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

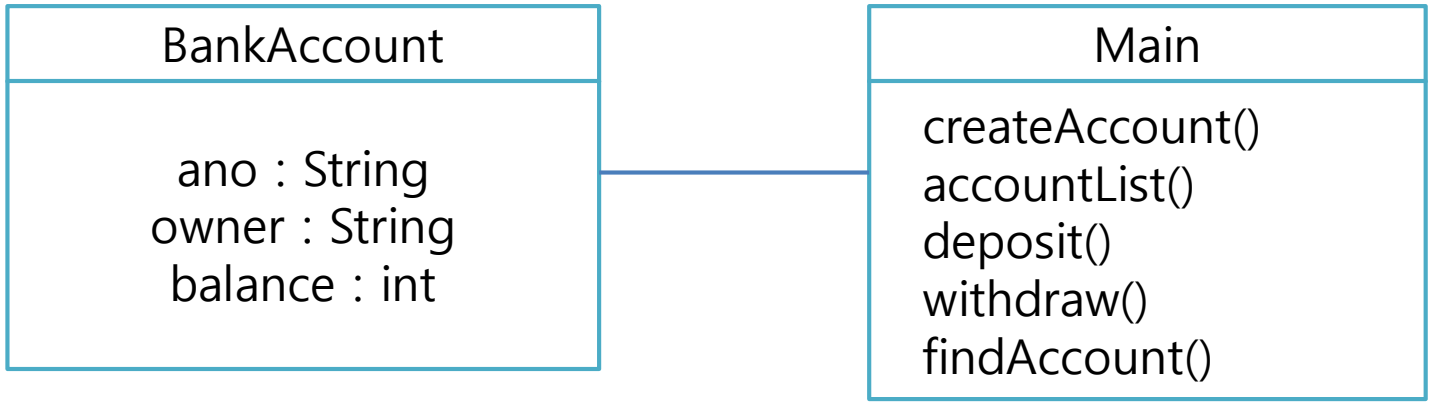
클래스 관계도 그리기

BankAccount 클래스

계좌 번호
계좌주
잔액

Main 클래스

계좌 생성
계좌 목록
입금
출금



step2. 클래스 정의하기

- **Account 클래스**

```java
package bankapp;

public class BankAccount {
    private String ano;    //계좌 번호
    private String owner; //계좌주
    private int balance;  //잔고

    public BankAccount(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**

```
public class BankMain {
    static BankAccount[] accounts = new BankAccount[100];
    static Scanner scan = new Scanner(System.in);

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

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

            int selectNum = Integer.parseInt(scan.nextLine()); //메뉴 선택
```

step3. 은행 업무 기능 설계, 구현

- **Main 클래스**

```
            if(selectNum == 1) {
                createAccount();
            }else if(selectNum == 2) {
                getAccountList();
            }else if(selectNum == 3) {
                deposit();
            }else if(selectNum == 4) {
                withdraw();
            }else if(selectNum == 5) {
                System.out.println("프로그램을 종료합니다.");
                sw = false;
            }else {
                System.out.println("지원되지 않는 기능입니다. 다시 입력해 주세요");
            }
        } //while() 끝
        scan.close();
    }//main() 닫기
```

step3. 은행 업무 기능 설계 , 구현

- 계좌 생성

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

    System.out.print("계좌번호: ");
    String ano = scan.nextLine();

    System.out.print("계좌주: ");
    String owner = scan.nextLine();

    System.out.print("초기입금액: ");
    int balance = Integer.parseInt(scan.nextLine());

    //첫번째 계좌 생성
    //accounts[0] = new BankAccount(ano, owner, balance);

    for(int i=0; i<accounts.length; i++) {
        if(accounts[i] == null) {
            accounts[i] = new BankAccount(ano, owner, balance);
            System.out.println("결과: 계좌가 생성되었습니다.");
            break;
        }
    }
}
```

step3. 은행 업무 기능 설계 , 구현

- 계좌 목록

```
private static void getAccountList() {
    for(int i=0; i<accounts.length; i++) {
        if(accounts[i] != null) {
            System.out.print("계좌번호: " + accounts[i].getAno() + "\t");
            System.out.print("계좌주: " + accounts[i].getOwner() + "\t");
            System.out.print("잔고: " + accounts[i].getBalance() + "\n");
        }
    }
}
```

step3. 은행 업무 기능 설계, 구현

- 예금

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

    System.out.print("계좌번호: ");
    String ano = scan.nextLine();

