Banking App

은행 거래

은행 업무 프로젝트 개요

◆ 은행 업무 프로젝트

은행 계좌 클래스를 만들고, 은행 업무 기능 만들기

- 학점 산출 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

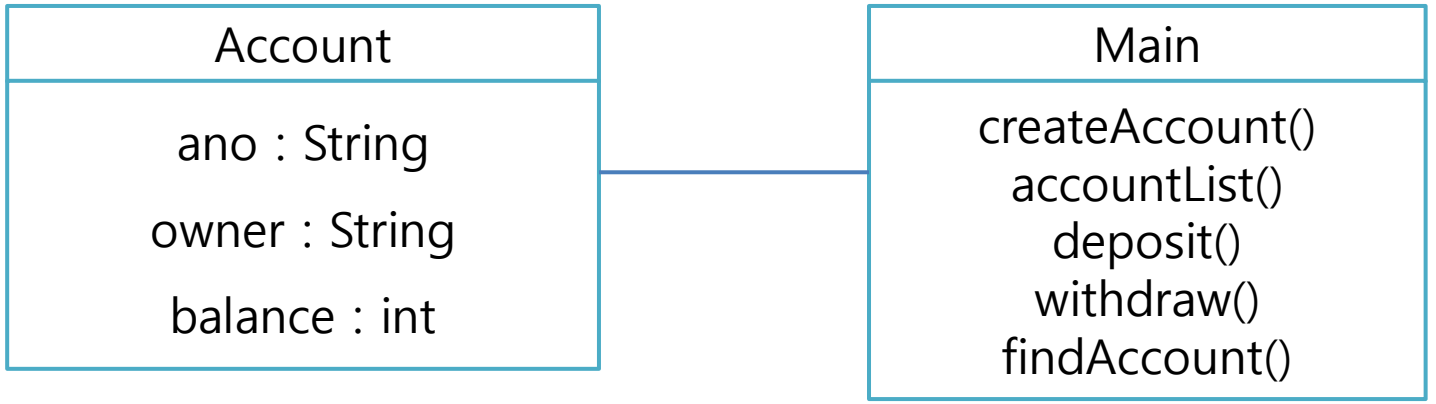
클래스 관계도 그리기

Account 클래스

계좌 번호
계좌주
잔액

Main 클래스

계좌 생성
계좌 목록
입금
출금



step2. 클래스 정의하기

클래스 정의하기

```java
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;
    }
```

```java
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 – 중복 계좌 체크, 초기 입금액 설정**

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

    //계좌번호(기본키) 중복 문제
    System.out.println("계좌 번호 :");
    String ano = scanner.next();
    if(findAccount(ano) != null) {
        System.out.println("중복 계좌입니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

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

step3. 은행 업무 기능 설계 , 구현

- **계좌 생성 – 중복 계좌 체크, 초기 입금액 설정**

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

        //초기 입금액 100원 이상 설정
        System.out.println("초기 입금액 : ");
        int balance = scanner.nextInt();
        if(balance < 100) {
            System.out.println("기본 입금액은 100원 이상입니다. 다시 입력하세요");
            System.out.println("초기 입금액 : ");
            balance = scanner.nextInt();
        }

        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;
            }
        }
    }
```

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() + "   ");
            System.out.print("계좌주 : " + account.getOwner() + "   ");
            System.out.println("잔액 : " + account.getBalance());
        }
    }
}
```

step3. 은행 업무 기능 설계, 구현

- **계좌 검색**

```
private static Account findAccount(String ano) {
    //계좌번호로 검색
    Account account = null;
    for(int i=0; i<accountArray.length; i++) {
        if(accountArray[i] != null) {
            String dbAno = accountArray[i].getAno();
            if(dbAno.equals(ano)) {
                account = accountArray[i];
                break;
            }
        }
    }
    return account;
}
```

step3. 은행 업무 기능 설계, 구현

- 예금 – 계좌가 없으면 재 입력

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

    System.out.println("계좌 번호 :");
    String ano = scanner.next();
    if(findAccount(ano) == null) {
        System.out.println("계좌가 없습니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

    Account account = findAccount(ano);

    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(인출) – 잔액 부족이 경우 오류 처리

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

    System.out.println("계좌 번호 :");
    String ano = scanner.next();
    if(findAccount(ano) == null) {
        System.out.println("계좌가 없습니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

    Account account = findAccount(ano);

    System.out.println("출금액 :");
    int money = scanner.nextInt();
    if(money > account.getBalance()) {
        System.out.println("잔액이 부족합니다. 다시 입력하세요");
        System.out.println("출금액 :");
        money = scanner.nextInt();
    }
    account.setBalance(account.getBalance() - money);
    System.out.println("결과 : 출금을 성공하였습니다.");
}
```

