Shape, square

Description automatically generated**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**UNIVERSITY OF TRANSPOST AND COMMUNICATIONS**

Logo

Description automatically generated

**BÁO CÁO**

**MÔN CÔNG NGHỆ JAVA**

***Đề tài:* Bank Management System**

**Giảng viên hướng dẫn : Vũ Huấn**

**Lớp : Công nghệ thông tin 6 - K62**

**Nhóm thực hiện : 18**

**Sinh viên thực hiện : Nguyễn Thế Trung**

**Mã sinh viên :211202747**

**Hà Nội 4 - 2023**

Mục lục

[I. Giới thiệu 1](#_Toc134040822)

[II.Chức năng 1](#_Toc134040823)

[1. SignIn, SignOut, SignUp 1](#_Toc134040824)

[2. Các chức năng khi đăng nhập tài khoản Employee: 2](#_Toc134040825)

[3. Các chức năng khi đăng nhập tài khoản Client. 3](#_Toc134040826)

[III. Thiết kế và cách tương tác giữa các lớp. 3](#_Toc134040827)

[1. Lớp Accountant 4](#_Toc134040828)

[2. Lớp Bank\_Account 5](#_Toc134040829)

[3. Lớp Client 5](#_Toc134040830)

[4. Lớp Login\_Account 6](#_Toc134040831)

[5. Lớp Manager 7](#_Toc134040832)

[6. Lớp Transaction\_History 8](#_Toc134040833)

[7. Lớp AccountsFromFile 9](#_Toc134040834)

[8. Lớp BM\_System 9](#_Toc134040835)

[9. Lớp DB\_Handler 10](#_Toc134040836)

[11. Lớp GUI 30](#_Toc134040837)

[IV. Mô hình Cơ sở dữ liệu 56](#_Toc134040838)

[V. Tài liệu tham khảo 57](#_Toc134040839)

# I. Giới thiệu

Trong thời đại công nghệ 4.0, ngân hàng trở thành một phần không thể thiếu trong cuộc sống của mọi người. Với vai trò quan trọng trong việc quản lý tài chính, ngân hàng đã ngày càng phát triển và đóng góp đáng kể vào sự phát triển kinh tế của mỗi quốc gia.

Trong bối cảnh đó, việc xây dựng và phát triển các hệ thống ngân hàng thông minh, tiên tiến đã trở thành một yêu cầu cấp bách. Hệ thống ngân hàng thông minh không chỉ giúp tăng cường hiệu suất và nâng cao chất lượng dịch vụ của ngân hàng mà còn đem lại lợi ích cho khách hàng khi sử dụng các sản phẩm và dịch vụ của ngân hàng.

Trong bài viết này, chúng tôi sẽ giới thiệu một hệ thống ngân hàng thông minh cơ bản, bao gồm các chức năng cơ bản như quản lý tài khoản, quản lý khách hàng, báo cáo và thống kê, đăng nhập và đăng xuất, tạo tài khoản, nạp tiền và rút tiền, chuyển khoản, quản lý tài sản, và các yêu cầu phi chức năng như bảo mật và an ninh, dễ sử dụng và tương tác.

Mục tiêu của hệ thống này là tạo ra một nền tảng tiên tiến để quản lý tài chính, cung cấp các sản phẩm và dịch vụ tài chính đa dạng và chất lượng cao, và nâng cao trải nghiệm khách hàng khi sử dụng các sản phẩm và dịch vụ của ngân hàng.

# II.Chức năng

Dự án Bank Management System bao gồm các chức năng sau :

## 1. SignIn, SignOut, SignUp

- SignIn : Để đăng nhập bạn cần phải có User Name và Password. Có 2 loại đăng nhập là: Employee hoặc Client, mỗi loại khi đăng nhập vào sẽ có được hiển thị những chức năng riêng.

- SignOut : Đăng xuất khỏi tài khoản.

- SignUp : Đăng kí tài khoản để đăng nhập ( chỉ được đăng kí tài khoản client(khách hàng). Các bước đăng kí:

+ Bước 1: Nhập **Account num** và **CIC**. Nếu xác minh thành công( đã có thông tin trong hệ thống ) thì chuyển sang bước 2.

+ Bước 2: Nhập **User Name**, **Password** và **Confirm Password**.

## 2. Các chức năng khi đăng nhập tài khoản Employee:

2.1 Manager Employee gồm:

- Create Account: Người quản lí có thể thêm thông tin tài khoản vào hệ thống bao gồm: **First Name**, **Last Name**, **Father Name**, **Mother Name**, **CIC**, **Date of Birth**, **Phone**, **Email**, **Address**, **Account Type** **(Current/Saving).**

- Block/ Unblock Account: Chức năng khóa/mở khóa (tạm thời) tài khoản khi nhập 2 thông tin **Account num** và **CIC.**

- Close Account: Chức năng đóng (vĩnh viễn) tài khoản khi nhập 2 thông tin **Account num** và **CIC**.

- Search Account: Chức năng tìm kiếm tài khoản khoản khi nhập Account num sẽ hiện ra các thông tin : **Name, CIC, Account num, Account Type, Balance, Status, Opening Date**.

- Manage Credit/Debit Card: Trong quản lí thẻ có chức năng khóa/mở khóa(tạm thời) thẻ khi nhập 3 thông tin: **Account num, Card num, CIC**.

- Update Client Info: Chức năng cập nhật thông tin khách hàng : chỉ có thể thay đổi 3 thông tin : **Phone, Email, Address**.

- Add Account From File: Chức năng thêm thông tin tài khoản từ file.

2.2 Accountant Employee gồm:

- Deposit Cash: Nhân viên kế toán có chức năng thêm số tiền vào tài khoản nếu khách hàng gửi tiền mặt khi nhập vào 2 thông tin: **Account num, CIC**.

- Withdraw Cash: Chức năng trừ số tiền trong tài khoản nếu khách hàng muốn rút tiền mặt khi nhập vào 2 thông tin: **Account num, CIC**.

- Cheque Deposit: Nhân viên kế toán có chức năng cho phép khách hàng gửi tiền bằng số séc vào tài khoản ngân hàng của họ khi nhập vào 3 thông tin:**Account num, Amount, Cheque num**. Khác so với Deposit Cash là khách hàng cần phải nhập CIC, với Chueque Deposit khách hàng gửi tiền thông qua số Séc sẽ an toàn hơn và họ cũng có thể lưu lại thông tin về số Séc, đảm bảo rằng việc thanh toán được ghi nhân rõ ràng và dễ quản lý hơn.

- Issue E-Statement: Nhân viên kế toán có thể tra cứu thông tin lịch sử các giao dịch (chuyển tiền, nhận tiền …) của một khách hàng khi nhập Account num và CIC. Tuy nhiên ở đây chúng tôi chỉ giới hạn tra cứu giao dịch từ năm 2018 đến năm 2023.

## 3. Các chức năng khi đăng nhập tài khoản Client.

- Account Info: Khách hàng có thể xem thông tin của mình : Name, Account Number, Account Type(Current/Saving), Account Current Balance, Account Opening Date.

- Cardless Cash Withdrawal: Khách hàng có thể rút tiền không cần thẻ khi 3 thông tin: Card Number, Amount, Pin Code. Khi nhập các thông tin này hệ thống sẽ trả về một mã OTP và khách hàng có thể thực hiện rút tiền ngoài cây ATM mà không cần thẻ.

- View EStatement: Khách hàng cũng có thể xem thông tin tất cả các giao dịch của mình (chuyển tiền, nhận tiền…) (giới hạn 2018 đến 2023).

- Change Password: Chức năng cuối cùng là đổi mật khẩu đăng nhập khi nhập 3 thông tin: Current Password, new Password, Confirm new Password.

Đó là tất cả các chức năng của Bank Management System. Tiếp theo chúng tôi sẽ đi đến cách thiết kế và cách tương tác giữa các lớp.

# III. Thiết kế và cách tương tác giữa các lớp.

Cách tương tác giữa các lớp chúng tôi sẽ trình bày ở trong lớp GUI.

## 1. Lớp Accountant

Lớp Accountant là một lớp đại diện cho các kế toán trong hệ thống quản lý ngân hàng. Lớp này chứa các phương thức để thực hiện các chức năng quản lý tài khoản, tìm kiếm thông tin tài khoản và khách hàng, kiểm tra khoản tiền gửi và lấy thông tin lịch sử giao dịch của một tài khoản trong một khoảng thời gian nhất định.

Cụ thể, lớp Accountant có các phương thức sau:

- Phương thức khởi tạo không tham số và khởi tạo có tham số, dùng để tạo đối tượng Accountant với tên người kế toán được cung cấp (nếu có).

- Phương thức **searchAccount1(String accountNum, String CIC)** dùng để tìm kiếm thông tin tài khoản bằng cách truyền vào số tài khoản và mã số CIC. Phương thức này sử dụng đối tượng DB\_Handler để kết nối với cơ sở dữ liệu và truy vấn dữ liệu tài khoản. Phương thức trả về đối tượng Bank\_Account chứa thông tin của tài khoản tìm kiếm được.

- Phương thức **searchClient1(String accountNum, String CIC)** tương tự như phương thức searchAccount1() nhưng dùng để tìm kiếm thông tin khách hàng. Phương thức này trả về đối tượng Client chứa thông tin của khách hàng tìm kiếm được.

- Phương thức **chequeDeposit(String accNum, String chequeNum, int amount )** được sử dụng để kiểm tra khoản tiền gửi. Phương thức này truyền vào số tài khoản, số phiếu gửi tiền(số Séc) và số tiền cần kiểm tra. Phương thức sử dụng đối tượng DB\_Handler để truy vấn cơ sở dữ liệu và trả về số tiền kiểm tra được.

- Phương thức **getTransactions( String acc\_num, String From, String To)** được sử dụng để lấy thông tin lịch sử giao dịch của một tài khoản trong một khoảng thời gian nhất định. Phương thức này truyền vào số tài khoản, thời gian bắt đầu và kết thúc của khoảng thời gian cần lấy thông tin. Phương thức sử dụng đối tượng DB\_Handler để truy vấn cơ sở dữ liệu và trả về một danh sách các đối tượng Transaction\_History chứa thông tin về các giao dịch của tài khoản đó trong khoảng thời gian được chỉ định.

## 2. Lớp Bank\_Account

Đây là một lớp đại diện cho một tài khoản ngân hàng. Lớp này có các thuộc tính để lưu trữ thông tài khoản như số tài khoản, mã khách hàng, mã đăng nhập, loại tài khoản, số dư, trạng thái, ngày mở tài khoản, ngày đóng tài khoản và số thẻ.

Lớp này cũng có một số phương thức để thực hiện các hoạt động trên tài khoản, bao gồm:

- Các phương thức getter/setter

- addInfo(…): Cập nhật lại thông tin tài khoản.

- addAmount(int b): Cộng thêm số tiền vào tài khoản.

- removeAmount(int b): Rút tiền khỏi tài khoản

- updateBalance(): Cập nhật lại số dư tìa khoản ngân hàng bằng cách truy vấn đến cơ sở dữ liệu.

- finalize(): Được gọi khi đối tượng tài khoản bị hủy để giải phóng tài nguyên.

## 3. Lớp Client

Đây là lớp đại diện cho khách hàng. Lớp này có các thuộc tính để lưu trữ thông tin khách hàng và cá phương thức để quản lí thông tin khách hàng.

Các thuộc tính của lớp Client bao gồm:

**client\_id**: mã số khách hàng

**f\_name**: tên

**l\_name**: họ

**father\_name**: tên cha

**mother\_name**: tên mẹ

**CIC**: số CIC của khách hàng

**DOB**: ngày tháng năm sinh

**phone**: số điện thoại

**email**: địa chỉ email

**address**: địa chỉ

Lớp Client có 3 phương thức khởi tạo (constructor), cho phép tạo ra một đối tượng khách hàng với các thông tin khác nhau.

Lớp Client cũng có một số phương thức, bao gồm:

showClientInfo(): hiển thị thông tin khách hàng ra màn hình

getClientID(), getFName(), getLName(), getFatherName(), getMotherName(), getCIC(), getDOB(), getPhone(), getEmail(), getAddress(): trả về các giá trị của các thuộc tính tương ứng của khách hàng.

transferMoney(): thực hiện việc chuyển tiền đến một tài khoản nhận được xác định

do\_card\_cash\_withdrawal(): thực hiện rút tiền từ tài khoản ngân hàng của khách hàng mà không cần sử dụng thẻ

change\_password(): thay đổi mật khẩu đăng nhập của khách hàng

getTransactions(): truy xuất các giao dịch của khách hàng trong một khoảng thời gian nhất định.

Lớp Client cũng tham chiếu đến một đối tượng DB\_Handler để thực hiện các thao tác trên cơ sở dữ liệu.

## 4. Lớp Login\_Account

Đây là lớp đại diện cho một tài khoản đăng nhập trong hệ thống.

Các thuộc tính của lớp Login\_Account bao gồm:

**login\_id:** Mã đăng nhập.

**username:** Tên tài khoản đăng nhập.

**password:** Mật khẩu.

**type:** Kiểu tài khoản (Client/Manager/Accountant).

Lớp Login\_Account có 2 phương thức khởi tạo (constructor), cho phép tạo ra tạo ra một đối tượng đăng nhập với các thông tin khác nhau.

Lớp Login\_Account có một số phương thức khác, bao gồm:

getLoginId(), getUserName(), getType(): trả về các giá trị thuộc tính tương ứng với tài khoản đăng nhập.

verify\_account(String acc\_num, String cic): Sử dụng để xác thực tài khoản bằng cách truy xuất đến cơ sở dữ liệu thông qua đối tượng DB\_Handler. Nếu tài khoản không tồn tại hoặc thông tin xác thực không chính xác, phương thức sẽ trả về một giá trị số nguyên âm để chỉ ra lỗi xảy ra.

signup(String username, String pass\_1, String pass\_2, String acc\_num): Sử dụng để đăng ký tài khoản bằng cách tạo một tài khoản mới trong cơ sở dữ liệu thông qua đối tượng DB\_Handler. Nếu mật khẩu nhập lại không khớp hoặc tạo tài khoản không thành công, phương thức sẽ trả về một giá trị số nguyên âm để chỉ ra lỗi xảy ra.

getName(): trả về tên của người dùng tương ứng với tài khoản đăng nhập, thông qua việc truy xuất cơ sở dữ liệu. Nó không trực tiếp trả về giá trị thuộc tính của đối tượng Login\_Account, mà thực hiện một hành động phức tạp hơn là truy xuất cơ sở dữ liệu để lấy thông tin tên của người dùng.

## 5. Lớp Manager

Đây là lớp đại diện cho một người quản lý trong hệ thống ngân hàng. Lớp này có các phương thức để thực hiện các chức năng quản lý tài khoản, thẻ, khách hàng, và cập nhật thông tin khách hàng.

Lớp Manager có hai phương thức khởi tạo, một có tham số tên và một không có tham số, và các phương thức getter để truy cập vào thuộc tính name: Tên người quản lí.

Các phương thức quản lý bao gồm:

createAccount: tạo một tài khoản mới cho khách hàng với loại tài khoản và trả về số tài khoản được tạo.

block\_account: khóa một tài khoản bằng cách sử dụng số tài khoản và số CIC, trả về 0 nếu thành công hoặc một mã lỗi nếu thất bại.

unlock\_account: mở khóa một tài khoản bằng cách sử dụng số tài khoản và số CIC, trả về 0 nếu thành công hoặc một mã lỗi nếu thất bại.

block\_card: khóa một thẻ bằng cách sử dụng số tài khoản, số CIC và số thẻ, trả về 0 nếu thành công hoặc một mã lỗi nếu thất bại.

unlock\_card: mở khóa một thẻ bằng cách sử dụng số tài khoản, số CIC và số thẻ, trả về 0 nếu thành công hoặc một mã lỗi nếu thất bại.

close\_account: đóng một tài khoản bằng cách sử dụng số tài khoản và số CIC, trả về 0 nếu thành công hoặc một mã lỗi nếu thất bại.

getClientInfo: lấy thông tin khách hàng bằng cách sử dụng số tài khoản và trả về một đối tượng khách hàng.

getAccountInfo: lấy thông tin tài khoản bằng cách sử dụng số tài khoản và trả về một đối tượng tài khoản.

updateClientInfo: cập nhật thông tin khách hàng với số ID khách hàng, số điện thoại, email và địa chỉ.

getAccNum: lấy số tài khoản bằng cách sử dụng số CIC.

Tất cả các phương thức trên của lớp Manager tương tác với cơ sở dữ liệu thông qua đối tượng DB\_Handler.

## 6. Lớp Transaction\_History

Đây là lớp đại diện cho lịch sử giao dịch, nơi lưu trữ thông tin về các giao dịch được thực hiện trên tài khoản ngân hàng.

Các thuộc tính bao gồm:

**serial\_no** - số serial của giao dịch.

**amount** - số tiền được giao dịch.

**type** - loại giao dịch (ví dụ: rút tiền, nộp tiền).

**date** - ngày thực hiện giao dịch.

**time** - thời gian thực hiện giao dịch.

**account\_num** - số tài khoản của người thực hiện giao dịch.

**recv\_acc\_num** - số tài khoản nhận tiền của người được nhận giao dịch.

**cheque\_num** - số séc được sử dụng để thực hiện giao dịch.

Các phương thức (methods) bao gồm:

Hai phương thức khởi tạo (constructor) để tạo mới đối tượng Transaction\_History mà không có tham số và có tham số với tất cả các thuộc tính được truyền vào.