    System.out.print("입금액: ");
    int amount = Integer.parseInt(scan.nextLine());

    if(findAccount(ano) != null) { //찾는 계좌가 있다면
        BankAccount account = findAccount(ano);
        //예금 = 잔고 + 입금액
        account.setBalance(account.getBalance() + amount);
        System.out.println("결과: 정상 입금되었습니다. 현재 잔액: " + account.getBalance());
    }else {
        System.out.println("결과: 계좌가 없습니다.");
    }
}
```

step3. 은행 업무 기능 설계, 구현

- 출금

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

    System.out.print("계좌번호: ");
    String ano = scan.nextLine();

    System.out.print("출금액: ");
    int amount = Integer.parseInt(scan.nextLine());

    if(findAccount(ano) != null) {
        BankAccount account = findAccount(ano);
        //출금 = 잔고 + 출금액
        account.setBalance(account.getBalance() - amount);
        System.out.println("결과: 정상 출금되었습니다. 현재 잔액: " + account.getBalance());
    }else {
        System.out.println("결과: 계좌가 없습니다.");
    }
}
```

step3. 은행 업무 기능 설계, 구현

- 계좌 검색

```
private static BankAccount findAccount(String ano) {
    BankAccount account = null; //BankAccoun 객체 선언

    for(int i=0; i<accounts.length; i++) {
        if(accounts[i] != null) {
            String dbAno = accounts[i].getAno(); //이미 저장된 계좌
            if(dbAno.equals(ano)) { //찾는 계좌와 일치한다면
                account = accounts[i];
                break;
            }
        }
    }
    return account;
}
```

step4. 프로그램 테스트 하기

1. 출금시 잔액 부족

```
if(findAccount(ano) != null) {
    BankAccount account = findAccount(ano);
    while(true) {
        System.out.print("출금액: ");
        int amount = Integer.parseInt(scan.nextLine());
        if(amount > account.getBalance()) {
            System.out.println("잔액이 부족합니다. 다시 입력하세요");
        }else {
            //예금 = 잔고 - 입금액
            account.setBalance(account.getBalance() - amount);
            System.out.println("결과: 정상 출금되었습니다. 현재 잔액: " + account.getBalance());
            break;
        }
    }
}else {
    System.out.println("결과: 계좌가 없습니다.");
}
```

step4. 프로그램 테스트 하기

2. 메뉴 선택시 문자 입력 예외 처리

```
public class BankMain {
    static BankAccount[] accounts = new BankAccount[100];
    static Scanner scan = new Scanner(System.in);

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

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

            try { //문자 입력시 예외 처리
                int selectNum = Integer.parseInt(scan.nextLine());

                switch(selectNum) {
                case 1:
                    createAccount();
                    break;
                case 2:
                    getAccountList();
                    break;
```

step4. 프로그램 테스트 하기

2. 메뉴 선택시 문자 입력 예외 처리

```
                case 3:
                    deposit();
                    break;
                case 4:
                    withdraw();
                    break;
                case 5:
                    System.out.println("프로그램을 종료합니다.");
                    sw = false;
                    break;
                default:
                    System.out.println("지원되지 않는 기능입니다. 다시 입력하세요");
                    break;
                }
            }catch(Exception e) {
                System.out.println("잘못된 입력입니다. 다시 입력하세요");
            }
        }//while() 닫기
        scan.close();
}//main() 닫기
```

ver2. ArrayList로 구현하기

- 메인 화면

```
public class Banking {
    //BankAccount를 저장할 ArrayList 자료구조 생성
    static List<BankAccount> accountList = new ArrayList<>();
    static Scanner scan = new Scanner(System.in);

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

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

            try { //문자 입력시 예외 처리
                int selectNum = Integer.parseInt(scan.nextLine());
```

ver2. ArrayList로 구현하기

- 메인 화면

```
                switch(selectNum) {
                case 1:
                    createAccount();
                    break;
                case 2:
                    getAccountList();
                    break;
                case 3:
                    deposit();
                    break;
                case 4:
                    withdraw();
                    break;
                case 5:
                    System.out.println("프로그램을 종료합니다.");
                    sw = false;
                    break;
                default:
                    System.out.println("지원되지 않는 기능입니다. 다시 입력하세요");
                    break;
                }
            }catch(Exception e) {
                System.out.println("잘못된 입력입니다. 다시 입력하세요");
            }
        }//while() 닫기
        scan.close();
}
```