Banking ver2. ArrayList로 구현하기

- **Main 클래스**

```
public class Main {
    private static ArrayList<Account> accountList = new ArrayList<>();
    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.계좌검색 | 6.계좌삭제 | 7.종료");
            System.out.println("-----------------------------------------------------------");
            System.out.print("선택> ");

            String selectNo = scanner.next();
            if(selectNo.equals("1")) {
                createAccount();
            }else if(selectNo.equals("2")){
                accountList();
            }else if(selectNo.equals("3")) {
                deposit();
            }else if(selectNo.equals("4")) {
                withdraw();
```

Banking ver2. ArrayList로 구현하기

- **Main 클래스**

```
            }else if(selectNo.equals("5")){
                searchAccount();
            }else if(selectNo.equals("6")){
                deleteAccount();
            }else if(selectNo.equals("7")){
                run = false;
            }else{
                System.out.println("지원되지 않는 기능입니다.");
            }
        }
        System.out.println("프로그램 종료!!");
    }
```

Banking ver2. ArrayList로 구현하기

- 계좌 생성

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

    //계좌번호(기본키) 중복 문제
    System.out.println("계좌 번호 :");
    String ano = scanner.next();
    if(findAccount(ano) != null) {
        System.out.println("중복 계좌입니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

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

    //초기 입금액 100원 이상 설정
    System.out.println("초기 입금액 : ");
    int balance = scanner.nextInt();
    if(balance < 100) {
        System.out.println("기본 입금액은 100원 이상입니다. 다시 입력하세요");
        System.out.println("초기 입금액 : ");
        balance = scanner.nextInt();
    }

    Account newAccount = new Account(ano, owner, balance); //계좌 생성
    accountlist.add(newAccount);
    System.out.println("결과 : 계좌가 생성되었습니다.");
}
```

Banking ver2. ArrayList로 구현하기

- 계좌 목록

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

    for(int i = 0; i < accountlist.size(); i++) {
        Account account = accountlist.get(i);
        System.out.print("계좌번호 : " + account.getAno() + "   ");
        System.out.print("계좌주 : " + account.getOwner() + "   ");
        System.out.println("잔액 : " + account.getBalance());
    }
}
```

Banking ver2. ArrayList로 구현하기

- 계좌 번호로 검색

```
private static Account findAccount(String ano) {
    //계좌 번호로 검색
    Account account = null;
    for(int i = 0; i < accountlist.size(); i++) {
        String dbAno = accountlist.get(i).getAno();
        if(dbAno.equals(ano)) {
            account = accountlist.get(i);
            break;
        }
    }
    return account;
}
```

Banking ver2. ArrayList로 구현하기

- 입금(예금)

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

    System.out.println("계좌 번호 :");
    String ano = scanner.next();
    if(findAccount(ano) == null) {
        System.out.println("계좌가 없습니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

    Account account = findAccount(ano);

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

}
```

Banking ver2. ArrayList로 구현하기

- 출금(인출)

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

    System.out.println("계좌 번호 :");
    String ano = scanner.next();
    if(findAccount(ano) == null) {
        System.out.println("계좌가 없습니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

    Account account = findAccount(ano);

    System.out.println("출금액 :");
    int money = scanner.nextInt();
    if(money > account.getBalance()) {
        System.out.println("잔액이 부족합니다. 다시 입력하세요");
        System.out.println("출금액 :");
        money = scanner.nextInt();
    }
    account.setBalance(account.getBalance() - money);
    System.out.println("결과 : 출금을 성공하였습니다.");
}
```

Banking ver2. ArrayList로 구현하기

- 특정 계좌 1개 검색

```
private static void searchAccount() {
    System.out.println("-------");
    System.out.println("계좌 검색");
    System.out.println("-------");

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

    if(findAccount(ano) == null) {
        System.out.println("계좌가 없습니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

    Account account = findAccount(ano);
    System.out.print("계좌 번호:" + account.getAno() + "   ");
    System.out.print("계좌주:" + account.getOwner() + "   ");
    System.out.println("잔액:" + account.getBalance());
}
```

