Banking App

은행 거래

은행 업무 프로젝트 개요

◆ 은행 업무 프로젝트

은행 계좌 클래스를 만들고, 은행 업무 기능 만들기

- 은행 업무 프로젝트 단계

step1. 문제 정의하기

step2. 클래스 정의하고 관계도 그리기

step3. 은행 업무 기능 설계하고 구현하기

step4. 프로그램 테스트하기

step5. 유지보수 - 업그레이드 하기

step1. 문제 정의하기

프로그램 시나리오

- 계정(Account) 클래스에는 계좌 번호, 계좌주, 잔액 속성으로 구성되어 있음.
- Account 배열을 100개 생성한다.
- Main 클래스에서 계좌 생성, 계좌 목록, 입금, 출금, 종료 등의 메뉴가 있다.

계좌 번호	계좌주	금액
1111	홍길동	1000
2222	성춘향	2000
3333	이몽룡	3000
4444	황진이	4000

step1. 문제 정의하기

메뉴별 결과 리포트

```
---------------------------------------
1.계좌생성 |2.계좌목록 |3.예금 |4.출금 |5.종료
---------------------------------------
선택> 1
-------
계좌 생성
-------
계좌번호 : 1111-222
계좌주 : 홍길동
초기입금액 : 10000
결과 : 계좌가 생성되었습니다.
```

```
---------------------------------------
1.계좌생성 |2.계좌목록 |3.예금 |4.출금 |5.종료
---------------------------------------
선택> 2
-------
계좌 목록
-------
1111-222      홍길동      10000
```

```
---------------------------------------
1.계좌생성 |2.계좌목록 |3.예금 |4.출금 |5.종료
---------------------------------------
선택> 3
-------
예금
-------
계좌번호: 1111-222
예금액: 50000
결과 : 입금을 성공하였습니다.
```

```
---------------------------------------
1.계좌생성 |2.계좌목록 |3.예금 |4.출금 |5.종료
---------------------------------------
선택> 4
-------
출금
-------
계좌번호: 1111-222
출금액: 30000
결과 : 출금을 성공하였습니다.
```

step2. 클래스 다이어그램

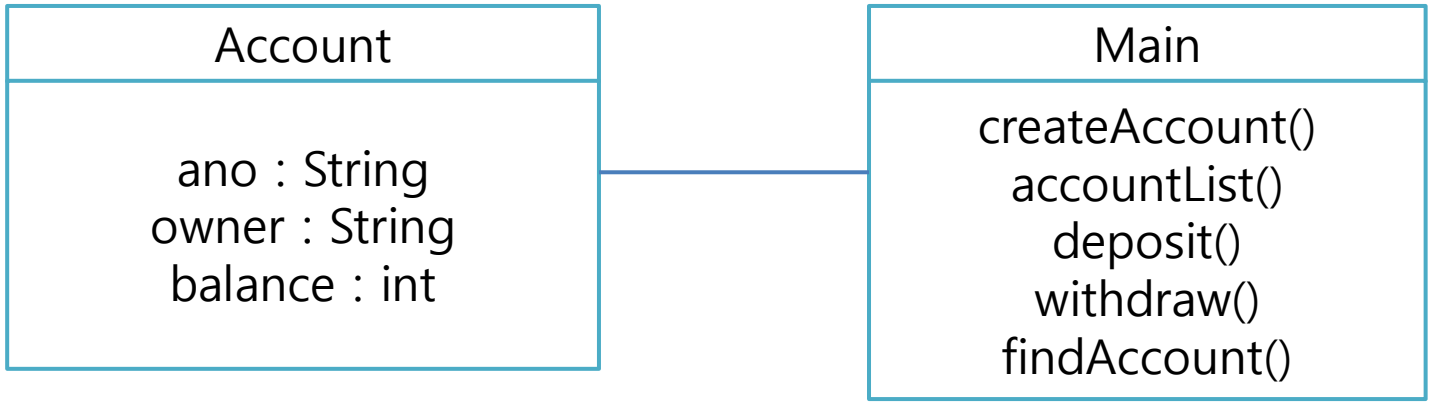
클래스 관계도 그리기

Account 클래스

계좌 번호
계좌주
잔액

Main 클래스

계좌 생성
계좌 목록
입금
출금



step2. 클래스 정의하기

Account 클래스(자료형-VO)

```
package bankapp;

public class Account {
    private String ano;
    private String owner;
    private int balance;

    public Account(String ano, String owner, int balance) {
        this.ano = ano;
        this.owner = owner;
        this.balance = balance;
    }
```