ver2. ArrayList로 구현하기

- 계좌 생성

```
private static void createAccount() {
    System.out.println("=============================================");
    System.out.println("                  계  좌  생  성                  ");
    System.out.println("=============================================");

    while(true) {
        System.out.print("계좌번호: ");
        String ano = scan.nextLine();

        if(findAccount(ano) != null) {
            System.out.println("이미 등록된 계좌입니다. 다른 계좌를 입력해 주세요.");
        }else {
            System.out.print("계좌주: ");
            String owner = scan.nextLine();

            System.out.println("초기입금액: ");
            int balance = Integer.parseInt(scan.nextLine());

            //신규 계좌 생성
            BankAccount newAccount = new BankAccount(ano, owner, balance);
            accountList.add(newAccount);  //리스트에 추가(저장)
            System.out.println("결과: 계좌가 생성되었습니다.");
            break;
        }
    }
}
```

ver2. ArrayList로 구현하기

- **계좌 검색**

```
private static Account findAccount(String ano) {
    Account account = null;  //빈 계좌 계정을 할당

    for(int i = 0; i < accountList.size(); i++) {
        String dbAno = accountList.get(i).getAno(); //이미 등록된 계좌번호
        if(dbAno.equals(ano)) { //등록된 계좌와 찾는 계좌가 일치하면
            account = accountList.get(i); //등록 계좌 객체 생성
            break;
        }
    }
    return account;
}
```

ver2. ArrayList로 구현하기

- **계좌 목록**

```
private static void getAccountList() {
    for(int i = 0; i < accountList.size(); i++) { //리스트를 순회하면서
        Account account = accountList.get(i);  //등록된 계좌를 가져옴
        System.out.print("계좌번호: " + account.getAno() + "\t");
        System.out.print("계좌주: " + account.getOwner() + "\t");
        System.out.println("잔액: " + account.getBalance());
    }
}
```

ver2. ArrayList로 구현하기

- 예 금

```
private static void deposit() {
    System.out.println("=============================================");
    System.out.println("                  예          금                  ");
    System.out.println("=============================================");

    System.out.print("계좌번호: ");
    String ano = scan.nextLine();

    System.out.print("입금액: ");
    int amount = Integer.parseInt(scan.nextLine());

    if(findAccount(ano) != null) {
        BankAccount account = findAccount(ano);
        account.setBalance(account.getBalance() + amount);
        System.out.println("결과: 정상 입금되었습니다. 현재 잔액: " + account.getBalance());
    }else {
        System.out.println("결과: 계좌가 없습니다.");
    }
}
```

ver2. ArrayList로 구현하기

- 출 금

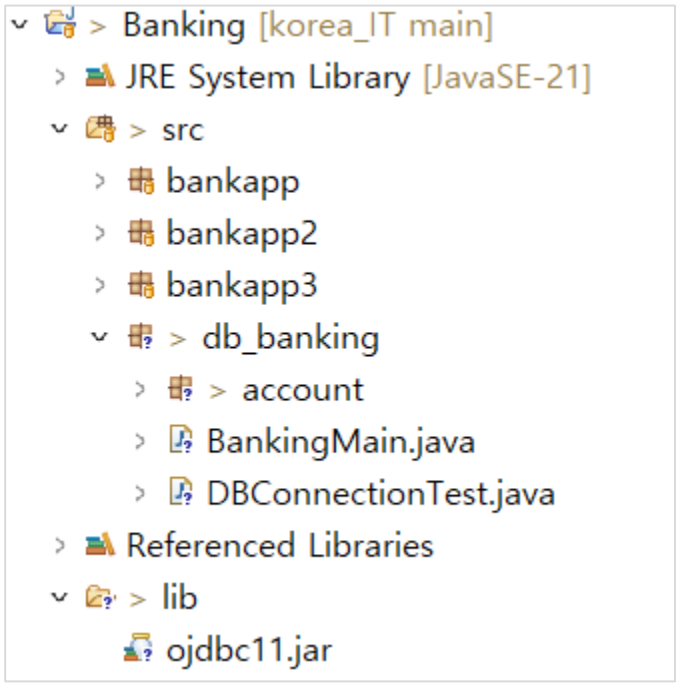
```
private static void withdraw() {
    System.out.println("=============================================");
    System.out.println("                  출          금                  ");
    System.out.println("=============================================");

    System.out.print("계좌번호: ");
    String ano = scan.nextLine();

    if(findAccount(ano) != null) {
        BankAccount account = findAccount(ano);
        while(true) {
            System.out.print("출금액: ");
            int amount = Integer.parseInt(scan.nextLine());
            if(amount > account.getBalance()) {
                System.out.println("잔액이 부족합니다. 다시 입력하세요");
            }else {
                account.setBalance(account.getBalance() - amount);
                System.out.println("결과: 정상 출금되었습니다. 현재 잔액: " + account.getBalance());
                break;
            }
        }
    }else {
        System.out.println("결과: 계좌가 없습니다.");
    }
}
```

