BANK SYSTEM

- **기획의도 및 개발환경**
  - 기획의도
    - 자바로 은행프로그램 구현을 통해 기획과 흐름을 알게된다.
  - 개발환경
    - Eclipse
    - MYSQL
  - 테스트 계정
    - 회원 > ID: 11111 PW: 1111
    - 관리자 > ID : 1111 PW: 1111

- **기획 INDEX**
  - **MAIN MENU**
    1. 회원 > CUSTOMER MAIN
    2. 관리자 > CUSTOMER MAIN

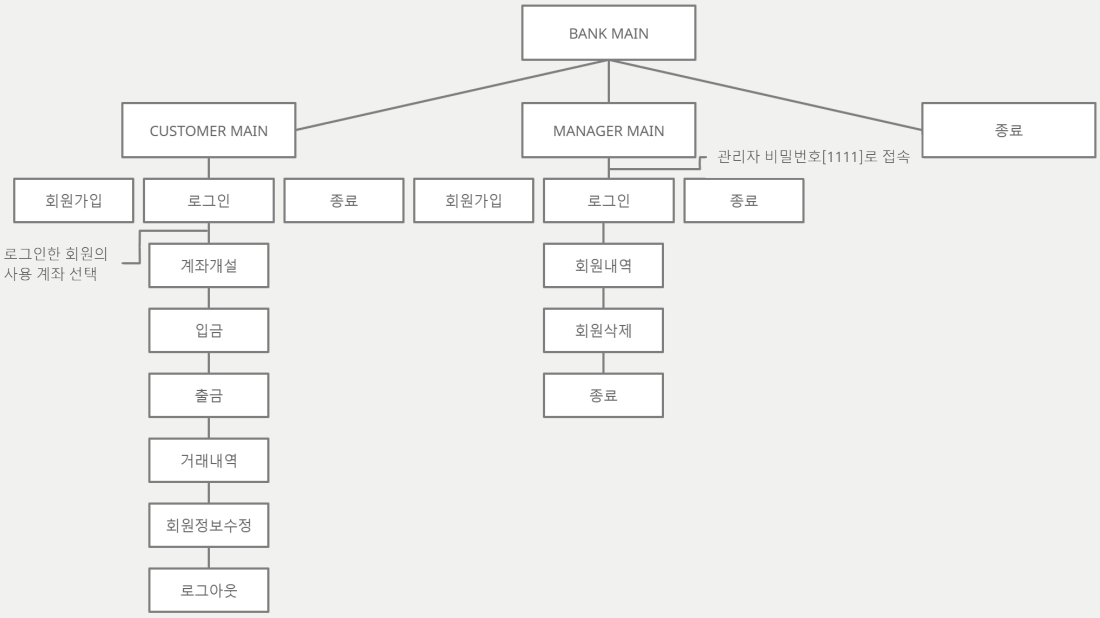
    0. 종료
  - **CUSTOMER MAIN**
    1. 회원가입
    2. 로그인 > 회원 계좌선택 > ACCOUNT MENU
       - ACCOUNT MENU
         1. 계좌개설
         2. 입금
         3. 출금
         4. 거래내역
         5. 회원정보수정

0. 로그아웃

3. 종료

- **CUSTOMER MAIN**
  - 관리자 모드 접속 (비밀번호)
  1. 회원가입
  2. 로그인 > MANAGER MENU
     - MANAGER MENU
       1. 회원내역
       2. 회원삭제
       0. 로그아웃
  0. 종료

- **WORKFLOW**



- **요구사항 분석**

기능	요구사항
은행 메인페이지(BANK MAIN)	회원/관리자 메뉴 구분 종료
회원 메인(CUSTOMER MAIN)	회원가입 로그인 종료
관리자 메인(MANAGER MAIN)	회원가입 로그인 종료

기능	요구사항
회원 메뉴(ACCOUNT MENU)	계좌개설 입/출금 거래내역 회원정보수정 로그아웃
관리자 메뉴(MANAGER MENU)	회원정보조회 회원삭제 종료
회원가입	고객번호 아이디 비밀번호 이름 생년월일 연락처
관리자 회원가입	아이디 비밀번호 이름
계좌개설	계좌번호(순서) 계좌번호 계좌비밀번호 잔고 고객번호
회원정보수정	회원정보조회 비밀번호변경 연락처변경 계좌삭제 종료

- 코드
  - Main
    - ▼ Start

```
package bank;

public class Start {

  public static void main(String[] args) throws Exception {
    new ConnectDB();
    new Menu();
  }
}
```

  - JDBC
    - ▼ ConnectDB

```
package bank;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ConnectDB {
  private static ConnectDB instance;
  Connection conn = null;
  Statement stmt = null;
  ResultSet rs = null;

  public ConnectDB() {
    try {
      // 1. JDBC 드라이버 로딩
      Class.forName("com.mysql.jdbc.Driver");
      // 2. Connection 객체 생성
      conn = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/ban
```

```
k?&allowPublicKeyRetrieval=true", "root", "1234");
      // 연결된 DB
      stmt = conn.createStatement();
      System.out.println("DB 연결 성공");
    } catch (ClassNotFoundException | SQLException e) {
      System.out.println("");
      System.out.println("DB접속 오류");
      e.printStackTrace(); // 오류 메시지 출력
    }
  }

  //getInstance 메소드를 통해 한번만 생성된 객체를 가져온다.
  public static ConnectDB getInstance() {
    if (instance == null)
      instance = new ConnectDB();
    return instance;
  }

  // 한 번 연결된 객체를 계속 사용
  // 즉, 연결되지 않은 경우에만 연결을 시도하겠다는 의미
  // → 싱글톤(디자인 패턴)
  public Connection getConnection() {
    return this.conn;
  }
}
```

- MAP 저장

  ▼ Session

```
package bank;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

public class Session {
  static int cno;
  static String pw;
  static String name;
  static String birth;
  static String tel;
  static int mno;
  static String mpw;
  static int ano;
  static String accountpw;
  static int balance;

  public Session() {
  }

  // 맵 생성
  static HashMap<String, Customer> map = new HashMap<String, Customer>();
```

```
// 고객
  static HashMap<String, Manager> map2 = new HashMap<String, Manager>();
 // 관리자
  static HashMap<String, Account> map3 = new HashMap<String, Account>();
 // 계좌
  static Map<String, List<Transaction>> map4 = new TreeMap<String, List<T
ransaction>>(); // 거래내역

  // 고객
  public static void put(String id, Customer ct) {
    map.put(id, new Customer(ct.getCno(), ct.getId(), ct.getPw(), ct.getN
ame(), ct.getBirth(), ct.getTel())); // 맵에 저장
  }

  public static Customer get1(String id) {
    return map.get(id);
  }

  // 관리자
  public static void put(String mid, Manager mg) {
    map2.put(mid, new Manager(mg.getMno(), mg.getMid(), mg.getMpw())); //
맵에 저장
  }

  // 키값 확인
  public static Manager get2(String mid) {
    return map2.get(mid);
  }

  // 계좌
  public static void put(String account, Account ac) {
    map3.put(account, new Account(ac.getAno(), ac.getAccount(), ac.getAcc
ountpw(), ac.getBalance(), ac.getCno())); // 맵에 저장
  }

  // 키값 확인
  public static Account get3(String account) {
    return map3.get(account);
  }

  // 거래내역
  public static void put(String account, List<Transaction> t) {
  map4.put(account, new ArrayList<Transaction>()); // 맵에 저장
  }

}
```

- DTO

  ▼ Customer

```
package bank;

public class Customer {
```

```
  private int cno; // 고객번호
  private String id; // 아이디
  private String pw; // 비밀번호
  private String name; // 이름
  private String birth; // 생년월일
  private String tel; // 연락처

  // 생성자
  public Customer() {
    super();
  }

  public Customer(int cno, String id, String pw, String name, String birt
h, String tel) {
    super();
    this.cno = cno;
    this.id = id;
    this.pw = pw;
    this.name = name;
    this.birth = birth;
    this.tel = tel;
  }

  public int getCno() {
    return cno;
  }

  public String getId() {
    return id;
  }

  public String getPw() {
    return pw;
  }

  public String getName() {
    return name;
  }

  public String getBirth() {
    return birth;
  }

  public String getTel() {
    return tel;
  }

  public void setCno(int cno) {
    this.cno = cno;
  }

  public void setId(String id) {
    this.id = id;
  }

  public void setPw(String pw) {
    this.pw = pw;
  }
```

```
  public void setName(String name) {
    this.name = name;
  }

  public void setBirth(String birth) {
    this.birth = birth;
  }

  public void setTel(String tel) {
    this.tel = tel;
  }

  @Override
  public String toString() {
    return  "┃NO :" + cno +"\t"+ "┃ID :" +id + "\t" + "┃PW :" + pw + "\t"
        + "┃이름 :"+ name + "\t" + "┃생년월일 :" + birth + "\t" + "┃연락처 :"
+ tel + "\t";
  }

}
```

▼ Manager

```
package bank;

public class Manager {

  private int mno; // 관리자 번호
  private String mid; // 아이디
  private String mpw; // 비밀번호

  // 생성자
  public Manager() {

  }

  public Manager(int mno, String mid, String mpw) {
    super();
    this.mno = mno;
    this.mid = mid;
    this.mpw = mpw;
  }

  public int getMno() {
    return mno;
  }

  public String getMid() {
    return mid;
  }

  public String getMpw() {
    return mpw;
  }
```

```
  public void setMno(int mno) {
    this.mno = mno;
  }

  public void setMid(String mid) {
    this.mid = mid;
  }

  public void setMpw(String mpw) {
    this.mpw = mpw;
  }


  @Override
  public String toString() {
    return  "┃NO :" + mno +" "+ "┃ID :" +mid + " " + "┃PW :" + mpw + " ";
  }

}
```

▼ Account

```
package bank;


public class Account {

  private int ano; // 계좌번호
  private String account; // 계좌
  private String accountpw; // 계좌 비밀번호
  private int balance; // 잔고
  private int cno; // 계좌번호

  // 기본 생성자
  public Account() {

  }

  public Account(int ano, String account, String accountpw, int balance,
 int cno) {
    this.ano = ano;
    this.account = account;
    this.accountpw = accountpw;
    this.balance = balance;
    this.cno = cno;

  }

  public String getAccount() {
    return account;
  }

  public String getAccountpw() {
    return accountpw;
```

```
  }

  public int getBalance() {
    return balance;
  }

  public void setAccount(String account) {
    this.account = account;
  }

  public void setAccountpw(String accountpw) {
    this.accountpw = accountpw;
  }

  public void setBalance(int balance) {
    this.balance = balance;
  }

  public int getAno() {
    return ano;
  }

  public void setAno(int ano) {
    this.ano = ano;
  }

  public int getCno() {
    return cno;
  }

  public void setCno(int cno) {
    this.cno = cno;
  }

  @Override
  public String toString() {
    return    "┃NO :" + ano +"\n"
        + "┃계좌번호 :" + account + "\n"
        + "┃계좌비밀번호 :" + accountpw + "\n"
        + "┃잔고 :" + balance + "\n";
  }

}
```

