JAVA 프로젝트

팀원 : 김정석, 이상민, 이효진, 최형진

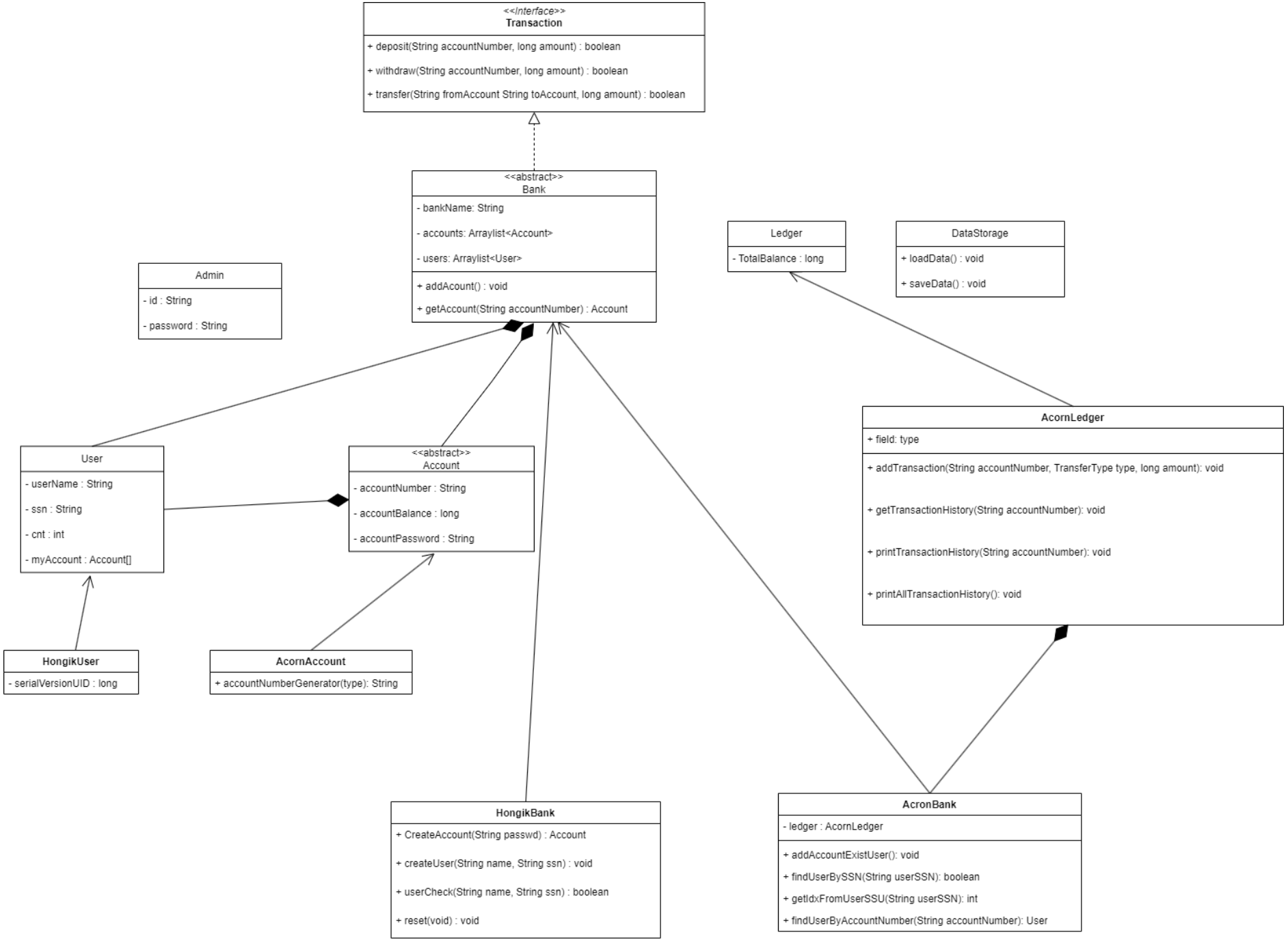
주제

은행 ATM프로그램

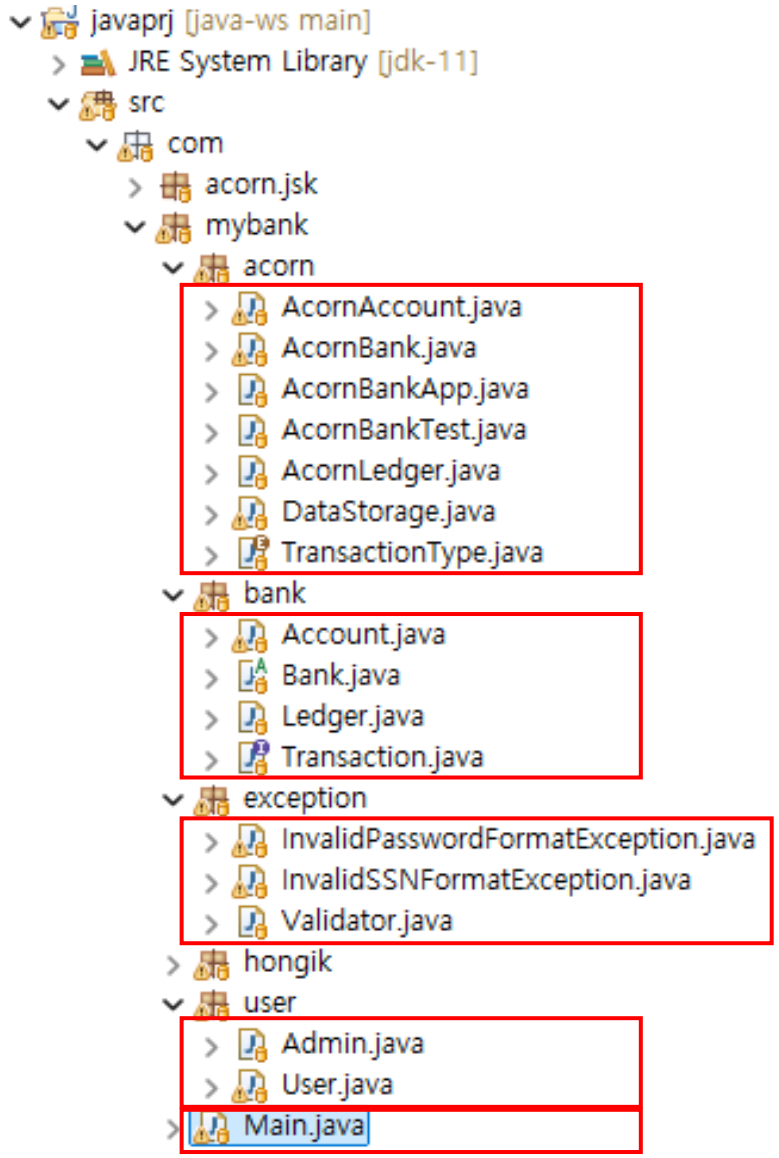
은행

-Acorn은행

-Hongik은행



Mybank 프로젝트 패키지 구조




```
Mybank
├- acorn
├- bank
├- exception
├- user
```

Interface: Transaction.java

```java
public interface Transaction {

    //입금
    public boolean deposit(String accountNumber, long amount);

    //출금
    public boolean withdraw(String accountNumber, long amount);

    //송금
    public boolean transfer(String fromAccount, String toAccount, long amount);
}
```

Enum:
TransactionType.java

```java
package com.mybank.acorn;

public enum TransactionType {
    DEPOSIT, WITHDRAW, TRNASFER
}
```

Public static final 생략된 형태

```java
                }
                ledger.addTransaction(accountNumber, TransactionType.DEPOSIT, amount);
                System.out.println("입금 완료");
                System.out.println("> 잔액 : " + a.getAccountBalance());
                return true;
            }
        }
```

AcornBank.java

[참고자료] Enum을 사용하는 이유 :
- 우아한 기술 블로그
https://techblog.woowahan.com/2527/
- 자바 Enum 열거형 타입 문법 & 응용 💯 정리
https://bit.ly/3XdtaP0
- 이것이 자바다 5-12 열거(Enum) 타입

Acorn은행

초기메뉴

```
에이콘은행 본점에 오신것을 환영합니다
메뉴선택 [1]관리자용 [2]고객용 [3]신규계좌 개설 [4]종료:
```

관리자 메뉴

```
메뉴를 선택해주세요 [1]거래내역조회, [2]전체고객조회, [3]은행잔액조회, [4]전단계
```

고객 메뉴

```
계좌번호 : 425936
비밀번호 : 111
111님 안녕하세요!
현재 계좌 잔액 : 0
메뉴를 선택해주세요 [1]추가 계좌개설, [2]입금, [3]출금, [4]송금, [5]전단계
```

```java
loop:while (true) {
    System.out.println(b.getBankName() + "에 오신것을 환영합니다");
    System.out.println("메뉴선택 [1]관리자용 [2]고객용 [3]신규계좌 개설 [4]종료: ");

    String select = sc.nextLine();

    switch (select) {
    case "1":
        System.out.println("관리자용");
        System.out.print("아이디 : ");
        String id = sc.nextLine();
        System.out.print("비밀번호 : ");
        String pw = sc.nextLine();
        if (id.equals(new Admin().getId()) && pw.equals(new Admin().getPassword())) {
            adminMenu();
        }else {
            System.out.println("아이디 혹은 비밀번호가 틀렸습니다");
        }
        break;
    case "2":
        System.out.println("고객용");
        System.out.println();
        System.out.print("계좌번호 : ");
        String checkid = sc.nextLine();
        for (int i = 0; i < b.getAccounts().size(); i++) {
            if (checkid.equals(b.getAccounts().get(i).getAccountNumber())) {
                System.out.print("비밀번호 : ");