Các phương thức getter để lấy giá trị của các thuộc tính.

## 7. Lớp AccountsFromFile

Lớp này có nhiệm vụ đọc dữ liệu từ một file chứa thông tin tài khoản của khách hàng, xử lý và tạo tài khoản cho từng khách hàng bằng cách sử dụng lớp Client và Manager.

Lớp AccountsFromFile có một phương thức là **importAccountsFromFile(String fileName)** để đọc file chứa thông tin tài khoản. Khi gọi phương thức này, lớp sẽ tạo một thread pool với số lượng luồng đọc file được giới hạn là 5. Sau đó, lớp sử dụng đối tượng BufferedReader để đọc từng dòng trong file. Mỗi dòng đọc được sẽ được đưa vào một Runnable task để xử lý dữ liệu, sau đó thêm vào thread pool để thực thi đa luồng.

Đối với từng dòng dữ liệu đọc được, lớp sẽ kiểm tra định dạng dữ liệu trước khi xử lý. Nếu dữ liệu không đúng định dạng, lớp sẽ ném ra một ngoại lệ RuntimeException. Sau khi kiểm tra định dạng dữ liệu, lớp sẽ xử lý dữ liệu và tạo một đối tượng Client mới với thông tin khách hàng được lấy từ file. Sau đó, lớp sẽ sử dụng đối tượng Manager để tạo tài khoản cho khách hàng đó với loại tài khoản được chỉ định trong dữ liệu.

## 8. Lớp BM\_System

Lớp BM\_System là một lớp kế thừa từ lớp JFrame trong thư viện Swing của Java. Nó đại diện cho hệ thống quản lý ngân hàng và cung cấp giao diện đồ họa cho người dùng. Lớp này khởi tạo một cửa sổ chương trình với tiêu đề "Bank Management System", biểu tượng ứng dụng và kích thước ban đầu là 800x500 pixel. Nó cũng tạo ra một đối tượng GUI để hiển thị các giao diện người dùng và một đối tượng Login\_Account để đăng nhập vào hệ thống. Cuối cùng, nó hiển thị cửa sổ chương trình lên màn hình và đóng cửa sổ khi người dùng tắt nó. Lớp BM\_System cũng chứa phương thức main() để khởi chạy ứng dụng.

## 9. Lớp DB\_Handler

Lớp này được sử dụng để xử lý các tương tác với cơ sở dữ liệu trong ứng dụng ngân hàng. Nó cung cấp các phương thức để thực hiện các hoạt động như đăng nhập, truy vấn thông tin khách hàng và tài khoản, kiểm tra trạng thái thẻ, và thực hiện giao dịch không thẻ. Các phương thức trong lớp này sử dụng JDBC để kết nối và truy cập cơ sở dữ liệu MySQL.

Lớp **DB\_Handler** có các thuộc tính sau:

- **url**: là URL để kết nối đến cơ sở dữ liệu MySQL.

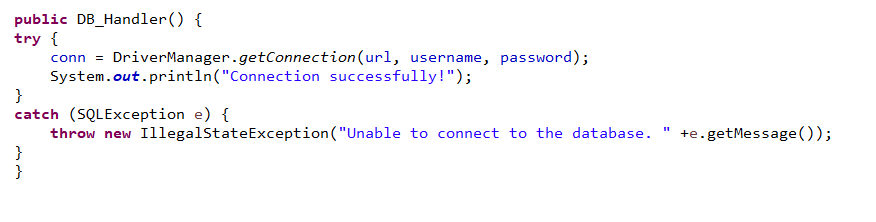
- **username**: là tên người dùng để kết nối đến cơ sở dữ liệu MySQL.

- **password**: là mật khẩu để kết nối đến cơ sở dữ liệu MySQL.

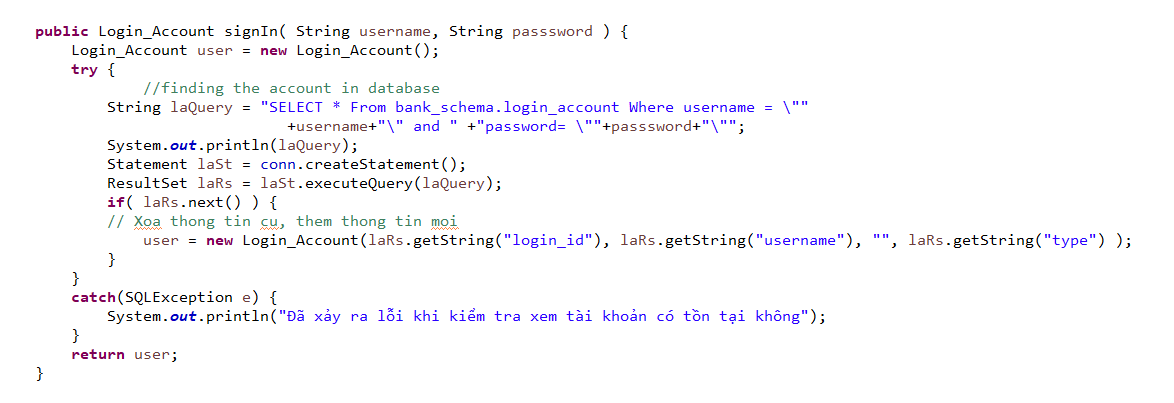
- **conn**: là một đối tượng Connection để kết nối đến cơ sở dữ liệu.

Các phương thức của **DB\_Handler** bao gồm:

**- DB\_Handler()**: Phương thức này tạo ra một đối tượng DB\_Handler và thiết lập kết nối đến cơ sở dữ liệu bằng cách sử dụng JDBC API và thông tin URL, tên đăng nhập và mật khẩu được cung cấp. Nếu không thể kết nối đến cơ sở dữ liệu, ngoại lệ SQLException sẽ được ném ra.



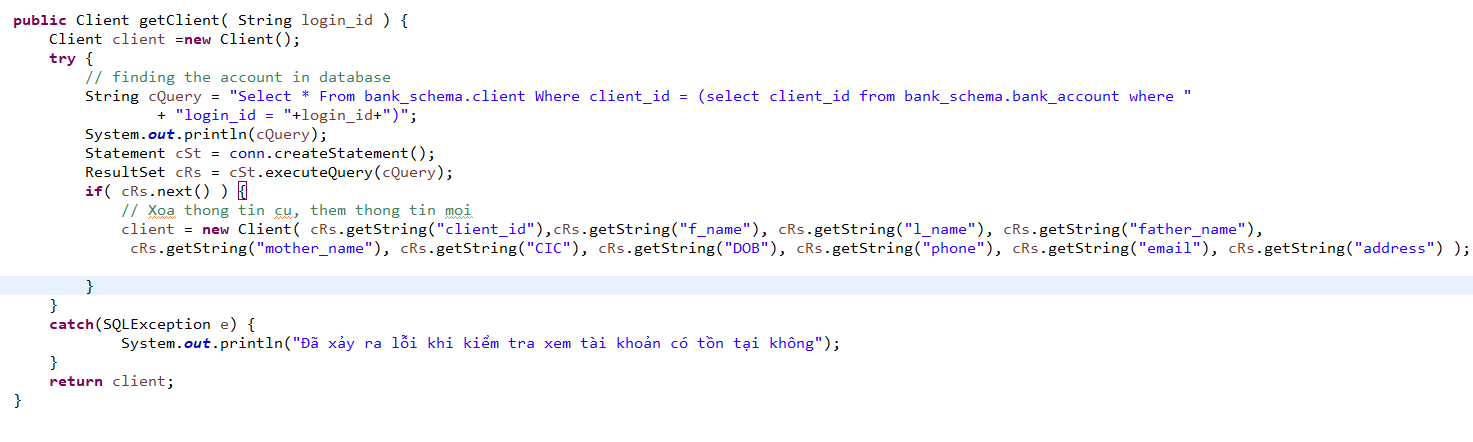
**- signIn( String username, String password )**: Phương thức **signIn()** trong lớp DB\_Handler có nhiệm vụ kiểm tra tài khoản đăng nhập trong cơ sở dữ liệu. Nó sử dụng các tham số đầu vào là tên đăng nhập và mật khẩu, tạo một đối tượng Login\_Account để đại diện cho người dùng đăng nhập thành công và trả về đối tượng đó. Phương thức này truy vấn cơ sở dữ liệu để tìm tài khoản với tên đăng nhập và mật khẩu tương ứng, nếu tìm thấy thì sẽ tạo đối tượng Login\_Account với các thông tin tài khoản từ kết quả truy vấn và trả về đối tượng đó. Nếu có lỗi trong quá trình truy vấn, phương thức sẽ in ra thông báo lỗi.

**- getClient( String login\_id) :** Phương thức getClient trong DB\_Handler có chức năng trả về thông tin của một khách hàng (client) trong cơ sở dữ liệu dựa trên login\_id. Đầu vào của phương thức là một chuỗi login\_id dạng String, đầu ra là một đối tượng Client.

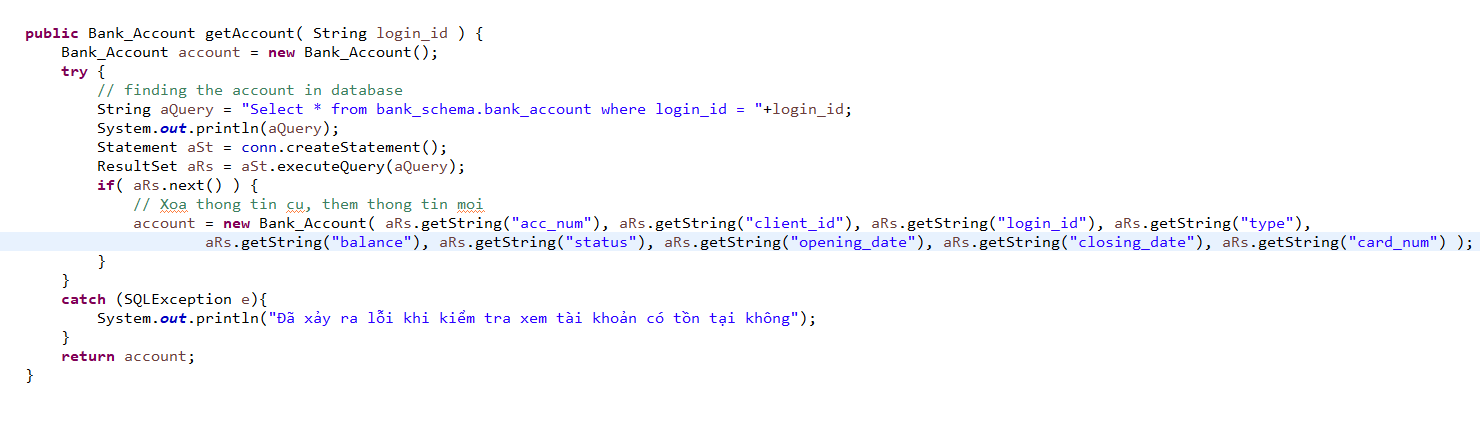
Phương thức này sẽ tạo ra một đối tượng Client rỗng, thực hiện truy vấn cơ sở dữ liệu để lấy thông tin khách hàng có client\_id tương ứng với login\_id được truyền vào.

Nếu tìm thấy khách hàng thì sẽ cập nhật thông tin vào đối tượng Client đã tạo, sau đó trả về đối tượng Client đã cập nhật.

Nếu không tìm thấy khách hàng, phương thức sẽ trả về đối tượng Client rỗng. Nếu có lỗi trong quá trình thực hiện truy vấn, phương thức sẽ in ra thông báo lỗi.



**- getAccount( String login\_id)**: Phương thức này tương tự như phương thức getClient ở trên nên chúng tôi sẽ không trình bày lại nữa.

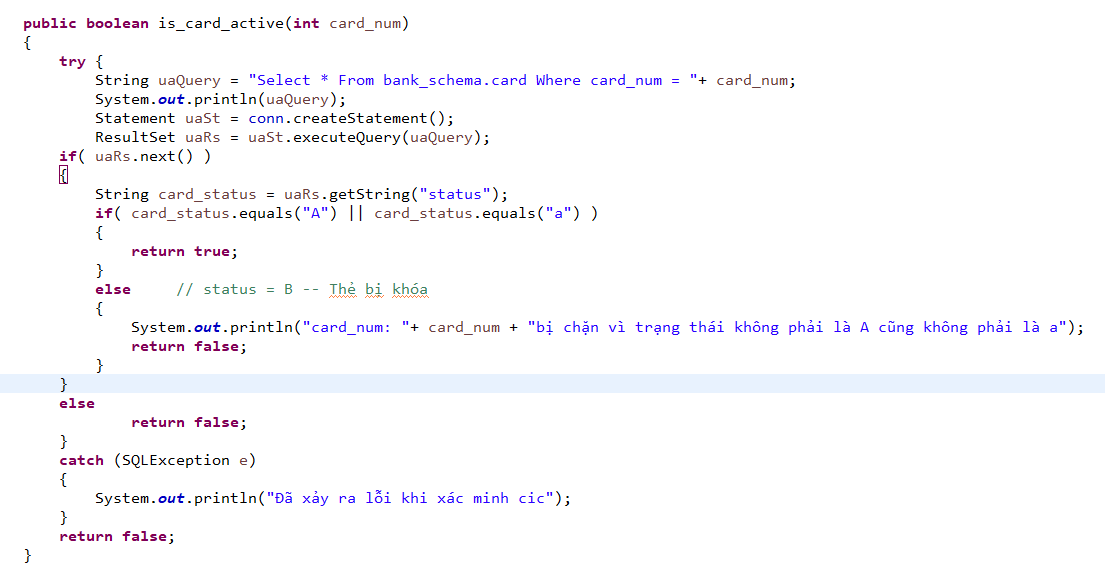


**- is\_card\_active(int card\_num)**: Phương thức is\_card\_active kiểm tra xem thẻ có số card\_num được cung cấp có đang hoạt động hay không.

Đầu tiên, nó thực hiện một truy vấn để tìm thẻ trong cơ sở dữ liệu. Nếu thẻ tồn tại, phương thức sẽ trả về true nếu trạng thái của thẻ là "A" hoặc "a", tức là thẻ đang hoạt động.

Nếu trạng thái là "B", tức là thẻ bị khóa, phương thức sẽ in một tin nhắn lỗi và trả về false.

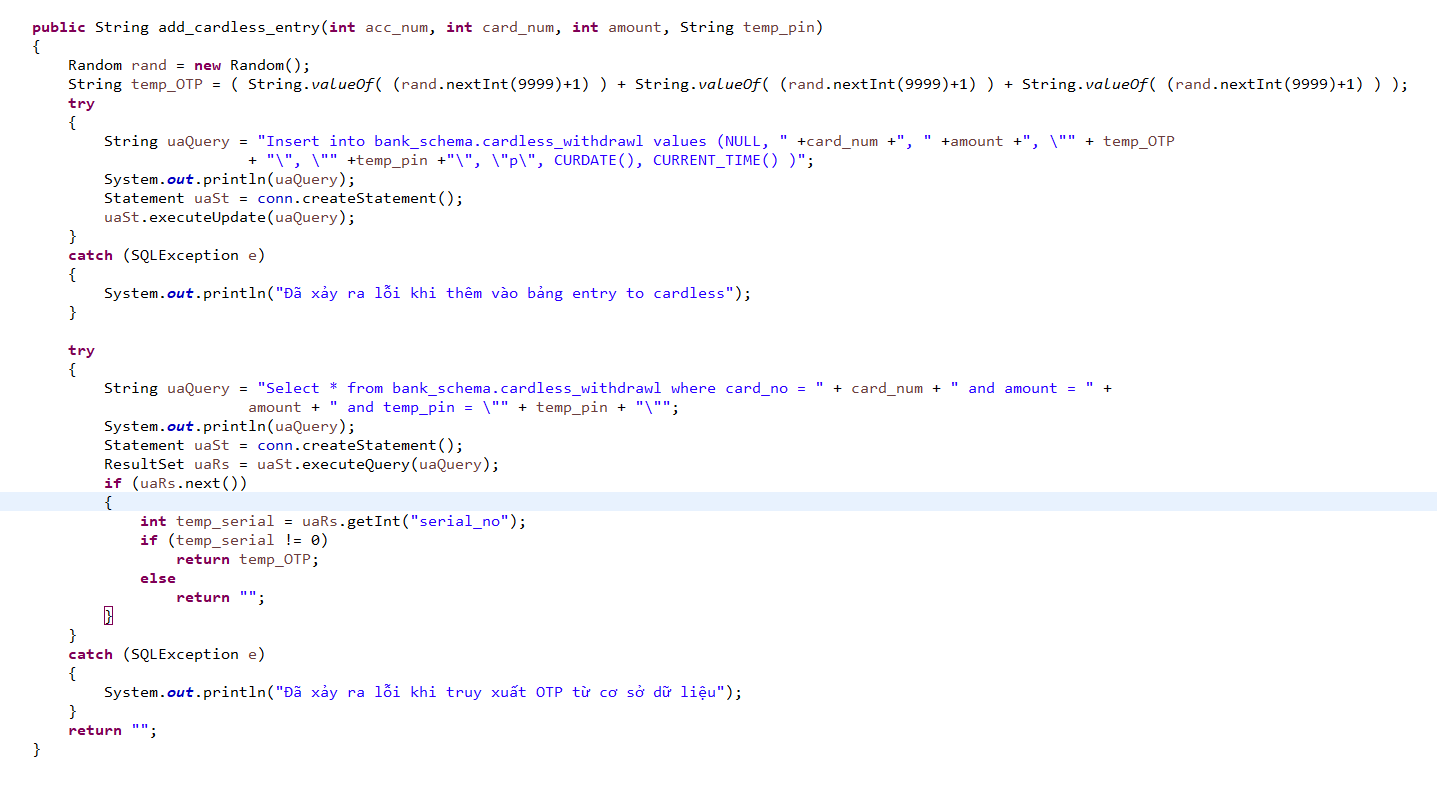
Nếu thẻ không tồn tại, phương thức sẽ trả về false. Nếu xảy ra lỗi trong khi thực hiện truy vấn, phương thức sẽ in một tin nhắn lỗi và trả về false.



