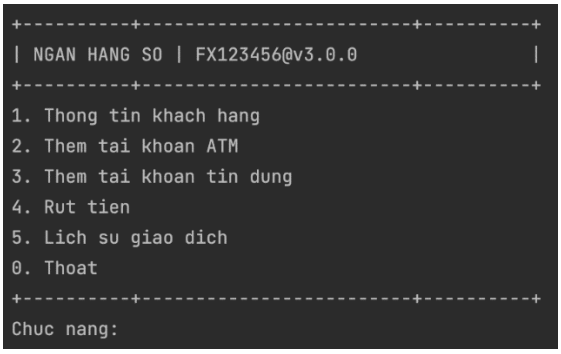
Lab 03: NGÂN HÀNG SỐ phiên bản v3.0.0

1. Tổng quan bài tập

Xây dựng phần mềm ngân hàng số cho phép kiểm tra mã số căn cước công dân từ đó biết được thông tin người cần truy cập, bao gồm các chức năng sau:



2. Mô tả ứng dụng

- Phần mềm ngân hàng số cho phép quản lý tài khoản khách hàng gồm tài khoản tiết kiệm **(SavingsAccount)**, và tài khoản vay **(LoanAccount)**, cho phép rút tiền, tra cứu lịch sử giao dịch.
- Ứng dụng cần đảm bảo các chức năng và yêu cầu cơ bản tuy nhiên sinh viên có thể thêm chức năng bổ sung vào theo ý thích.

3. Thiết kế giao diện

- Yêu cầu giao diện dạng console đơn giản.
- Hiển thị thông tin thành khối rõ ràng và dễ nhìn.

```
+----------+------------------------+----------+
| NGAN HANG SO | FX123456@v3.0.0 |
+----------+------------------------+----------+
1. Thong tin khach hang
2. Them tai khoan ATM
3. Them tai khoan tin dung
4. Rut tien
5. Lich su giao dich
0. Thoat
+----------+------------------------+----------+
Chuc nang:
```

4. Yêu cầu chức năng cơ bản

- Khi chương trình được khởi chạy đầu tiên sẽ hiển thị mô tả ngắn gọn về chương trình để người dùng hiểu được hệ thống này sẽ hoạt động thế nào, có chức năng gì ví dụ như hiển thị tên phần mềm viết in hoa:

  NGAN HANG SO

- Hiển thị tên tác giả (mã số sinh viên) và phiên bản (version) phần mềm theo mẫu

  1912001@v3.0.0

Trong đó:

  - 1912001: là mã số học viên.
  - 3.0.0: Phiên bản phần mềm.

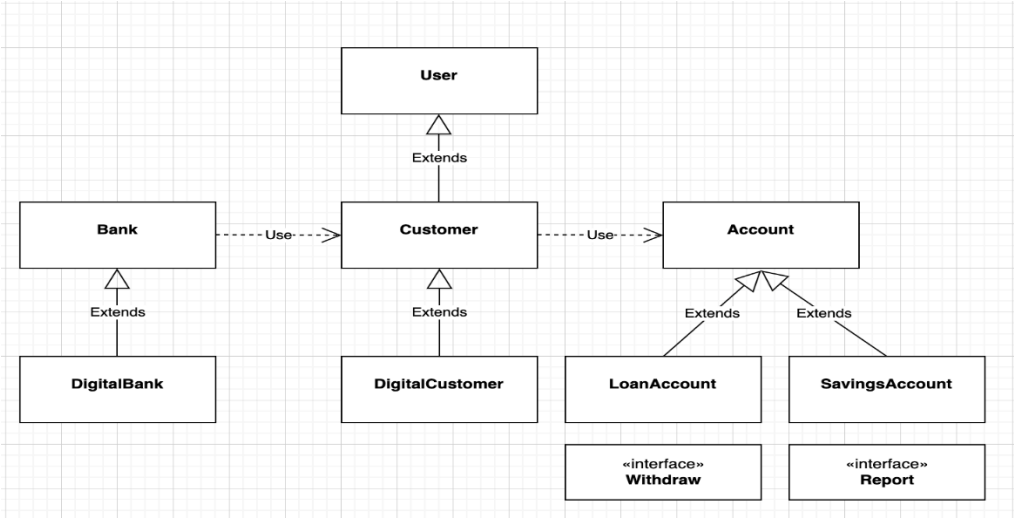
- Hiển thị dường phân cách giữa tiêu đề và nội dung chức năng.
- Hiển thị menu cho người dùng chọn theo mẫu ví dụ.
  - 1. Thông tin khách hàng
  - 2. Thêm tài khoản ATM
  - 3. Thêm tài khoản tín dụng
  - 4. Rút tiền
  - 5. Lịch sử giao dịch
  - 0. Thoát

