를 입력해주세요.");
        return;
    }
    try {
        DBM.dbOpen();
        String query = "SELECT * FROM Account WHERE account_number = ? AND password = ?";
        PreparedStatement stmt = DBM.DB_con.prepareStatement(query);
        stmt.setString(1, account);
        stmt.setString(2, new String(currentPassword));
```

```
            DBM.DB_rs = stmt.executeQuery();
            if (DBM.DB_rs.next()) {
                changePasswordButton.setEnabled(false);
                CaptchaDialog captchaDialog = new CaptchaDialog(this, resultTextArea);
                captchaDialog.setVisible(true);
                // 캡챠 인증 완료 후 버튼 상태 업데이트
                if (captchaDialog.isCaptchaValid()) {
                    changePasswordButton.setEnabled(true);
                    resultTextArea.setText("본인 인증 되었습니다.");
                } else {
                    resultTextArea.setText("캡챠 인증에 실패했습니다.");
                    changePasswordButton.setEnabled(false);
                }
            } else {
                resultTextArea.setText("사용자 정보를 찾을 수 없습니다.");
                changePasswordButton.setEnabled(false);
            }
            DBM.dbClose();
        } catch (SQLException e) {
            resultTextArea.setText("DB 오류: " + e.getMessage());
        } catch (IOException ex) {
            Logger.getLogger(ChangePassword.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public static void main(String args[]) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            ChangePassword cp = new ChangePassword();
            cp.setVisible(true);
        }
    });
}

}
```

10. CaptchaDialog.java

본인 인증을 위한 캡챠 입력 대화 상자 생성.

소스코드

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class CaptchaDialog extends JDialog {
    private JTextField captchaInputField;
    private JLabel captchaLabel;
    private JButton submitButton, refreshButton;
    private String generatedCaptchaText;
    private JTextArea resultTextArea;
    private boolean captchaValid = false;
    public CaptchaDialog(JFrame parent, JTextArea resultTextArea) {
        super(parent, "본인 인증", true);
        this.resultTextArea = resultTextArea;
        setLayout(new BorderLayout());
        generatedCaptchaText = CaptchaGenerator.generateCaptchaText();

        ImageIcon captchaImage = new ImageIcon(CaptchaGenerator.generateCaptchaImage(generatedCaptchaText));
        captchaLabel = new JLabel(captchaImage, SwingConstants.CENTER);
        captchaInputField = new JTextField(10);
        submitButton = new JButton("확인");
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String userCaptchaInput = captchaInputField.getText().trim();
                if (userCaptchaInput.equals(generatedCaptchaText)) {
                    resultTextArea.setText("본인 인증 되었습니다.");
                    captchaValid = true;
                    dispose();
                } else {
                    resultTextArea.setText("캡챠 인증에 실패했습니다.");
                    refreshCaptcha();
                }
            }
        });
        refreshButton = new JButton("새로고침");
        refreshButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                refreshCaptcha();
            }
        });
        JPanel buttonPanel = new JPanel(new FlowLayout());
        buttonPanel.add(submitButton);
        buttonPanel.add(refreshButton);
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 1));
        panel.add(captchaLabel);
        panel.add(captchaInputField);
        panel.add(buttonPanel);
```

```
        add(panel, BorderLayout.CENTER);
        setSize(captchaImage.getIconWidth() + 50, 200);
        setLocationRelativeTo(parent);
    }
    private void refreshCaptcha() {
        generatedCaptchaText = CaptchaGenerator.generateCaptchaText();
        ImageIcon captchaImage = new ImageIcon(CaptchaGenerator.generateCaptchaImage(generatedCaptchaText));
        captchaLabel.setIcon(captchaImage);
        captchaInputField.setText("");
    }
    public boolean isCaptchaValid() {
        return captchaValid;
    }
}
```

11. CaptchaGenerator.java
캡챠 텍스트 및 이미지 생성.

소스코드

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.Random;
import javax.imageio.ImageIO;
import javax.swing.*;

public class CaptchaGenerator {
    public static String generateCaptchaText() {
        String characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
        Random random = new Random();
        StringBuilder captchaText = new StringBuilder(5);

        for (int i = 0; i < 5; i++) {
            int randomIndex = random.nextInt(characters.length());
            captchaText.append(characters.charAt(randomIndex));
        }
        return captchaText.toString();
    }
    public static BufferedImage generateCaptchaImage(String captchaText) {
        int width = 250;
        int height = 100;