                String checkpw = sc.nextLine();
                if (checkpw.equals(b.getAccounts().get(i).getAccountPassword())) {
                    clientMenu(checkid);
                    checkid = null;
                    break;
                }else {
                    System.out.println("비밀번호가 일치하지 않습니다");
                    checkid = null;
                    break;
                }
            }

        }
```

클래스

계좌

```java
package com.mybank.bank;

import java.io.Serializable;

public class Account implements Serializable {
    private String accountNumber; // 계좌번호
    private long accountBalance; // 계좌금액
    private String accountPassword; // 비밀번호

    public String getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }

    public long getAccountBalance() {
        return accountBalance;
    }

    public void setAccountBalance(long accountBalance) {
        this.accountBalance = accountBalance;
    }

    public String getAccountPassword() {
        return accountPassword;
    }

    public void setAccountPassword(String accountPassword) {
        this.accountPassword = accountPassword;
    }

    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return "계좌번호 : " + this.accountNumber + "\n계좌금액 : " + this.accountBalance + "\n";
    }
}
```

사용자

```java
package com.mybank.user;

import java.io.Serializable;

public class User implements Serializable{

    //field
    private String userName;
    private String ssn;
    private int cnt = 0;
    private Account[] myAccount;

    public User(String userName, String ssn) {
        this.userName = userName;
        this.ssn = ssn;
        this.myAccount = new Account[3];
    }

    public String getUserName() {
        return userName;
    }
    public String getSSN() {
        return ssn;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public void setSSN(String ssn) {
        this.ssn = ssn;
    }

    public int getCnt() {
        return cnt;
    }

    public void setCnt(int cnt) {
        this.cnt = cnt;
    }

    public Account[] getMyAccount() {
        return myAccount;
    }

    public void setMyAccount(Account[] myAccount) {
        this.myAccount = myAccount;
    }

    @Override
    public String toString() {
        return "User [userName=" + userName + ", ssn=" + ssn + ", cnt=" + cnt + ", myAccount="
                + Arrays.toString(myAccount) + "]";
    }

}
```