5. Yêu cầu chức năng nâng cao

- Hiển thị menu cho người dùng chọn theo mẫu ví dụ.

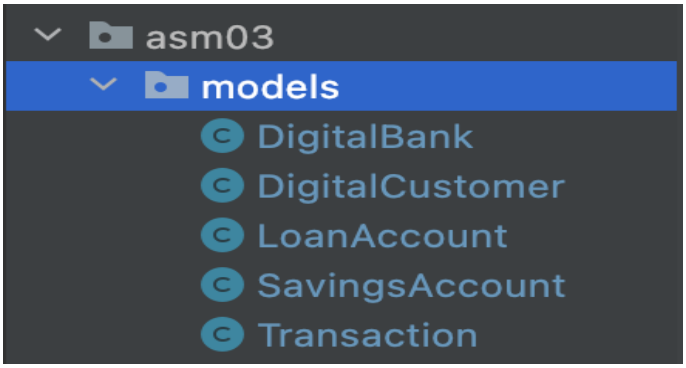
```
Chuc nang: 5
+----------+-------------------------------+----------+
| LICH SU GIAO DICH                                   |
+----------+-------------------------------+----------+
001215000001 |        FUNIX |  Premium |  87,500,000đ
1     123456 |          ATM |             40,000,000đ
2     234567 |       CREDIT |             47,500,000đ
001215000001 | 123456 | -10,000,000đ | 31/03/2022 10:56:28
001215000001 | 234567 | -50,000,000đ | 31/03/2022 10:56:45
+----------+-------------------------------+----------+
```

Tổ chức mô hình quan hệ các lớp như sau:



6. Tổ chức code

- Dùng lại folder của Lab số 2 để lưu trữ toàn bộ kết quả làm bài



- Tạo folder **asm03** dùng để chứa source code của bài tập 03, theo hình trên.
- File **Asm03.java** chứa mã nguồn hàm main, để khởi chạy chức năng.
- File **ReportService.java**: định nghĩa interface cho nghiệp vụ báo cáo
- Hàm log, không có kiểu trả về, có tham số **amount**, kiểu **double**, dùng để In ra biên lai theo từng loại tài khoản mỗi khi rút tiền (Xem yêu cầu chức năng 4)

```
public interface ReportService {
    void log(double amount);
}
```

Ví dụ:

```java
    @Override
    public void log(double amount) {
        System.out.println(Utils.getDivider());
        System.out.printf("%30s%n", getTitle());
        System.out.printf("NGAY G/D: %28s%n", Utils.getDateTime());
        System.out.printf("ATM ID: %30s%n", "DIGITAL-BANK-ATM 2022");
        System.out.printf("SO TK: %31s%n", account.getAccountNumber());
        System.out.printf("SO TIEN: %29s%n", Utils.formatBalance(amount));
        System.out.printf("SO DU: %31s%n", Utils.formatBalance(account.getBalance()));
        System.out.printf("PHI + VAT: %27s%n", Utils.formatBalance(account.getTransactionFee() * amount));
        System.out.println(Utils.getDivider());
    }

}
```