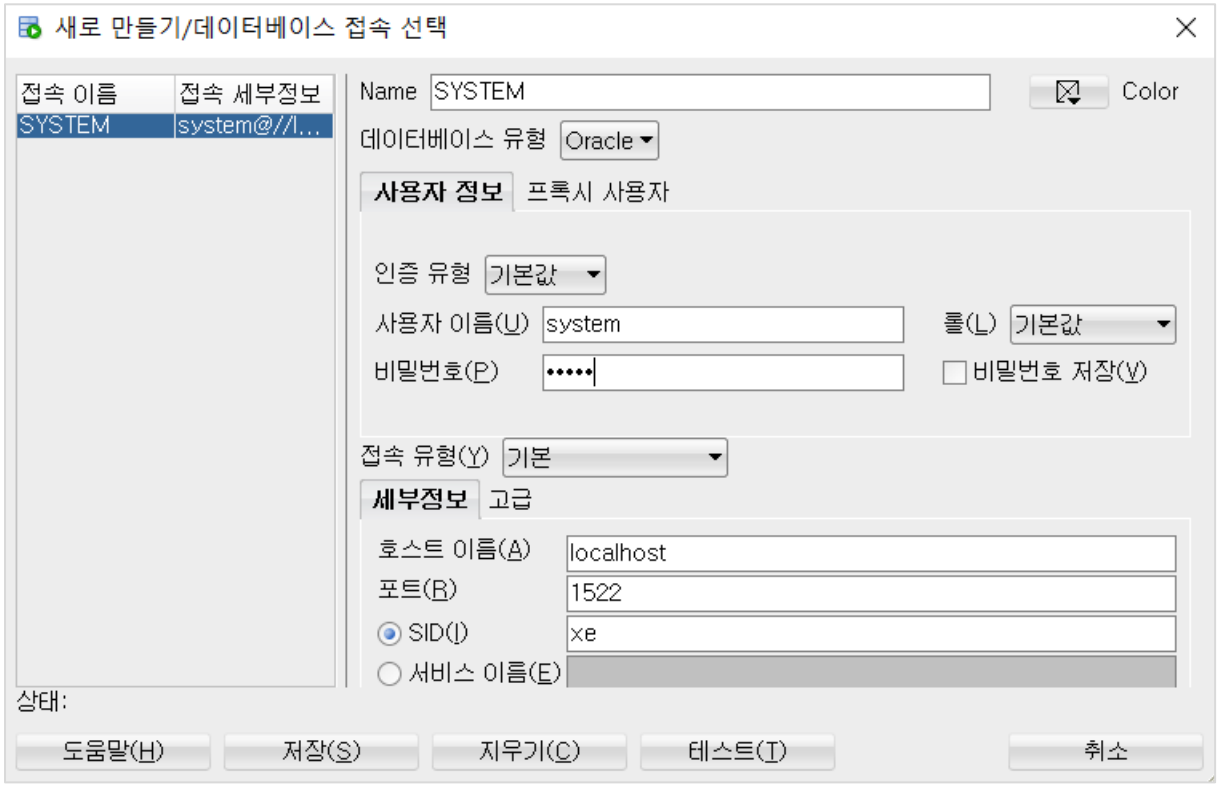
Banking ver2. ArrayList로 구현하기

- 계좌 삭제

```
private static void deleteAccount() {
    System.out.println("-------");
    System.out.println("계좌 삭제");
    System.out.println("-------");

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

    if(findAccount(ano) == null) {
        System.out.println("계좌가 없습니다. 다시 입력하세요");
        System.out.println("계좌 번호 :");
        ano = scanner.next();
    }

    Account account = findAccount(ano);

    accountlist.remove(account);
    System.out.println("계좌가 삭제되었습니다.");

}
```

Banking ver3. 오라클DB 연동

- **SqlDeveloper – 데이터베이스 사용(system)**



Banking ver3. 오라클DB 연동

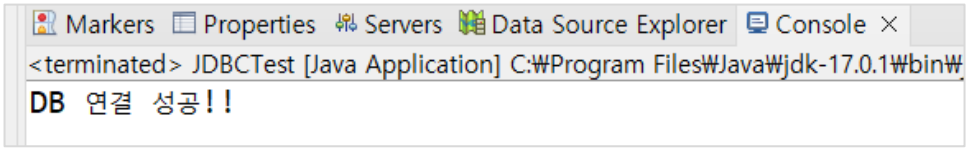
- **AccountDAO 클래스 – connDB() 메서드**

```
public class AccountDAO {
    private static final String driver="oracle.jdbc.driver.OracleDriver";
    private static final String url="jdbc:oracle:thin:@localhost:1522:xe";
    private static final String username="system";
    private static final String password="54321";

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

    public void connDB() {
        try {
            Class.forName(driver);
            conn = DriverManager.getConnection(url, username, password);
            System.out.println("DB 연결 성공!!");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

Banking ver3. 오라클DB 연동

- **JDBC 테스트 – DB 연결 확인**