step2. 클래스 정의하기

```
public String getAno() {
    return ano;
}
public void setAno(String ano) {
    this.ano = ano;
}
public String getOwner() {
    return owner;
}
public void setOwner(String owner) {
    this.owner = owner;
}
public int getBalance() {
    return balance;
}
public void setBalance(int balance) {
    this.balance = balance;
}
```

step3. 은행 업무 기능 설계, 구현

- **Main 클래스**

```java
public class Main {
    //Account형 배열 공간 100개 준비
    private static Account[] accountArray = new Account[100];
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        boolean run = true;

        while(run) {
            System.out.println("-----------------------------------------");
            System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.종료");
            System.out.println("-----------------------------------------");
            System.out.print("선택> ");

            String selectNo = scanner.next();
            if(selectNo.equals("1")) {
                createAccount();  //계좌 생성
            }else if(selectNo.equals("2")){
                accountList();    //계좌 목록
            }else if(selectNo.equals("3")) {
                deposit();        //예금
            }else if(selectNo.equals("4")) {
                withdraw();       //출금
            }else if(selectNo.equals("5")){
                run = false;      //프로그램 종료
            }else{
                System.out.println("지원되지 않는 기능입니다.");
            }
        }
        System.out.println("프로그램 종료!!");
    }
```

step3. 은행 업무 기능 설계 , 구현

- 계좌 생성

```
private static void createAccount() {
    System.out.println("--------------------------");
    System.out.println("계좌 생성");
    System.out.println("--------------------------");

    System.out.println("계좌 번호: ");
    String ano = scanner.next();

    System.out.println("계좌주: ");
    String owner = scanner.next();

    System.out.println("초기 입금액: ");
    int balance = scanner.nextInt();

    //계좌 객체 생성
    Account newAccount = new Account(ano, owner, balance);
    for(int i=0; i<accountArray.length; i++) {
        if(accountArray[i] == null) {
            accountArray[i] = newAccount;
            System.out.println("결과: 계좌가 생성되었습니다.");
            break;
        }
    }
}
```

step3. 은행 업무 기능 설계 , 구현

- **계좌 목록**

```
private static void accountList() {
    System.out.println("-----------------------------------------");
    System.out.println("계좌 목록");
    System.out.println("-----------------------------------------");

    for(int i=0; i<accountArray.length; i++) {
        Account account = accountArray[i];
        if(account != null) {
            System.out.print("계좌번호: " + account.getAno() + "\t");
            System.out.print("계좌주: " + account.getOwner() + "\t");
            System.out.println("잔액: " + account.getBalance());
        }
    }
}
```

step3. 은행 업무 기능 설계, 구현

- 예금

```
//예금
private static void deposit() {
    System.out.println("-----------------------------");
    System.out.println("예금");
    System.out.println("-----------------------------");

    System.out.println("계좌 번호: ");
    String ano = scanner.next();

    Account account = findAccount(ano);  //검색된 계좌 반환

    System.out.println("입금액: ");
    int money = scanner.nextInt();
    account.setBalance(account.getBalance() + money);
    System.out.println("결과: 입금을 성공하였습니다.");
}
```

step3. 은행 업무 기능 설계, 구현

- **출금**

```
//출금
private static void withdraw() {
    System.out.println("---------------------------");
    System.out.println("출금");
    System.out.println("---------------------------");

    System.out.println("계좌 번호: ");
    String ano = scanner.next();

    Account account = findAccount(ano);

    System.out.println("출금액: ");
    int money = scanner.nextInt();
    account.setBalance(account.getBalance() - money);
    System.out.println("결과: 출금을 성공하였습니다.");
}
```