클래스

은행

```java
package com.mybank.bank;

import com.mybank.user.User;

public abstract class Bank implements Transaction{

    //field
    private String bankName;
    protected ArrayList<Account> accounts;
    protected ArrayList<User> users;

    //constructor
    public Bank(String bankName) {
        this.bankName = bankName;
        this.accounts = new ArrayList<Account>();
        this.users = new ArrayList<User>();
    }

    //method
    public abstract void addAccount();
    public abstract Account getAccount();

    public String getBankName() {
        return bankName;
    }

    public ArrayList<Account> getAccounts() {
        return accounts;
    }

    public ArrayList<User> getUsers() {
        return users;
    }



}
```

장부

```java
package com.mybank.acorn;

import java.text.SimpleDateFormat;

public class AcornLedger extends Ledger {

    private ArrayList<String> transactions;

    public AcornLedger() {
        super();
        this.transactions = new ArrayList<>();
    }

    public ArrayList<String> getTransactions() {
        return transactions;
    }

    public void setTransactions(ArrayList<String> transactions) {
        this.transactions = transactions;
    }

    @Override
    public long getTotalBalance() {
        return super.getTotalBalance();
    }

    @Override
    public void setTotalBalance(long totalBalance) {
        super.setTotalBalance(totalBalance);
    }
```

신규계좌개설

```
에이콘은행 본점에 오신것을 환영합니다
메뉴선택 [1]관리자용 [2]고객용 [3]신규계좌 개설 [4]종료:
3
신규계좌 개설
이름을 입력해주세요 : 최형진
주민번호를 입력해주세요 : 020812-3XXXXXX
계좌비밀번호를 입력해주세요 : 0812
가입을 환영합니다 당신의 계좌번호는 633255 입니다
```

- 주민번호로 등록된 고객인지 if문으로 확인
- 입력받은 후 생성자 생성

```java
@Override
    public void addAccount() {

        System.out.print("이름을 입력해주세요 : ");
        String userName = sc.nextLine();
        System.out.print("주민번호를 입력해주세요 : ");
        String ssn = sc.nextLine();

        boolean isUserExist = false;

        for(int i = 0; i < users.size(); i++) {
            String searchSSN = users.get(i).getSSN();
            if (ssn.equals(searchSSN)) {
                System.out.println("이미 존재하는 고객입니다!!!(직원 창구 문의)");
                isUserExist = true;
            }
            if (isUserExist) return;
        }

        System.out.print("계좌비밀번호를 입력해주세요 : ");
        String password = sc.nextLine();

        String newAccountNumber = AcornAccount.accountNumberGenerator();
        AcornAccount newAccount = new AcornAccount();
        newAccount.setAccountNumber(newAccountNumber);
        newAccount.setAccountPassword(password);


        User newUser = new User(userName, ssn);
        accounts.add(newAccount);
        users.add(newUser);

        newUser.getMyAccount()[newUser.getCnt()] = newAccount;
        newUser.setCnt(newUser.getCnt() + 1);

        System.out.println("가입을 환영합니다 당신의 계좌번호는 " + newAccount.getAccountNumber() + " 입니다");
        //testPrintAllUser();
        return;
    }
```

고객 로그인

```
에이콘은행 본점에 오신것을 환영합니다
메뉴선택 [1]관리자용 [2]고객용 [3]신규계좌 개설 [4]종료:
2
고객용

계좌번호 : 633255
비밀번호 : 0812
최형진님 안녕하세요!
현재 계좌 잔액 : 0
메뉴를 선택해주세요 [1]추가 계좌개설, [2]입금, [3]출금, [4]송금, [5]전단계
```

- 계좌번호가 있고 비밀번호가 일치하는지 Account객체에서 가져와 확인

```
System.out.println("고객용");
System.out.println();
System.out.print("계좌번호 : ");
String checkid = sc.nextLine();
for (int i = 0; i < b.getAccounts().size(); i++) {
    if (checkid.equals(b.getAccounts().get(i).getAccountNumber())) {
        System.out.print("비밀번호 : ");
        String checkpw = sc.nextLine();
        if (checkpw.equals(b.getAccounts().get(i).getAccountPassword())) {
            clientMenu(checkid);
            checkid = null;
            break;
        }else {
            System.out.println("비밀번호가 일치하지 않습니다");
            checkid = null;
            break;
        }
    }

}
if (checkid != null) {
    System.out.println("계좌번호를 잘못 입력하셨습니다");
}

break;
```

추가 계좌 개설

```
메뉴를 선택해주세요 [1]추가 계좌개설, [2]입금, [3]출금, [4]송금, [5]전단계
1
추가 계좌개설 진행
추가 계좌 개설을 진행합니다
주민번호를 입력해주세요 : 020812-3XXXXXX
새로운 계좌의 비밀번호를 입력해주세요0000
계좌개설이 완료되었습니다 당신의 계좌번호는 718879 입니다
최형진님 안녕하세요!
현재 계좌 잔액 : 0
메뉴를 선택해주세요 [1]추가 계좌개설, [2]입금, [3]출금, [4]송금, [5]전단계
```