        BufferedImage captchaImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        Graphics g = captchaImage.getGraphics();

        g.setColor(Color.WHITE);
        g.fillRect(0, 0, width, height);

        g.setColor(Color.BLACK);
        g.setFont(new Font("Arial", Font.BOLD, 40));
        g.drawString(captchaText, 50, 60);
        g.setColor(Color.GRAY);
        for (int i = 0; i < 5; i++) {
            int x1 = (int) (Math.random() * width);
            int y1 = (int) (Math.random() * height);
            int x2 = (int) (Math.random() * width);
            int y2 = (int) (Math.random() * height);
            g.drawLine(x1, y1, x2, y2);
        }
        g.dispose();
        return captchaImage;
    }
}
```

12. ReturnFrame.java

거래 완료 후 5초 뒤 메인 화면(MainFrame)으로 자동 전환.

사용자에게 거래 완료 메시지와 화면 전환 안내를 제공.

소스코드

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ReturnFrame extends javax.swing.JFrame {
    public ReturnFrame() {
        initComponents();
        int delay = 5000; // 5초 (밀리초)
        Timer timer = new Timer(delay, new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // MainFrame으로 전환
                new MainFrame().setVisible(true);
                dispose(); // 현재 프레임 닫기
            }
        });
        timer.setRepeats(false); // 한 번만 실행
        timer.start();
    }

    public static void main(String args[]) {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
              java.util.logging.Logger.getLogger(ReturnFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
              java.util.logging.Logger.getLogger(ReturnFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
              java.util.logging.Logger.getLogger(ReturnFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
              java.util.logging.Logger.getLogger(ReturnFrame.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ReturnFrame().setVisible(true);
            }
        });
    }
}
```

13. DB_MAN.java
MySQL 데이터베이스 연결 및 쿼리 실행 관리.

주요 메서드:
- dbOpen(): DB 연결.
- dbClose(): DB 연결 해제.

소스코드

```
import java.sql.*;
import java.io.*;
import java.sql.Statement;

public class DB_MAN {
    String strDriver = "com.mysql.cj.jdbc.Driver";
    String strURL = "jdbc:mysql://3.27.185.193:3306/BankSys?characterEncoding=UTF-8&serverTimezone=UTC";
    String strUser = "minjae";
    String strPWD = "qwer1234";

    Connection DB_con;
    Statement DB_stmt;
    ResultSet DB_rs;

    public void dbOpen() throws IOException {
        try {
            System.setOut(new java.io.PrintStream(System.out, true, "UTF-8"));
        } catch (java.io.UnsupportedEncodingException ex) {
            ex.printStackTrace();
        }
        try {
            Class.forName(strDriver);
            DB_con = DriverManager.getConnection(strURL, strUser, strPWD);
            DB_stmt = DB_con.createStatement();
        } catch (Exception e) {
            System.out.println("SQLException : " + e.getMessage());
        }
    }

    public void dbClose() throws IOException {
        try {
            System.setOut(new java.io.PrintStream(System.out, true, "UTF-8"));
        } catch (java.io.UnsupportedEncodingException ex) {
            ex.printStackTrace();
        }
        try {
            DB_stmt.close();
            DB_con.close();
        } catch (SQLException e) {
            System.out.println("SQLException : " + e.getMessage());
        }
    }
}
```

• **사용자 매뉴얼**

1. 프로그램 개요

목적: 노인친화적 ATM 프로그램으로, 간단하고 안전하게 금융 거래를 처리할 수 있습니다.

사용 대상: 디지털 금융에 익숙하지 않은 노년층 사용자.

2. 주요 기능

- 계좌 개설:
  - 이름, 전화번호, 이메일 입력 후 계좌 생성.
  - 비밀번호는 4자리 숫자만 가능하며, 비밀번호 확인 필요.

- 조회:
  - 이름과 전화번호를 입력하면 계좌와 현재 잔액이 표시됩니다.

- 입금 및 출금:
  - 계좌 번호와 금액을 입력하면 입출금이 완료되며, 잔액이 실시간 갱신됩니다.

- 송금:
  - 송금 계좌 번호와 금액 입력 후 수취인 정보를 검증하여 송금 완료.

- 거래 내역 조회:
  - 날짜별 또는 특정 거래 내역을 확인.

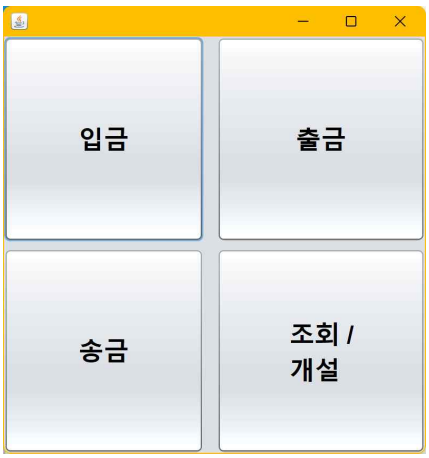
- 비밀번호 변경:
  - 현재 비밀번호 입력 후 새 비밀번호 설정 가능.

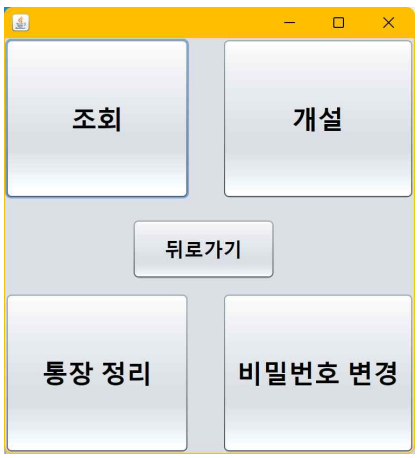
- 자동 화면 전환 및 로그아웃:
  - 거래 완료 후 일정 시간 후 자동으로 메인 화면으로 이동.