▼ Transaction

```
package bank;

public class Transaction {


  private boolean transType; // 입금 or 출금
  private int amount; // 거래금액
  private int balance; // 거래일자
  private int commission; // 거래일자
```

```
  private String transactionTime; // 거래시간

  public Transaction() {}


  public Transaction(boolean transType, int amount, int balance, int comm
ission, String transactionTime) {
    this.transType = transType;
    this.amount = amount;
    this.balance = balance;
    this.commission = commission;
    this.transactionTime = transactionTime;
  }

  public boolean isTransType() {
    return transType;
  }


  public void setTransType(boolean transType) {
    this.transType = transType;
  }

  public int getAmount() {
    return amount;
  }


  public void setAmount(int amount) {
    this.amount = amount;
  }


  public int getBalance() {
    return balance;
  }


  public int getCommission() {
    return commission;
  }


  public String getTransactionTime() {
    return transactionTime;
  }

  public void setBalance(int balance) {
    this.balance = balance;
  }


  public void setCommission(int commission) {
    this.commission = commission;
  }


  public void setTransactionTime(String transactionTime) {
```

```
    this.transactionTime = transactionTime;
  }

}
```

- DAO

▼ CheckCustomer

```
package bank;

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

public class CheckCustomer {
  PreparedStatement pstmt;
  Statement stmt;
  ResultSet rs;
  Customer ct;

  public CheckCustomer() {

  }

  // 회원 정보 아이디로 조회
  public void getCustomer(String id) {
    Connection conn = ConnectDB.getInstance().getConnection();
    List<Customer> ctlist = new ArrayList<>();
    String sql = "SELECT * FROM CUSTOMER WHERE ID=?";
    try {
      // 3. Statement 객체 생성
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, id);
      // 4. SQL 문장을 실행하고 결과를 리턴
      rs = pstmt.executeQuery();
      if (rs == null) {
        System.out.println("저장된 데이터가 없습니다.");
      } else {
        // 5. ResultSet에 저장된 데이터 얻기
        while (rs.next()) {
          // Customer객체에 저장
          Customer ct = new Customer();
          int cno = rs.getInt("cno");
          String pw = rs.getString("pw");
          String name = rs.getString("name");
          String birth = rs.getString("birth");
          String tel = rs.getString("tel");
          ct = new Customer(cno, id, pw, name, birth, tel);

          // 리스트에 추가
```

```
          ctlist.add(ct);
        }
      }
      // 결과물 출력
      for (Customer ct : ctlist) {
        System.out.println("┃NO :"+ ct.getCno() + " " + "┃ID :" + ct.getI
d() + " " + "┃PW :" + ct.getPw()+ " " + "┃NAME :" + ct.getName() + " " +
 "┃BIRTH :" + ct.getBirth() + " " + "┃TEL :" + ct.getTel());
      }
    } catch (SQLException e) {
      System.out.println("[정보 조회 실패 : " + e.getMessage() + "]");
    }
  }

  // 회원번호에 해당하는 한명의 회원정보 조회
  public void showCustomer(int cno) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "SELECT * FROM CUSTOMER WHERE CNO=?";
    Customer ct = null;
    ResultSet rs = null;
    PreparedStatement pstmt = null;
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setInt(1, cno);
      rs = pstmt.executeQuery();
      if(rs.next()) {
        ct = new Customer();
        ct.setCno(rs.getInt("cno"));
        ct.setId(rs.getString("id"));
        ct.setPw(rs.getString("pw"));
        ct.setName(rs.getString("name"));
        ct.setBirth(rs.getString("birth"));
        ct.setTel(rs.getString("tel"));

        System.out.println("                          [회원정보]");
        System.out.println("=========================================
==========================");
        System.out.println("┃No. : " +ct.getCno() + "\t" + "┃ID : " + ct.
getId() + "\t" + "┃PW : " + ct.getPw() + "\t" + "┃이름 : "
            + ct.getName() + "\t" +"┃생일 : " +ct.getBirth() + "\t"+ "┃연락
처 : " +ct.getTel() + "\t");
        System.out.println("=========================================
==========================");
      }

    } catch (Exception e) {
      System.out.println("예외발생: " + e.getMessage());
    }
  }

  // 고객 정보 입력
  public int join(Customer ct) {
    Connection conn = ConnectDB.getInstance().getConnection();
    int key = 0;
    String sql = " INSERT INTO CUSTOMER(cno, id, pw, name, birth, tel) ";
    sql += " VALUES(?, ?, ?, ?, ?, ?)";
    try {
      new ConnectDB(); // DriverManager.getConnection 관련 초기화 코드
```

```
      pstmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEY
S);
      pstmt.setInt(1, 0);
      pstmt.setString(2, ct.getId());
      pstmt.setString(3, ct.getPw());
      pstmt.setString(4, ct.getName());
      pstmt.setString(5, ct.getBirth());
      pstmt.setString(6, ct.getTel());
      key = pstmt.executeUpdate();
      rs = pstmt.getGeneratedKeys(); // 쿼리 실행 후 생성된 키 값 반환
      if (rs.next()) {
        if (key != 0) {
          key = rs.getInt(1); // 키값 초기화
          System.out.println("[" + ct.getName() + "]님,회원가입에 성공하였습니
다.");
          System.out.println("*****회원정보와 고객번호를 확인하세요.*****");
          System.out.println("┃NO :" + key +" "+ "┃ID :" + ct.getId() + "
" + "┃PW :" + ct.getPw() + " " + "┃이름 :"
              + ct.getName() + " " + "┃생년월일 :" + ct.getBirth() + " " +
 "┃연락처 :" + ct.getTel() + " ");
          System.out.println(">> 로그인 화면으로 이동합니다.");
        } else {
          System.out.println("[회원가입 실패 : 다시 시도해 주세요.]");
        }
      }
      String msg = key > -1 ? "성공" : "실패";
      System.out.println(msg);
      return key;
    } catch (SQLException e) {
      System.out.println("[로그인 실패 : " + e + "]");
      key = -1;
    }
    return key;
  }

  // 아이디 체크
  public boolean chkid(String id) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "SELECT ID FROM CUSTOMER WHERE ID=?";
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, id);
      rs = pstmt.executeQuery();
      if (rs.next()) {
        int key = rs.getInt(1);
        if (key > 0) {
          return true;
        }
      }
    } catch (SQLException e) {
      System.out.println("[아이디 체크 실패 : " + e + "]");
      e.printStackTrace();
    }
    return false;
  }

  // 연락처 체크
  public boolean chktel(String tel) {
```

```
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "SELECT TEL FROM CUSTOMER WHERE TEL=?";
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, tel);
      rs = pstmt.executeQuery();
      if (rs.next()) {
        int key = rs.getInt(1);
        if (key > 0) {
          return true;
        }
      }
    } catch (SQLException e) {
      System.out.println("[연락처 체크 실패 : " + e + "]");
    }
    return false;
  }

  // 로그인
  public boolean login(Customer ct) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "SELECT * FROM CUSTOMER WHERE ID=? AND PW=?";
    String id = ct.getId();
    String pw = ct.getPw();
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, id);
      pstmt.setString(2, pw);
      rs = pstmt.executeQuery();
      if (rs.next()) {
        if ((rs.getString("id").equals(id)) == true && rs.getString("p
w").equals(pw)) {
          ct.setCno(rs.getInt(1));
          ct.setName(rs.getString("name"));
          ct.setBirth(rs.getString("birth"));
          ct.setTel(rs.getString("tel"));
          Session.put(id, ct); // 세션에 정보 저장
          return true;
        } else {
          System.out.println("[로그인 실패 | 비밀번호가 일치하지 않습니다.]");
          return false;
        }
      }
    } catch (SQLException e) {
      System.out.println("[로그인  실패 | " + e + "]");
      e.printStackTrace();
    } finally {
      try {
        if (pstmt != null)
          pstmt.close();
      } catch (Exception e2) {
      }
    }
    return false;
  }

}
```

▼ CheckManager

```
package bank;

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

public class CheckManager {
  PreparedStatement pstmt;
  static Statement stmt;
  static ResultSet rs;
  Manager mg;
  Customer ct;

  public CheckManager() {

  }
  // 관리자 정보 입력
  public int join(Manager mg) {
    Connection conn = ConnectDB.getInstance().getConnection();
    int key = 0;
    String sql = " INSERT INTO MANAGER(mno, mid, mpw) ";
    sql += " VALUES(?, ?, ?)";
    try {
      new ConnectDB(); // DriverManager.getConnemgion 관련 초기화 코드
      pstmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEY
S);
      pstmt.setInt(1, 0);
      pstmt.setString(2, mg.getMid());
      pstmt.setString(3, mg.getMpw());
      key = pstmt.executeUpdate();
      rs = pstmt.getGeneratedKeys(); // 쿼리 실행 후 생성된 키 값 반환
      if (rs.next()) {
        key = rs.getInt(1); // 키값 초기화
        System.out.println(" 회원번호 : " + key); // 출력
      }
      String msg = key > -1 ? "성공" : "실패";
      System.out.println(msg);
      return key;
    } catch (SQLException e) {
      System.out.println("[회원가입 실패 : " + e + "]");
      key = -1;
    }
    return key;
  }

  // 아이디 체크
  public boolean chkmid(String mid) {
    Connection conn = ConnectDB.getInstance().getConnection();
```

```
    String sql = "SELECT MID FROM MANAGER WHERE MID=?";
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, mid);
      rs = pstmt.executeQuery();
      if (rs.next()) {
        int key = rs.getInt(1);
        if (key > 0) {
          return true;
        }
      }
    } catch (SQLException e) {
      System.out.println("[비밀번호 체크 실패 : " + e + "]");
    }
    return false;
  }

  // 관리자 로그인
  public boolean login(Manager mg) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "SELECT * FROM MANAGER WHERE MID=? AND MPW=?";
    String mid = mg.getMid();
    String mpw = mg.getMpw();
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, mid);
      pstmt.setString(2, mpw);
      rs = pstmt.executeQuery();
      if (rs.next()) {
        if ((rs.getString("mid").equals(mid)) == true && rs.getString("mp
w").equals(mpw)) {
          mg = new Manager(1, mg.getMid(), mg.getMpw());
          mg.setMno(rs.getInt(1));
          Session.put(mid, mg);
          return true;
        } else {
          System.out.println("[로그인 실패 | 비밀번호가 일치하지 않습니다.]");
          return false;
        }
      }
    } catch (SQLException e) {
      System.out.println("[로그인  실패 | " + e + "]");
      e.printStackTrace();
    }
    return false;
  }

  // 저장된 회원 목록
  public List<Customer> customerlist() {
    List<Customer> ctlist = new ArrayList<Customer>();
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "SELECT * FROM CUSTOMER ORDER BY CNO";
    try {
      pstmt = conn.prepareStatement(sql);
      rs = pstmt.executeQuery();
      // 5. ResultSet에 저장된 데이터 얻기
      while (rs.next()) {
        // Customer객체에 저장
```

```
        ct = new Customer();
        ct.setCno(rs.getInt(1));
        ct.setId(rs.getString("id"));
        ct.setPw(rs.getString("pw"));
        ct.setName(rs.getString("name"));
        ct.setBirth(rs.getString("birth"));
        ct.setTel(rs.getString("tel"));

        // 리스트에 추가
        ctlist.add(ct);

      }
    } catch (SQLException e) {
      System.out.println("[정보 불러오기 실패 : " + e.getMessage() + "]");
    }
    return ctlist;
  }

  // 회원삭제
  public boolean delete(String id) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "DELETE FROM CUSTOMER WHERE ID=?";
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, id);
      int key = pstmt.executeUpdate();
      if (key > 0) {
        System.out.println("[" + id + "]회원 정보 삭제 완료");
        return true;
      } else {
        System.out.println("[" + id + "]회원 정보 삭제 실패");
        return false;
      }
    } catch (Exception e) {
      System.out.println("예외발생: " + e.getMessage());
    }
    return false;
  }

}
```