- 고객당 최대 3개까지의 계좌를 만들 수 있게 제한을 두고 새로 계좌를 만들때마다 고객계좌의 수를 1씩 늘리고 3이상이면 못만들게 if문으로 구현

```
public void addAccountExistUser() {
    System.out.println("추가 계좌 개설을 진행합니다");
     System.out.print("주민번호를 입력해주세요 : ");
     String searchUserSSN = sc.nextLine();

     boolean isUserExist = findUserBySSN(searchUserSSN);

     if(isUserExist) {
         int foundIdx = getIdxFromUserSSN(searchUserSSN);
         if(foundIdx != -1) {
             int cntUsrAccount = users.get(foundIdx).getCnt();
             if(cntUsrAccount >= 3) {
                 System.out.println("계좌 계설은 최대 3개까지 가능합니다 ");
                 return;
             }

             User foundUser = users.get(foundIdx);

             System.out.print("새로운 계좌의 비밀번호를 입력해주세요");
             String password = sc.nextLine();
             AcornAccount newAccount = new AcornAccount();
             String newAccountNumber = AcornAccount.accountNumberGenerator();
             newAccount.setAccountNumber(newAccountNumber);
             newAccount.setAccountPassword(password);
             accounts.add(newAccount);

             foundUser.getMyAccount()[foundUser.getCnt()] = newAccount;
             foundUser.setCnt(foundUser.getCnt() + 1);

             System.out.println("계좌개설이 완료되었습니다 당신의 계좌번호는 " + newAccount.getAccountNumber() + " 입니다");
         }
     } else {
         System.out.println("주민번호가 일치하지 않습니다");
     }
}
```

입금, 출금, 송금

```
메뉴를 선택해주세요 [1]추가 계좌개설, [2]입금, [3]출금, [4]송금, [5]전단계
2
입금하실 금액을 입력해주세요 : 10000
입금성공
잔액 10000
최형진님 안녕하세요!
현재 계좌 잔액 : 10000
```

```
메뉴를 선택해주세요 [1]추가 계좌개설, [2]입금, [3]출금, [4]송금, [5]전단계
3
출금하실 금액을 입력해주세요 : 5000
출금성공
잔액 5000
최형진님 안녕하세요!
현재 계좌 잔액 : 5000
```

- 입금 시 금액을 입력받고 계좌 객체의 돈을 추가하고 고객 객체의 돈을 추가하고 장부에 기록함
- 출금 시에는 금액보다 많은 돈이 계좌에 있는지 if문으로 확인함

```
@Override
public boolean deposit(String accountNumber, long amount) {

    Account a;
    for (int i = 0; i < accounts.size(); i++) {
        if (accounts.get(i).getAccountNumber().equals(accountNumber)) {
            a = accounts.get(i);
            long money = a.getAccountBalance();
            money += amount;
            a.setAccountBalance(money);
            for (int j = 0; j < users.size(); j++) {
                for (int j2 = 0; j2 < users.get(j).getCnt(); j2++) {
                    if (accountNumber.equals(users.get(j).getMyAccount()[j2].getAccountNumber())) {
                        users.get(j).getMyAccount()[j2].setAccountBalance(money);
                    }
                }
            }
            ledger.addTransaction(accountNumber, TransactionType.DEPOSIT, amount);
            System.out.println("입금성공");
            System.out.println("잔액 " + a.getAccountBalance());
            return true;
        }
    }

    System.out.println("입금실패");
    return false;
}
```

```
@Override
public boolean withdraw(String accountNumber, long amount) {
    Account a;
    for (int i = 0; i < accounts.size(); i++) {
        if (accounts.get(i).getAccountNumber().equals(accountNumber)) {
            a = accounts.get(i);
            long money = a.getAccountBalance();
            if (money >= amount) {
                money -= amount;
                a.setAccountBalance(money);
                for (int j = 0; j < users.size(); j++) {
                    for (int j2 = 0; j2 < users.get(j).getCnt(); j2++) {
                        if (accountNumber.equals(users.get(j).getMyAccount()[j2].getAccountNumber())) {
                            users.get(j).getMyAccount()[j2].setAccountBalance(money);
                        }
                    }
                }
                ledger.addTransaction(accountNumber, TransactionType.WITHDRAW, amount);
                System.out.println("출금성공");
                System.out.println("잔액 " + a.getAccountBalance());
                return true;
            }else {
                System.out.println("잔액 부족");
                return false;
            }

        }
    }

    System.out.println("출금실패");
```

입금, 출금, 송금

```
메뉴를 선택해주세요 [1]추가 계좌개설, [2]입금, [3]출금, [4]송금, [5]전단계
4
송금하실 계좌를 입력해주세요 : 718879
송금하실 금액을 입력해주세요 : 5000
송금 완료
최형진님 안녕하세요!
현재 계좌 잔액 : 0
```

- 송금시에도 계좌에 송금할 돈 이상의 돈이 있는지 if문으로 확인함

```
@Override
public boolean transfer(String fromAccount, String toAccount, long amount) {

    Account a = null, b = null;

    for (int i = 0; i < accounts.size(); i++) {
        if (accounts.get(i).getAccountNumber().equals(fromAccount)) {
            a = accounts.get(i);
        }

        if (accounts.get(i).getAccountNumber().equals(toAccount)) {
            b = accounts.get(i);
        }
    }

    if (a == null || b == null) {
        System.out.println("존재하지 않는 계좌");
        return false;
    } else {

        long fromMoney = a.getAccountBalance();
        long toMoney = b.getAccountBalance();
        if (fromMoney >= amount) {
            fromMoney -= amount;
            toMoney += amount;
            a.setAccountBalance(fromMoney);
            for (int j = 0; j < users.size(); j++) {
                for (int j2 = 0; j2 < users.get(j).getCnt(); j2++) {
                    if (fromAccount.equals(users.get(j).getMyAccount()[j2].getAccountNumber())) {
                        users.get(j).getMyAccount()[j2].setAccountBalance(fromMoney);
                    }
                }
            }
            b.setAccountBalance(toMoney);
            for (int j = 0; j < users.size(); j++) {
                for (int j2 = 0; j2 < users.get(j).getCnt(); j2++) {
                    if (toAccount.equals(users.get(j).getMyAccount()[j2].getAccountNumber())) {
                        users.get(j).getMyAccount()[j2].setAccountBalance(toMoney);
                    }
                }
            }

            ledger.addTransaction(fromAccount, TransactionType.TRNASFER, amount);
            ledger.addTransaction(toAccount, TransactionType.TRNASFER, amount);
            System.out.println("송금 완료");
            return true;
        }else {
            System.out.println("잔액 부족");
            return false;
        }


    }
}
```