**- add\_cardless\_entry(int acc\_num, int card\_num, int amount, String temp\_pin)** : Phương thức add\_cardless\_entry trong lớp DB\_Handler được sử dụng để thêm thông tin vào bảng cardless\_withdrawl trong cơ sở dữ liệu. Thông tin bao gồm card\_num, amount, temp\_OTP và temp\_pin.

Sau khi thêm thông tin, phương thức sẽ truy xuất thông tin vừa được thêm vào để lấy temp\_OTP và trả về cho người dùng. temp\_OTP được tạo ra bằng cách sử dụng một đối tượng Random để tạo ra ba số ngẫu nhiên trong khoảng từ 1 đến 9999 và ghép lại thành một chuỗi.

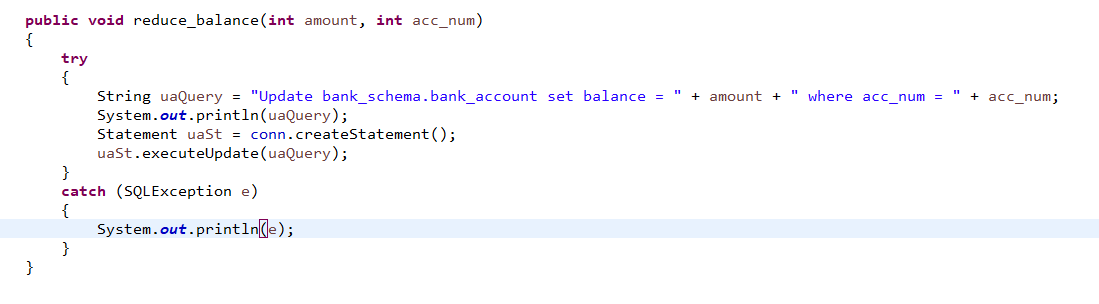
Nếu có lỗi xảy ra trong quá trình thêm hoặc truy xuất thông tin, phương thức sẽ trả về một chuỗi rỗng.



**- reduce\_balance(int amount, int acc\_num)**: Phương thức reduce\_balance trong DB\_Handler dùng để cập nhật số dư tài khoản ngân hàng của khách hàng sau khi thực hiện giao dịch rút tiền hoặc chuyển khoản.

Phương thức này nhận đầu vào là số tiền amount và số tài khoản acc\_num cần cập nhật số dư. Sau đó, phương thức sử dụng câu lệnh SQL để cập nhật số dư trong cơ sở dữ liệu cho tài khoản acc\_num. Cụ thể, câu lệnh UPDATE được sử dụng để cập nhật số dư balance của tài khoản acc\_num thành amount.

Nếu trong quá trình cập nhật số dư xảy ra lỗi, phương thức sẽ in ra thông báo lỗi. Không có giá trị trả về từ phương thức này.



**- get\_client\_id(int acc\_num):** Phương thức get\_client\_id trong DB\_Handler có chức năng lấy client\_id của tài khoản ngân hàng thông qua số tài khoản. Nó thực hiện truy vấn SQL để tìm tài khoản trong cơ sở dữ liệu và trả về giá trị client\_id tương ứng. Nếu tài khoản không tồn tại, hoặc có lỗi khi truy vấn, phương thức sẽ trả về -1.

Cụ thể, phương thức này thực hiện các bước sau:

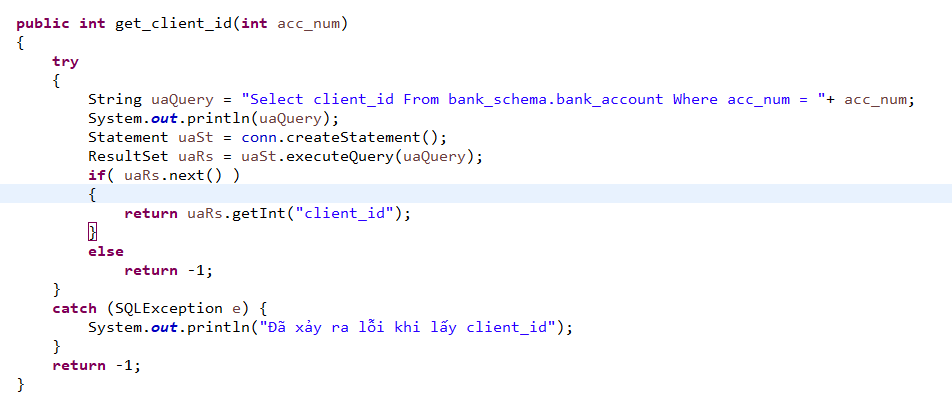
+ Tạo một truy vấn SQL để lấy client\_id của tài khoản với số tài khoản đã cho.

+ Thực thi truy vấn và lấy kết quả trả về dưới dạng ResultSet.

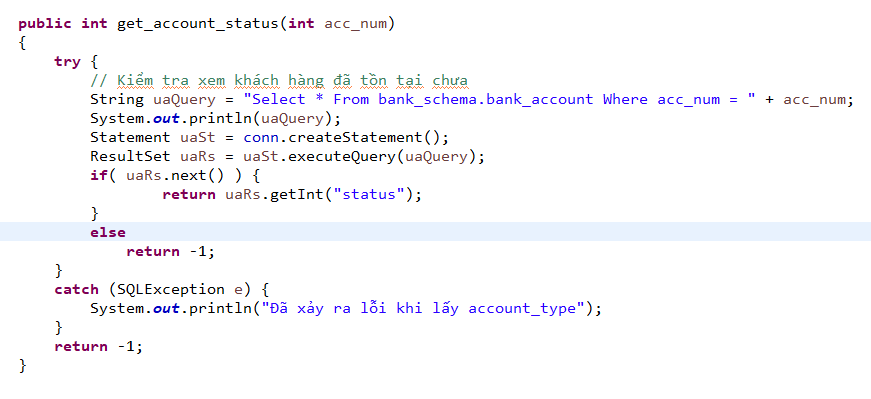
+ Nếu ResultSet chứa ít nhất một bản ghi, lấy giá trị client\_id từ bản ghi đó và trả về.

+ Nếu không có bản ghi nào trong ResultSet, trả về -1.

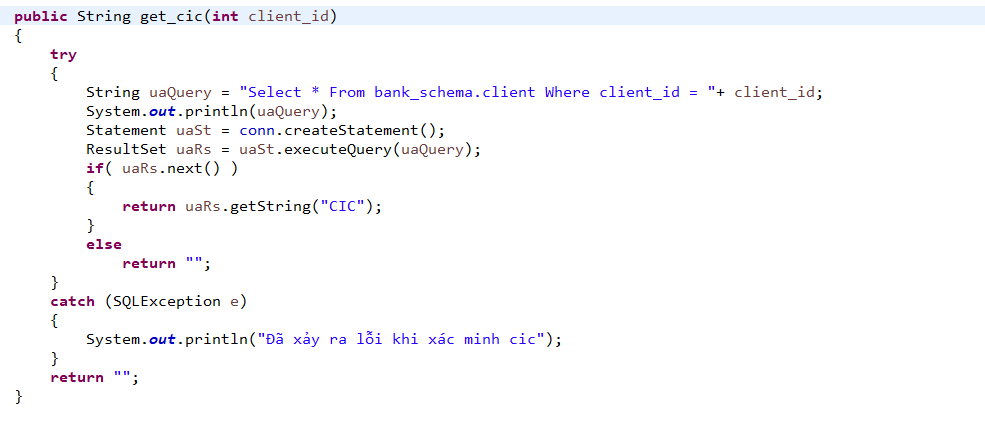
+ Nếu có lỗi khi thực hiện truy vấn SQL, hiển thị thông báo lỗi và trả về -1.

****

**- get\_account\_status(int acc\_num)**: Tương tự như phương thức get\_client\_id trên nên chúng tôi sẽ không trình bày lại nữa.

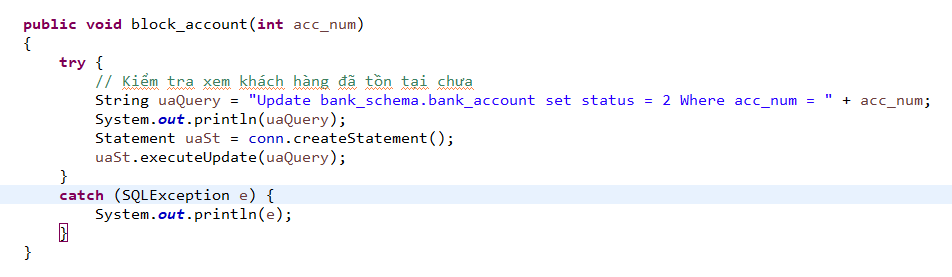


**- String get\_cic(int client\_id)**: Tương tự như 2 phương thức trên nên chúng tôi sẽ không trình bày lại nữa.

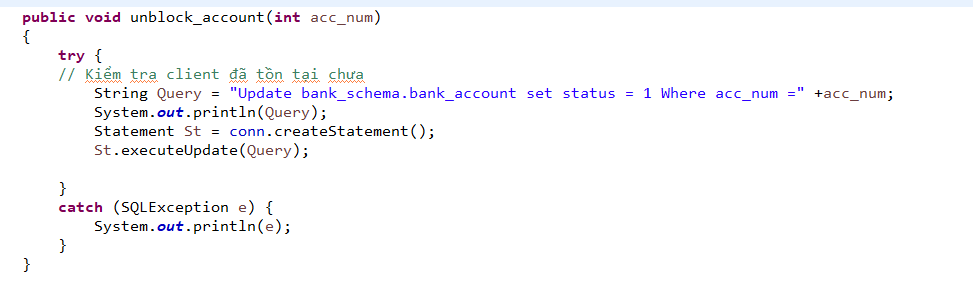


**- block\_account(int acc\_num)**: Phương thức block\_account(int acc\_num) trong DB\_Handler được sử dụng để khóa tài khoản ngân hàng của một khách hàng với mã số tài khoản acc\_num được chỉ định. Cụ thể, phương thức này thực hiện câu lệnh SQL để cập nhật trạng thái của tài khoản ngân hàng trong bảng bank\_account của cơ sở dữ liệu.

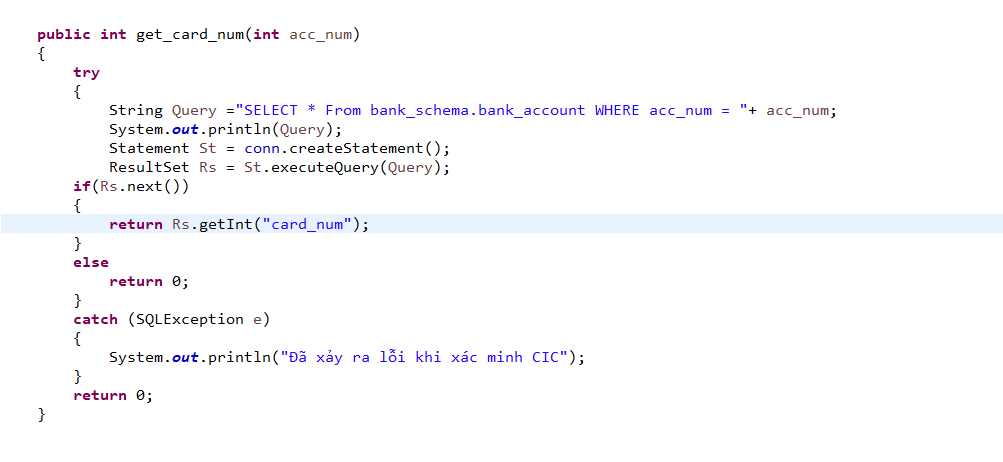
Trong đó, câu lệnh SQL được sử dụng để cập nhật trạng thái tài khoản thành 2 (đã bị khóa) và chỉ áp dụng cho tài khoản có mã số acc\_num tương ứng. Nếu câu lệnh SQL thực thi không thành công, phương thức sẽ in ra một thông báo lỗi tương ứng.



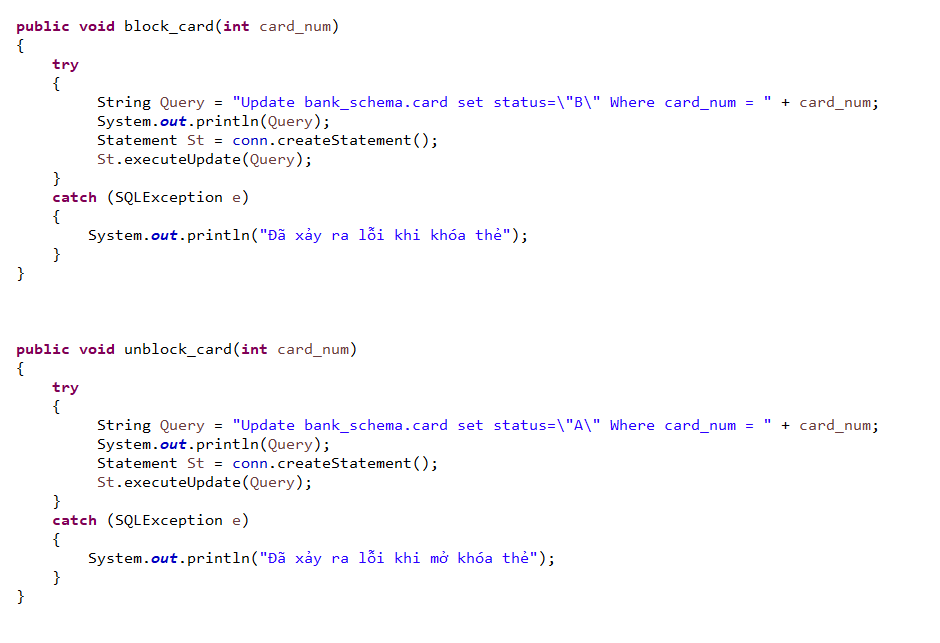
**- unblock\_account(int acc\_num)**: Tương tự như phương thức block\_account chỉ khác là cập nhật lại trạng thái tài khoản là 1(đã mở).



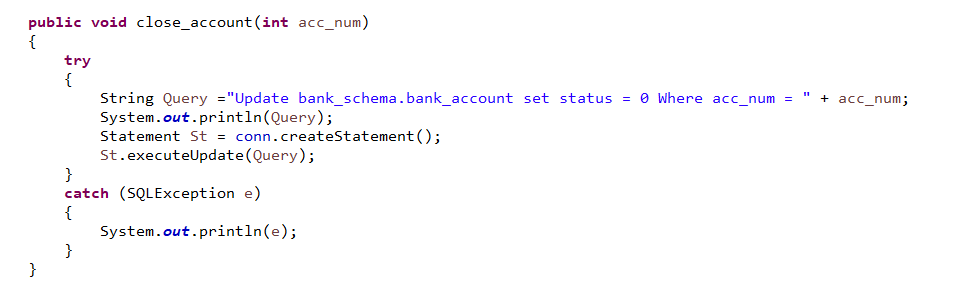
**- get\_card\_num(int acc\_num)**: Tương tự như các get\_client\_id, get\_account\_status… ở trên phương thức này tạo truy vấn để lấy số thẻ. Nếu không có bản ghi nào sẽ trả về 0. Nếu lỗi xảy ra sẽ in ra lỗi và trả về 0.



- **block\_card(int card\_num)** và **unblock\_card(int card\_num)**: Tương tự như 2 phương thức block/unblock\_account. Các phương thức này sẽ cập nhật lại status = B (khóa thẻ) hoặc status=A (Mở khóa thẻ).