▼ CheckAccount

```
package bank;

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

public class CheckAccount {
```

```
  private Connection conn;
  private Statement stmt;
  private static PreparedStatement pstmt;
  private ResultSet rs;
  private Customer ct = new Customer();
  static String id = CustomerHandler.id2;

  // 계좌 정보 계좌번호로 불러오기
  public void getAccount(String account) {
    Connection conn = ConnectDB.getInstance().getConnection();
    List<Account> aclist = new ArrayList<>();
    String sql = "SELECT ANO,ACCOUNT,ACCOUNTPW,BALANCE,CNO FROM ACCOUNT W
HERE ACCOUNT=?";
    try {
      /// 3. Statement 객체 생성
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1,account);
      // 4. SQL 문장을 실행하고 결과를 리턴
      rs = pstmt.executeQuery();
      if (rs == null) {
        System.out.println("저장된 데이터가 없습니다.");
      } else {
        while (rs.next()) {
          Account ac = new Account();
          int ano = rs.getInt(1);
          String accountpw = rs.getString("accountpw");
          int balance = rs.getInt("balance");
          int cno = rs.getInt("cno");
          ac = new Account(ano, account, accountpw, balance, cno);

          Session.put(account, ac);
          aclist.add(ac);

        }
        // 결과물 출력
        for (Account ac : aclist) {
          System.out.println("No. : " + ac.getAno() + "\t" + "계좌번호 : "
 + ac.getAccount() + "\t" + "계좌 비밀번호 : "
              + ac.getAccountpw() + "\t" + "잔고 : " + ac.getBalance() +
 "\t" + "고객번호 : " + ac.getCno() + "\t");
        }
      }
    } catch (SQLException e) {
      System.out.println("[정보 조회 실패 : " + e.getMessage() + "]");
    } finally {
      try {
        if (stmt != null)
          stmt.close();
      } catch (Exception e) {
      }
    }
  }

  // 계좌 정보 회원번호로 불러오기
    public void showAccount(int cno) {
      Connection conn = ConnectDB.getInstance().getConnection();
      List<Account> aclist = new ArrayList<>();
      String sql = "SELECT ANO,ACCOUNT,ACCOUNTPW,BALANCE FROM ACCOUNT WHE
```

```
RE CNO=?";
      try {
        /// 3. Statement 객체 생성
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1,cno);
        // 4. SQL 문장을 실행하고 결과를 리턴
        rs = pstmt.executeQuery();
        if (rs == null) {
          System.out.println("저장된 데이터가 없습니다.");
        } else {
          while (rs.next()) {
            Account ac = new Account();
            int ano = rs.getInt(1);
            String account = rs.getString("account");
            String accountpw = rs.getString("accountpw");
            int balance = rs.getInt("balance");
            ac = new Account(ano, account, accountpw, balance, cno);

            aclist.add(ac);
          }
        }
        // 결과물 출력
        System.out.println("                              [계좌정보]");
        System.out.println("=============================================
============================");
        for (Account ac : aclist) {
          System.out.println("No. : " + ac.getAno() + "\t" + "계좌번호 : "
 + ac.getAccount() + "\t" + "계좌 비밀번호 : "
              + ac.getAccountpw() + "\t" + "잔고 : " + ac.getBalance() +
 "\t");
        }
        System.out.println("=============================================
============================");
      } catch (SQLException e) {
        System.out.println("[정보 조회 실패 : " + e.getMessage() + "]");
      } finally {
        try {
          if (stmt != null)
            stmt.close();
        } catch (Exception e) {
        }
      }
    }

  // 계좌 정보 입력
  public int join(Account ac) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String id = CustomerHandler.id2;
    int key = 0;
    String sql = "INSERT INTO ACCOUNT(ano,account,accountpw,balance,cno)
 ";
    sql += " VALUES(?, ?, ?, ?, ?)";
    try {
      new ConnectDB(); // DriverManager.getConnection 관련 초기화 코드
      pstmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEY
S);
      pstmt.setInt(1, 0);
      pstmt.setString(2, ac.getAccount());
```

```
      pstmt.setString(3, ac.getAccountpw());
      pstmt.setInt(4, ac.getBalance());
      pstmt.setInt(5, Session.get1(id).getCno());
      key = pstmt.executeUpdate();
      rs = pstmt.getGeneratedKeys(); // 쿼리 실행 후 생성된 키 값 반환
      if (rs.next()) {
        key = rs.getInt(1); // 키값 초기화
        System.out.println(" 회원(계좌)번호 : " + key); // 출력
      }
      String msg = key > -1 ? "성공" : "실패";
      System.out.println(msg);
      return key;
    } catch (SQLException e) {
      System.out.println("[회원가입 실패 : " + e + "]");
      e.printStackTrace();
      key = -1;
    } finally {
      try {
        if (rs != null)
          rs.close();
        if (pstmt != null)
          pstmt.close();
      } catch (SQLException e) {
        System.out.println("[로그인 실패 : " + e + "]");
      }
    }
    return key;
  }

  // 계좌 체크
  public boolean chkac(String account) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "SELECT ACCOUNT FROM ACCOUNT WHERE ACCOUNT=?";
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, account);
      rs = pstmt.executeQuery();
      if (rs.next()) {
        String key = rs.getString("account");
        if (key != null) {
          return true;
        }
      }
    } catch (SQLException e) {
      System.out.println("계좌 체크 실패 > " + e);
      e.printStackTrace();
    }
    return false;
  }

  // 회원 수정(잔고)
      public static void updateAccount(String account, int balance) {
        Scanner sc = new Scanner(System.in);
        Connection conn = ConnectDB.getInstance().getConnection();
        String sql = "UPDATE ACCOUNT SET BALANCE=? where ACCOUNT=?";
        try {
          Account ac = Session.get3(account);
          pstmt = conn.prepareStatement(sql);
```

```
          pstmt.setInt(1, balance);
          pstmt.setString(2, account);
          int key = pstmt.executeUpdate();
          ac = new Account(ac.getAno(), account, ac.getAccountpw(), balan
ce, ac.getCno());
          //System.out.println("번호:"+ac.getAno()+"계좌:"+account+"비번:"+a
c.getAccountpw()+"잔고:"+balance+"고객번호:"+ac.getCno());
        } catch (Exception e) {
          System.out.println("예외발생: " + e.getMessage());
          e.printStackTrace();
        }
      }
//  // 회원정보 + 계좌정보
//  public void selectjoin(int cno) {
//
//    Connection conn = ConnectDB.getInstance().getConnection();
//    String sql = "SELECT C.CNO,C.ID,C.PW,C.NAME,C.BIRTH,C.TEL,A.ANO,A.A
CCOUNT,A.ACCOUNTPW,A.BALANCE ";
//    sql += "FROM CUSTOMER C JOIN ACCOUNT A";
//    sql += "ON C.CNO = A.CNO WHERE C.ID=" + id + "";
//    try {
//      stmt = conn.createStatement();
//      // pstmt.setInt(1, cno);
//      rs = stmt.executeQuery(sql);
//      if (rs == null) {
//        System.out.println("저장된 데이터가 없습니다.");
//      } else {
//        if (rs.next()) {
//          AccountDTO dto = new AccountDTO();
//          // cno = rs.getInt("cno");
//          String id = rs.getString("id");
//          String pw = rs.getString("pw");
//          String name = rs.getString("name");
//          String bitrh = rs.getString("birth");
//          String tel = rs.getString("tel");
//
//          int ano = rs.getInt("ano");
//          String account = rs.getString("account");
//          String accountpw = rs.getString("accountpw");
//          int balance = rs.getInt("balance");
//          dto = new AccountDTO(dto.getCno(), dto.getId(), dto.getPw(),
 dto.getName(), dto.getBirth(),
//              dto.getTel(), dto.getAno(), dto.getAccount(), dto.getAcco
untpw(), dto.getBalance());
//
//          System.out.println(cno + "\t" + id + "\t" + pw + "\t" + name
 + "\t" + bitrh + "\t" + tel + "\t"
//              + accountpw + "\t" + balance);
//
//        } else {
//          System.out.println("[정보조회  실패 | 정보가 일치하지 않습니다.]");
//        }
//      }
//    } catch (SQLException e) {
//      System.out.println("[정보조회  실패 | " + e + "]");
//      e.printStackTrace();
//    } finally {
//      try {
```

```
//        if (rs != null)
//          rs.close();
//        if (stmt != null)
//          stmt.close();
//      } catch (Exception e) {
//        System.out.println("[정보조회  실패 | " + e + "]");
//      }
//    }
//  }

  // 회원 수정(비밀번호)
  public static void updatepw(String id, String pw) {
    Connection conn = ConnectDB.getInstance().getConnection();
    Scanner sc = new Scanner(System.in);
    String sql = "UPDATE CUSTOMER SET PW=? where id=?";
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, pw);
      pstmt.setString(2, id);
      int key = pstmt.executeUpdate();
      System.out.println(Session.get1(id).getName() + "님의 정보를 수정하였습
니다.");
    } catch (Exception e) {
      System.out.println(Session.get1(id).getName() + "님의 정보 수정하지 못하
였습니다.");
      System.out.println("예외발생: " + e.getMessage());
      e.printStackTrace();
    } finally {
      System.out.println("[변경된 패스워드 : " + pw + "]");
    }
  }

  // 회원 수정(연락처)
  public static void updatetel(String id, String tel) {
    Scanner sc = new Scanner(System.in);
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "UPDATE CUSTOMER SET TEL=? where id=?";
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, tel);
      pstmt.setString(2, id);
      int key = pstmt.executeUpdate();
      System.out.println(Session.get1(id).getName() + "님의 정보를 수정하였습
니다.");
    } catch (Exception e) {
      System.out.println(Session.get1(id).getName() + "님의 정보 수정하지 못하
였습니다.");
      System.out.println("예외발생: " + e.getMessage());
      e.printStackTrace();
    } finally {
      System.out.println("[변경된 연락처 : " + tel + "]");
    }
  }

  // 계좌정보 삭제
  public boolean deleteAccount(String account) {
    Connection conn = ConnectDB.getInstance().getConnection();
    String sql = "DELETE FROM ACCOUNT WHERE ACCOUNT=?";
```

```
    int key = 0;
    try {
      pstmt = conn.prepareStatement(sql);
      pstmt.setString(1, account);
      key = pstmt.executeUpdate();
      if (key > 0) {
        System.out.println("[" + account + "계좌 정보 삭제 완료]");
        return true;
      } else {
        System.out.println("[" + account + "계좌 정보 삭제 실패]");
        return false;
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
    return false;
  }
}
```