3. 사용 방법

초기화면:

- 프로그램 실행 시 메인 메뉴에서 원하는 기능을 선택.



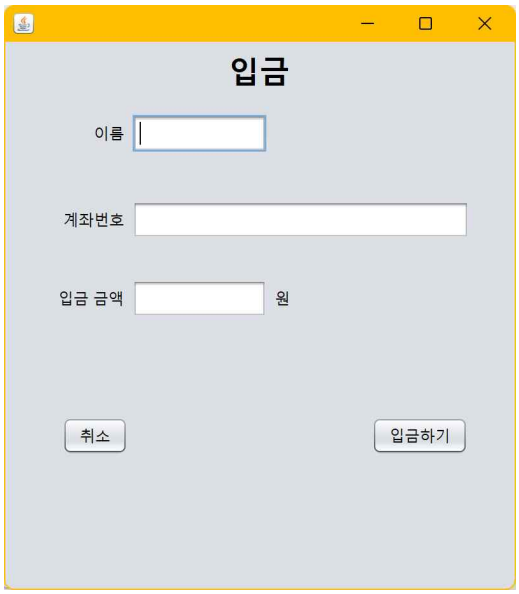


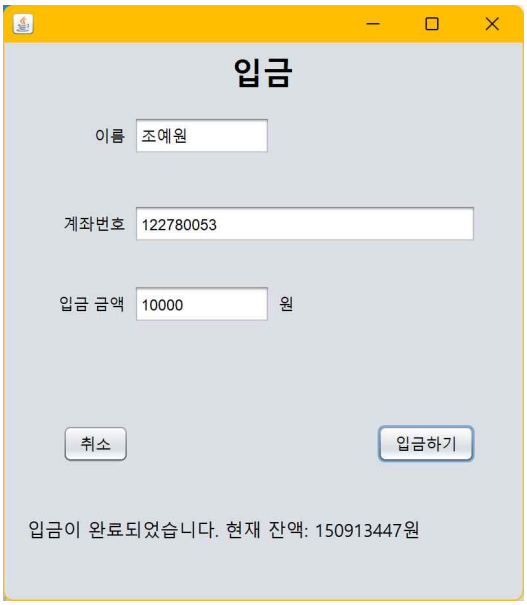
각 기능 사용:

- 계좌 개설:
  - 이름, 전화번호, 이메일 입력 → 비밀번호 설정 → "개설하기" 버튼 클릭.

- 조회:
  - 이름, 전화번호 입력 → "잔고 조회" 버튼 클릭.

- 입금 및 출금:
  - 계좌 번호 및 금액 입력 → "입금하기" 또는 "출금하기" 버튼 클릭.

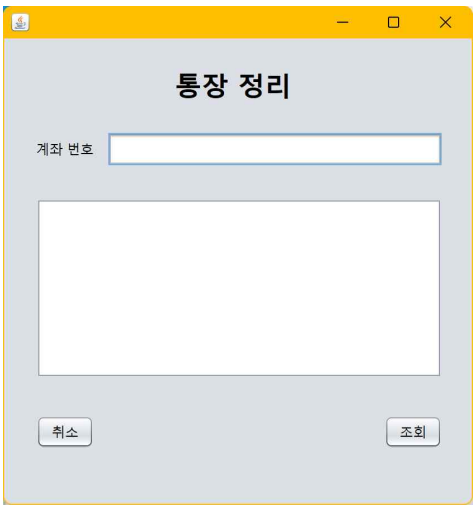


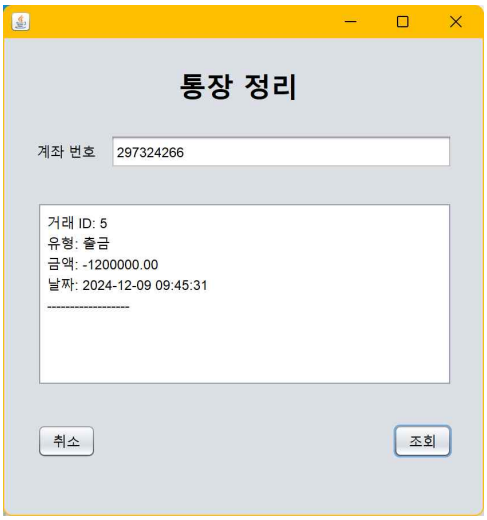


- 송금:
  - 송금 계좌 번호와 금액 입력 → "송금하기" 버튼 클릭.

- 비밀번호 변경:
  - 현재 비밀번호와 새 비밀번호 입력 → "변경하기" 버튼 클릭.

- 거래 내역 조회:
  - 계좌 입력 후 "조회하기" 버튼 클릭.





- 화면 전환
  - 기능 수행 후 일정 시간 뒤에 자동 로그아웃 및 화면 전환