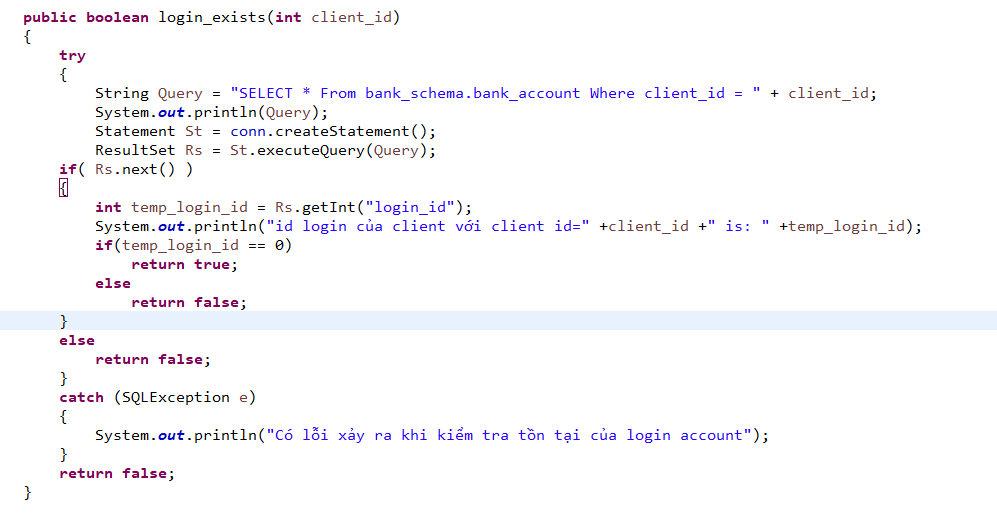
**- close\_account(int acc\_num)**: Phương thức này sẽ đóng tài khoản(vĩnh viễn) bằng cách cập nhật status trong bảng bank\_account trong CSDL bằng 0.



**- login\_exists(int client\_id):** Phương thức login\_exists trong DB\_Handler có nhiệm vụ kiểm tra xem một khách hàng đã có tài khoản đăng nhập hay chưa bằng cách kiểm tra xem có bản ghi nào trong bảng bank\_account có client\_id trùng với client\_id đầu vào và login\_id bằng 0 hay không.

Nếu có, tức là khách hàng chưa có tài khoản đăng nhập và phương thức sẽ trả về true.

Nếu không có bản ghi phù hợp hoặc login\_id khác 0, phương thức sẽ trả về false. Nếu trong quá trình thực thi xảy ra lỗi, phương thức sẽ in ra thông báo lỗi và trả về false.



**- verify\_cic(int client\_id, String cic)**: Phương thức verify\_cic trong DB\_Handler nhận vào 2 tham số là client\_id và cic và trả về giá trị boolean xác định xem cic có đúng cho client\_id đã cho hay không.

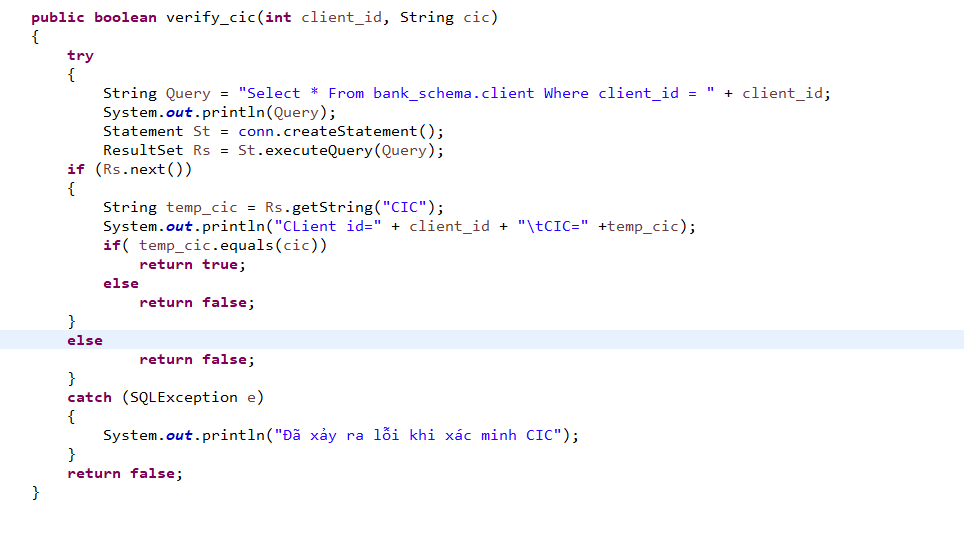
Trong phương thức này, đầu tiên nó tạo câu truy vấn SQL để lấy thông tin khách hàng với client\_id tương ứng.

Sau đó nó thực thi câu truy vấn và lấy kết quả trả về. Nếu kết quả trả về có chứa thông tin của khách hàng đó thì nó lấy giá trị của trường CIC và so sánh với giá trị cic đã cho.

Nếu giá trị bằng nhau, thì phương thức sẽ trả về true, ngược lại trả về false.

Nếu kết quả truy vấn không có bất kỳ khách hàng nào, phương thức sẽ trả về false.

Nếu trong quá trình thực thi câu truy vấn xảy ra lỗi SQL, phương thức sẽ bắt lỗi và in ra thông báo lỗi tương ứng và trả về false.



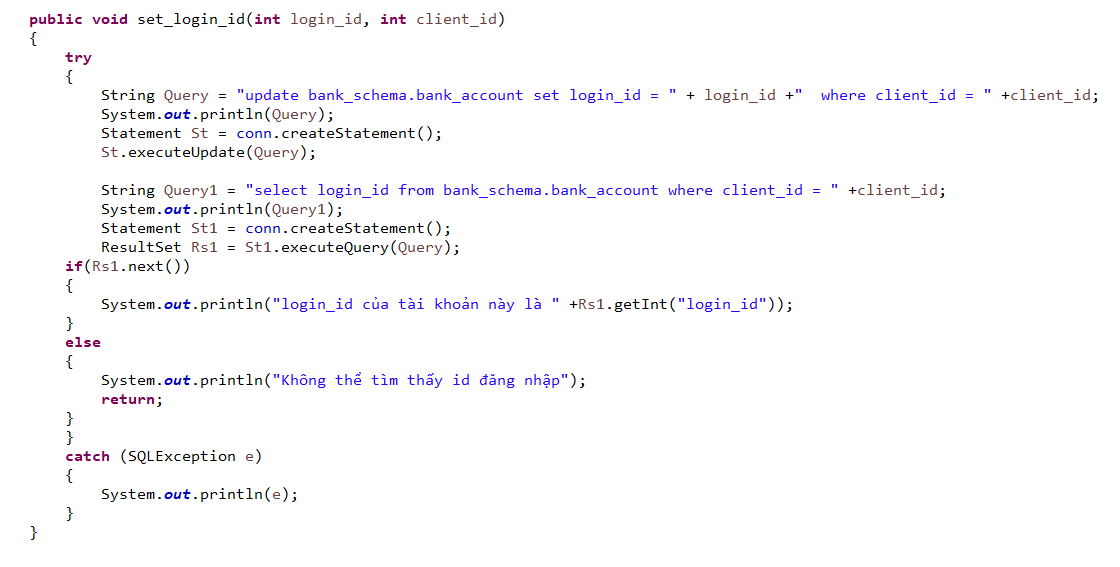
**- create\_login(String username, String password)**: Phương thức này để thêm tài khoản đăng nhập vào bảng login\_account trong CSDL. Nếu có lỗi sẽ ném ra thông báo lỗi.

Sau khi thêm xong sẽ kiểm tra lại xem đã thêm thành công chưa bằng cách truy vấn lại username và password đó. Nếu có sẽ trả về id\_login, nếu không sẽ trả về -1.



**- set\_login\_id(int login\_id, int client\_id)**: Phương thức này sẽ cập nhật login\_id trong bảng bank\_account với client\_id tương ứng.

Sau khi cập nhật xong sẽ kiểm tra lại xem đã cập nhật thành công chưa bằng cách truy vấn ra login\_id của client\_id tương ứng.



**- DB\_CreateAccount( Client new\_client, String aType)**: Phương thức DB\_CreateAccount được sử dụng để tạo tài khoản mới cho khách hàng. Phương thức này có các tham số là một đối tượng Client mới được truyền vào và một chuỗi loại tài khoản aType.

Phương thức đầu tiên kiểm tra xem khách hàng đã tồn tại trong cơ sở dữ liệu hay chưa bằng cách tìm kiếm theo số CIC của khách hàng. Nếu không tìm thấy bản ghi trùng lặp, phương thức tiếp tục thêm thông tin khách hàng mới vào bảng client. Sau đó, nó tìm kiếm client\_id cho khách hàng mới được thêm vào cơ sở dữ liệu và sử dụng nó để thêm thông tin tài khoản mới vào bảng bank\_account.

Nếu khách hàng đã tồn tại trong cơ sở dữ liệu, phương thức trả về giá trị 2 để báo hiệu rằng tài khoản không thể được tạo. Nếu quá trình tạo tài khoản gặp lỗi, phương thức sẽ trả về giá trị 0. Phương thức trả về giá trị 1 để báo hiệu rằng tài khoản đã được tạo thành công.



**- TransferMoney( Client client, String rAccNum, int amount)**: Phương thức TransferMoney trong DB\_Handler được sử dụng để thực hiện giao dịch chuyển tiền từ tài khoản của một khách hàng đến tài khoản của một người nhận khác.

Phương thức này có các đối số đầu vào là **Client client**, **String rAccNum**, **int amount**. Trong đó, **client** là đối tượng khách hàng, **rAccNum** là số tài khoản của người nhận và **amount** là số tiền cần chuyển.

Các bước thực hiện chuyển tiền như sau:

- Kiểm tra xem tài khoản của người nhận đã tồn tại chưa bằng câu lệnh SQL: SELECT acc\_num, balance From bank\_schema.bank\_account Where acc\_num = rAccNum and status = 1.

- Nếu tài khoản của người nhận tồn tại, lấy số dư của người nhận và kiểm tra xem tài khoản của khách hàng có đủ số dư để thực hiện giao dịch hay không.

- Nếu khách hàng có đủ số dư, trừ số tiền cần chuyển từ tài khoản của khách hàng và cộng số tiền đó vào tài khoản của người nhận. Cập nhật lại số dư của hai tài khoản bằng câu lệnh SQL: Update bank\_schema.bank\_account Set balance = snd\_balance where client\_id=client.getClientID() và Update bank\_schema.bank\_account Set balance = recv\_balance where acc\_num = rAccNum.

- Thêm giao dịch vào lịch sử giao dịch bằng câu lệnh SQL: Insert into bank\_schema.transaction\_history values(NULL, amount, "transfer", CURDATE(), CURRENT\_TIME(), snd\_acc\_num, rAccNum , NULL ).

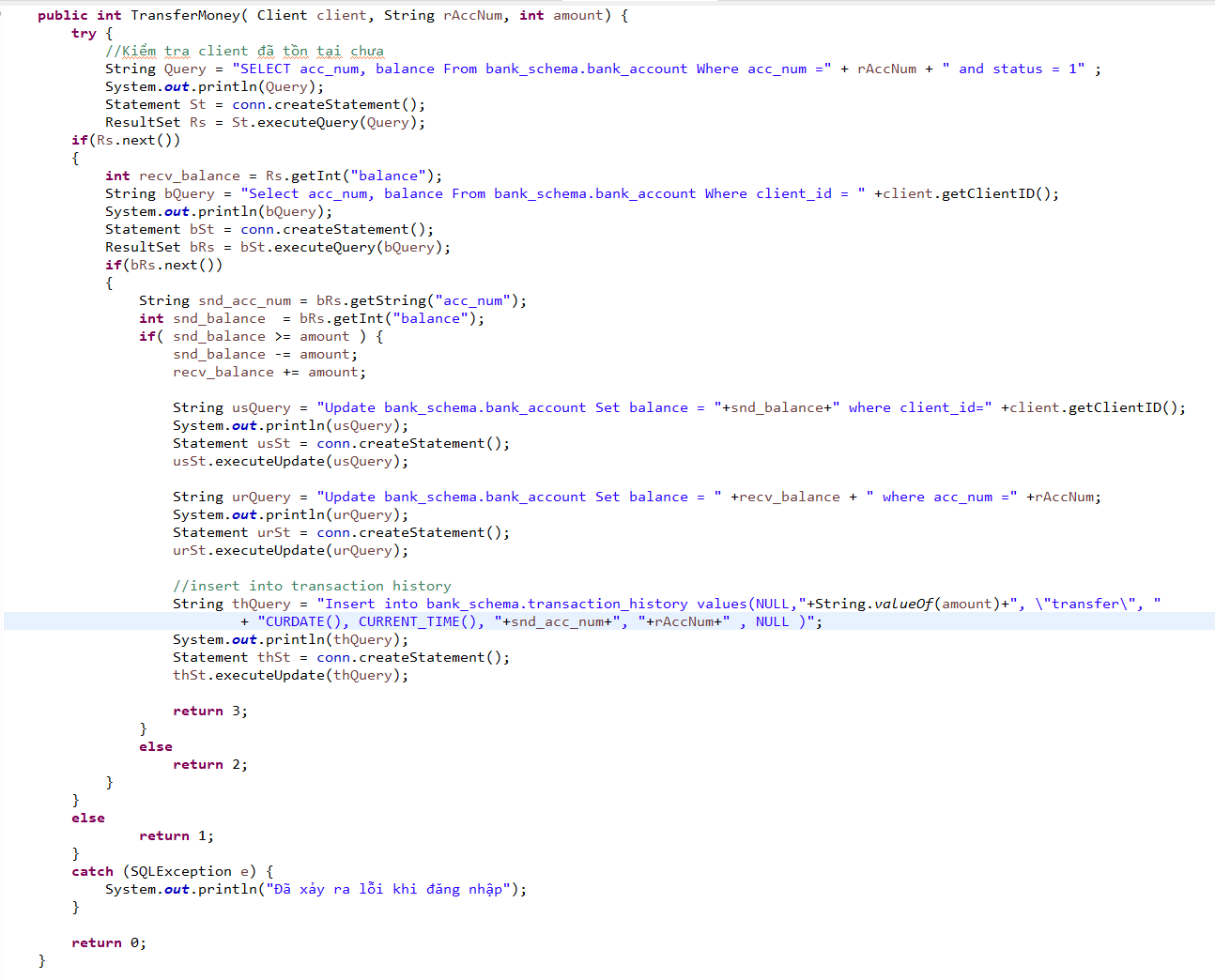
- Cuối cùng, phương thức trả về giá trị khác nhau tùy thuộc vào kết quả kiểm tra:

+ Nếu tài khoản người nhận không tồn tại, trả về giá trị là 1.

+ Nếu tài khoản khách hàng không đủ số dư để thực hiện giao dịch, trả về giá trị là 2.

+ Nếu giao dịch thành công, trả về giá trị là 3.

- Nếu xảy ra lỗi sẽ thông báo lỗi và trả về giá trị là 0.

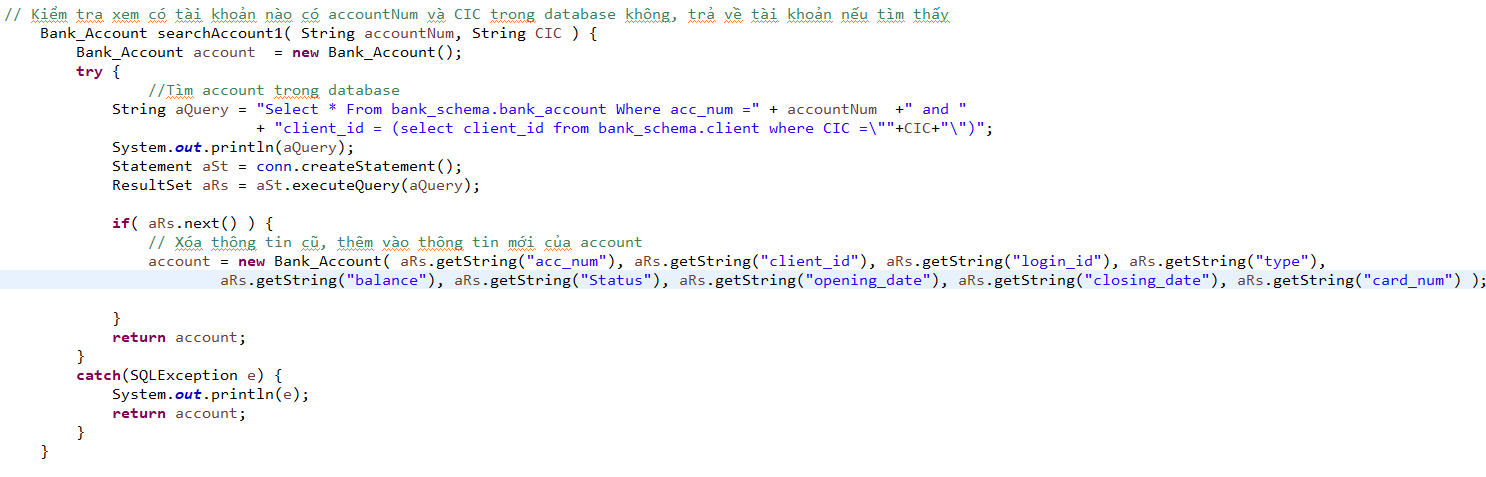


**- searchAccount1(String accountNum,String CIC)**: Phương thức searchAccount1 trong DB\_Handler được sử dụng để tìm kiếm thông tin tài khoản trong database dựa trên số tài khoản và mã số CIC của khách hàng.