- 기능구현 클래스(View)

▼ Menu

```
package bank;

import java.util.Scanner;

public class Menu {
  static Scanner sc = new Scanner(System.in); // 입력
  static boolean run = true; // 실행
  static Customer ct = new Customer();
  static CheckAccount ca = new CheckAccount();
  static Account ac = new Account();
  static Manager mg;
  static CheckCustomer cc;
  static CheckManager cm;
  static String account2;

  public Menu() {

    while (true) {
      System.out.println("**BANK MAIN**");
      System.out.println("------------------------");
      System.out.println("1.회원 | 2.관리자 |0.종료");
      System.out.println("------------------------");
      System.out.print("번호선택 : ");
      int num = sc.nextInt(); // 변수 입력
      switch (num) { // 변수 선택
      case 1:
        run = true;
        while (run) {
          System.out.println("**CUSTMOER MAIN**");
          System.out.println("------------------------");
          System.out.println("1.회원가입 | 2.로그인 | 0.종료");
          System.out.println("------------------------");
```

```
          System.out.print("번호선택 : ");
          num = sc.nextInt(); // 변수 입력
          switch (num) { // 변수 선택
          case 1:
            CustomerHandler.join();
            break;
          case 2:
            CustomerHandler.login();
            break;
          case 0:
            run = false;
            System.out.println("***coming back to the main***");
            break;
          default:
            System.out.println("알수없는 입력입니다.");
          }
        }
        break;
      case 2:
        run = true;
        ManagerHandler.managerconnect();
        while (run) {
          System.out.println("**MANAGER MAIN**");
          System.out.println("------------------");
          System.out.println("1.회원가입 | 2.로그인 | 0.종료");
          System.out.println("------------------");
          System.out.print("번호선택 : ");
          num = sc.nextInt(); // 변수 입력
          switch (num) { // 변수 선택
          case 1:
            ManagerHandler.join();
            break;
          case 2:
            ManagerHandler.login();
            break;
          case 0:
            run = false;
            System.out.println("***coming back to the main***");
            break;
          default:
            System.out.println("알수없는 입력입니다.");
          }
        }
        break;
      case 0:
        System.out.println("***프로그램 종료***");
        System.exit(0);
        break;
      default:
        System.out.println("알수없는 입력입니다.");
      }
    }
  }

  public static void Menuchk() {
    String id = CustomerHandler.id2; // 아이디 넘기기
    while (run) {
      try {
```

```
        if (Session.get1(id) == null) {
          System.out.println("[로그인 후 이용하세요.]");
          ManagerHandler.login();
        } else {
          ca.showAccount(Session.get1(id).getCno());
          System.out.print("계좌번호 입력(계좌가 없을 경우 아무번호나 입력해주세
요.):");
          String account = sc.next();
          // 계좌 확인;
          if (ca.chkac(account)) {
            System.out.println("                              ["+Session.ge
t1(id).getName()+"님의 계좌정보]");
            System.out.println("==========================================
================================");
            System.out.println("reg.No\t  계좌번호 \t\t계좌비밀번호\t\t잔고\t\t
고객번호");
            System.out.println("==========================================
================================");
            ca.getAccount(account);
            System.out.println("==========================================
================================");
            account2 = Session.get3(account).getAccount(); // account값 넘
기기
            System.out.println("*****'" + Session.get1(id).getName() +
 "'" + "님 은행에 오신 것을 환영합니다.*****");
            AccountMenu();
          } else {
            System.out.println("[" + Session.get1(id).getName() + "]님의 정
보가 없거나 일치하지 않습니다.");
            System.out.println("메뉴를 선택하세요>> |1.계좌개설 | 2.다시입력 | 3.
나가기 |");
            System.out.print("번호 입력 :");
            int num = sc.nextInt(); // 변수 입력
            switch (num) { // 변수 선택
            case 1:
              System.out.println("계좌개설로 이동>>");
              AccountHandler.addaccount();
              break;
            case 2:
              continue;
            case 3:
              run = false;
              break;
            }
          }
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
  }

  public static void AccountMenu() {
    while (run) {
      System.out.println("**MENU(ACCOUNT)**");
      System.out.println("----------------------------------------------
----------");
      System.out.println("1.계좌개설 | 2.입금 | 3.출금 | 4.거래내역 | 5.회원정보
```

```
수정 | 0.로그아웃");
      System.out.println("-----------------------------------------------
----------");
      System.out.print("번호선택 : ");
      int num = sc.nextInt(); // 변수 입력
      switch (num) { // 변수 선택
      case 1:
        AccountHandler.addaccount();
        break;
      case 2:
        AccountHandler.deposit();
        break;
      case 3:
        AccountHandler.withdraw();
        break;
      case 4:
        AccountHandler.transInfo();
        break;
      case 5:
        AccountHandler.updateInfo();
        break;
      case 0:
        AccountHandler.logout();
        run = false;
        System.out.println("***coming back to the main***");
        break;
      default:
        System.out.println("알수없는 입력입니다.");
      }

    }
  }

  public static void ManagerMenu() {
    String mid = ManagerHandler.mid2; // 아이디 넘기기
    while (run) {
      if (Session.get2(mid) == null) {
        System.out.println("[로그인 후 이용하세요.]");
        ManagerHandler.login();
      } else {
        System.out.println("**MENU(MANAGER)**");
        System.out.println("------------------------");
        System.out.println("1.회원내역 | 2.회원삭제 | 0.로그아웃");
        System.out.println("------------------------");
        System.out.print("번호선택 : ");
        int num = sc.nextInt(); // 변수 입력
        switch (num) { // 변수 선택
        case 1:
          ManagerHandler.showCustomerList();
          break;
        case 2:
          ManagerHandler.deletecustomer();
          break;
        case 0:
          ManagerHandler.managerlogout();
          run = false;
          System.out.println("***coming back to the main***");
          break;
```

```
        default:
          System.out.println("알수없는 입력입니다.");
        }
      }
    }

  }
}
```

▼ CustomerHandler

```
package bank;

import java.util.Scanner;

public class CustomerHandler {
  static boolean run = true; // 실행
  static Scanner sc = new Scanner(System.in);
  static Customer ct = new Customer(); // DB를 dto에 저장
  static CheckCustomer cc = new CheckCustomer();
  static String id, pw, name, birth, tel = null;
  static String id2; // 다른클래스로 넘길 아이디값

  public CustomerHandler() {
  }

  // 회원가입
  public static void join() {
    // ID
    while (run) {
      System.out.print("ID : ");
      ct.setId(sc.next());
      if (cc.chkid(ct.getId())) {
        System.out.println("[중복된 아이디입니다. 다시 입력하세요.]");
        continue;
      }
      if (ct.getId().length() < 5 || ct.getId().length() > 15) {
        System.out.println("5~15자 이내의 영어,숫자아이디만 가능합니다");
        continue;
      } else {
        int c1 = 0, c2 = 0;
        for (int i = 0; i < ct.getId().length(); i++) {
          char c = ct.getId().charAt(i);
          if ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z')) {
            c1++;
          }else if (c >= '0' && c <= '9') {
            c2++;
          }
        }
      }
      break;
    }
    // PW
    while (run) {
      System.out.print("PW : ");
```

```
        ct.setPw(sc.next());
        // 아스키코드로 숫자 유효성검사
        int i = 0;
        for (i = 0; i < ct.getPw().length(); i++) {
          char a = ct.getPw().charAt(i);
          if(!(a >= 48 && a<=57) || (a >= 96 && a <= 105)) {
            break;
          }
        }
        // 비밀번호 유효성 검사(4자리/공백/숫자/확인)
        if (i == ct.getPw().length()) {
          if (ct.getPw().length() != 4) {
            System.out.println("[패스워드는 4자리로 입력해주세요.]");
            continue;
          }
          System.out.print("PW 확인 : ");
          String pwchk = sc.next();
          if (ct.getPw().trim().isEmpty() || pwchk.trim().isEmpty()) {
            System.out.println("[패스워드 또는 패스워드 확인이 공백입니다.]");
            continue;
          } else if (!ct.getPw().equals(pwchk)) {
            System.out.println("[패스워드와 패스워드 확인이 일치하지 않습니다.]");
            continue;
          }
        } else {
          System.out.println("[숫자만 입력해주세요.]");
          continue;
        }
        break;
      }

      // 이름
      while (run) {
        System.out.print("이름 : ");
        ct.setName(sc.next());
        int i = 0;
        for (i = 0; i < ct.getName().length(); i++) {
          char a = ct.getName().charAt(i);
          if ((a >= 'a' && a <= 'z') || (a >= 'A' && a <= 'Z')) {
            break;
          }
        }
        if (i == ct.getName().length()) {
          break;
        } else {
          System.out.println("[한글만 입력해주세요.]");
        }
      }

      // 생년월일
      while (run) {
        System.out.print("생년월일(6자리): ");
        ct.setBirth(sc.next());
        // 아스키코드로 숫자 유효성검사
        int i = 0;
        for (i = 0; i < ct.getBirth().length(); i++) {
          char a = ct.getBirth().charAt(i);
          if(!(a >= 48 && a<=57) || (a >= 96 && a <= 105)) {
```

```
            break;
          }
        }
        // 생년월일 6자리 입력
        if (i == ct.getBirth().length()) {
          if (ct.getBirth().length() != 6) {
            System.out.println("[생년월일은 6자리로 입력하세요.]");
            continue;
          }
          break;
        } else {
          System.out.println("[숫자만 입력해주세요.]");
        }
      }

      // 연락처
      while (run) {
        String b, c = null;
        System.out.print("연락처(010제외 8자리 입력): ");
        ct.setTel(sc.next());
        // 아스키코드로 숫자 유효성검사
        int i = 0;
        for (i = 0; i < ct.getTel().length(); i++) {
          char a = ct.getTel().charAt(i);
          if(!(a >= 48 && a<=57) || (a >= 96 && a <= 105)) {
            break;
          }
        }
        // 연락처 8자리 입력
        if (i == ct.getTel().length()) { // for문 빠져나오도록 입력
          if (ct.getTel().length() != 8) {
            System.out.println("[연락처 8자리를 입력하세요.]");
            continue;
          }
          if (!cc.chktel(ct.getTel())) {
            // 연락처 유효성 검사
            b = ct.getTel().substring(0, 4);
            c = ct.getTel().substring(4, 8);
            ct.setTel("010-" + b + "-" + c);
            break;
          } else {
            System.out.println("[중복된 핸드폰 번호입니다.]");
          }
        } else {
          System.out.println("[숫자만 입력해주세요.]");
        }
      }
      System.out.println("휴대폰 번호 : " + ct.getTel());

      ct = new Customer(1, ct.getId(), ct.getPw(), ct.getName(), ct.getBirt
h(), ct.getTel());
      Session.map.put(id, ct);

      cc.join(ct);

    }

    // 로그인
```

```
  public static void login() {
    while (run) {
      try {
        System.out.println("|로그인|");
        System.out.print("ID : ");
        ct.setId(sc.next());
        System.out.print("PW : ");
        ct.setPw(sc.next());
        if (cc.login(ct) != false) {
          System.out.println("'" + ct.getName() + "'" + "님 환영합니다.");
          System.out.println("[로그인에 성공하였습니다.] \n");
          id2 = ct.getId(); // id값 넘기기
          break;
        } else {
          System.out.println("[로그인 실패했습니다.]");
          continue;
        }
      } catch (Exception e) {
        System.out.println("[로그인 실패 :" + e.getMessage() + "]");
        e.printStackTrace();
      }
    }
    Menu.Menuchk(); // 메뉴체크 이동
  }
}
```