- File **Withdraw.java**: định nghĩa interface cho nghiệp vụ rút tiền.
- Hàm **withdraw** có kiểu trả về là **boolean**, có tham số **amount**, kiểu **double**. Xử lý nghiệp vụ rút tiền (Chức năng 4)
- Hàm **isAccepted** có kiểu trả về là **boolean**, có tham số **amount**, kiểu **double**, xác định xem giá trị có thoả điều kiện rút tiền hay không. Xử lý nghiệp vụ rút tiền (Chức năng 4)

```java
public interface Withdraw {
    boolean withdraw(double amount);

    boolean isAccepted(double amount);
}
```

Ví dụ:

```java
public class LoanAccount extends Account implements Withdraw {

    @Override
    public boolean withdraw(double amount) {
        double newBalance = 0.0; // ..
        if (isAccepted(newBalance)) {
            setBalance(newBalance);
            // ..
            return true;
        }
        // ..
        return false;
    }
```

- File **SavingsAccount.java**: định nghĩa cho class quản lý tài khoản tiết kiệm của khách hàng.
    - Kế thừa lại class **Account** của bài 2.
    - Implement interface **Report** để xử lý nghiệp vụ báo cáo cho loại Savings.
    - Implement interface **Withdraw** để xử lý nghiệp vụ rút tiền cho loại Savings.
    - Hằng số: **SAVINGS_ACCOUNT_MAX_WITHDRAW,** định nghĩa số tiền tối đa mà khách hàng có thể rút cho 1 lần giao dịch là **5.000.000đ**
    - Đối tượng này khi tạo ra sẽ được lưu vào **DigitalBank** thông qua hàm **addAccount,** nếu bài tập 2 chưa làm tiêu chí nâng cao thì có thể tham khảo đoạn code dưới đây.

```java
public void addAccount(String customerId, Account account) {
    for (Customer customer : customers) {
        if (isCustomerExisted(customerId)) {
            customer.addAccount(account);
        }
    }
}

public boolean isCustomerExisted(String customerId) {
    for (Customer customer : customers) {
        if (customer.getCustomerId().equals(customerId)) {
            return true;
        }
    }
    return false;
}
```

- File **LoansAccount.java**: định nghĩa cho class quản lý tài khoản của khách hàng.
    - Kế thừa lại class **Account** của bài 2.
    - **Implement interface Report** để xử lý nghiệp vụ báo cáo cho loại Credit.
    - **Implement interface Withdraw** để xử lý nghiệp vụ rút tiền cho loại Credit.
    - Hằng số: **LOAN_ACCOUNT_WITHDRAW_FEE** định nghĩa phí rút tiền cho tài khoản thường là 0.05.
    - Hằng số: **LOAN_ACCOUNT_WITHDRAW_PREMIUM_FEE** định nghĩa phí rút tiền tài khoản premium là 0.01.
    - Hằng số: **LOAN_ACCOUNT_MAX_BALANCE** định nghĩa hạn mức tối đa cho loại tài khoản này là: 100.000.000đ.
    - Đối tượng này khi tạo ra sẽ được lưu vào **DigitalBank** thông qua hàm **addAccount**
- File **DigitalCustomer.java**: kế thừa lại **class Custome**r ở bài tập 02, định nghĩa cho class quản lý thông tin khách hàng tiềm năng.
    - Thêm hàm **withdraw** dùng để cho phép khách hàng rút tiền theo tài khoản gồm 2 tham số:
        - account **Number** kiểu String là mã tài khoản của khách hàng.
        - amount kiểu **double** là số tiền mà khách hàng muốn rút.
    - Hàm này kiểm tra nếu **accountNumber** có tồn tại thì truy xuất đối tượng đó ra và gọi hàm **withdraw** của đối tượng đó.
- File **DigitalBank.java**: kế thừa lại **class Bank** ở bài tập 02, định nghĩa cho class quản lý ngân hàng tiềm năng.
    - Hàm **getCustomerById** dùng để tìm khách hàng theo CCCD, gồm 1 tham số:
        - customerId kiểu **String** là CCCD của khách hàng.
        - Kiểu trả về **Customer**.
        - Chỉ trả về khách hàng nếu như tồn tại CCCD trong hệ thống ngân hàng, ngược lại trả về null.