Đầu tiên là thực hiện truy vấn để tìm các thông tin Account trong bảng bank\_account với điều kiện acc\_num=accountNum và client\_id = (Truy vấn con tìm client\_id trong bảng client có CIC tương ứng).

Nếu có thì sẽ tạo đối tượng account và trả về account đó.

Nếu lỗi xảy ra sẽ thông báo lỗi và trả về account.

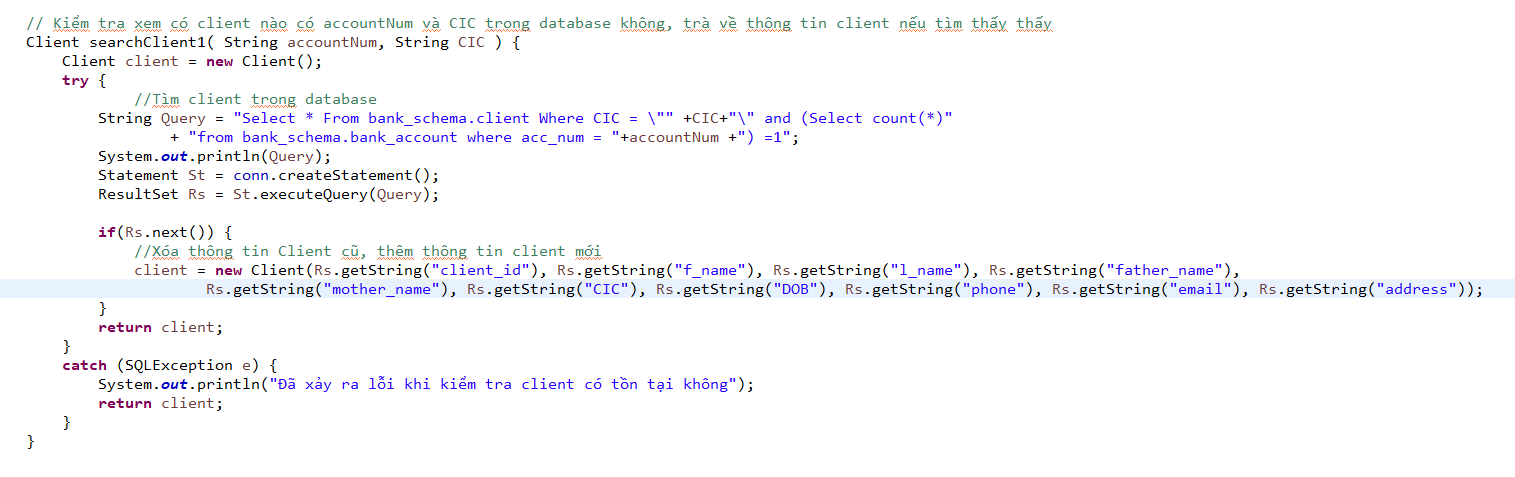


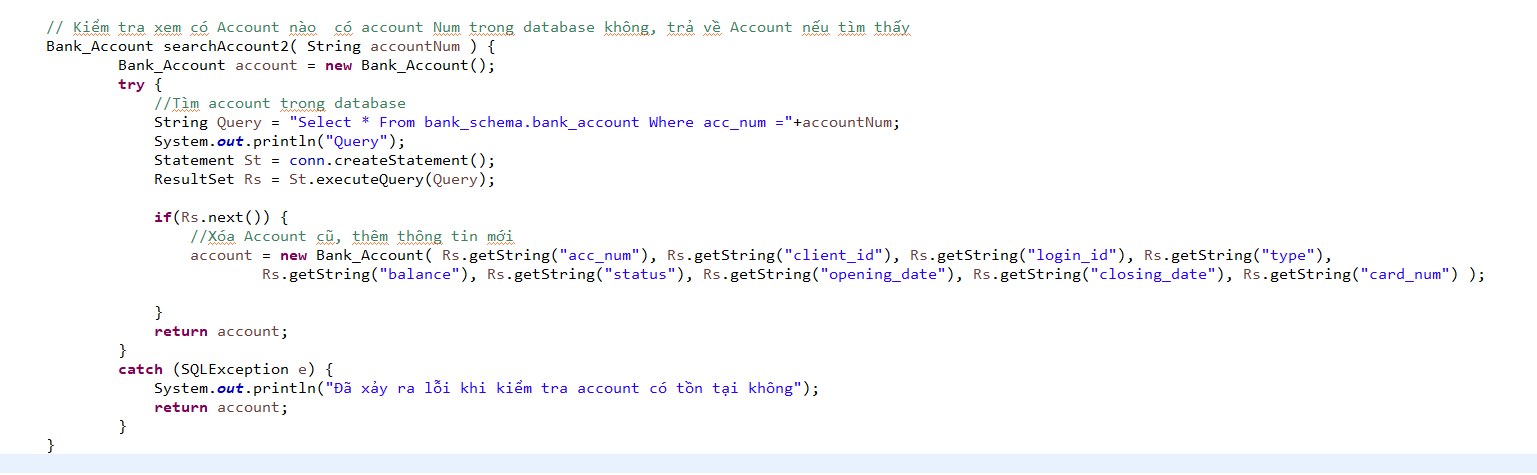
- **searchClient1( String accNum, String CIC)**: Phương thức searchClient1 trong DB\_Handler được sử dụng để tìm kiếm thông tin khách hàng trong database dựa trên số tài khoản và mã số CIC của khách hàng.

Tương tự như trên thực hiện truy vấn để tìm các thông tin Client trong bảng Client với điều kiện CIC tương ứng và số lượng tài khoản chỉ bằng 1).

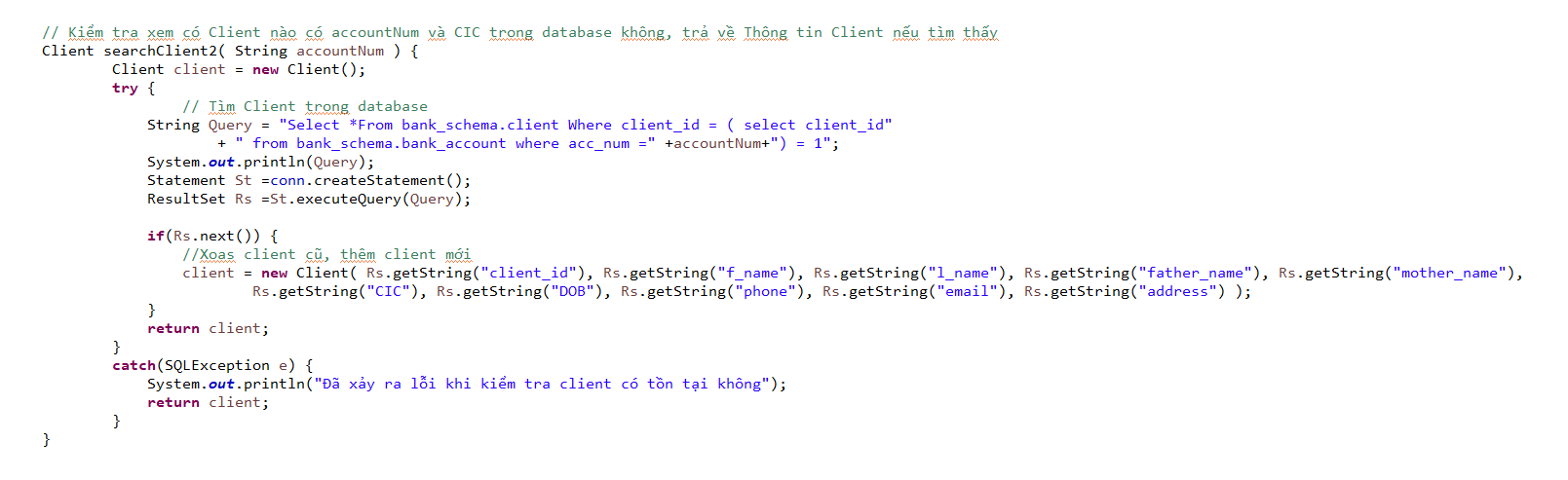
Nếu tìm thấy sẽ tạo đối tượng client và trả về client đó.

Nếu xảy ra lỗi sẽ thông báo lỗi và trả về client.



**- searchAccount2(String accountNum)**: tương tự như phương thức searchAcount1 nhưng ở đây chỉ tìm theo accountNum. Dùng cho bên Manager sử dụng. 

**-searchClient2(String accountNum):** tương tự như phương thức searchclient1 nhưng ở đây chỉ tìm theo accountNum. Dùng cho bên Manager sử dụng.



**-** **updateBalance( Bank\_Account account, int balance, int t )**: Phương thức updateBalance trong DB\_Handler được sử dụng để cập nhật số dư tài khoản trong database. Nó nhận vào một đối tượng Bank\_Account, số dư cần cập nhật và một biến t để xác định nếu đó là giao dịch rút tiền.

Cụ thể, phương thức này thực hiện các bước sau:

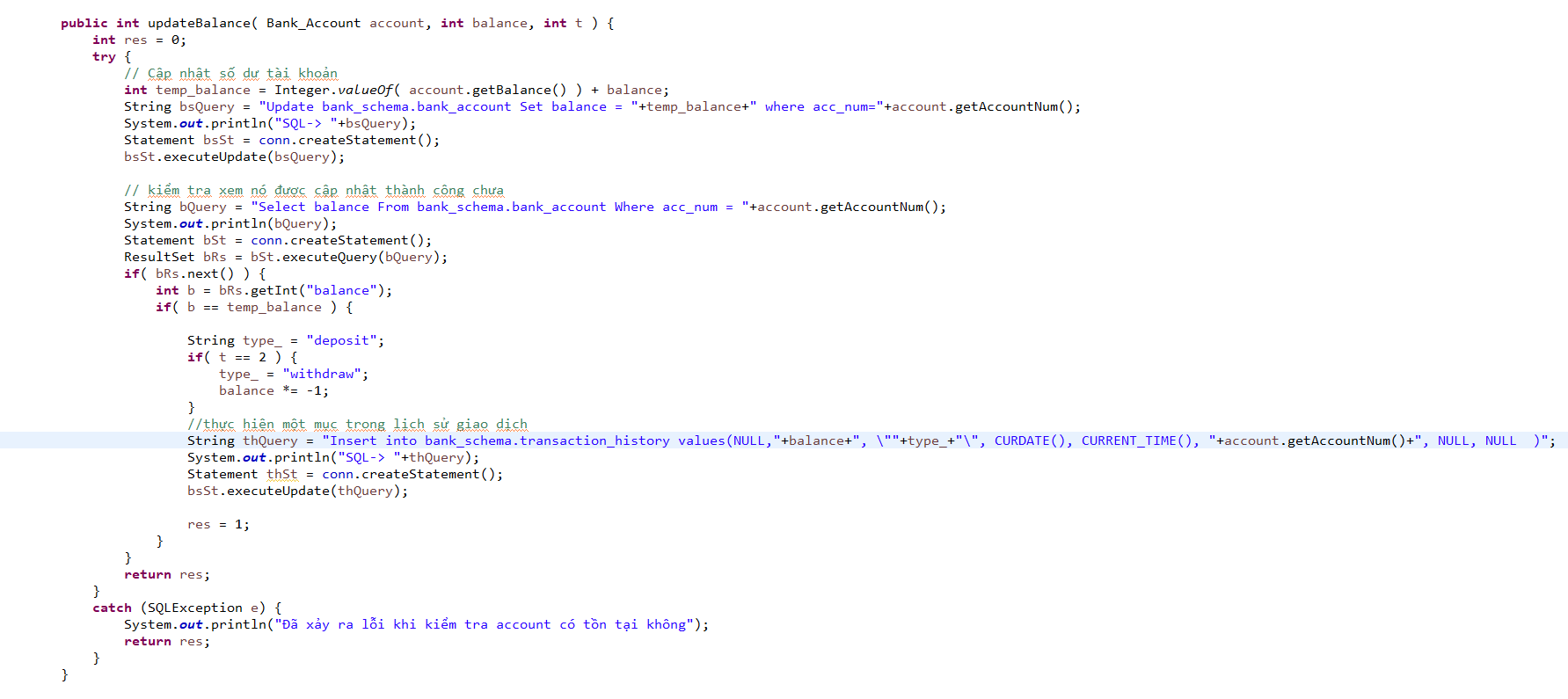
- Tính số dư mới của tài khoản bằng cách cộng số dư hiện tại của tài khoản với số dư cần cập nhật.

- Thực hiện truy vấn SQL để cập nhật số dư mới của tài khoản trong database.

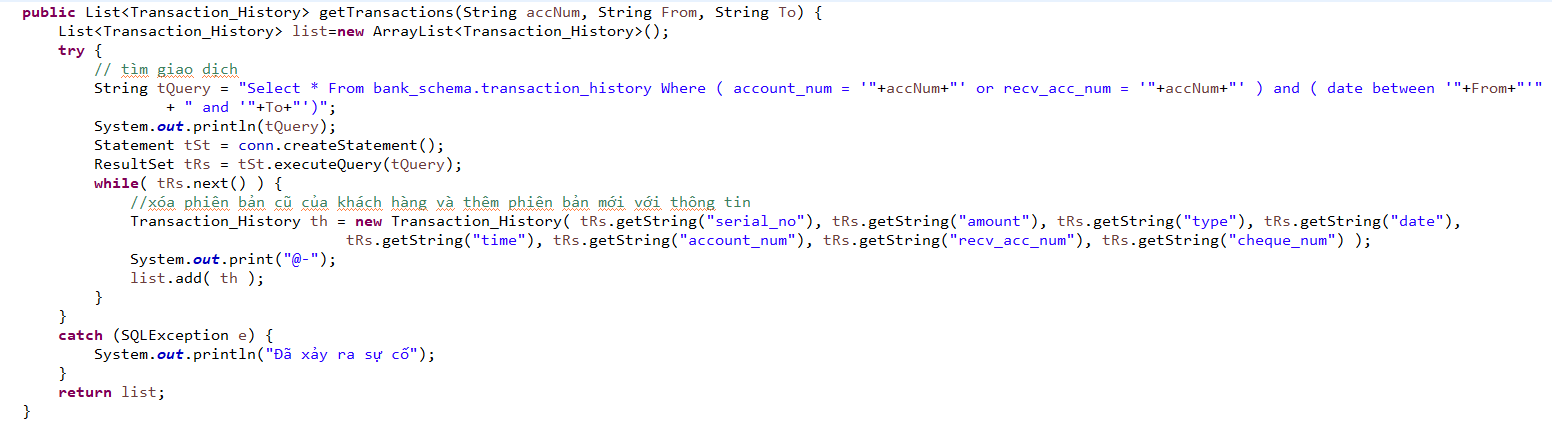
- Thực hiện truy vấn SQL để kiểm tra xem số dư của tài khoản đã được cập nhật thành công hay không.

- Nếu số dư của tài khoản đã được cập nhật thành công, phương thức thực hiện một mục trong lịch sử giao dịch của tài khoản (với thông tin về số tiền, loại giao dịch và ngày giờ).

- Trả về giá trị 1 nếu cập nhật số dư thành công và 0 nếu có lỗi xảy ra trong quá trình cập nhật.



**- getTransactions(String accNum, String From, String To)**: Phương thức này thực hiện truy vấn vào bảng transaction\_history trong CSDL trả về danh sách lịch sử giao dịch tương ứng với Số tài khoản và khoảng thời tương ứng.



**-** **chequeDeposit( String accNum, String chequeNum, int amount ):** Phương thức chequeDeposit trong DB\_Handler thực hiện việc gửi tiền vào tài khoản ngân hàng từ một số Séc. Phương thức này có các tham số đầu vào bao gồm số tài khoản accNum, số Séc chequeNum và số tiền amount được gửi. Các bước thực hiện của phương thức như sau:

Bước 1: Tìm tài khoản trong cơ sở dữ liệu bằng cách truy vấn từ khóa SELECT với điều kiện là số tài khoản trùng với tham số đầu vào accNum.

Bước 2: Nếu tìm thấy tài khoản, kiểm tra trạng thái tài khoản có phải là closed hay blocked hay không. Nếu tài khoản có trạng thái này, trả về giá trị tương ứng (res = 4 nếu closed, res = 3 nếu blocked). Nếu không, tiếp tục bước 3.

Bước 3: Tính toán số dư mới của tài khoản bằng cách cộng thêm số tiền amount với số dư hiện có của tài khoản. Sau đó, cập nhật số dư mới này vào cơ sở dữ liệu bằng câu lệnh UPDATE với điều kiện là số tài khoản trùng với tham số đầu vào accNum.