관리자 로그인

```
에이콘은행 본점에 오신것을 환영합니다
메뉴선택 [1]관리자용 [2]고객용 [3]신규계좌 개설 [4]종료:
1
관리자용
아이디 : admin
비밀번호 : qwer1234
메뉴를 선택해주세요 [1]거래내역조회, [2]전체고객조회, [3]은행잔액조회, [4]전단계
```

```
System.out.println("관리자용");
System.out.print("아이디 : ");
String id = sc.nextLine();
System.out.print("비밀번호 : ");
String pw = sc.nextLine();
if (id.equals(new Admin().getId()) && pw.equals(new Admin().getPassword())) {
    adminMenu();
}else {
    System.out.println("아이디 혹은 비밀번호가 틀렸습니다");
}
break;
```

전체고객조회

```
메뉴를 선택해주세요 [1]거래내역조회, [2]전체고객조회, [3]은행잔액조회, [4]전단계
2
>>>>>>>>>>>>>>>전체 고객조회
---------------------------------------------------------
[1]
고객이름 : 111
주민번호 : 111
<<<<고객계좌 목록>>>>
1개 계좌 조회되었습니다
계좌번호 : 425936 계좌잔액 : 0
---------------------------------------------------------
[2]
고객이름 : 222
주민번호 : 222
<<<<고객계좌 목록>>>>
2개 계좌 조회되었습니다
계좌번호 : 359145 계좌잔액 : 19000
계좌번호 : 677184 계좌잔액 : 5000
---------------------------------------------------------
[3]
고객이름 : 최형진
주민번호 : 020812-3XXXXXX
<<<<고객계좌 목록>>>>
2개 계좌 조회되었습니다
계좌번호 : 633255 계좌잔액 : 0
계좌번호 : 718879 계좌잔액 : 5000
```

```java
public void printAllUsers() {

    ArrayList<User> userList = b.getUsers();
    if (!userList.isEmpty()) {
        for (int i = 0; i < userList.size(); i++) {
            System.out.println("---------------------------------------------------------");
            System.out.println("[" + (i + 1) + "]");
            System.out.println("고객이름 : " + userList.get(i).getUserName());
            System.out.println("주민번호 : " + userList.get(i).getSSN());
            System.out.println("<<<<고객계좌 목록>>>>");
            Account[] userAccounts = userList.get(i).getMyAccount();

            int numOfAccounts = userList.get(i).getCnt();
            System.out.println(numOfAccounts + "개 계좌 조회되었습니다");
            for (Account account : userAccounts) {
                if (account == null)
                    break;
                String accountNumber = account.getAccountNumber();
                long accountBalance = account.getAccountBalance();
                System.out.println("계좌번호 : " + accountNumber + " 계좌잔액 : " + accountBalance);
            }
        }
    } else {

        System.out.println("고객정보가 없습니다");
    }
}
```

- 모든 고객과 그 고객이 가지고 있는 계좌를 for문을 통해 출력

거래내역조회

```
메뉴를 선택해주세요 [1]거래내역조회, [2]전체고객조회, [3]은행잔액조회, [4]전단계
1
거래내역조회
2024-08-29 오전 10:57:29 | 359145 | DEPOSIT | 10000
2024-08-29 오전 11:08:47 | 677184 | DEPOSIT | 5000
2024-08-29 오전 11:09:04 | 359145 | DEPOSIT | 9000
2024-08-29 오후 02:53:43 | 633255 | DEPOSIT | 10000
2024-08-29 오후 02:53:52 | 633255 | WITHDRAW | 5000
2024-08-29 오후 02:54:05 | 633255 | TRNASFER | 5000
2024-08-29 오후 02:54:05 | 718879 | TRNASFER | 5000
```

```java
public void addTransaction(String accountNumber, TransactionType type, long amount) {
    String timestamp = new SimpleDateFormat("yyyy-MM-dd a hh:mm:ss").format(Calendar.getInstance().getTime());
    String transaction = String.format("%s | %s | %s | %d", timestamp, accountNumber, type, amount);
    transactions.add(transaction);

    // 잔액 업데이트
    if (type == TransactionType.DEPOSIT) {
        setTotalBalance(getTotalBalance() + amount);
    } else if (type == TransactionType.WITHDRAW) {
        setTotalBalance(getTotalBalance() - amount);
    }
}

//계좌번호별 트렌젝션
public ArrayList<String> getTransactionHistory(String accountNumber) {
    ArrayList<String> accountTransactions = new ArrayList<>();
    for (String transaction : transactions) {
        if (transaction.contains(accountNumber)) {
            accountTransactions.add(transaction);
        }
    }
    return accountTransactions;
}

//계좌번호의 트렌젝션 히스토리 출력
public void printTransactionHistory(String accountNumber) {
    ArrayList<String> history = getTransactionHistory(accountNumber);
    for (String transaction : history) {
        System.out.println(transaction);
    }
}

//전체 트렌젝션 출력

public void printAllTransactionHistory() {
    for(String transaction : transactions) {
        System.out.println(transaction);
    }
}
```