- Hàm **addCustomer** dùng để tạo khách hàng cho ngân hàng, gồm 2 tham số:
    - **customerId** kiểu **String** là **CCCD** của khách hàng.
    - **name** kiểu **String** là tên của khách hàng.
- **Thêm** hàm **withdraw** dùng để cho phép khách hàng rút tiền theo tài khoản gồm 3 tham số:
    - **customerId** kiểu **String** là CCCD của khách hàng.
    - **accountNumber** kiểu **String** là mã tài khoản của khách hàng.
    - **amount** kiểu **double** là số tiền mà khách hàng muốn rút.
    - Hàm này kiểm tra nếu **customerId** không tồn tại thì trả về, ngược lại có tồn tại thì gọi hàm **withdraw** của đối tượng khách hàng

7. Yêu cầu chức năng cơ bản

Chức năng 1: Thông tin khách hàng

- Khởi tạo sẵn đối tượng **DigitalBank** trong **Ams03**.
- Dùng hàng số để khởi tạo sẵn mã CUSTOMER_ID (CCCD) và CUSTOMER_NAME (tên) cho tài khoản khách hàng của mình. Tham khảo code sau:

```
public class Asm03 {

    private static final int EXIT_COMMAND_CODE = 0;
    private static final int EXIT_ERROR_CODE = -1;
    private static final Scanner scanner = new Scanner(System.in);
    private static final DigitalBank activeBank = new DigitalBank();
    private static final String CUSTOMER_ID = "001215000001";
    private static final String CUSTOMER_NAME = "FUNIX";
```

- Dùng hàm **getCustomerById** để lấy đối tượng khách hàng, sau đó in thông tin của khách hàng gồm:

```java
private static void showCustomer() {
    Customer customer = activeBank.getCustomerById(CUSTOMER_ID);
    if (customer != null)
        customer.displayInformation();
}
```

- Override lại hàm **displayInformation** ở bài tập 02 để có được thêm loại tài khoản của khách hàng, tham khảo ví dụ sau:

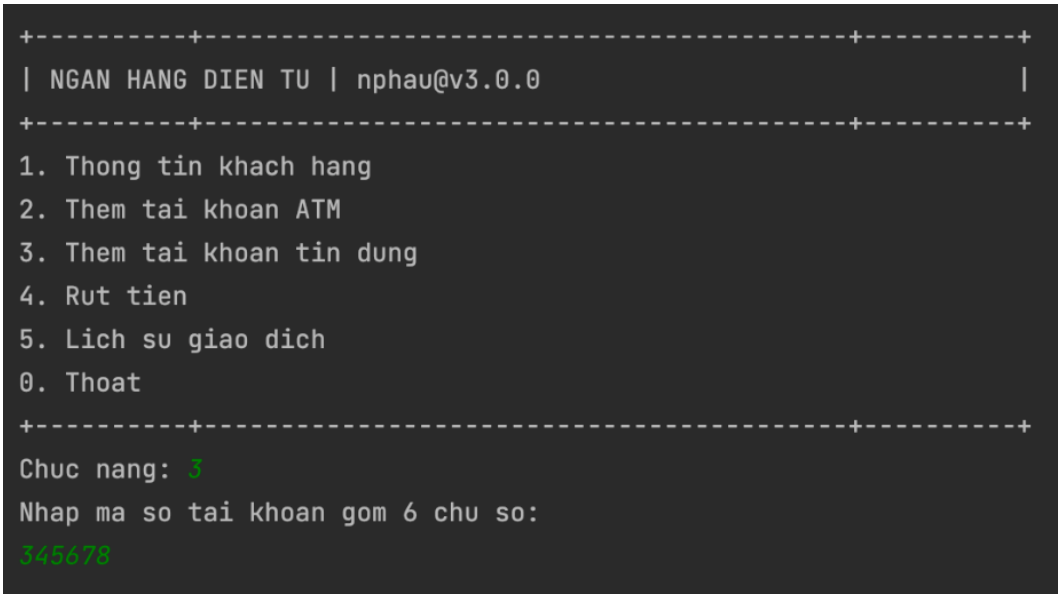
```
001215000001 |        FUNIX |  Premium |  11,000,000đ
1     123456 |      SAVINGS |           1,000,000đ
2     234567 |         LOAN |          10,000,000đ
```