▼ ManagerHandler

```
package bank;

import java.util.List;
import java.util.Scanner;

public class ManagerHandler {
  static boolean run = true; // 실행
  static Scanner sc = new Scanner(System.in);
  static Manager mg = new Manager(); // DB를 dto에 저장
  static CheckCustomer cc = new CheckCustomer();
  static CheckManager cm = new CheckManager();
  static Customer ct;
  static String mno, mid, mpw = null;
  static String mid2; // 다른클래스로 넘길 아이디값

  public ManagerHandler() {

  }

  // 관리자 접속 로그인 (비밀번호 1111)
  public static void managerconnect() {
    int cpw;
    while (run) {
      try {
        System.out.println("|관리자 모드 접속|");
        System.out.print("PW > ");
        cpw = sc.nextInt();
```

```
        if (cpw == 1111) {
          System.out.println("[접속 성공하였습니다.] \n");
          break;
        } else {
          System.out.println("[접속 실패했습니다.]");
          continue;
        }
      } catch (Exception e) {
        System.out.println("[접속 실패 :" + e.getMessage() + "]");
        e.printStackTrace();
      }
    }
  }

  // 회원가입
  public static void join() {
    // ID
    while (run) {
      System.out.print("ID : ");
      mg.setMid(sc.next());
      if (cm.chkmid(mg.getMid())) {
        System.out.println("[중복된 아이디입니다. 다시 입력하세요.]");
        continue;
      }
      break;
    }
    // PW
    while (run) {
      System.out.print("PW : ");
      mg.setMpw(sc.next());
      // 아스키코드로 숫자 유효성검사
      int i = 0;
      for (i = 0; i < mg.getMpw().length(); i++) {
        char a = mg.getMpw().charAt(i);
        if (a <= 48 || a >= 57) {
          break;
        }
      }
      // 비밀번호 유효성 검사(4자리/공백/숫자/확인)
      if (i == mg.getMpw().length()) {
        if (mg.getMpw().length() != 4) {
          System.out.println("[패스워드는 4자리로 입력해주세요.]");
          continue;
        }
        System.out.print("PW 확인 : ");
        String pwchk = sc.next();
        if (mg.getMpw().trim().isEmpty() || pwchk.trim().isEmpty()) {
          System.out.println("[패스워드 또는 패스워드 확인이 공백입니다.]");
          continue;
        } else if (!mg.getMpw().equals(pwchk)) {
          System.out.println("[패스워드와 패스워드 확인이 일치하지 않습니다.]");
          continue;
        }
      } else {
        System.out.println("[숫자만 입력해주세요.]");
        continue;
      }
      break;
```

```
      }

      mg = new Manager(1, mg.getMid(), mg.getMpw());
      Session.map2.put(mid, mg);

      int key = cm.join(mg);
      if (key != 0) {
        System.out.println("[관리자]님,회원가입에 성공하였습니다.");
        System.out.println(">> 로그인 화면으로 이동합니다.");
      } else {
        System.out.println("[회원가입 실패 : 다시 시도해 주세요.]");
      }
    }

    // 로그인
    public static void login() {
      while (run) {
        try {
          System.out.println("|로그인|");
          System.out.print("ID : ");
          mg.setMid(sc.next());
          System.out.print("PW : ");
          mg.setMpw(sc.next());
          if (cm.login(mg) != false) {
            System.out.println("['관리자' 로그인에 성공하였습니다.] \n");
            mid2 = mg.getMid(); // id값 넘기기
            break;
          } else {
            System.out.println("[로그인 실패했습니다.]");
            continue;
          }
        } catch (Exception e) {
          System.out.println("[로그인 실패 :" + e.getMessage() + "]");
          e.printStackTrace();
        }
      }
      Menu.ManagerMenu(); // 관리자메뉴 이동
    }

    // 저장된 회원 목록
    public static void showCustomerList() {
      List<Customer> list = cm.customerlist();
      System.out.println("                             Customer List");
      System.out.println("━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━");
      if (list != null && list.size() > 0) {
        System.out.println("reg.No\t  아이디 \t\t비밀번호\t\t생년월일\t\t연락처");
        System.out.println("━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━");

         for (Customer ct : list){
                    System.out.println(ct);
                }

      } else {
        System.out.println("저장된 데이터가 없습니다. ");
      }
      System.out.println("━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
```

```
──────────────────────────────────────────────────총 "
        + ((list == null) ? "0" : list.size()) + " 명=\n");
  }

  // 회원 삭제
  public static void deletecustomer() {
    while (run) {
      System.out.println("삭제할 회원의 아이디를 입력해주세요");
      String id = sc.next();
      cc.getCustomer(id);
      if (id != null) {
        System.out.println("해당 회원의 정보를 삭제하시겠습니까?(Y/N) : ");
        String input = sc.next();
        if (input.equalsIgnoreCase("y")) {
          boolean r = cm.delete(id);
          if (r) {
            System.out.println("["+ id +"]회원의 정보가 정상적으로 삭제되었습니
다.");
            break;
          } else {
            System.out.println("회원의 정보가 정상적으로 삭제 되지 않았습니다.");
            continue;
          }
        } else {
          System.out.println("삭제를 취소하였습니다.");
          continue;
        }
      } else {
        System.out.println("입력하신 회원번호에 해당하는 회원이 존재하지 않습니다.");
        continue;
      }

    }
  }
  // 로그아웃
    public static void managerlogout() {
      Session.map2.put(mid, null);
      System.out.println("[로그아웃되었습니다.]");
    }
}
```