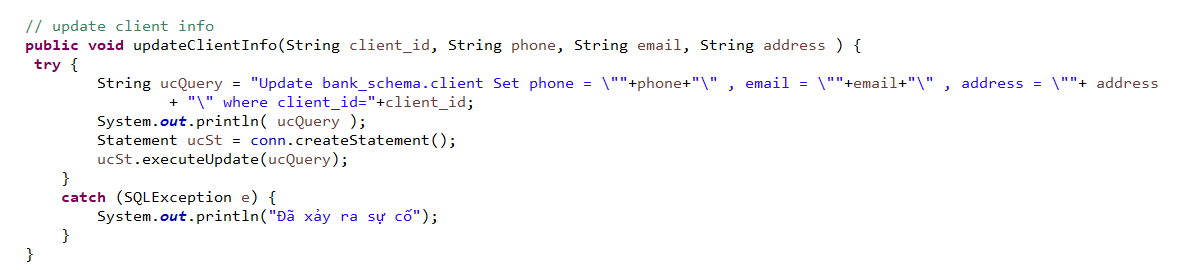
Bước 4: Thêm một mục vào lịch sử giao dịch của tài khoản bằng cách thực hiện câu lệnh INSERT vào bảng transaction\_history với các thông tin như số tiền gửi (amount), loại giao dịch ("deposit"), ngày và giờ giao dịch (sử dụng hàm CURDATE() và CURRENT\_TIME()), số tài khoản (accNum) và số Séc (chequeNum, có giá trị NULL nếu không được cung cấp).

Bước 5: Trả về giá trị res = 1 để thể hiện giao dịch thành công.

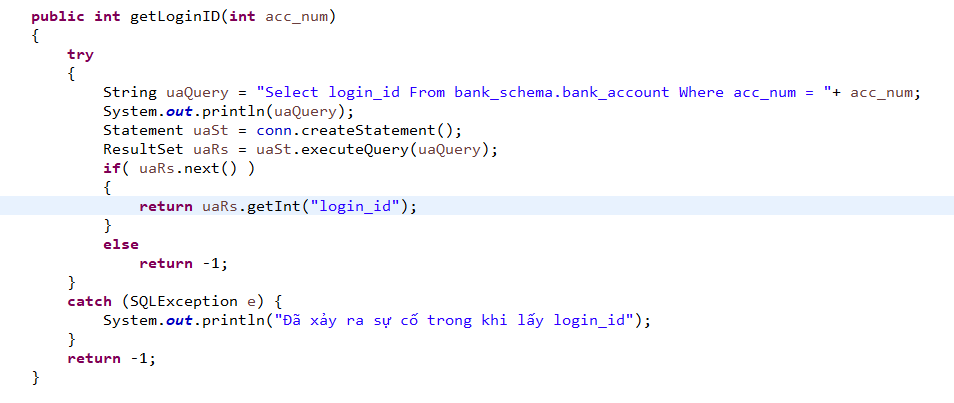
Nếu trong quá trình thực hiện có bất kỳ lỗi nào, phương thức sẽ in ra thông báo lỗi và trả về giá trị res = 0 để thể hiện giao dịch không thành công.



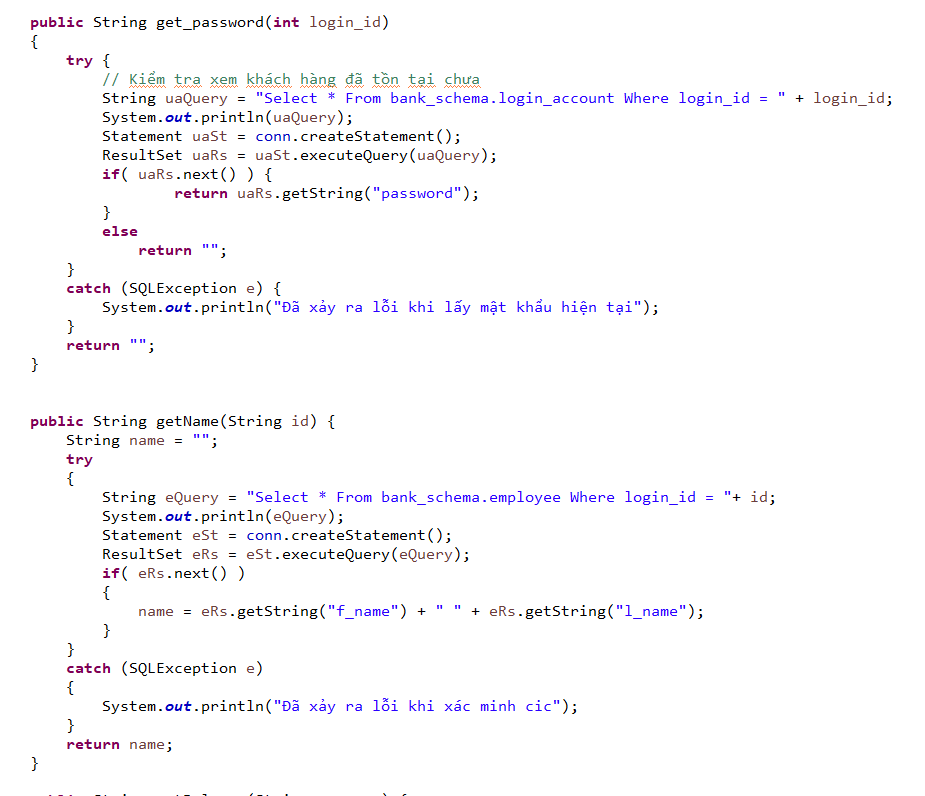
**-** **updateClientInfo(String client\_id, String phone, String email, String address)**: Phương thức này cập nhật lại thông tin phone, email, address của client tương ứng.

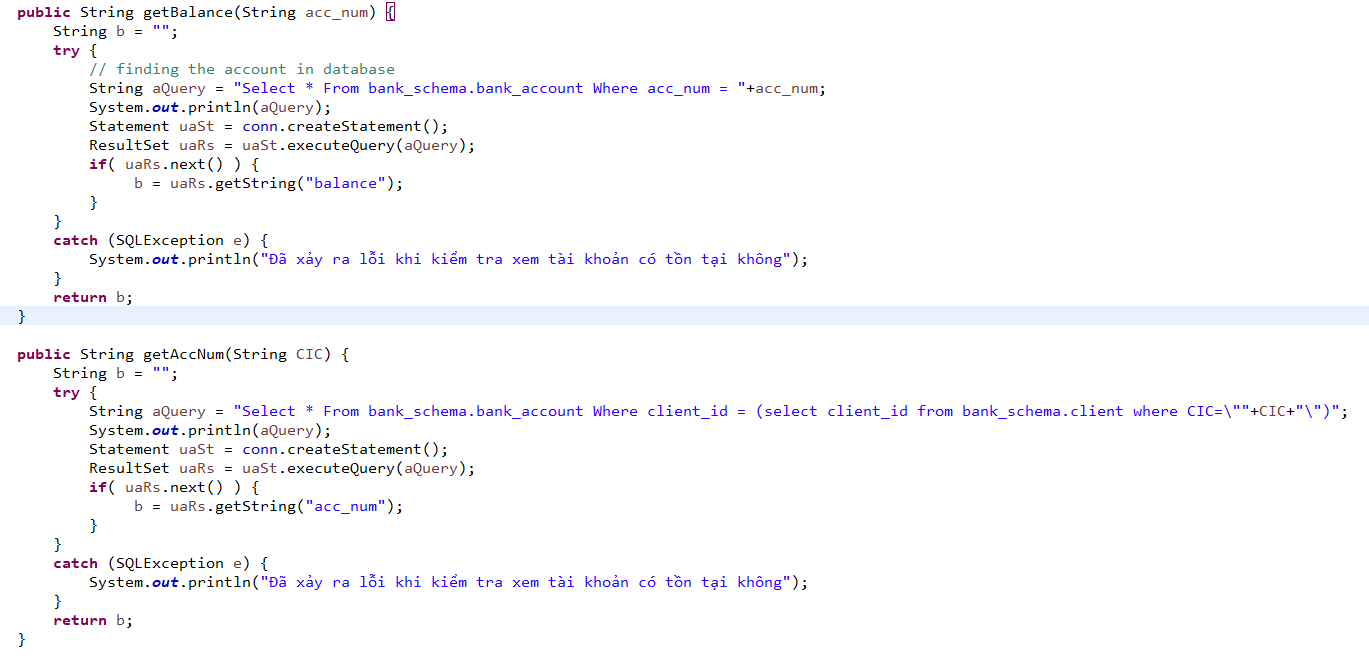


**-** **getLoginID(int acc\_num)**: phương thức này trả về Id\_login tương ứng với số tài khoản trong bảng bank\_account. Trả về -1 nếu không tìm thấy hoặc xảy ra lỗi.

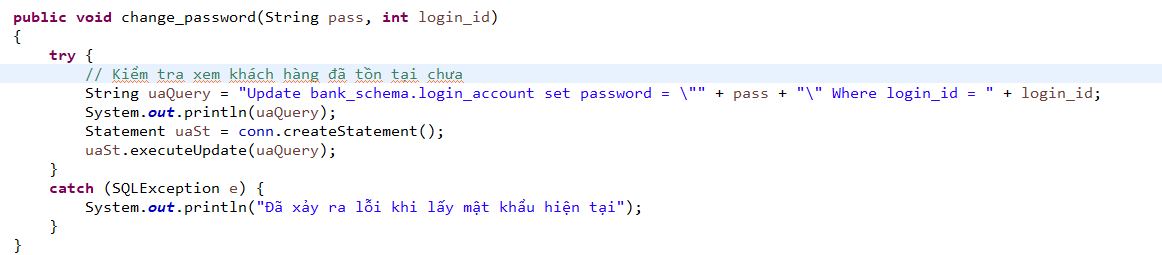


- Các phương thức **get\_password(int login\_id), getName(String id), getBalance(String acc\_num), getAccNum(String CIC)**: Tương tự trả về lần lượt password, Name, Balance, AccNum.





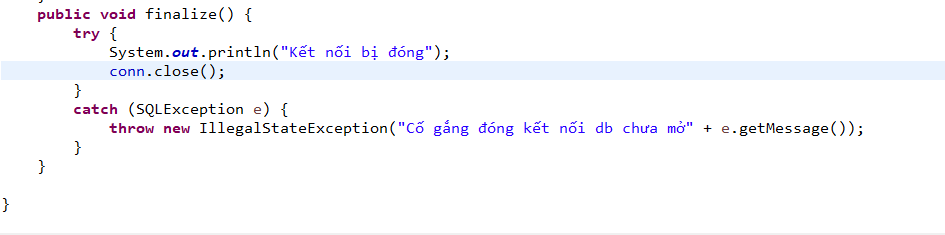
**- change\_password(String pass, int login\_id)**: Phương thức này cập nhật lại mật khẩu trong bảng login\_account với login\_id tương ứng.



**-finalize()**: Phương thức finalize() là một phương thức được kế thừa từ lớp Object trong Java và nó được sử dụng để giải phóng tài nguyên hoặc thực hiện các hành động cuối cùng trước khi đối tượng bị thu hồi bởi bộ dọn rác.

Trong trường hợp của lớp DB\_Handler, khi một đối tượng được tạo ra để kết nối với cơ sở dữ liệu, phương thức finalize() được sử dụng để đóng kết nối với cơ sở dữ liệu khi đối tượng được thu hồi bởi bộ dọn rác. Phương thức này gọi phương thức close() của đối tượng kết nối (Connection) để đóng kết nối và giải phóng tài nguyên được sử dụng bởi kết nối này. Nếu kết nối đã được đóng trước đó, phương thức sẽ báo lỗi bằng cách ném ra một ngoại lệ IllegalStateException.

Vì vậy, phương thức finalize() trong lớp DB\_Handler có chức năng đảm bảo rằng kết nối đến cơ sở dữ liệu được đóng trước khi đối tượng của lớp này bị thu hồi bởi bộ dọn rác để tránh lãng phí tài nguyên và tránh xảy ra lỗi hoặc ngoại lệ trong quá trình kết nối với cơ sở dữ liệu.

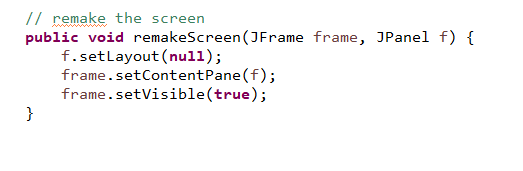


## 11. Lớp GUI

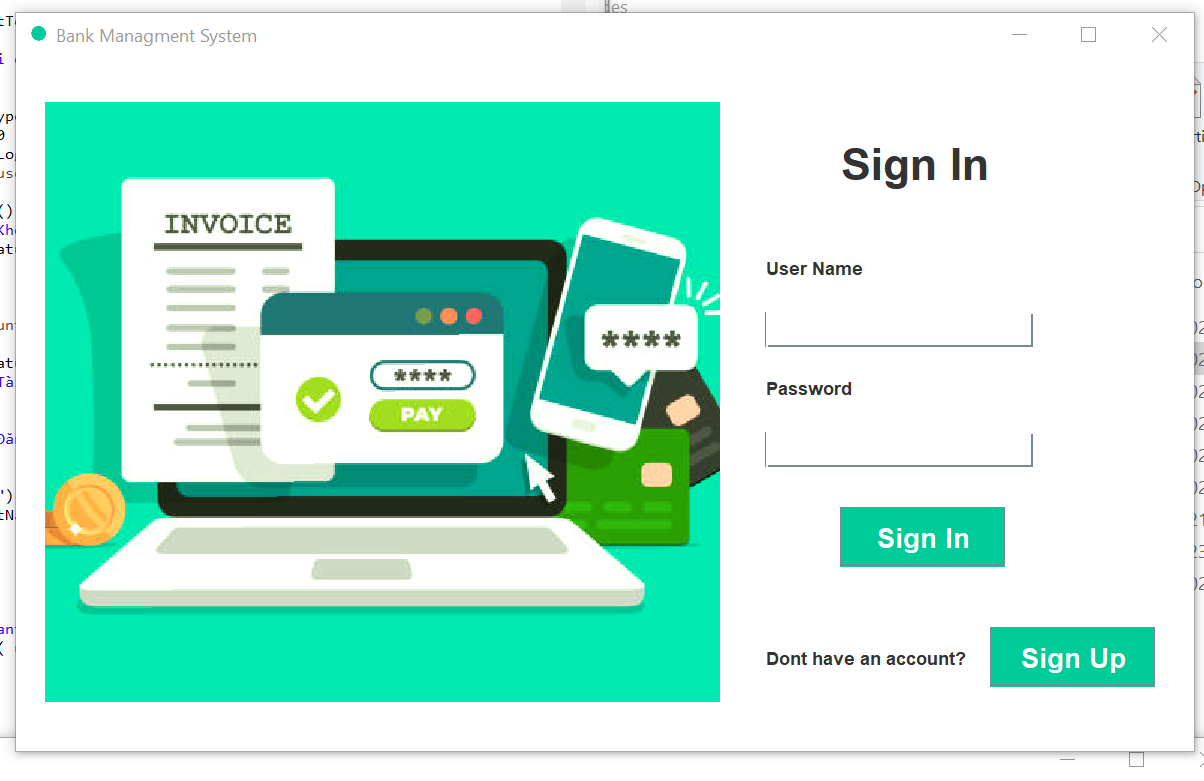
Lớp này đại diện để chứa các phương thức tạo ra giao diện hình ảnh

Các phương thức bao gồm :

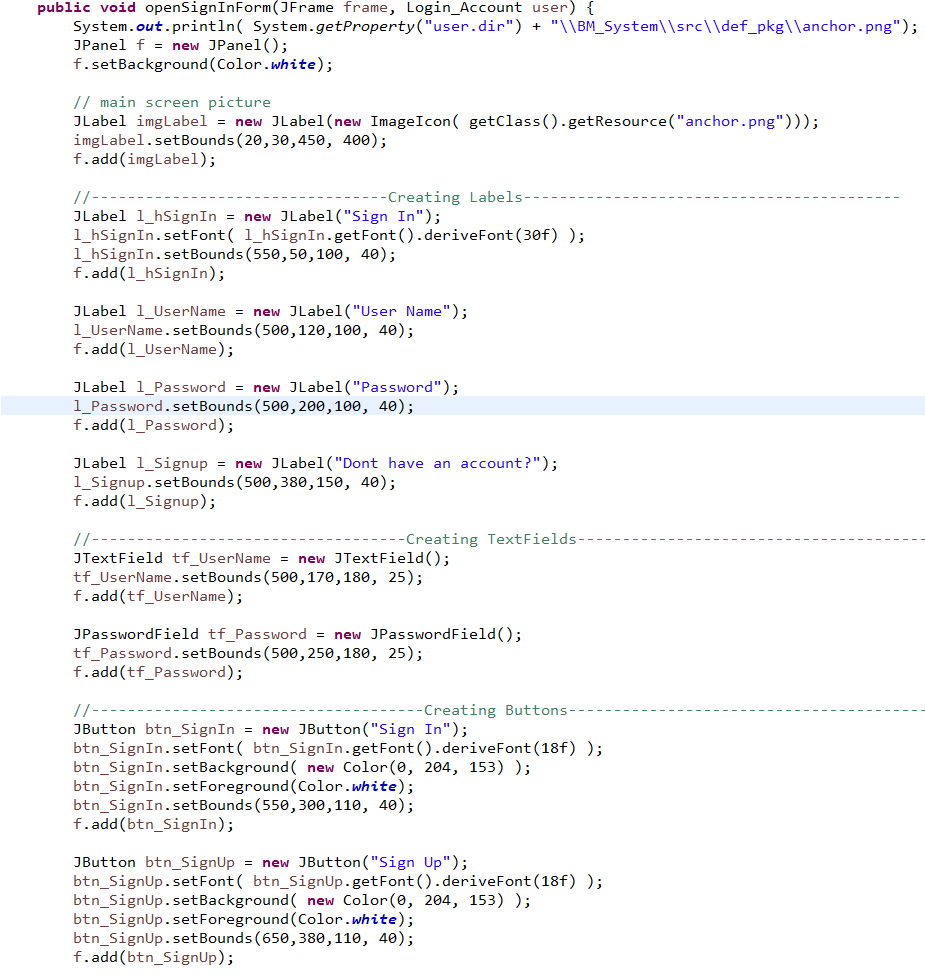
**11.1 remakeScreen(JFrame frame, JPanel f)**: Phương thức này dùng để làm mới lại màn hình hiển thị.