Banking ver3. 오라클DB 연동

- **프로젝트 구조**



- 상위 패키지: db_banking
- 하위 패키지: account

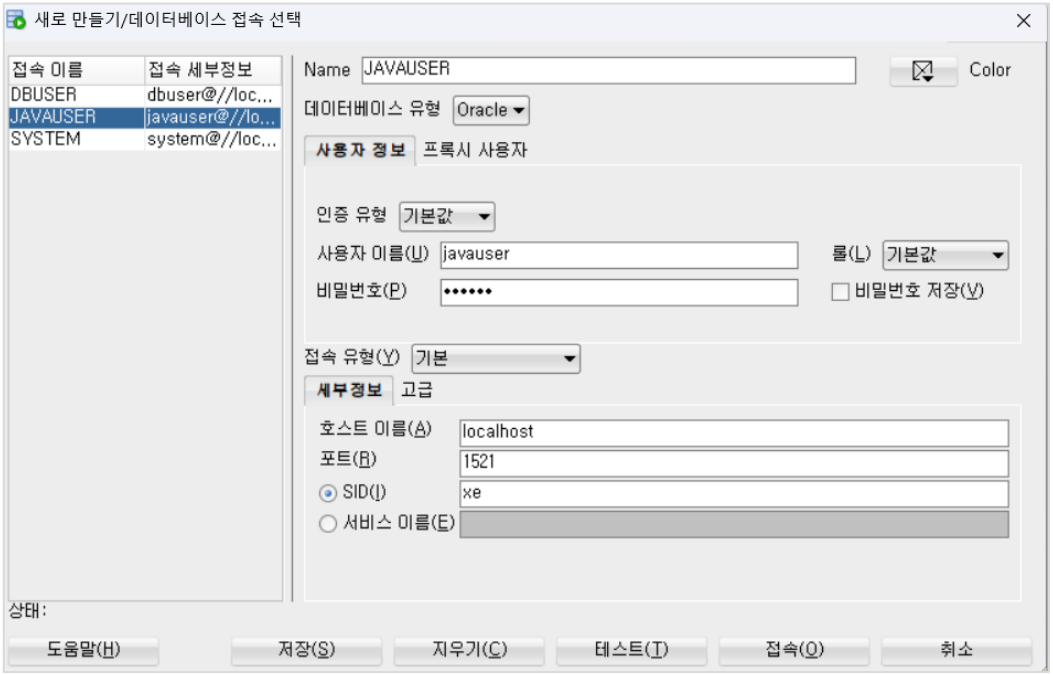
  (BankAccount.java, BankAccountDAO)
- 메인클래스: BankingMain.java
- db 연결 테스트: DBConnectTest..java

Banking ver3. 오라클DB 연동

- **SQL developer – 데이터베이스 사용(system)**

  사용자 이름(계정) : javauser

  비밀번호: pwjava



Banking ver3. 오라클DB 연동

- **DB 연결 테스트**

```
public class DBConnectionTest {
    //static{} - 정적 초기화 블럭
    //매번 연결시 마다 드라이버를 로드할 필요가 없어짐
    static {
        try {
            // 클래스 로딩시 드라이버 등록
            Class.forName("oracle.jdbc.OracleDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    static String url = "jdbc:oracle:thin:@localhost:1521/xe"; //db 경로
    static String username = "system"; //사용자 계정명
    static String password = "pw1234"; //사용자 비밀번호

    public static void main(String[] args) {

        try(Connection conn = DriverManager.getConnection(url, username, password)){
            System.out.println(conn + ": DB 연결 성공!");
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
    }
}
```