step3. 은행 업무 기능 설계, 구현

- **계좌 검색**

```
//계좌 찾기
private static Account findAccount(String ano) {
    Account account = null;  //찾는 계좌 객체 선언
    for(int i=0; i<accountArray.length; i++) {
        if(accountArray[i] != null) {
            String dbAno = accountArray[i].getAno(); //이미 저장된 계좌를 가져와서
            if(dbAno.equals(ano)) { //찾을 계좌(ano)와 일치한다면
                account = accountArray[i]; //배열에 저장된 계좌 대입
                break;
            }
        }
    }
    return account;
}
```

step4. 프로그램 테스트 하기

1.입, 출금시 계좌 비교

2. 출금시 잔액 부족

```
while(true) {
    System.out.println("계좌 번호: ");
    String ano = scanner.next();

    //입력한 계좌가 없을때 처리
    if(findAccount(ano) == null) {
        System.out.println("계좌가 없습니다. 다시 입력하세요");
    }else {
        Account account = findAccount(ano);
        while(true) {
            System.out.println("출금액: ");
            int money = scanner.nextInt();
            if(money > account.getBalance()) {
                System.out.println("잔액이 부족합니다. 다시 입력하세요");
            }else if(money < 0) {
                System.out.println("잘못된 입력입니다. 다시 입력하세요");
            }else {
                account.setBalance(account.getBalance() - money);
                System.out.printf("%,d원 정상 출금되었습니다.\n", money);
                break;
            }
        }//안쪽 while
        break;
    }
}//바깥쪽 while
```

step4. 프로그램 테스트 하기

1.계좌 생성시
- 중복 계좌 오류 처리
- 초기입금액 100원이상 설정

```
while(true) {
    System.out.println("계좌 번호: ");
    String ano = scanner.next();
    if(findAccount(ano) != null) {
        System.out.println("중복 계좌입니다. 다시 입력하세요");
    }else {
        System.out.println("계좌주: ");
        String owner = scanner.next();

        while(true) {
            System.out.println("초기 입금액 : ");
            int balance = scanner.nextInt();
            if(balance < 100) {
                System.out.println("기본 입금액은 100원 이상입니다. 다시 입력하세요");
            }else {
                Account newAccount = new Account(ano, owner, balance); //계좌 생성
                for(int i=0; i<accountArray.length; i++) {
                    if(accountArray[i] == null) {
                        accountArray[i] = newAccount;
                        System.out.println("결과 : 계좌가 생성되었습니다.");
                        break;
                    }
                }
                break;
            } //안쪽 while
        }
        break;
    }
```

step4. 프로그램 테스트 하기

1.계좌 생성시
- 중복 계좌 오류 처리
- 초기입금액 100원이상 설정

```
while(true) {
    System.out.println("계좌 번호: ");
    String ano = scanner.next();
    if(findAccount(ano) != null) {
        System.out.println("중복 계좌입니다. 다시 입력하세요");
    }else {
        System.out.println("계좌주: ");
        String owner = scanner.next();

        while(true) {
            System.out.println("초기 입금액 : ");
            int balance = scanner.nextInt();
            if(balance < 100) {
                System.out.println("기본 입금액은 100원 이상입니다. 다시 입력하세요");
            }else {
                Account newAccount = new Account(ano, owner, balance); //계좌 생성
                for(int i=0; i<accountArray.length; i++) {
                    if(accountArray[i] == null) {
                        accountArray[i] = newAccount;
                        System.out.println("결과 : 계좌가 생성되었습니다.");
                        break;
                    }
                }
                break;
            } //안쪽 while
        }
        break;
    }
```