Chức năng 2: Thêm tài khoản Savings

```
+----------+-------------------------------------------+----------+
| NGAN HANG DIEN TU | nphau@v3.0.0                                 |
+----------+-------------------------------------------+----------+
1. Thong tin khach hang
2. Them tai khoan ATM
3. Them tai khoan tin dung
4. Rut tien
5. Lich su giao dich
0. Thoat
+----------+-------------------------------------------+----------+
Chuc nang: 2
Nhap ma so tai khoan gom 6 chu so:
123456
Nhap so du:
50000000
```

- Nhập số tài khoản gồm 6 chữ số, thông báo lỗi nếu người dùng nhập không đúng hoặc tài khoản đã tồn tại.
- Nhập số dư ban đầu cho tài khoản.

Chức năng 3: Thêm tài khoản Loan



- Nhập số tài khoản gồm 6 chữ số, thông báo lỗi nếu người dùng nhập không đúng hoặc tài khoản đã tồn tại.

Chức năng 4: Rút tiền

- Cho người dùng nhập số tiền cần rút
  - Đối với tài khoản **Savings**
    - Số tiền rút phải lớn hơn hoặc bằng **50.000đ**
    - Số tiền 1 lần rút không được quá **5.000.000đ** đối với tài khoản thường, và không giới hạn đối với tài khoản **Premium**.
    - Số dư còn lại sau khi rút phải lớn hơn hoặc bằng **50.000đ**
    - Số tiền rút phải là bội số của **10.000đ**
  - Đối với tài khoản **Loan**
    - Hạn mức không được qúa giới hạn **100.000.000đ**
    - Phí cho mỗi lần giao dịch là **0.05** trên số tiền giao dịch đối với tài khoản thường và **0.01** trên số tiền giao dịch đối với tài khoản **Premium**.
    - Hạn mức còn lại sau khi rút tiền không được nhỏ hơn **50.000đ**

- Gọi làm **log** để in ra biên lai theo từng loại tài khoản mỗi khi rút tiền thành công. Định dạng theo mẫu dưới đây.

```
+----------+-------------------------+----------+
       BIEN LAI GIAO DICH LOAN
NGAY G/D:          12/09/2022 23:19:28
ATM ID:          DIGITAL-BANK-ATM 2022
SO TK:                          234567
SO TIEN:                      500,000đ
SO DU:                      9,495,000đ
PHI + VAT:                      5,000đ
+----------+-------------------------+----------+
```

```
+----------+-------------------------+----------+
    BIEN LAI GIAO DICH SAVINGS
NGAY G/D:          26/04/2022 00:29:06
ATM ID:          DIGITAL-BANK-ATM 2022
SO TK:                          123456
SO TIEN:                      500,000đ
SO DU:                        500,000đ
PHI + VAT:                          0đ
+----------+-------------------------+----------+
```

8. Yêu cầu chức năng nâng cao

Chức năng 5: Lịch sử giao dịch

- Tạo file **Transaction.jav**a định nghĩa cho class quản lý giao dịch gồm:
  - id kiểu String là mã random ngẫu nhiên (Xem lại cách tạo mã ngẫu nhiên ở bài tập 02)
  - **accountNumber kiểu String** là mã tài khoản của khách hàng.
  - Chỉ trả về khách hàng nếu như tồn tại CCCD trong hệ thống ngân hàng, ngược lại trả về null.
  - **amount kiểu double** là số tiền mà khách hàng muốn rút.