▼ AccountHandler

```
package bank;

import java.util.ArrayList;
import java.util.InputMismatchException;
import java.util.List;
import java.util.Random;
import java.util.Scanner;

public class AccountHandler {
  static boolean run = true; // 실행
  static Account ac; // DB를 dto에 저장
  static Customer ct;
```

```
  static CheckAccount ca = new CheckAccount();
  static CheckCustomer cc = new CheckCustomer();
  static Transaction ts;
  static CheckManager cm;
  static Scanner sc = new Scanner(System.in);
  static Random r = new Random(); // 계좌번호 랜덤 생성
  static String id = CustomerHandler.id2; // 로그인에서 아이디 값 넘김
  //static String account = Menu.account2; // 계좌인증에서 계좌번호 넘김
  static int commission, money = 0;
  static List<Transaction> tslist = new ArrayList<>();

  public AccountHandler() {
  }

  // 계좌 개설
  public static void addaccount() {
    ac = new Account();
    // 계좌번호
    // 9자리 랜덤 계좌번호 생성
    while (run) {
      ac.setAccount(r.nextInt(999999999) + 1 + "");
      for (int i = 0; i < 9 - ac.getAccount().length(); i++) {
        break;
      }
      if (ac.getAccount().length() == 9) {
        String e = ac.getAccount().substring(0, 3);
        String f = ac.getAccount().substring(3, 9);
        ac.setAccount("1002-" + e + "-" + f);
        // 계좌번호 유효성 검사
        if (!ca.chkac(ac.getAccount())) {
          break;
        } else {
          System.out.println("[중복된 계좌 번호입니다.]");
          continue;
        }
      }
    }
    System.out.println(Session.get1(id).getName() + "님의 계좌번호 : " + ac.
getAccount());

    // 계좌 비밀번호
    while (run) {
      System.out.print("ACCOUNT PW : ");
      ac.setAccountpw(sc.next());
      // 아스키코드로 숫자 유효성검사
      int i = 0;
      for (i = 0; i < ac.getAccountpw().length(); i++) {
        char a = ac.getAccountpw().charAt(i);
        if (a <= 48 || a >= 57) {
          break;
        }
      }
      // 비밀번호 유효성 검사(4자리/공백/숫자/확인)
      if (i == ac.getAccountpw().length()) {
        if (ac.getAccountpw().length() != 4) {
          System.out.println("[패스워드는 4자리로 입력해주세요.]");
          continue;
        }
```

```
        System.out.print("ACCOUNT PW 확인 : ");
        String pwchk = sc.next();
        if (ac.getAccountpw().trim().isEmpty() || pwchk.trim().isEmpty())
{
          System.out.println("[패스워드 또는 패스워드 확인이 공백입니다.]");
          continue;
        } else if (!ac.getAccountpw().equals(pwchk)) {
          System.out.println("[패스워드와 패스워드 확인이 일치하지 않습니다.]");
          continue;
        }
      } else {
        System.out.println("[숫자만 입력해주세요.]");
        continue;
      }
      break;
    }
    // 잔고
    while (run) {
      System.out.print("입금액 :");
      ac.setBalance(sc.nextInt());
      try {
        if (ac.getBalance() < 10000) {
          System.out.println("[10,000원이상의 금액을 입금하셔야합니다.]");
          continue;
        }
        break;
      } catch (Exception e) {
        System.out.println("다시 입력해 주세요.");
        continue;
      }
    }

    ac = new Account(1, ac.getAccount(), ac.getAccountpw(), ac.getBalance
(), ac.getCno());
    Session.put(ac.getAccount(), ac);

    int key = ca.join(ac);
    if (key != 0) {
      System.out.println("[" + Session.get1(id).getName() + "]님,계좌개설에
 성공하였습니다.");
      Menu.Menuchk();
    } else {
      System.out.println("[계좌개설 실패 : 다시 시도해 주세요.]");
    }
  }

  // 입금
  public static void deposit() {
    ac = Session.get3(Menu.account2);
    int amount = 0;
    while (run) {
      try {
        System.out.println("[계좌 비밀번호 인증]");
        System.out.println("계좌번호 : " + ac.getAccount());
        System.out.print("계좌비밀번호 : ");
        String accountpw = sc.next();
        // System.out.println(Session.get3(account));
        // 계좌번호와 비밀번호 확인
```

```
        if (ac.getAccountpw().equals(accountpw)) {
          System.out.println("[인증 성공]");
        } else {
          System.out.println("[인증 실패]");
          continue;
        }
        // 입금
        try {
          System.out.print("입금액 : ");
          // 해당 객체가 Member의 자식 객체면 재정의된 메소드로 실행된다.
          amount = sc.nextInt();
          int balance = ac.getBalance();
          balance += amount;
          ac.setBalance(balance);
          savelog(true, ac, amount, 0);
          ca.updateAccount(ac.getAccount(), balance); // 잔고수정
          System.out.println("입금 되었습니다.");
          break;
        } catch (Exception e) {
          System.out.println("[입금 실패 : 정보 오류]");
          e.printStackTrace();
          continue;
        }
      } catch (InputMismatchException e) {
        System.out.println("[다시 입력해주세요.]");
      }

    }
    // 입금 영수증
    System.out.println("");
    System.out.println("  ━━━━* receipt *━━━━");
    System.out.println("   계좌번호 : " + ac.getAccount());
    System.out.println("      성함 : " + Session.get1(id).getName());
    System.out.println("      입금액 : " + amount + "원");
    System.out.println("      잔액 : " + ac.getBalance() + "원");
    System.out.println("  ━━━━━━━━━━━━━━━━━━\n");
    Menu.AccountMenu();
  }

  // 출금
  public static void withdraw() {
    ac = Session.get3(Menu.account2);
    int amount = 0;
    while (run) {
      try {
        System.out.println("[계좌 비밀번호 인증]");
        System.out.println("계좌번호 : " + ac.getAccount());
        System.out.print("비밀번호 : ");
        String accountpw = sc.next();
        // 계좌번호와 비밀번호 확인;
        if (Session.get3(ac.getAccount()).getAccountpw().equals(accountp
w)) {
          System.out.println("[인증 성공]");
        } else {
          System.out.println("[인증 실패]");
          continue;
        }
        // 출금
```

```
        try {
          System.out.print("출금액 : ");
          amount = sc.nextInt();
          // 출금액이 음수이거나 잔고보다 많으면 출금 실패
          if (amount < 0 || amount > Session.get3(ac.getAccount()).getBal
ance()) {
            System.out.println("[출금 실패 : 수수료를 합산한 잔액이 부족합니
다.]");
            continue;
          }
          // 잔고가 0이면 출금 실패
          if (Session.get3(ac.getAccount()).getBalance() <= 0) {
            System.out.println("[출금 실패 : 다시 입력하세요.]");
            continue;
          }
          commission = (int) (amount * 0.1); // 수수료10%
          money = amount + commission; // 잔고 + 수수료 = 출금액
          int balance = Session.get3(ac.getAccount()).getBalance();
          balance -= money;
          ac.setBalance(balance);
          savelog(false, ac, amount, commission);
          ca.updateAccount(ac.getAccount(), balance); // 잔고수정
          System.out.println("출금 되었습니다.");
          break;
        } catch (Exception e) {
          System.out.println("[출금 실패 : 정보 오류]");
          e.printStackTrace();
          continue;
        }
      } catch (InputMismatchException e) {
        System.out.println("[다시 입력해주세요.]");
      }
    }
    // 출금 영수증
    System.out.println("");
    System.out.println("  ━━━━* receipt *━━━━");
    System.out.println("   계좌번호 : " + ac.getAccount());
    System.out.println("      성함 : " + Session.get1(id).getName());
    System.out.println("      출금액 : " + amount + "원");
    System.out.println("      수수료 : " + commission + "원");
    System.out.println("      잔액 : " + ac.getBalance() + "원");
    System.out.println("  ━━━━━━━━━━━━━━━━━━\n");
    Menu.AccountMenu();
  }

  // 거래내역
  public static void transInfo() {
    System.out.println("[" + Session.get1(id).getName() + "님의"+ac.getAcc
ount()+"계좌 거래내역]");
    displaylog();
  }

  // 회원정보 조회 및 변경
  public static void updateInfo() {
    boolean run = true; // 실행
    String id = CustomerHandler.id2;
    while (run) { // 실행하는 동안(while)
      // InputMismatchException 예외처리
```

```
      try {
        System.out.println("**메뉴 선택*");
        System.out.println("1.회원정보 조회 | 2.회원비밀번호 변경 | 3.연락처 변경
 | 4.계좌삭제 | 0.종료 ");
        System.out.print("번호선택 : ");
        int num = sc.nextInt(); // 변수 입력

        switch (num) { // 변수 선택
        case 1:
          cc = new CheckCustomer();
          ca = new CheckAccount();
          while (run) {
            cc.showCustomer(Session.get1(id).getCno());
            ca.showAccount(Session.get1(id).getCno());
            break;
          }
          break;
        case 2:
          cc.showCustomer(Session.get1(id).getCno());
          while (run) {
            // 회원정보 출력
            System.out.println("새로운 패스워드를 입력하세요.");
            System.out.print("PW : ");
            String pw = sc.next();
            // 아스키코드로 숫자 유효성검사
            int i = 0;
            for (i = 0; i < pw.length(); i++) {
              char a = pw.charAt(i);
              if (a <= 48 || a >= 57) {
                break;
              }
            }
            if (i == pw.length()) {
              if (i == pw.length()) {
                if (pw.length() != 4) {
                  System.out.println("[패스워드는 4자리로 입력해주세요.]");
                  continue;
                }
              }
              System.out.print("PW 확인 : ");
              String pwchk = sc.next();
              if (pw.trim().isEmpty() || pwchk.trim().isEmpty()) {
                System.out.println("[패스워드 또는 패스워드 확인이 공백입니다.]");
                continue;
              } else if (!pw.equals(pwchk)) {
                System.out.println("[패스워드와 패스워드 확인이 일치하지 않습니
다.]");
                continue;
              }
            } else {
              System.out.println("[숫자만 입력해주세요.]");
              continue;
            }
            // 비밀번호 변경
            ca.updatepw(id, pw);
            break;
          }
          continue;
```

```
        case 3:
          cc.showCustomer(Session.get1(id).getCno());
          while (run) {
            System.out.println("새로운 연락처를 입력하세요.");
            System.out.print("연락처(010제외 8자리 입력): ");
            String tel = sc.next();
            // 아스키코드로 숫자 유효성검사
            int i = 0;
            for (i = 0; i < tel.length(); i++) {
              char a = tel.charAt(i);
              if (a <= 48 || a >= 57) {
                break;
              }
            }
            // 연락처 8자리 입력
            if (i == tel.length()) { // for문 빠져나오도록 입력
              if (tel.length() != 8) {
                System.out.println("[연락처 8자리를 입력하세요.]");
                continue;
              }
              // 연락처 유효성 검사
              if (!cc.chktel(tel)) {
                String b = tel.substring(0, 4);
                String c = tel.substring(4, 8);
                tel = "010-" + b + "-" + c;
              } else {
                System.out.println("[중복된 핸드폰 번호입니다.]");
                continue;
              }
            } else {
              System.out.println("[숫자만 입력해주세요.]");
              continue;
            }
            // 휴대폰 번호 변경
            ca.updatetel(id, tel);
            break;
          }
          continue;
        case 4:
          while (run) {
            ca.showAccount(Session.get1(id).getCno());
            System.out.print("삭제할 계좌 번호 : ");
            String account = sc.next();
            if (account != null) {
              System.out.println("해당 회원의 정보를 삭제하시겠습니까?(Y/N) : ");
              String input = sc.next();
              if (input.equalsIgnoreCase("y")) {
                boolean r = ca.deleteAccount(account);
                if (r) {
                  System.out.println("[" + Session.get1(id).getId() + "]회
원의 계좌정보가 정상적으로 삭제되었습니다.");
                  break;
                } else {
                  System.out.println("회원의 계좌정보가 정상적으로 삭제 되지 않았습
니다.");
                  continue;
                }
              } else {
```

```
                System.out.println("삭제를 취소하였습니다.");
                continue;
              }
            } else {
              System.out.println("입력하신 계좌번호에 해당하는 회원이 존재하지 않습
니다.");
              continue;
            }

          }
          continue;
        case 0:
          run = false;
          break;
        default:
          System.out.println("알수없는 입력입니다.");
        }
      } catch (InputMismatchException e) {
        System.out.println("[ERROR : 정수만 입력 가능합니다. 다시시작하세요]");
        System.exit(0);
      }
    }
  }

  // 거래내역 저장
  public static void savelog(boolean transType, Account ac, int amount, i
nt commission) {
    // List<Transaction> tslist = new ArrayList<>();
    DataSource ds = new DataSource();
    String date = ds.getYear() + ds.getMonth() + ds.getDate() + " " + ds.
getDay() + " " + ds.getCurTime();
    String transactionTime = date;
    if (ac != null) {
      ts = new Transaction(transType, amount, ac.getBalance(), commissio
n, transactionTime);
      Session.map4.put(Session.map3.get(Menu.account2).getAccount(), tsli
st);
      Session.map4.get(Session.map3.get(Menu.account2).getAccount()).add
(ts);
    }
  }

  // 거래내역 출력
  public static void displaylog() {
    try {
      for (Transaction ts : tslist) {
        if (ts.isTransType() == true) {
          System.out.println("|입금액 :" + ts.getAmount() + "원\t" + "|잔고
 :" + ts.getBalance() + "원\t" + "|수수료 :"
              + "없음" + "\t" + "|거래시간 :" + ts.getTransactionTime());
        } else {
          System.out.println("|출금액 :" + ts.getAmount() + "원\t" + "|잔고
 :" + ts.getBalance() + "원\t" + "|수수료 :"
              + ts.getCommission() + "원\t" + "|거래시간 :" + ts.getTransac
tionTime());
        }
      }
    } catch (NullPointerException e) {
```

```
        System.out.println("거래내역이 존재하지 않습니다.");
      }
    }

    // 로그아웃
    public static void logout() {
      Session.map.put(id, null);
      System.out.println("[로그아웃되었습니다.]");
    }
  }
```