ver2. ArrayList로 구현하기

```
while(run) {
    System.out.println("-----------------------------------------------------------------");
    System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.계좌검색 | 6.계좌삭제 | 7.종료");
    System.out.println("-----------------------------------------------------------------");
    System.out.print("선택> ");

    String selectNo = scanner.next();
    if(selectNo.equals("1")) {
        createAccount();      //계좌 생성
    }else if(selectNo.equals("2")) {
        accountList();        //계좌 목록
    }else if(selectNo.equals("3")) {
        deposit();            //예금
    }else if(selectNo.equals("4")) {
        withdraw();           //출금
    }else if(selectNo.equals("5")) {
        viewAccount();          //계좌 검색
    }else if(selectNo.equals("6")) {
        removeAccount();        //계좌 삭제
    }else if(selectNo.equals("7")) {
        run = false;          //종료
    }else {
        System.out.println("지원되지 않는 기능입니다.");
    }
}//while 닫기
System.out.println("프로그램 종료!");
```

ver3. jdbc 연동

- **account 테이블 생성**

```
-- account 테이블 생성
CREATE TABLE account(
    ano     VARCHAR2(20) PRIMARY KEY,
    owner   VARCHAR2(20) NOT NULL,
    balance NUMBER
);

INSERT INTO account VALUES ('111-222-3333', '김기용', 10000);
```

ver3. jdbc 연동

```
package banking_db;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        AccountDao dao = new AccountDao();
        Scanner scanner = new Scanner(System.in);

        boolean run = true;

        while(run) {
            System.out.println("-----------------------------------------------------------------------");
            System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.계좌검색 | 6.계좌삭제 | 7.종료");
            System.out.println("-----------------------------------------------------------------------");
            System.out.print("선택> ");

            String selectNo = scanner.next();
            if(selectNo.equals("1")) {
                dao.createAccount();
            }
```

ver3. jdbc 연동

```
            else if(selectNo.equals("2")){
                dao.getAccountList();
            }
            else if(selectNo.equals("3")) {
                dao.deposit();
            }
            else if(selectNo.equals("4")) {
                dao.withdraw();
            }
            else if(selectNo.equals("5")){
                dao.viewAccount();
            }
            else if(selectNo.equals("6")){
                dao.deleteAccount();
            }
            else if(selectNo.equals("7")){
                run = false;
            }
            else{
                System.out.println("지원되지 않는 기능입니다.");
            }
        }
        System.out.println("프로그램 종료!!");
        scanner.close();
    }
}
```

JDBCUtil – DB 연결

```java
package banking_db.common;

import java.sql.Connection;

public class JDBCUtil {
    private static String driverClass = "oracle.jdbc.OracleDriver";
    private static String url = "jdbc:oracle:thin:@localhost:1521:xe";
    private static String username = "system";
    private static String password = "12345";

    //DB 연결 메서드
    public static Connection getConnention() {
        try {
            Class.forName(driverClass);
            return DriverManager.getConnection(url, username, password);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

JDBCUtil – 연결 종료

```
//DB 연결 종료 메서드
public static void close(Connection conn, PreparedStatement pstmt) {
    if(pstmt != null) {
        try {
            pstmt.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            pstmt = null;
        }
    }

    if(conn != null) {
        try {
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            conn = null;
        }
    }
}
```

JDBCUtil – 연결 종료

```java
//연결 종료(ResultSet이 있는 경우)
public static void close(Connection conn, PreparedStatement pstmt, ResultSet rs) {
    if(rs != null) {
        try {
            rs.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    if(pstmt != null) {
        try {
            pstmt.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            pstmt = null;
        }
    }

    if(conn != null) {
        try {
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }finally {
            conn = null;
        }
    }
}
```

AccountDao

```java
package banking_db;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import banking_db.common.JDBCUtil;

public class AccountDao {
    //JDBC 관련 변수
        private Connection conn = null;
        private PreparedStatement pstmt = null;
        private ResultSet rs = null;
        private Scanner scanner = new Scanner(System.in);

        //계좌 생성
        public void createAccount() {
            System.out.println("---------------------------");
            System.out.println("계좌 생성");
            System.out.println("---------------------------");
```

AccountDao – 계좌 생성