파일 저장, 불러오기

- File Output/Input Stream과 Object Output/Input Stream을 사용하여 Arraylist 객체 전체를 저장

```java
//데이터 저장하기
public void saveData() throws EOFException{

    try (OutputStream out = new FileOutputStream(filePathArr[0]);
            ObjectOutputStream oout = new ObjectOutputStream(out);
            ){
        // 고객정보 저장하기
        ArrayList<User> ulist = bank.getUsers();
        oout.writeObject(ulist);
    } catch (IOException e) {
        System.out.println("파일저장 오류");
    }
    try (OutputStream out = new FileOutputStream(filePathArr[1]);
            ObjectOutputStream oout = new ObjectOutputStream(out);
            ){
        // 은행계좌 저장하기
        ArrayList<Account> alist = bank.getAccounts();
        oout.writeObject(alist);
    } catch (IOException e) {
        System.out.println("파일저장 오류");
    }
    try (OutputStream out = new FileOutputStream(filePathArr[2]);
            ObjectOutputStream oout = new ObjectOutputStream(out);
            ){
        // 은행장부 저장하기
        ArrayList<String> transactions = ledger.getTransactions();
        oout.writeObject(transactions);
    } catch (IOException e) {
        System.out.println("파일저장 오류");
    }
}
```

```java
//method
public void loadData() throws EOFException, ClassNotFoundException {
    try (InputStream in = new FileInputStream(filePathArr[0]);
            ObjectInputStream oin = new ObjectInputStream(in);){
        // 고객정보 불러오기
        ArrayList<User> ulist = new ArrayList<>();
        ulist = (ArrayList<User>) oin.readObject();
        bank.setUsers(ulist);
    } catch (IOException e) {
        System.out.println("파일 불러오기 오류");
    }
    try (InputStream in = (new FileInputStream(filePathArr[1]));
            ObjectInputStream oin = new ObjectInputStream(in);){
        // 은행계좌 불러오기
        ArrayList<Account> alist = new ArrayList<>();
        alist = (ArrayList<Account>) oin.readObject();
        bank.setAccounts(alist);
    } catch (IOException e) {
        System.out.println("파일 불러오기 오류");
    }
    try (InputStream in = (new FileInputStream(filePathArr[2]));
            ObjectInputStream oin = new ObjectInputStream(in);){
        // 은행장부 불러오기
        ArrayList<String> transactions = new ArrayList<>();
        transactions = (ArrayList<String>) oin.readObject();
        ledger.setTransactions(transactions);
    } catch (IOException e) {
        System.out.println("파일 불러오기 오류");
    }
}
```

Exception : Validator.java

```java
package com.mybank.exception;

import java.util.regex.Matcher;

public class Validator {
    public static boolean ssnValidator(String ssn) throws InvalidSSNFormatException{

        // "\d" 숫자, "{6}" 앞의 패턴이 6번 반복, "[-]?" -가 있을수도 있고 아닐수도 있음
        // "[1-4]" 1에서 4까지의 범위 숫자(성별), "\d" 숫자, "{6}" 앞의 패턴이 6번 반복
        String patternType = "\\d{6}[-]?[1-4]\\d{6}";

        //정규식 입력
        Pattern pattern = Pattern.compile(patternType);

        //입력된 주민번호가 정규식과 일치하는지 검사
        Matcher matcher = pattern.matcher(ssn);


        //패턴 전체 문자열이 일치하면 true를 반환
        if(matcher.matches()) {
            return true;
        } else {
            throw new InvalidSSNFormatException();
        }
```

주민번호 13자리
xxxxxx-[1~4]xxxxx
xxxxxx[1~4]xxxxx

} boolean ssnValidator(String ssn) : 유저가 입력한 주민번호 형식이 일치하면 true 반환, 일치하지 않으면 InvalidSSNFormatException 예외를 발생시킴

Exception : Validator.java

```
    public static boolean passwordValidator(String password) throws InvalidPasswordFormatException{

        // "\d" 숫자, "{4}" 앞의 패턴이 4번 반복
        String patternType = "\\d{4}";

        //정규식 입력
        Pattern pattern = Pattern.compile(patternType);

        //입력된 password가 정규식과 일치하는지 검사
        Matcher matcher = pattern.matcher(password);

        if(matcher.matches()) {
            return true;
        } else {
            throw new InvalidPasswordFormatException();
        }


    }
}
```

boolean passwordValidator(String password) : 유저가 입력한 비밀번호가 숫자 4자리이면 true 반환, 일치하지 않으면 InvalidPasswordFormatException 예외를 발생시킴

Exception

- InvalidSSNFormatException

```
public final class InvalidSSNFormatException extends Exception{
    public InvalidSSNFormatException () {
        super("잘못된 형식의 주민번호 입니다");
    }
}
```

- InvalidPasswordFormatException

```
public class InvalidPasswordFormatException extends Exception {
    public InvalidPasswordFormatException () {
        super("잘못된 형식의 비밀번호 입니다");
    }
}
```

Hongik 은행

데이터 직렬화, 반직렬화

```java
public void loadingData() {
    try {
        FileInputStream fis = new FileInputStream(FILE_NAME);
        ObjectInputStream ois = new ObjectInputStream(fis);

        bank = (HongikBank) ois.readObject();
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        bank = new HongikBank();
    } catch (IOException ie) {
        ie.printStackTrace();
    } catch (ClassNotFoundException cfe) {
        cfe.printStackTrace();
    }
}

public void savingData() {
    try {
        FileOutputStream fos = new FileOutputStream(FILE_NAME);
        ObjectOutputStream oos = new ObjectOutputStream(fos);

        oos.writeObject(bank);
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException ie) {
        ie.printStackTrace();
    }
}
```

```java
public void run() {
    this.loadingData();
    while (true) {
        // 시작 메뉴창
        displayMenu1();
        int input = sc.nextInt();
        sc.nextLine();

        switch(input) {
        case 1:
            System.out.println();
            managerLogin();
            System.out.println();
            break;
        case 2:
            System.out.println();
            clientLogin();
            System.out.println();
            break;
        case 3:
            System.out.println();
            createAccount();
            System.out.println();
            break;
        case 0:
            this.savingData();
            System.exit(0);
            break;
        }
    }
}
```