Banking ver3. 오라클DB 연동

- **bank_account 테이블 생성**

```
-- BankAccount 테이블 생성
CREATE TABLE bank_account(
    ano     VARCHAR2(10) PRIMARY KEY,
    owner   VARCHAR2(20) NOT NULL,
    balance NUMBER(10) NOT NULL
);

SELECT * FROM bank_account;

INSERT INTO bank_account VALUES ('1111', '김기용', 10000);

COMMIT;
```

BankAccountDAO()

- **BankAccountDAO() 클래스**

```
package db_banking.account;

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


public class BankAccountDAO {
    static {
        try {
            // 클래스 로딩시 드라이버 등록
            Class.forName("oracle.jdbc.OracleDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    static String url = "jdbc:oracle:thin:@localhost:1521/xe"; //db 경로
    static String username = "javauser";  //사용자 계정명
    static String password = "pwjava";  //사용자 비밀번호
```

BankAccountDAO()

- **계좌 생성**

```
//계좌 생성
public void createAccount(BankAccount account) {
    //SQL - DML(삽입 구문)
    String sql = "INSERT INTO bank_account VALUES (?, ?, ?)";

    //Connection 객체(db 연결), PreparedStatement 객체(sql 처리) 생성
    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql)){

        pstmt.setString(1, account.getAno());
        pstmt.setString(2, account.getOwner());
        pstmt.setInt(3, account.getBalance());

        pstmt.executeUpdate();  //sql 실행(커밋)
    }catch (SQLException e) {
        e.printStackTrace();
    }
}
```

BankAccountDAO()

- **계좌 목록 보기**

```
public List<BankAccount> getAccountList() {
    //SQL - DML(검색 구문)
    String sql = "SELECT * FROM bank_account";
    //검색된 user를 저장할 리스트 객체 생성
    List<BankAccount> accountList = new ArrayList<>();

    //ResultSet rs 객체(db에서 가져온 자료)
    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql);
        ResultSet rs = pstmt.executeQuery()){

        while(rs.next()) { //검색된 자료가 있는 동안 계속
            String ano = rs.getString("ano");
            String owner = rs.getString("owner");
            int balance = rs.getInt("balance");

            BankAccount account = new BankAccount(ano, owner, balance);
            accountList.add(account); //user 객체를 리스트에 저장
        }
    }catch(SQLException e) {
        e.printStackTrace();
    }
    return accountList;  //리스트를 반환함
}
```

BankAccountDAO()

- **계좌 검색(상세 보기)**

```
public BankAccount findAccount(String ano) {
    //SQL - DML(검색 구문)
    String sql = "SELECT * FROM bank_account WHERE ano = ?";
    //검색된 account를 저장할 리스트 객체 생성
    BankAccount account = null;

    //ResultSet rs 객체(db에서 가져온 자료)
    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql)){
        pstmt.setString(1, ano);

        try(ResultSet rs = pstmt.executeQuery()){
            if(rs.next()) { //검색된 자료가 있는 동안 계속
                ano = rs.getString("ano");
                String owner = rs.getString("owner");
                int balance = rs.getInt("balance");
                account = new BankAccount(ano, owner, balance); //계좌 생성
            }
        }
    }catch(SQLException e) {
        e.printStackTrace();
    }
    return account;  //리스트를 반환함
}
```

BankAccountDAO()

- 입금

```
public void deposit(String ano, int money) { //계좌번호, 입금액
    BankAccount account = findAccount(ano);
    String owner = account.getOwner();
    int balance = account.getBalance() + money; //잔고 + 입금액

    //sql - 계좌 수정
    String sql = "UPDATE bank_account SET owner = ?, balance = ? WHERE ano = ?";
    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql)){

        pstmt.setString(1, owner);
        pstmt.setInt(2, balance);
        pstmt.setString(3, ano);

        pstmt.executeUpdate();  //sql 실행(커밋)
    }catch (SQLException e) {
        e.printStackTrace();
    }
}
```

BankAccountDAO()

- 출금

```
public void withdraw(String ano, int money) { //계좌번호, 출금액
    BankAccount account = findAccount(ano);
    String owner = account.getOwner();
    int balance = account.getBalance() - money; //잔고 - 출금액

    //sql - 계좌 수정
    String sql = "UPDATE bank_account SET owner = ?, balance = ? WHERE ano = ?";
    try(Connection conn = DriverManager.getConnection(url, username, password);
        PreparedStatement pstmt = conn.prepareStatement(sql)){

        pstmt.setString(1, owner);
        pstmt.setInt(2, balance);
        pstmt.setString(3, ano);

        pstmt.executeUpdate();  //sql 실행(커밋)
    }catch (SQLException e) {
        e.printStackTrace();
    }
}
```