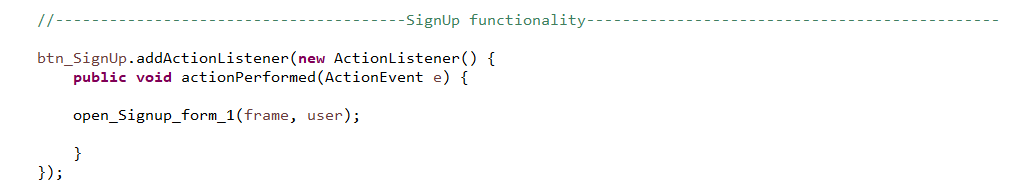
**11.2 openSignInForm(JFrame frame, Login\_Account user)**: Phương thức này hiển thị giao diện đăng nhập như hình dưới đây:



Code viết ra giao diện trên :



Ở đây chúng tôi sẽ không trình bày về cách viết ra giao diện mà sẽ trình bày về bắt sự kiện khi click vào các button :



Đầu tiên là bắt sự kiện khi click vào Sign Up thì màn hình hiển thị sẽ chuyển sang form Signup1.



Tiếp theo bắt sự kiện khi click vào Sign In :

- Đầu tiên là sẽ tạo một đối tượng DB\_Handler và gọi đến phương thức SignIn trong DB\_Handler để tìm kiếm user tương ứng.

- Nếu không tìm thấy thì sẽ hiện ra một Message thông báo “ Tên người dùng hoặc mật khẩu sai”.

- Nếu tìm thấy thì sẽ kiểm tra tiếp xem tài khoản đó thuộc Type nào:

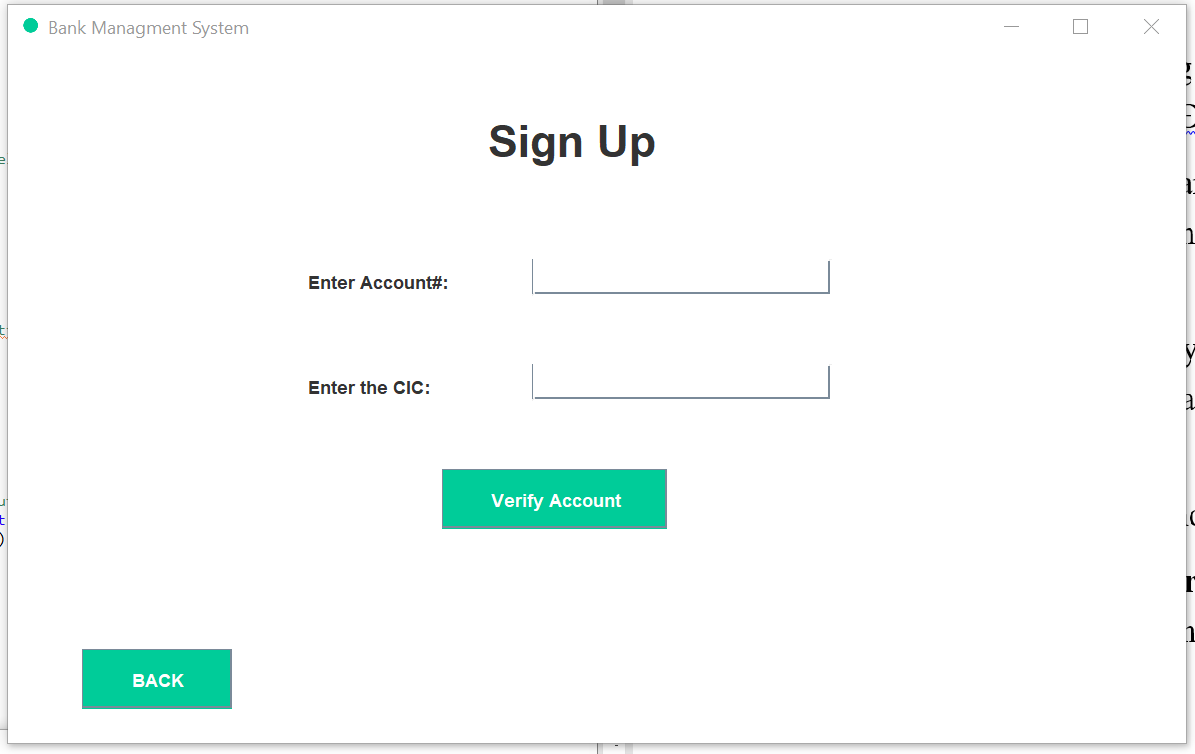
+ Nếu là “Client” thì sẽ : Lấy LoginId của client tương ứng trong CSDL, tiếp theo lấy thông tin account tương ứng LoginId đó. Kiểm tra trạng thái tài khoản: Nếu bằng 0 thì hiện thông báo “Tài khoản bị đóng vĩnh viễn”. Nếu bằng 1 thì chuyển sang màn hình hiển thị của Client. Nếu bằng 2 thì thông báo “ Tài khoản đã bị chặn”. Các trường hợp còn lại thì thông báo “ Đăng nhập không thành công”

+ Nếu là “Manager” thì sẽ chuyển sang màn hình hiển thị các chức năng của Manager (Vì tài khoản Manager là tài khoản nhân viên nên không bao giờ bị khóa, đóng ở ứng dụng này).

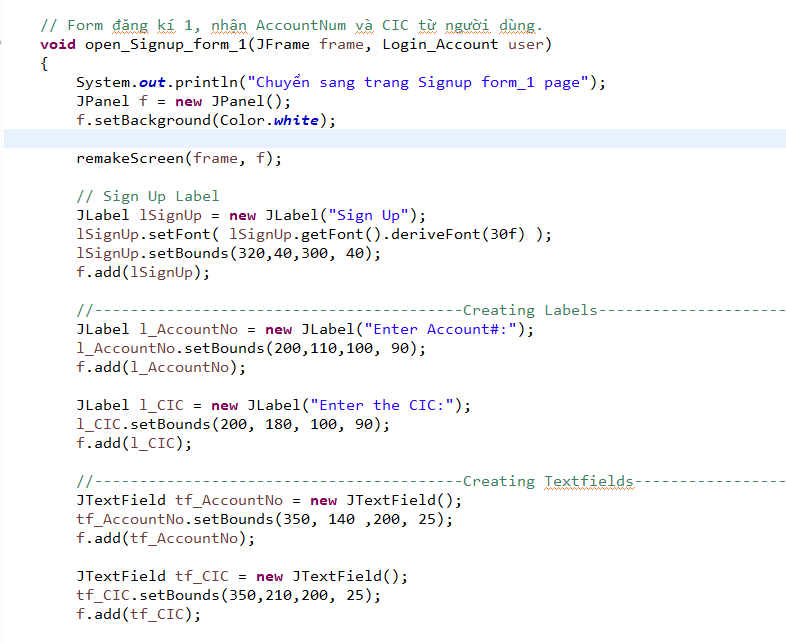
+ Nếu là “Accountant” thì sẽ chuyển sang màn hình hiển thị các chức năng của Accountant (Vì tài khoản Accountant là tài khoản nhân viên nên không bao giờ bị khóa, đóng ở ứng dụng này).

- Cuối cùng, sau khi xong sẽ làm mới lại màn hình.

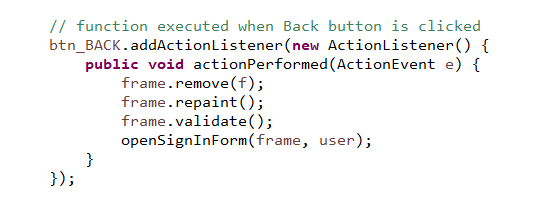
**11.3 open\_Signup\_form\_1(JFrame frame, Login\_Account user)**: Phương này dùng để hiển thị Form đăng kí 1 như hình dưới đây:



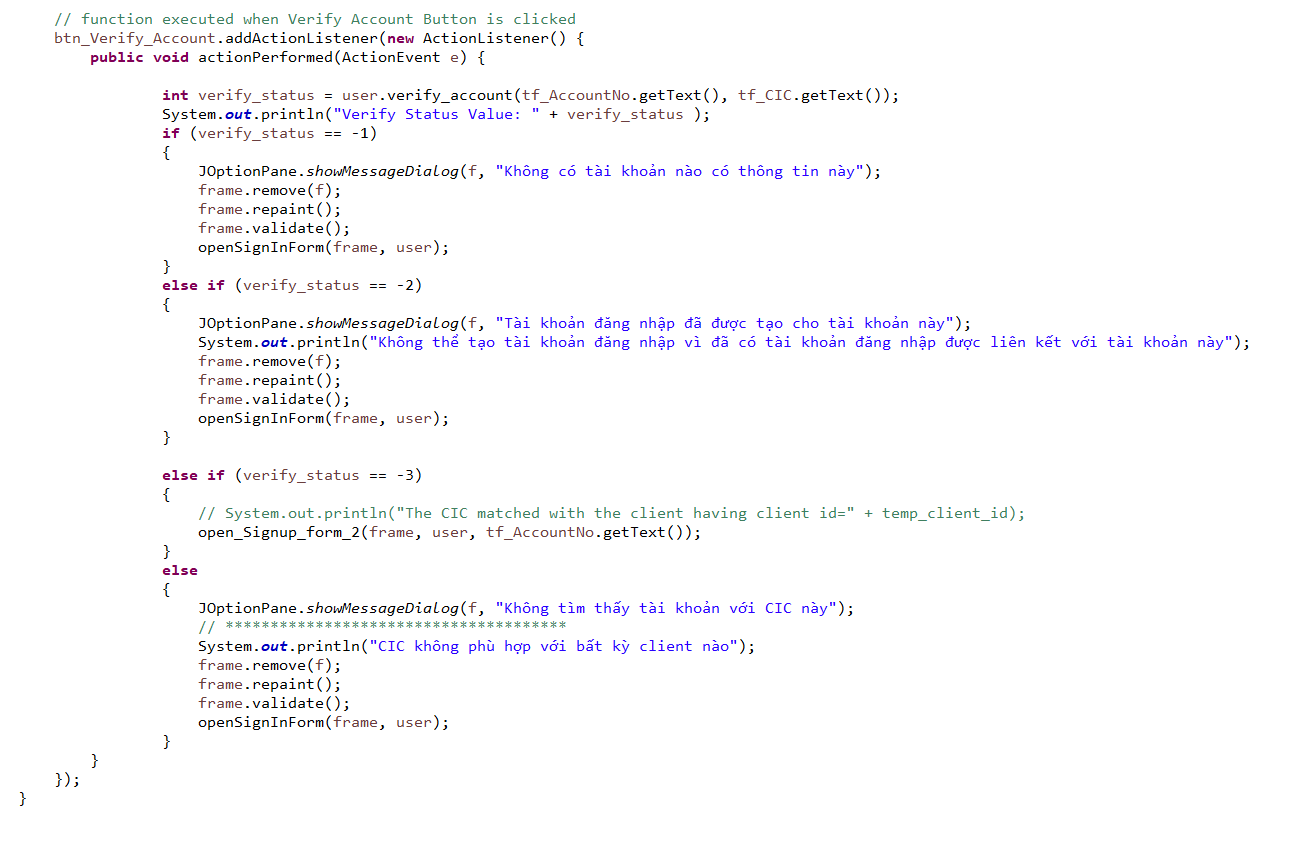
Đây là đoạn code trên hiển thị giao diện trên :



Giống như trên thì ở đây chúng tôi sẽ không trình bày về cách tạo ra giao diện này mà chúng tôi sẽ trình bày bắt sự kiện khi click vào các button.

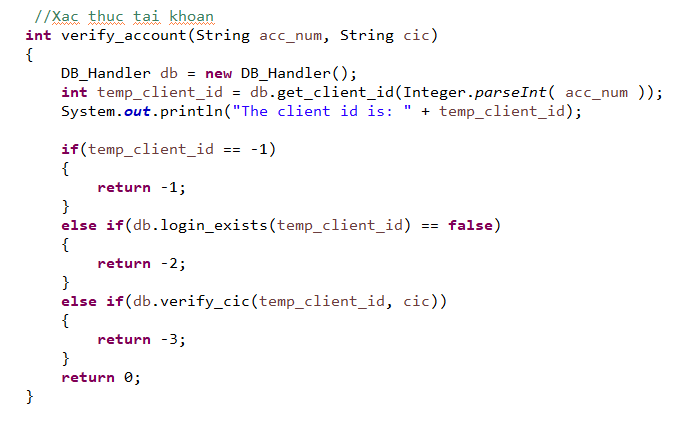


**Button Back:** Khi click vào Button Back thì giao diện sẽ trở về giao diện SignIn

****

**Button Verify Account:** khi click vào Button Verify Account:

- Đầu tiên là đối tượng Login\_Account user sẽ gọi đến phương thức verify\_account(…).



Trong verify\_account(…) của Login\_Account sẽ tạo đối tượng gọi đến phương thức lấy client\_id của trong Database thông qua đối tượng DB\_\_Handler và trả về client\_id để kiểm tra các điều kiện : nếu client\_id =-1 thì trả về cho giá trị -1 lưu vào biến **verify\_status ở GUI**. Nếu tài khoản có client\_id tương ứng đã tồn tại sẽ trả về -2 lưu vào biến **verify\_status ở GUI.** Nếu kiểm tra CIC tương ứng với client\_id thì trả về giá trị -3 **verify\_status ở GUI**.

Trong GUI:

Nếu biến verify\_status = -1 thì hiện ra message “Không tồn tài tài khoản này”.

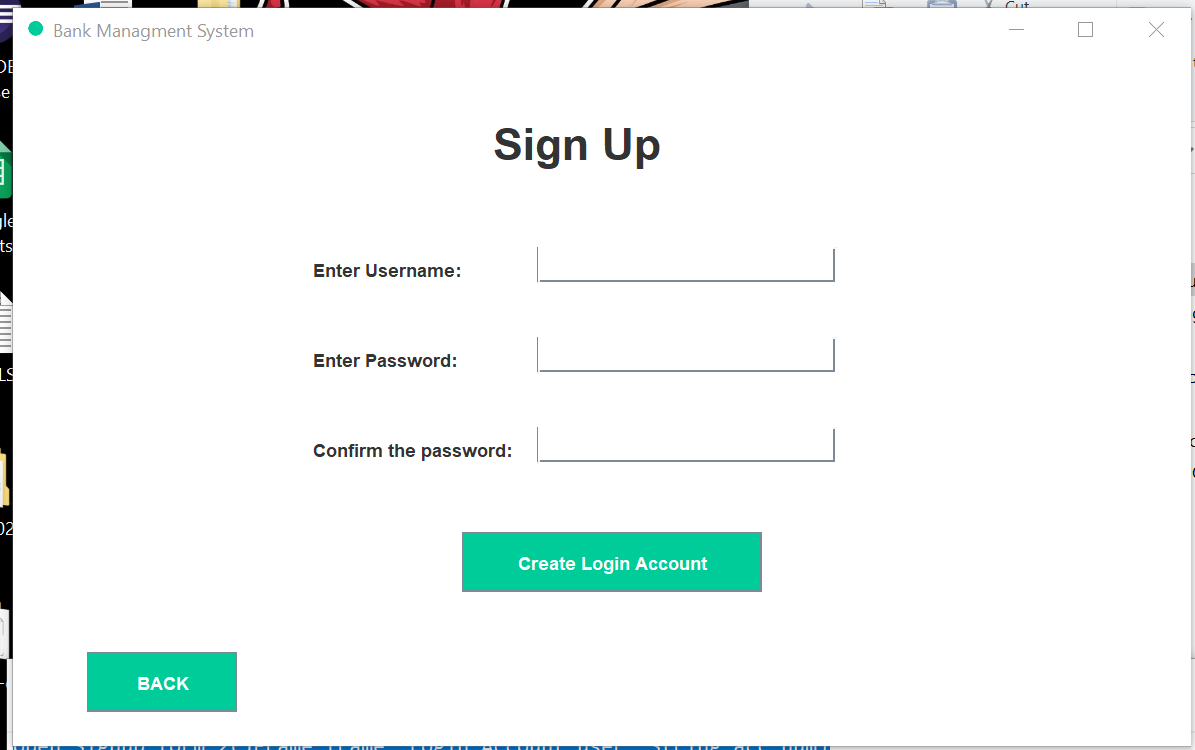
Nếu biến verify\_status = -2 thì hiện ra message “Tài khoản đăng nhập đã được tạo cho tài khoản này”.