계좌 생성

```
public void createAccount() {
    System.out.print(" 이름을 입력해주세요 : ");
    String name = sc.nextLine();
    System.out.print(" 주민번호를 입력해주세요 : ");
    String ssn = sc.nextLine();
    // true -> 이미 있는 고객, false -> 없는 고객
    if (bank.userCheck(name, ssn)) {
        System.out.print(" 설정할 비밀번호를 입력해주세요 : ");
        String passwd = sc.nextLine();
        Account account = bank.createAccount(passwd);
        if (account != null) {
            bank.addAccount(account);
            System.out.println();
            System.out.println(" 계좌가 성공적으로 생성되었습니다");
            System.out.println(" 계좌번호는 "  + bank.getAccountNum() + "입니다.");
        } else {
            // null이 반환된 경우 처리 로직
            System.out.println();
            System.out.println(" 계좌가 이미 최대로 존재합니다");
            System.out.println(" 개설이 취소되었습니다.");
        }
    } else {
        bank.createUser(name, ssn);
        System.out.print(" 설정할 비밀번호를 입력해주세요 : ");
        String passwd = sc.nextLine();
        bank.addAccount(bank.createAccount(passwd));
        System.out.println(" 계좌가 성공적으로 생성되었습니다");
        System.out.println(" 계좌번호는 "  + bank.getAccountNum() + "입니다.");
    }
}
```

프로그램 실행 코드

```
public void run() {
    this.loadingData();
    while (true) {
        // 시작 메뉴창
        displayMenu1();
        int input = sc.nextInt();
        sc.nextLine();

        switch(input) {
        case 1:
            System.out.println();
            managerLogin();
            System.out.println();
            break;
        case 2:
            System.out.println();
            clientLogin();
            System.out.println();
            break;
        case 3:
            System.out.println();
            createAccount();
            System.out.println();
            break;
        case 0:
            this.savingData();
            System.exit(0);
            break;
        }
    }
}
```

로그인 (고객, 관리자)

```java
public void clientLogin() {

    int check = 0;

    System.out.println("----------------------------");
    System.out.print(" 계좌번호 입력 : ");
    String id = sc.nextLine();
    System.out.print(" 비밀번호 입력 : ");
    String passwd = sc.nextLine();

    System.out.println("----------------------------");

    if (!(bank.getAccount(id) == null)) {
        if (bank.getAccount(id).getAccountPassword().equals(passwd)) {
            check = 2;
        }
        else check = 1;
    } else {
        check = 0;
    }

    switch (check) {
    case 0:
        System.out.println();
        System.out.println(" 일치하는 계좌가 존재하지 않습니다.");
        break;
    case 1:
        System.out.println();
        System.out.println(" 계좌 비밀번호가 틀렸습니다.");
        break;
    case 2:
        System.out.println();
        System.out.println(" 로그인 성공");
        System.out.println();
        this.accountNum = id;
        clientDisplay();
        break;
    }
}
```

```java
public void managerLogin() {
    System.out.println("----------------------------");
    System.out.print(" 아이디 입력 : ");
    String id = sc.nextLine();
    System.out.print(" 비밀번호 입력 : ");
    String passwd = sc.nextLine();

    System.out.println("----------------------------");

    Admin admin = new Admin();

    if (id.equals(admin.getId()) && passwd.equals(admin.getPassword())) {
        System.out.println();
        managerDisplay();
    } else {
        System.out.println();
        System.out.println(" 잘못된 접근입니다.");
    }
}
```

선택 화면 (고객, 관리자)

```
public void clientDisplay() {
    loop : while (true) {
        // 메뉴창
        displayMenu3();
        int input = 0;
            input = sc.nextInt();

            sc.nextLine();

            switch (input) {
            case 1:
                System.out.println();
                System.out.print("입금하실 금액을 입력해주세요 : ");
                try {
                    long balance = sc.nextLong();
                    sc.nextLine();
                    System.out.println();
                    if (balance <= 0) {
                        System.out.println("0 초과의 금액을 입력해주세요");
                    } else {
                        if (bank.deposit(this.accountNum, balance)) {
                            System.out.println("정상적으로 입금 완료되었습니다.");
                            System.out.println("입금 후 잔액 : " + bank.getBalance());
                        } else {
                            System.out.println("비정상 오류");
                        }
                    }
                } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    sc.nextLine();
                }
                break;
```

```
public void managerDisplay() {
    loop : while (true) {
        // 메뉴창
        displayMenu2();
        int input = sc.nextInt();
        sc.nextLine();

        switch (input) {
        case 1:
            bank.allAccountReference();
            break;
        case 0:
            break loop;
        }
    }
}
```

User 구현부

```java
public class HongikUser extends User implements Serializable{

    //field
    private static final long serialVersionUID = 1L;
    private String userName; // 이름
    private String ssn; // 주민번호
    public int cnt; // 계좌개수 체크
    private Account[] myAccount;



    public HongikUser(String userName, String ssn) {
        super();
        this.userName = userName;
        this.ssn = ssn;
        this.cnt = 0;
        this.myAccount = new Account[3];
    }
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getSsn() {
        return ssn;
    }
    public void setSsn(String ssn) {
        this.ssn = ssn;
    }
    public int getCnt() {
        return cnt;
    }
    public void setCnt(int cnt) {
        this.cnt = cnt;
    }
    public Account[] getMyAccount() {
        return myAccount;
    }
    public void setMyAccount(Account[] myAccount) {
        this.myAccount = myAccount;
    }
```

```java
public class User implements Serializable{

    //field
    private static final long serialVersionUID = 1L;
    private String userName; // 이름
    private String ssn; // 주민번호
    int cnt = 0; // 계좌개수 체크
    private Account[] myAccount = new Account[3];

    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getSsn() {
        return ssn;
    }
    public void setSsn(String ssn) {
        this.ssn = ssn;
    }
    public int getCnt() {
        return cnt;
    }
    public void setCnt(int cnt) {
        this.cnt = cnt;
    }
    public Account[] getMyAccount() {
        return myAccount;
    }
    public void setMyAccount(Account[] myAccount) {
        this.myAccount = myAccount;
    }
    @Override
    public String toString() {
        // TODO Auto-generated method stub
        return super.toString();
    }

}
```