```
public class JDBCTest {

    public static void main(String[] args) {
        AccountDAO dao = new AccountDAO();
        dao.connDB();
    }

}
```



Banking ver3. 오라클DB 연동

- **account 테이블 생성**

```
CREATE TABLE account(
    ano VARCHAR2(10) PRIMARY KEY,
    owner VARCHAR2(20) NOT NULL,
    balance NUMBER(10) NOT NULL
);

INSERT INTO account(ano, owner, balance) VALUES ('101', '김기용', 100);

SELECT * FROM account;
```

Banking ver3. 오라클DB 연동

- JDBC 테스트 – 계좌 생성, 검색, 삭제

```
//계좌 생성
Account newAccount = new Account("103", "park", 5000);
dao.createAccount(newAccount);

//계좌 목록
ArrayList<Account> accountList = dao.accountList();
for(int i = 0; i < accountList.size(); i++) {
    Account account = accountList.get(i);
    String ano = account.getAno();
    String owner = account.getOwner();
    int balance = account.getBalance();

    System.out.print("계좌번호 : " + ano + "   ");
    System.out.print("계좌주 : " + owner + "   ");
    System.out.println("잔액 : " + balance);
}
```

Banking ver3. 오라클DB 연동

- **JDBC 테스트 – 계좌 검색, 삭제, 입,출금**

```
//계좌 1개 상세 정보
Account account = dao.searchAccount("103");
String ano = account.getAno();
String owner = account.getOwner();
int balance = account.getBalance();

System.out.print("계좌번호 : " + ano + "   ");
System.out.print("계좌주 : " + owner + "   ");
System.out.println("잔액 : " + balance);

//계좌 삭제
dao.deleteAccount("102");

//예금
dao.deposit("101", 200);

//출금
dao.withdraw("103", 1000);
```

Banking ver3. AccountDAO() 작성

- **disconnect() – 연결 끊기**

```
private void disconnect() {
    if(pstmt != null) {
        try {
            pstmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if(conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```
private void disconnectRS() {
    if(rs != null) {
        try {
            rs.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if(pstmt != null) {
        try {
            pstmt.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if(conn != null) {
        try {
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Banking ver3. AccountDAO() 작성

- **createAccount()**

```
//계좌 생성
public void createAccount(Account account) {
    String ano = account.getAno();
    String owner = account.getOwner();
    int balance = account.getBalance();

    connDB();
    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();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        disconnect();
    }
}
```

Banking ver3. AccountDAO() 작성

- **accountList()**

```
//계좌 목록
public ArrayList<Account> accountList(){
    ArrayList<Account> accountList = new ArrayList<>();
    connDB();
    String sql = "SELECT * FROM account";
    try {
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        while(rs.next()) {
            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);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        disconnectRS();
    }

    return accountList;
}
```

Banking ver3. AccountDAO() 작성

- **accountList() – 계좌 전체 목록**

```
//계좌 목록
public ArrayList<Account> accountList(){
    ArrayList<Account> accountList = new ArrayList<>();
    connDB();
    String sql = "SELECT * FROM account";
    try {
        pstmt = conn.prepareStatement(sql);
        rs = pstmt.executeQuery();
        while(rs.next()) {
            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);
        }

    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        disconnectRS();
    }

    return accountList;
}
```

Banking ver3. AccountDAO() 작성

- **searchAccount() – 특정 계좌 상세 정보**

```
//계좌 검색
public Account searchAccount(String ano) {
    Account account = null;
    connDB();
    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 {
        disconnectRS();
    }

    return account;
}
```

Banking ver3. AccountDAO() 작성

- **deposit() – 입금**

```
//예금
public void deposit(String ano, int money) {
    Account account = searchAccount(ano);  //입금할 계좌 가져옴
    String owner = account.getOwner();
    int balance = account.getBalance() + money;  //잔액 = 잔액 + 입금액

    connDB();
    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();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        disconnect();
    }
}
```

Banking ver3. AccountDAO() 작성

- **withdraw() – 출금**

```
//출금
public void withdraw(String ano, int money) {
    Account account = searchAccount(ano);
    String owner = account.getOwner();
    int balance = account.getBalance() - money;

    connDB();
    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();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        disconnect();
    }
}
```

Banking ver3. 오라클DB 연동

- **Main 클래스**

```
package banking3;

import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    private static AccountDAO dao = new AccountDAO();
    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.계좌검색 | 6.계좌삭제 | 7.종료");
            System.out.println("-----------------------------------------------------------------");
            System.out.print("선택> ");

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

Banking ver3. 오라클DB 연동

- **Main 클래스**

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