- **time kiểu String** là thời điểm giao dịch (tham khảo hướng dẫn cách dùng Date)
- **status kiểu Boolean. True** là giao dịch thành công, False là giao dịch thất bại

- Lưu danh sách giao dịch trong mỗi loại account. Khi người dùng thực hiện giao dịch rút tiền thì lưu lại lịch sử.
- In ra thông tin giao dịch gồm có thông tin của khách hàng và thông tin giao dịch, tham khảo ví dụ mẫu

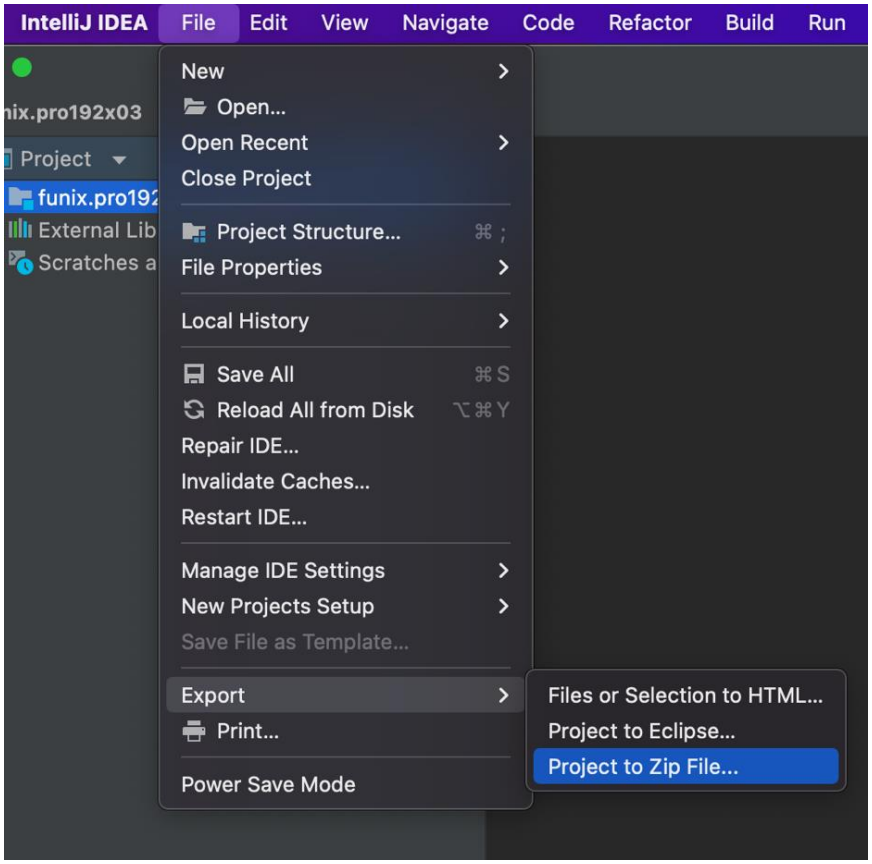
```
+----------+--------------------------+----------+
| LICH SU GIAO DICH                              |
+----------+--------------------------+----------+
001215000001 |        FUNIX |   Normal |  14,899,000đ
1     123456 |      SAVINGS |             5,000,000đ
2     234567 |         LOAN |             9,899,000đ
123456 |   5,000,000đ | 26/04/2022 00:39:40
234567 |    -100,000đ | 26/04/2022 00:39:40
+----------+--------------------------+----------+
```

Cách dùng Date để tạo thời điểm giao dịch

```
public static String getDateTime() {
    DateFormat df = new SimpleDateFormat( pattern: "dd/MM/yyyy HH:mm:ss");
    Date today = Calendar.getInstance().getTime();
    return df.format(today);
}
```

9. Hướng dẫn nộp bài

- Nén toàn bộ thư mục và các tài nguyên cần thiết của dự án để giảng viên chấm điểm bằng file .zip
- Chọn File -> Export -> Project to Zip file.
- Nộp bài trên hệ thống LMS


IntelliJ IDEA
File
Edit
View
Navigate
Code
Refactor
Build
Run
New
Open...
Open Recent
Close Project
Project Structure... ⌘ ;
File Properties
Local History
Save All ⌘ S
Reload All from Disk ⌥ ⌘ Y
Repair IDE...
Invalidate Caches...
Restart IDE...
Manage IDE Settings
New Projects Setup
Save File as Template...
Export
Print...
Power Save Mode
Files or Selection to HTML...
Project to Eclipse...
Project to Zip File...
Project
External Lib
Scratches a