Bank 구현부

```java
public abstract class Bank implements Transaction{


    //field
    private String bankName;
    private ArrayList<User> users;

    public Bank() {
        super();
    }

    //constructor
    public Bank(String bankName){
        this.bankName = bankName;
    }

    public String getName() {
        return bankName;
    }

    //method
    public abstract void addAccount(Account account);
    public abstract Account getAccount(String accountNumber);

}
```

```java
public class HongikBank extends Bank implements Serializable{

    private static final long serialVersionUID = 1L;
    private String bankName;
    private ArrayList<HongikUser> users; // User 동적배열
    // 현재 유저의 선택된 계좌의 인덱스값
    private int accountIndex;
    // 현재 유저의 인덱스위치
    private int userIndex;
    private String accountNum;

    public HongikBank() {
        this.bankName = "홍익";
        this.users = new ArrayList<HongikUser>();
    }

    public HongikBank(String bankName) {
        super(bankName);
        this.bankName = bankName;
        // TODO Auto-generated constructor stub
    }

    @Override
    public boolean deposit(String accountNumber, long amount) {
        // TODO Auto-generated method stub
        // 현재 계좌에 있는 잔액
        long preBalance = users.get(userIndex).getMyAccount()[accountIndex].getAccountBalance();
        users.get(userIndex).getMyAccount()[accountIndex].setAccountBalance(amount + preBalance);
        return true;
    }

    @Override
    public boolean withdraw(String accountNumber, long amount) {
        // TODO Auto-generated method stub
        long preBalance = users.get(userIndex).getMyAccount()[accountIndex].getAccountBalance();
        if (preBalance < amount) {
            return false;
        } else {
            users.get(userIndex).getMyAccount()[accountIndex].setAccountBalance(preBalance - amount);
            return true;
        }
    }
```

Bank 구현부2

```java
@Override
public boolean transfer(String fromAccount, String toAccount, long amount) {
    // TODO Auto-generated method stub
    if (withdraw(fromAccount, amount)) {
        for (int i = 0; i < users.size(); i++) {
            for (int j = 0; j < users.get(i).getCnt(); j++) {
                if (users.get(i).getMyAccount()[j].getAccountNumber().equals(toAccount)) {
                    long preBalance = users.get(i).getMyAccount()[j].getAccountBalance();
                    users.get(i).getMyAccount()[j].setAccountBalance(preBalance + amount);
                    return true;
                }
            }
        }
    }
    return false;
}

@Override
public void addAccount(Account account) {
    // TODO Auto-generated method stub
    users.get(userIndex).getMyAccount()[users.get(userIndex).cnt - 1] = account;
}

// 계좌를 뽑아내면서 초기값도 동시에 설정
@Override
public Account getAccount(String accountNumber) {
    // TODO Auto-generated method stub
    // 유저의 수만큼 반복
    for (int i = 0; i < users.size(); i++) {
        // 해당 유저의 계좌의 개수만큼 반복
        for (int j = 0; j < users.get(i).getCnt(); j++) {
            if (users.get(i).getMyAccount()[j].getAccountNumber().equals(accountNumber)) {
                userIndex = i;
                accountIndex = j;
                return users.get(i).getMyAccount()[j];
            }
        }
    }
    // 해당하는것이 없다면
    return null;
}
```

```java
// 현재 포인팅된 계좌의 잔액 조회
public long getBalance() {
    return users.get(userIndex).getMyAccount()[accountIndex].getAccountBalance();
}

public Account[] getAccount() {
    return users.get(userIndex).getMyAccount();
}

public void AccountReference() {
    // 현재 유저의 계좌를 모두 가져와서 출력
    for (Account i : this.getAccount()) {
        if (i != null) {
            System.out.println();
            System.out.println("----------------------------");
            System.out.println(" 계좌번호 : " + i.getAccountNumber());
            System.out.println(" 잔액 : " + i.getAccountBalance());
            System.out.println("----------------------------");
        }
    }
}

public void allAccountReference() {
    // 모든 유저로부터
    for (int i = 0; i < users.size(); i++) {
        // 해당 유저의 모든 계좌를 가져와서 출력
        for (Account j : users.get(i).getMyAccount()) {
            if (j != null) {
                System.out.println();
                System.out.println("----------------------------");
                System.out.println(" 이름 : " + users.get(i).getUserName());
                System.out.println(" 주민번호 : " + users.get(i).getSsn());
                System.out.println(" 계좌번호 : " + j.getAccountNumber());
                System.out.println(" 잔액 : " + j.getAccountBalance());
                System.out.println("----------------------------");
            }
        }
    }
}
```

Bank 구현부3

```java
// createAccount는 이미 user에 대한 유효성체크가 지난 시점이니 편하게 cnt값만 체크해주면 됨
public Account createAccount(String passwd) {
    if (users.get(userIndex).cnt == 3) return null;
    else {
        Account account = new Account();
        // 랜덤숫자로 accountNumber 생성 (100000 ~ 999999)
        String accountNum = String.valueOf(999999 - (int)(Math.random() * 900000));
        this.accountNum = accountNum;
        account.setAccountNumber(accountNum);
        account.setAccountBalance(0);
        account.setAccountPassword(passwd);
        // 현재 계좌개수 증가
        users.get(userIndex).cnt++;
        return account;
    }
}

// 해당 유저가 이미 존재하는지 체크
// 이걸 HongikApp에서 if, else문으로 써서 사용
public boolean userCheck(String name, String ssn) {
    if (users == null || users.isEmpty()) return false;
    for (int i = 0; i < users.size(); i++) {
        if (users.get(i).getUserName().equals(name) && users.get(i).getSsn().equals(ssn)) {
            this.userIndex = i;
            return true;
        }
    }
    return false;
}

public String getAccountNum() {
    return accountNum;
}

public void createUser(String name, String ssn) {
    HongikUser user = new HongikUser(name, ssn);
    this.userIndex = users.size();
    users.add(user);
}
```