- 기능구현
  - 메인페이지

```
DB 연결 성공
**BANK MAIN**
------------------------
1.회원 | 2.관리자 |0.종료
------------------------
번호선택 :
```

>> 1번 선택시 회원메인으로 이동

```
번호선택 : 1
**CUSTMOER MAIN**
------------------------
1.회원가입 | 2.로그인 | 0.종료
------------------------
번호선택 :
```

>> 2번 선택시 관리자 메인으로 이동

```
번호선택 : 2
|관리자 모드 접속|
PW > 1111
[접속 성공하였습니다.]

**MANAGER MAIN**
------------------
1.회원가입 | 2.로그인 | 0.종료
------------------
번호선택 :
```

💡 관리자 선택시 위와 같이 관리자모드 접속 비밀번호 [1111]을 입력 후 접속이 가능함.

>> 0번 선택시 프로그램 종료

```
**BANK MAIN**
-------------------------
1.회원 | 2.관리자 |0.종료
-------------------------
번호선택 : 0
***프로그램 종료***
```

- 관리자 메인
  - 회원가입

```
**MANAGER MAIN**
------------------
1.회원가입 | 2.로그인 | 0.종료
------------------
번호선택 : 1
ID : 1111
DB 연결 성공
[중복된 아이디입니다. 다시 입력하세요.]
ID : 1112
PW : 11
[패스워드는 4자리로 입력해주세요.]
PW : 11ㅇ
[숫자만 입력해주세요.]
PW : 1111
PW 확인 : 1111
DB 연결 성공
 관리자번호 : 6
성공
[관리자]님,회원가입에 성공하였습니다.
>> 로그인 화면으로 이동합니다.
```

💡 아이디 중복검사
비밀번호 유효성검사(숫자/4자리/비밀번호확인)
회원가입 성공시 관리자번호 자동생성후 화면이동

- 로그인

```
**MANAGER MAIN**
-------------------
1.회원가입 | 2.로그인 | 0.종료
-------------------
번호선택 : 2
|로그인|
ID : 22
PW : 2222
DB 연결 성공
[로그인 실패했습니다.]
|로그인|
ID : 1112
PW : 1111
['관리자' 로그인에 성공하였습니다.]
```

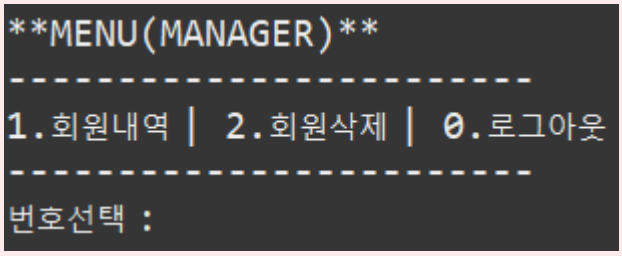
💡 DB에 저장된 관리자 계정정보와 중복검사

- 종료

```
**MANAGER MAIN**
-------------------
1.회원가입 | 2.로그인 | 0.종료
-------------------
번호선택 : 0
***coming back to the main***
**BANK MAIN**
-------------------------
1.회원 | 2.관리자 |0.종료
-------------------------
번호선택 :
```

종료시 메인페이지로 이동

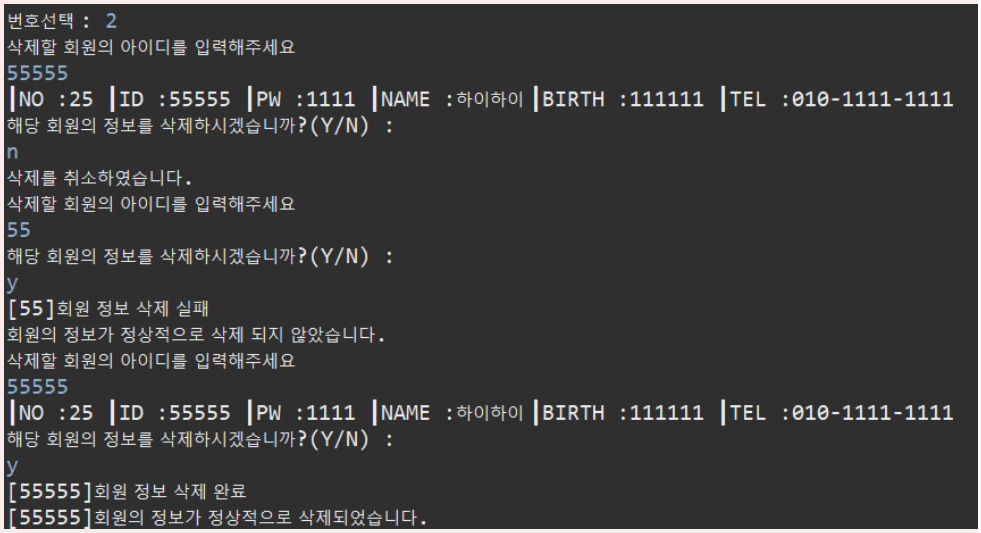
- 관리자 메뉴
  - 로그인 성공하면 관리자 메뉴로 이동



- 회원내역

회원내역 클릭시 가입되어있는 회원정보 모두 출력

- 회원삭제



회원아이디 일치 실패시 삭제 실패메세지 출력
삭제하시겠습니까?에서 Y를 제외한 문자 입력시 삭제실패
Y를 입력해야 삭제가능

- 로그아웃

```
**MENU(MANAGER)**
-------------------------
1.회원내역 | 2.회원삭제 | 0.로그아웃
-------------------------
번호선택 : 0
***coming back to the main***
```

로그아웃시 메인으로 이동

- 회원 메인
  - 회원가입

```
번호선택 : 1
ID : 11111
DB 연결 성공
[중복된 아이디입니다. 다시 입력하세요.]
ID : 11112
PW : 11111
[패스워드는 4자리로 입력해주세요.]
PW : 1111
PW 확인 : 1112
[패스워드와 패스워드 확인이 일치하지 않습니다.]
PW : 1111
PW 확인 : 1111
이름 : 하이이
생년월일(6자리): 111122
연락처(010제외 8자리 입력): 11111111
휴대폰 번호 : 010-1111-1111
DB 연결 성공
[하이이]님,회원가입에 성공하였습니다.
*****회원정보와 고객번호를 확인하세요.*****
|NO :26 |ID :11112 |PW :1111 |이름 :하이이 |생년월일 :111122 |연락처 :010-1111-1111
>> 로그인 화면으로 이동합니다.
```

💡 아이디중복검사 실행
비밀번호 유효성검사(4자리/비밀번호확인) 실행
생년월일 유효성검사(6자리) 실행
휴대폰번호 유효성검사(8자리) 실행
회원가입 성공시 회원정보 출력후 회원 메인으로 이동

- 로그인

```
번호선택 : 2
|로그인|
ID : 11111
PW : 11111
[로그인 실패했습니다.]
|로그인|
ID : 11111
PW : 1111
'김애나'님 환영합니다.
[로그인에 성공하였습니다.]

                                [계좌정보]
================================================================================
No. : 7 계좌번호 : 1002-448-661881 계좌 비밀번호 : 1111 잔고 : 34567
No. : 8 계좌번호 : 1002-284-605270 계좌 비밀번호 : 1111 잔고 : 11468
No. : 9 계좌번호 : 1002-427-726824 계좌 비밀번호 : 1111 잔고 : 24333
No. : 10        계좌번호 : 1002-211-632250 계좌 비밀번호 : 1111 잔고 : 151017
No. : 11        계좌번호 : 1002-492-118926 계좌 비밀번호 : 1111 잔고 : 115558
No. : 12        계좌번호 : 1002-693-218689 계좌 비밀번호 : 1111 잔고 : 12211
No. : 13        계좌번호 : 1002-720-198481 계좌 비밀번호 : 1111 잔고 : 12468
No. : 14        계좌번호 : 1002-770-444696 계좌 비밀번호 : 1111 잔고 : 112345
No. : 15        계좌번호 : 1002-602-942886 계좌 비밀번호 : 1111 잔고 : 12345
No. : 16        계좌번호 : 1002-685-273403 계좌 비밀번호 : 1111 잔고 : 1112308
No. : 17        계좌번호 : 1002-869-499289 계좌 비밀번호 : 1111 잔고 : 111123
No. : 19        계좌번호 : 1002-372-169901 계좌 비밀번호 : 1234 잔고 : 146222
No. : 20        계좌번호 : 1002-998-167068 계좌 비밀번호 : 2345 잔고 : 44779
No. : 28        계좌번호 : 1002-543-902648 계좌 비밀번호 : 1111 잔고 : 14323
No. : 31        계좌번호 : 1002-590-969441 계좌 비밀번호 : 1111 잔고 : 11029
No. : 32        계좌번호 : 1002-460-231386 계좌 비밀번호 : 1111 잔고 : 159293
No. : 33        계좌번호 : 1002-133-418371 계좌 비밀번호 : 1111 잔고 : 46402
================================================================================
```

💡 로그인시 중복검사 실행
로그인 성공시 회원의 계좌가 출력

>> 계좌번호 불일치 해당메뉴 출력

```
계좌번호 입력(계좌가 없을 경우 아무번호나 입력해주세요.):1
[김애나]님의 정보가 없거나 일치하지 않습니다.
메뉴를 선택하세요>> |1.계좌개설 | 2.다시입력 | 3.나가기 |
번호 입력 :2
```