BankingMain 클래스

- 메인 클래스

```
public class BankingMain {
    //BankAccountDAO 객체 생성
    static BankAccountDAO accountDAO = new BankAccountDAO();
    static Scanner scan = new Scanner(System.in);

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

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

            try { //문자 입력시 예외 처리
                int selectNum = Integer.parseInt(scan.nextLine());

                switch(selectNum) {
                case 1:
                    createAccount();
                    break;
```

BankingMain 클래스

- 메인 클래스

```
                case 2:
                    getAccountList();
                    break;
                case 3:
                    deposit();
                    break;
                case 4:
                    withdraw();
                    break;
                case 5:
                    System.out.println("프로그램을 종료합니다.");
                    sw = false;
                    break;
                default:
                    System.out.println("지원되지 않는 기능입니다. 다시 입력하세요");
                    break;
                }
            }catch(Exception e) {
                System.out.println("잘못된 입력입니다. 다시 입력하세요");
            }
        }//while() 닫기
        scan.close();
}
```

BankingMain 클래스

- **createAccount()**

```
private static void createAccount() {
    System.out.println("---------------------------------------------");
    System.out.println("                계 좌 생 성                  ");
    System.out.println("---------------------------------------------");

    while(true) {
        System.out.print("계좌번호: ");
        String ano = scan.nextLine();

        if(accountDAO.findAccount(ano) != null) {
            System.out.println("이미 등록된 계좌입니다. 다른 계좌를 입력해 주세요.");
        }else {
            System.out.print("계좌주: ");
            String owner = scan.nextLine();

            System.out.println("초기입금액: ");
            int balance = Integer.parseInt(scan.nextLine());

            //신규 계좌 생성
            BankAccount newAccount = new BankAccount(ano, owner, balance);
            accountDAO.createAccount(newAccount);
            System.out.println("결과: 계좌가 생성되었습니다.");
            break;
        }
    }
}
```

BankingMain 클래스

- **getAccountList()**

```
private static void getAccountList() {
    System.out.println("=============================================");
    System.out.println("                  계  좌  목  록                  ");
    System.out.println("=============================================");

    //accountList 가져오기
    List<BankAccount> accountList = accountDAO.getAccountList();
    for(int i = 0; i < accountList.size(); i++) {
        BankAccount account = accountList.get(i);
        System.out.print("계좌번호: " + account.getAno() + "\t");
        System.out.print("계좌주: " + account.getOwner() + "\t");
        System.out.println("잔고: " + account.getBalance());
    }
}
```

BankingMain 클래스

- **deposit()**

```
private static void deposit() {
    System.out.println("=============================================");
    System.out.println("                  예          금                  ");
    System.out.println("=============================================");

    System.out.print("계좌번호: ");
    String ano = scan.nextLine();

    System.out.print("입금액: ");
    int amount = Integer.parseInt(scan.nextLine());

    if(accountDAO.findAccount(ano) != null) {
        accountDAO.deposit(ano, amount);  //입금
        BankAccount account = accountDAO.findAccount(ano);  //계좌 가져와서
        System.out.println("결과: 정상 입금되었습니다. 현재 잔액: " + account.getBalance());
    }else {
        System.out.println("결과: 계좌가 없습니다.");
    }
}
```

BankingMain 클래스

- **deposit()**

```
private static void withdraw() {
    System.out.println("=============================================");
    System.out.println("                  출          금                  ");
    System.out.println("=============================================");

    System.out.print("계좌번호: ");
    String ano = scan.nextLine();

    if(accountDAO.findAccount(ano) != null) {
        while(true) {
            System.out.print("출금액: ");
            int amount = Integer.parseInt(scan.nextLine());
            accountDAO.withdraw(ano, amount);  //출금

            BankAccount account = accountDAO.findAccount(ano); //계좌 가져오기
            if(amount > account.getBalance()) {
                System.out.println("잔액이 부족합니다. 다시 입력하세요");
            }else {
                System.out.println("결과: 정상 출금되었습니다. 현재 잔액: " +
                        account.getBalance());
                break;
            }
        }
    }else {
        System.out.println("결과: 계좌가 없습니다.");
    }
```