```
while(true) {
    System.out.println("계좌 번호: ");
    String ano = scanner.next();
    if(findAccount(ano) != null) {
        System.out.println("중복된 계좌입니다. 다시 입력하세요");
    }else {
        System.out.println("계좌주: ");
        String owner = scanner.next();
        while(true) {
            System.out.println("초기 입금액: ");
            int balance = scanner.nextInt();
            if(balance < 100) {
                System.out.println("초기 입금액은 100원 이상입니다.");
            }else {
                //계좌 객체 생성
                conn = JDBCUtil.getConnention();
                String sql = "INSERT INTO account(ano, owner, balance) VALUES (?, ?, ?)";
                try {
                    pstmt = conn.prepareStatement(sql);
                    pstmt.setString(1, ano);
                    pstmt.setString(2, owner);
                    pstmt.setInt(3, balance);
                    pstmt.executeUpdate();
                    System.out.println("결과 : 계좌가 생성되었습니다.");
                } catch (SQLException e) {
                    e.printStackTrace();
                } finally {
                    JDBCUtil.close(conn, pstmt);
                }
                break;
            }
        }//while 닫기
        break;
    }
}//while 닫기
```

AccountDao – 목록 보기

```java
//계좌 목록 보기
public List<Account> getAccountList(){
    System.out.println("-----------------------------------------------------");
    System.out.println("2. 계좌 목록");
    System.out.println("-----------------------------------------------------");

    List<Account> accountList = new ArrayList<>();
    conn = JDBCUtil.getConnention();
    String sql = "SELECT * FROM account";
    try {
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        while(rs.next()) {              //자료가 있다면 계속 반복
            String ano = rs.getString("ano");
            String owner = rs.getString("owner");
            int balance = rs.getInt("balance");

            Account account = new Account(ano, owner, balance);
            accountList.add(account);
        }

        for(int i = 0; i < accountList.size(); i++) {
            Account account = accountList.get(i);
            System.out.print("계좌번호 : " + account.getAno() + "\t");
            System.out.print("계좌주 : " + account.getOwner() + "\t");
            System.out.println("잔액 : " + account.getBalance());
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(conn, pstmt, rs);
    }
    return accountList;
}
```

AccountDao – 예금

```java
//예금
public void deposit() {
    System.out.println("--------------------------");
    System.out.println("예금");
    System.out.println("--------------------------");

    while(true) {
        System.out.println("계좌 번호: ");
        String ano = scanner.next();
        //입력한 계좌가 없을때 처리
        if(findAccount(ano) == null) {
            System.out.println("계좌가 없습니다. 다시 입력하세요");
        }else {
            Account account = findAccount(ano);  //검색된 계좌 반환
            while(true) {
                System.out.println("입금액: ");
                int money = scanner.nextInt();
                String owner = account.getOwner();
                int balance = account.getBalance() + money;  //잔액 = 잔액 + 입금액
                if(money < 0) {
                    System.out.println("잘못된 입력입니다. 다시 입력하세요");
                }else {
                    conn = JDBCUtil.getConnection();
                    String sql = "UPDATE account SET owner = ?, balance = ?  WHERE ano = ? ";
                    try {
                        pstmt = conn.prepareStatement(sql);
                        pstmt.setString(1, owner);
                        pstmt.setInt(2, balance);
                        pstmt.setString(3, ano);
                        pstmt.executeUpdate();
                        System.out.printf("%,d원 정상 입금되었습니다.\n", money);
                    } catch (SQLException e) {
                        e.printStackTrace();
                    } finally {
                        JDBCUtil.close(conn, pstmt);
```

AccountDao – 출금

```java
//출금
public void withdraw() {
    System.out.println("--------------------------");
    System.out.println("출금");
    System.out.println("--------------------------");

    while(true) {
        System.out.println("계좌 번호: ");
        String ano = scanner.next();