>> 계좌번호 일치시 해당 계좌정보 출력후 회원메뉴이동

```
계좌번호 입력(계좌가 없을 경우 아무번호나 입력해주세요.):1002-133-418371
                              [김애나님의 계좌정보]
================================================================================
reg.No    계좌번호          계좌비밀번호                  잔고              고객번호
================================================================================
No. : 33        계좌번호 : 1002-133-418371 계좌 비밀번호 : 1111 잔고 : 46402      고객번호 : 14
================================================================================
*****'김애나'님 은행에 오신 것을 환영합니다.*****
```

- 회원 메뉴

```
**MENU(ACCOUNT)**
-------------------------
1.계좌개설 | 2.입금 | 3.출금 | 4.거래내역 | 5.회원정보수정 | 0.로그아웃
-------------------------
번호선택 :
```

  - 계좌개설

```
번호선택 : 1
김애나님의 계좌번호 : 1002-980-281076
ACCOUNT PW : 1111
ACCOUNT PW 확인 : 1111
입금액 :100
[10,000원이상의 금액을 입금하셔야합니다.]
입금액 :100000
DB 연결 성공
 회원(계좌)번호 : 36
성공
[김애나]님,계좌개설에 성공하였습니다.
```

계좌번호는 랜덤으로 생성,중복검사 실행
비밀번호 유효성검사(4자리/비밀번호 확인)
입금액 10,000원이상 미입금시 계좌개설 불가

- 입금

```
번호선택 : 2
[계좌 비밀번호 인증]
계좌번호 : 1002-634-588898
계좌비밀번호 : 1111
[인증 성공]
입금액 : 111
입금 되었습니다.

  ———* receipt *———
 계좌번호 : 1002-634-588898
   성함 : 김애나
   입금액 : 111원
   잔액 : 100011원
  ————————————————
```

입금시 계좌 비밀번호 인증 실행
인증 성공시 입금가능
입금 후에 영수증 출력(계좌번호,이름,입금액,잔액)

- 출금

```
번호선택 : 3
[계좌 비밀번호 인증]
계좌번호 : 1002-634-588898
비밀번호 : 1111
[인증 성공]
출금액 : 1000
출금 되었습니다.

  ———* receipt *———
 계좌번호 : 1002-634-588898
   성함 : 김애나
   출금액 : 1000원
   수수료 : 100원
   잔액 : 99900원
  ————————————————
```

💡 출금시 계좌 비밀번호 인증 실행
인증 성공시 출금가능
출금시 수수료 발생
출금후 영수증 출력(계좌번호,이름,출금액,수수료,잔액)

- 거래내역

```
번호선택 : 4
[김애나님의 거래내역]
|입금액 :11122원    |잔고 :126987원    |수수료 :없음       |거래시간 :2022.9.25 Sat 09시 51분 11초
|출금액 :42425원    |잔고 :80320원     |수수료 :4242원     |거래시간 :2022.9.25 Sat 09시 51분 18초
|입금액 :1111원     |잔고 :81431원     |수수료 :없음       |거래시간 :2022.9.25 Sat 09시 51분 29초
```

💡 boolean값 true시 입금액,잔고,거래시간 출력
boolean값 false시 출금액,잔고,수수료,거래시간 출력

- 회원정보수정

```
번호선택 : 5
**메뉴 선택*
1.회원정보 조회 | 2.비밀번호 변경 | 3.연락처 변경 | 4.계좌삭제 | 0.종료
번호선택 :
```

  - 회원정보 조회

```
번호선택 : 1
                        [회원정보]
=========================================================================
|No. : 14       |ID : 11111      |PW : 1111       |이름 : 김애나       |생일 : 111111     |연락처 : 010-1111-2222
=========================================================================
                        [계좌정보]
=========================================================================
No. : 7 계좌번호 : 1002-448-661881 계좌 비밀번호 : 1111 잔고 : 34567
No. : 8 계좌번호 : 1002-284-605270 계좌 비밀번호 : 1111 잔고 : 11468
No. : 9 계좌번호 : 1002-427-726824 계좌 비밀번호 : 1111 잔고 : 24333
No. : 10        계좌번호 : 1002-211-632250 계좌 비밀번호 : 1111 잔고 : 151017
No. : 11        계좌번호 : 1002-492-118926 계좌 비밀번호 : 1111 잔고 : 115558
No. : 12        계좌번호 : 1002-693-218689 계좌 비밀번호 : 1111 잔고 : 12211
No. : 13        계좌번호 : 1002-720-198481 계좌 비밀번호 : 1111 잔고 : 12468
No. : 14        계좌번호 : 1002-770-444696 계좌 비밀번호 : 1111 잔고 : 112345
No. : 15        계좌번호 : 1002-602-942886 계좌 비밀번호 : 1111 잔고 : 12345
No. : 16        계좌번호 : 1002-685-273403 계좌 비밀번호 : 1111 잔고 : 1112308
No. : 17        계좌번호 : 1002-869-499289 계좌 비밀번호 : 1111 잔고 : 111123
No. : 19        계좌번호 : 1002-372-169901 계좌 비밀번호 : 1234 잔고 : 146222
No. : 20        계좌번호 : 1002-998-167068 계좌 비밀번호 : 2345 잔고 : 44779
No. : 28        계좌번호 : 1002-543-902648 계좌 비밀번호 : 1111 잔고 : 14323
No. : 31        계좌번호 : 1002-590-969441 계좌 비밀번호 : 1111 잔고 : 11029
No. : 32        계좌번호 : 1002-460-231386 계좌 비밀번호 : 1111 잔고 : 159293
No. : 33        계좌번호 : 1002-133-418371 계좌 비밀번호 : 1111 잔고 : 46402
No. : 35        계좌번호 : 1002-634-588898 계좌 비밀번호 : 1111 잔고 : 100011
No. : 36        계좌번호 : 1002-980-281076 계좌 비밀번호 : 1111 잔고 : 100000
=========================================================================
```

회원의 회원정보 및 계좌정보 출력

- 비밀번호 수정

```
번호선택 : 2
                        [회원정보]
=========================================================================
|No. : 14       |ID : 11111     |PW : 1111      |이름 : 김애나      |생일 : 111111    |연락처 : 010-1111-2222
=========================================================================
새로운 패스워드를 입력하세요.
PW : 1234
PW 확인 : 1234
김애나님의 정보를 수정하였습니다.
[변경된 패스워드 : 1234]
```

회원정보 출력
새로운 비밀번호 및 비밀번호 확인 입력후 수정완료

- 연락처 수정

```
번호선택 : 3
                        [회원정보]
=========================================================================
|No. : 14       |ID : 11111     |PW : 1234      |이름 : 김애나      |생일 : 111111    |연락처 : 010-1111-2222
=========================================================================
새로운 연락처를 입력하세요.
연락처(010제외 8자리 입력): 11111111
김애나님의 정보를 수정하였습니다.
[변경된 연락처 : 010-1111-1111]
```

회원정보 출력
새로운 연락처 및 연락처확인 입력후 수정완료

- 계좌 삭제

```
                                [계좌정보]
========================================================================
No. : 7 계좌번호 : 1002-448-661881 계좌 비밀번호 : 1111 잔고 : 34567
No. : 8 계좌번호 : 1002-284-605270 계좌 비밀번호 : 1111 잔고 : 11468
No. : 9 계좌번호 : 1002-427-726824 계좌 비밀번호 : 1111 잔고 : 24333
No. : 10        계좌번호 : 1002-211-632250 계좌 비밀번호 : 1111 잔고 : 151017
No. : 11        계좌번호 : 1002-492-118926 계좌 비밀번호 : 1111 잔고 : 115558
No. : 12        계좌번호 : 1002-693-218689 계좌 비밀번호 : 1111 잔고 : 12211
No. : 13        계좌번호 : 1002-720-198481 계좌 비밀번호 : 1111 잔고 : 12468
No. : 14        계좌번호 : 1002-770-444696 계좌 비밀번호 : 1111 잔고 : 112345
No. : 15        계좌번호 : 1002-602-942886 계좌 비밀번호 : 1111 잔고 : 12345
No. : 16        계좌번호 : 1002-685-273403 계좌 비밀번호 : 1111 잔고 : 1112308
No. : 17        계좌번호 : 1002-869-499289 계좌 비밀번호 : 1111 잔고 : 111123
No. : 19        계좌번호 : 1002-372-169901 계좌 비밀번호 : 1234 잔고 : 146222
No. : 20        계좌번호 : 1002-998-167068 계좌 비밀번호 : 2345 잔고 : 44779
No. : 28        계좌번호 : 1002-543-902648 계좌 비밀번호 : 1111 잔고 : 14323
No. : 31        계좌번호 : 1002-590-969441 계좌 비밀번호 : 1111 잔고 : 11029
No. : 32        계좌번호 : 1002-460-231386 계좌 비밀번호 : 1111 잔고 : 159293
No. : 33        계좌번호 : 1002-133-418371 계좌 비밀번호 : 1111 잔고 : 46402
No. : 35        계좌번호 : 1002-634-588898 계좌 비밀번호 : 1111 잔고 : 100011
No. : 36        계좌번호 : 1002-980-281076 계좌 비밀번호 : 1111 잔고 : 100000
========================================================================
삭제할 계좌 번호 : 1002-980-281076
해당 회원의 정보를 삭제하시겠습니까?(Y/N) :
y
[1002-980-281076계좌 정보 삭제 완료]
[11111]회원의 계좌정보가 정상적으로 삭제되었습니다.
```

>> 계좌번호 불일치시 삭제 취소

```
삭제할 계좌 번호 : 1
해당 회원의 정보를 삭제하시겠습니까?(Y/N) :
ㅜ
삭제를 취소하였습니다.
```

회원의 계좌정보 출력후, 삭제할 계좌 입력
삭제시 Y를 누르면 삭제, 다른 문자를 입력하면 삭제 실패

- 종료

```
번호선택 : 0
**MENU(ACCOUNT)**
-------------------------
1.계좌개설 | 2.입금 | 3.출금 | 4.거래내역 | 5.회원정보수정 | 0.로그아웃
-------------------------
번호선택 :
```

- 결론
  - 고쳐야할 점

        - 변수의 약어 설정을 헷갈리지 않게 알아볼수 있는 의미로 설정해야된다.
        - 코드의 구현방식을 좀 더 고민하여 깔끔하고 쉽게 구현할 수 있도록 노력해야 할 것 같다.
    - 느낀점
        - 은행 프로그램 구현을 하면서 자바에 대한 이해도가 생겼고, 흐름의 중요성을 깨닫는 계기가 됐다.