        //입력한 계좌가 없을때 처리
        if(findAccount(ano) == null) {
            System.out.println("계좌가 없습니다. 다시 입력하세요");
        }else {
            Account account = findAccount(ano);
            while(true) {
                System.out.println("출금액: ");
                int money = scanner.nextInt();
                String owner = account.getOwner();
                int balance = account.getBalance() - money;  //잔액 = 잔액 - 입금액
                if(money > account.getBalance()) {
                    System.out.println("잔액이 부족합니다. 다시 입력하세요");
                }else if(money < 0) {
                    System.out.println("잘못된 입력입니다. 다시 입력하세요");
                }else {
                    conn = JDBCUtil.getConnention();
                    String sql = "UPDATE account SET owner = ?, balance = ?  WHERE ano = ? ";
                    try {
                        pstmt = conn.prepareStatement(sql);
                        pstmt.setString(1, owner);
                        pstmt.setInt(2, balance);
                        pstmt.setString(3, ano);
                        pstmt.executeUpdate();
                        System.out.printf("%,d원 정상 출금되었습니다.\n", money);
                    } catch (SQLException e) {
```

AccountDao – 계좌 상세보기

```java
//계좌 상세보기
public void viewAccount() {
    System.out.println("--------------------------");
    System.out.println("계좌 검색");
    System.out.println("--------------------------");

    while(true) {
        System.out.println("계좌 번호: ");
        String ano = scanner.next();

        //입력한 계좌가 없을때 처리
        if(findAccount(ano) == null) {
            System.out.println("계좌가 없습니다. 다시 입력하세요");
        }else {
            Account account = findAccount(ano);

            System.out.print("계좌번호: " + account.getAno() + "\t");
            System.out.print("계좌주: " + account.getOwner() + "\t");
            System.out.println("잔액: " + account.getBalance());
            break;
        }
    }
}
```

AccountDao – 계좌 삭제

```java
//계좌 삭제
public void deleteAccount() {
    System.out.println("---------------------------");
    System.out.println("계좌 삭제");
    System.out.println("---------------------------");

    while(true) {
        System.out.println("계좌 번호: ");
        String ano = scanner.next();

        //입력한 계좌가 없을때 처리
        if(findAccount(ano) == null) {
            System.out.println("계좌가 없습니다. 다시 입력하세요");
        }else {
            conn = JDBCUtil.getConnection();
            String sql = "DELETE FROM account WHERE ano = ?";
            try {
                pstmt = conn.prepareStatement(sql);
                pstmt.setString(1, ano);
                pstmt.executeUpdate();
                System.out.println("결과 : 계좌가 삭제되었습니다.");
            } catch (SQLException e) {
                e.printStackTrace();
            } finally {
                JDBCUtil.close(conn, pstmt, rs);
            }
            break;
        }
    }
}
```

AccountDao – 계좌 찾기

```
//계좌 찾기 메서드(전체 메서드에서 사용됨)
public Account findAccount(String ano) {
    Account account = null;
    conn = JDBCUtil.getConnention();
    String sql = "SELECT * FROM account WHERE ano = ?";
    try {
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, ano);
        rs = pstmt.executeQuery();
        if(rs.next()) {
            ano = rs.getString("ano");
            String owner = rs.getString("owner");
            int balance = rs.getInt("balance");

            account = new Account(ano, owner, balance);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(conn, pstmt, rs);
    }
    return account;
}
```

Main

```java
package banking_db;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        AccountDao dao = new AccountDao();
        Scanner scanner = new Scanner(System.in);

        boolean run = true;

        while(run) {
            System.out.println("-----------------------------------------------------------------");
            System.out.println("1.계좌생성 | 2.계좌목록 | 3.예금 | 4.출금 | 5.계좌검색 | 6.계좌삭제 | 7.종료");
            System.out.println("-----------------------------------------------------------------");
            System.out.print("선택> ");

            String selectNo = scanner.next();
            if(selectNo.equals("1")) {
                dao.createAccount();
            }
            else if(selectNo.equals("2")){
                dao.getAccountList();
            }
```

Main

```
        else if(selectNo.equals("3")) {
            dao.deposit();
        }
        else if(selectNo.equals("4")) {
            dao.withdraw();
        }
        else if(selectNo.equals("5")){
            dao.viewAccount();
        }
        else if(selectNo.equals("6")){
            dao.deleteAccount();
        }
        else if(selectNo.equals("7")){
            run = false;
        }
        else{
            System.out.println("지원되지 않는 기능입니다.");
        }
    }
    System.out.println("프로그램 종료!!");
    scanner.close();
}
```