Nếu biến verify\_status = -3 thì chuyển ra giao diện SignUpForm2.

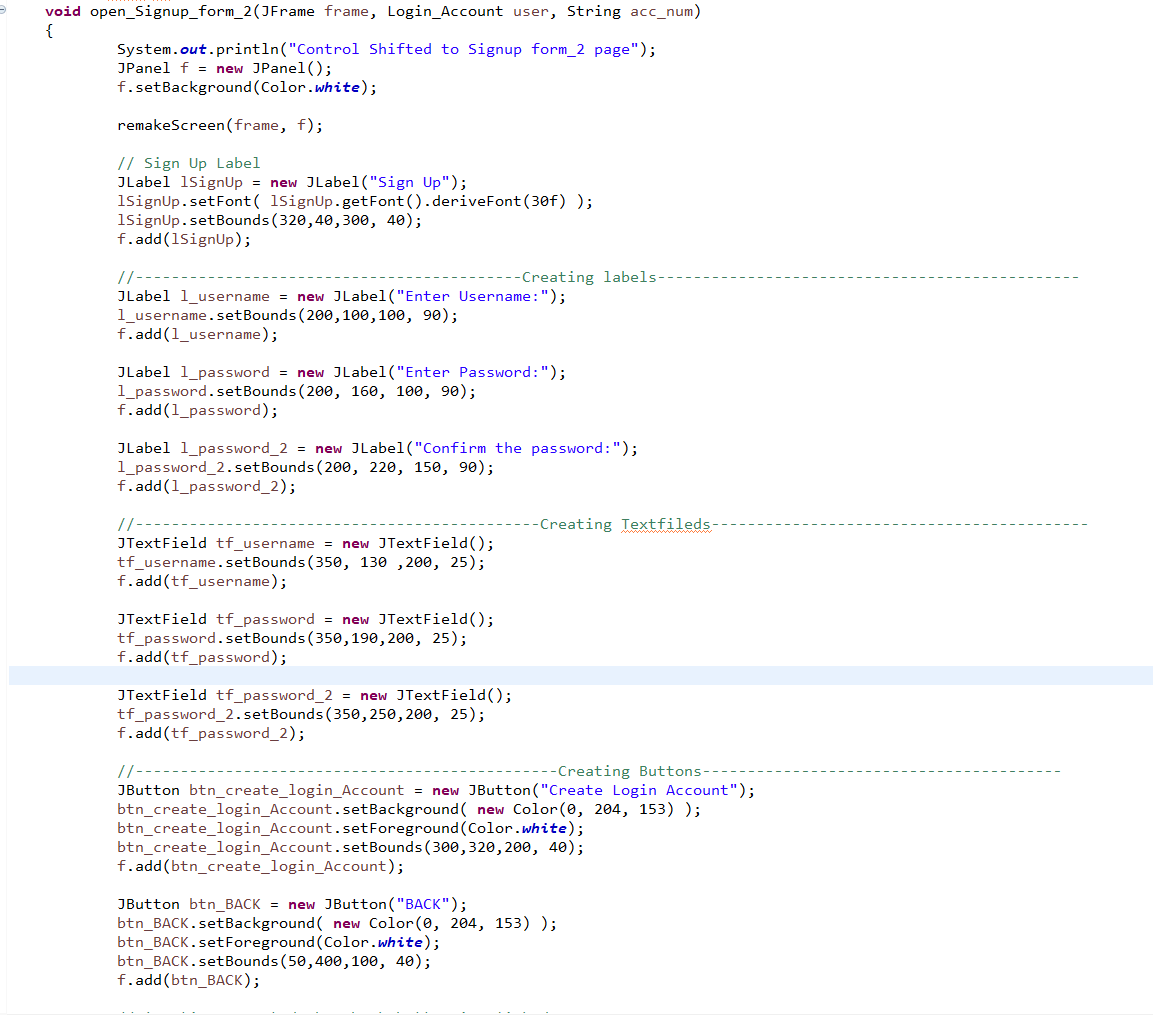
Các trường hợp còn lại sẽ hiển thị message “Không tìm thấy tài khoản với CIC này”.

*Đây chính là cách tương tác giữa các lớp*

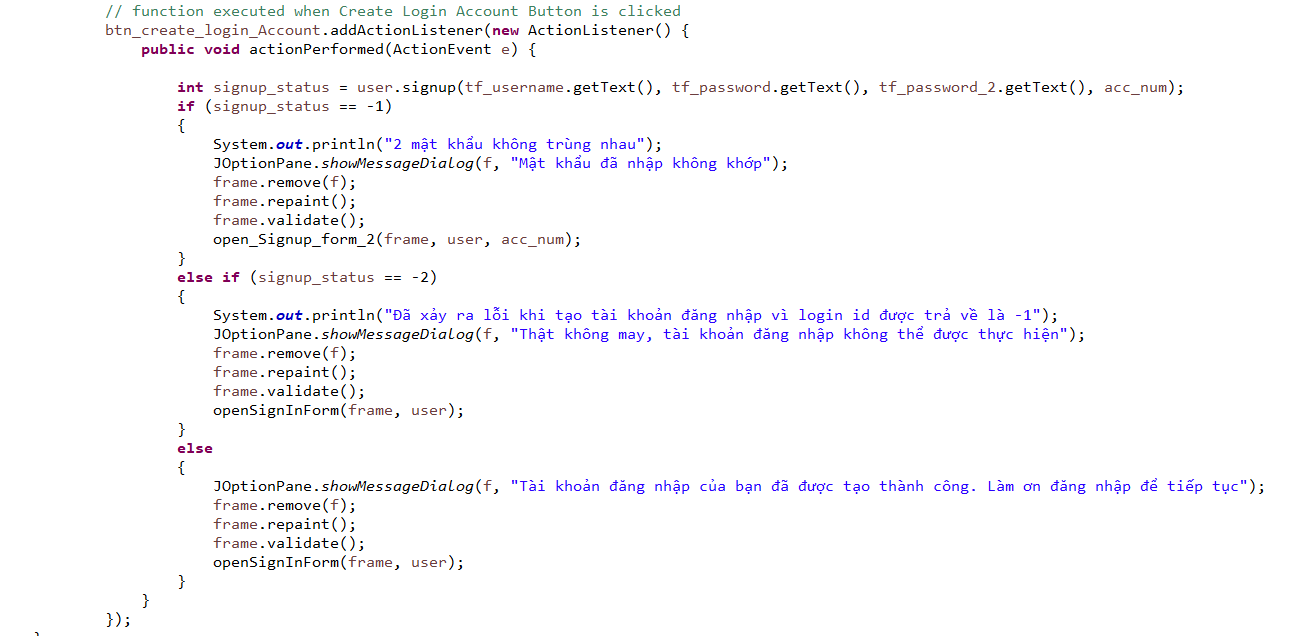
**11.4** **open\_Signup\_form\_2(JFrame frame, Login\_Account user, String acc\_num):** Phương thức này để hiển thị Form SignUp 2 như hình dưới đây:



Code để tạo ra giao diện này:



Chúng tôi sẽ bắt sự kiện nổi bật khi click vào các button, những button giống nhau chúng tôi đã nói trước đó sẽ bỏ qua:



**Button Create Login Account:** Khi click vào button này sự kiện sẽ được bắt và thực hiện các thao tác sau:

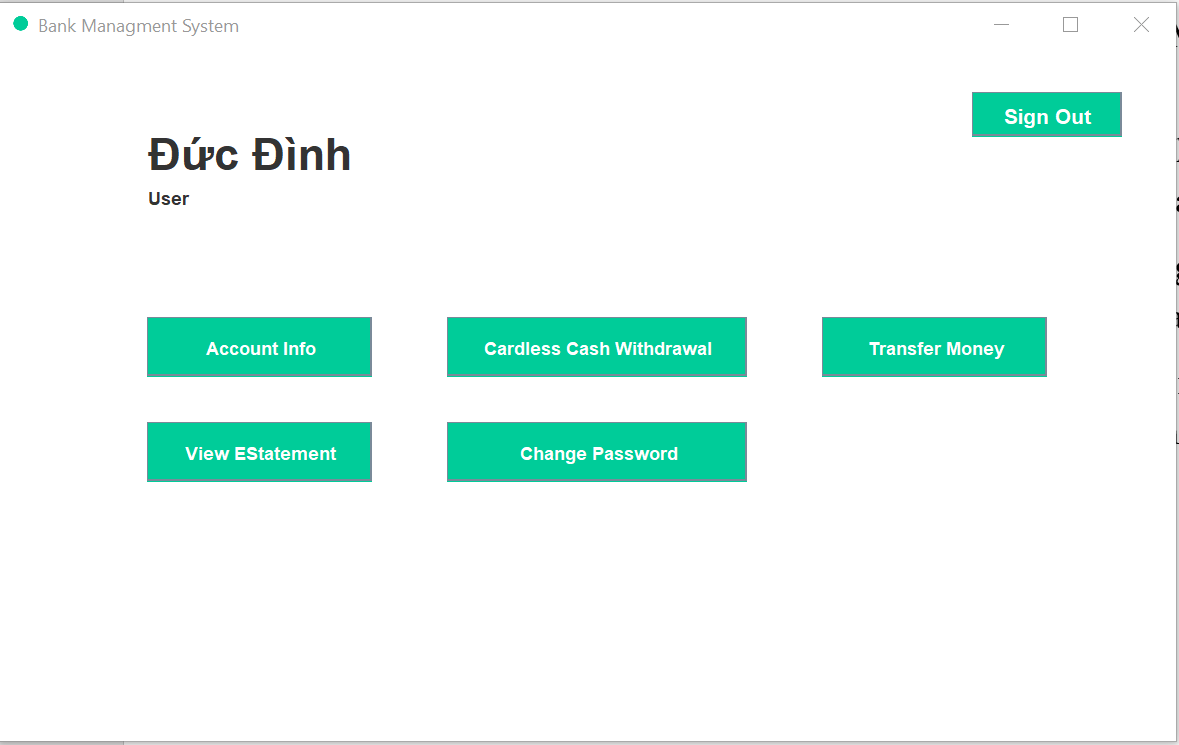
- Gọi đến đối tượng user sẽ gọi đến phương thức signup trong Login\_Account để trả về trạng thái đăng kí lưu vào biến signup\_status.

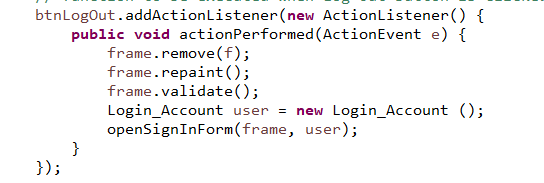
- Kiểm tra nếu signup\_status = -1 thì sẽ hiển thị thông báo “Mật khẩu đã nhập không khớp”. Quay trở về giao diện SignUpForm2.

- Nếu signup\_status=-2 (Đã có lỗi xảy ra trong hệ thống) thì sẽ hiển thị thông báo “Tài khoản đăng nhập không thể thực hiện”. Quay trở về giao diện SignIn.

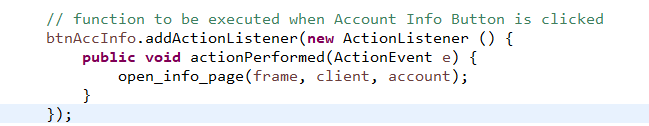
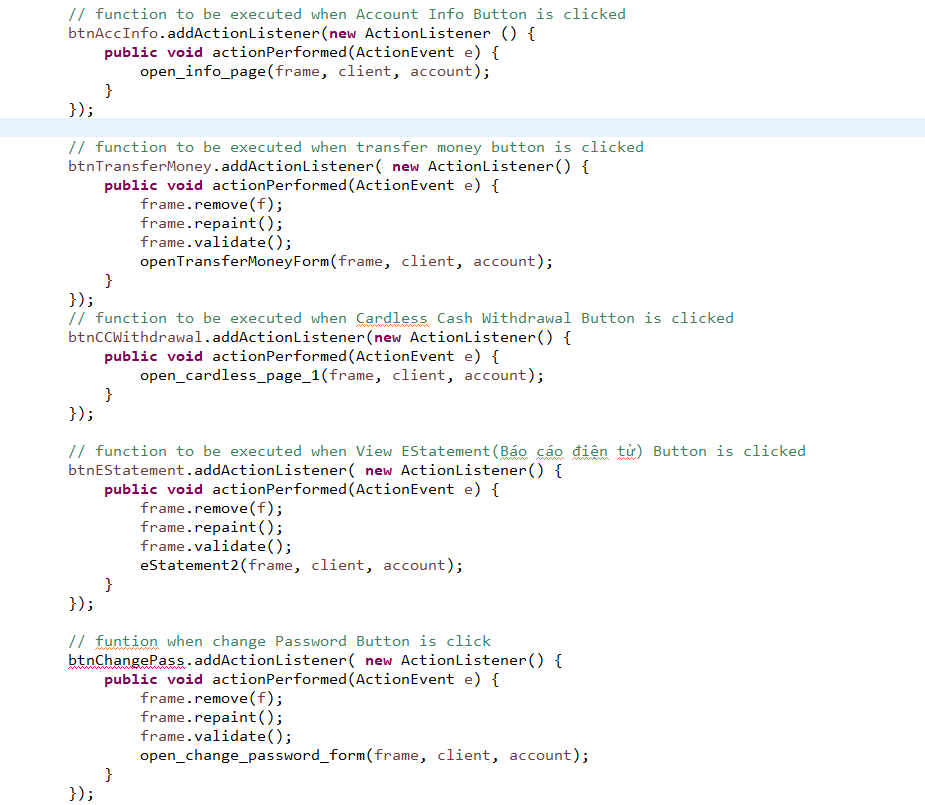
- Trường hợp còn lại sẽ hiển thị thông báo “Tài khoản đăng nhập của bạn đã được tạo thành công. Hãy đăng nhập để tiếp tục”. Quay trở về giao diện SignIn.

**11.5 openClientMenu(JFrame frame, Client client, Bank\_Account account):** Phương thức này hiển thị giao diện khi đăng nhập tài khoản client như hình dưới:

  
Chúng tôi sẽ nói đến bắt sự kiện khi click vào các button :

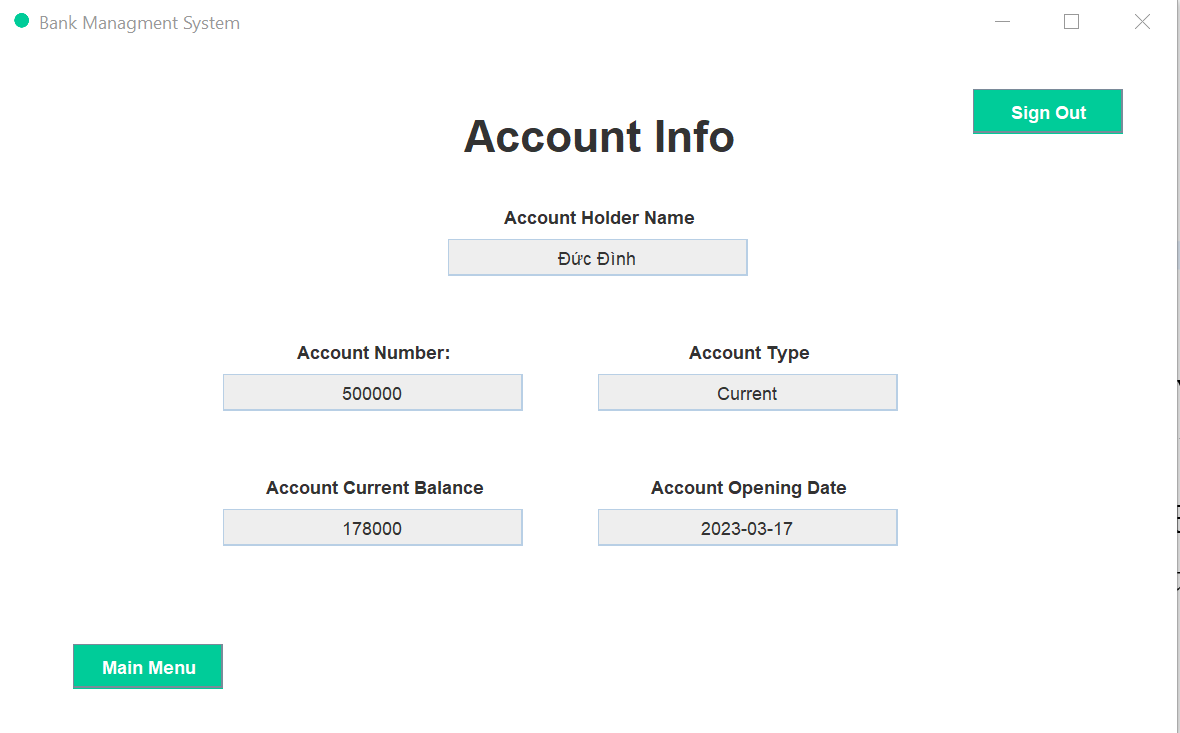
****

**Button Sign Out**: Bắt sự kiện khi click vào button này sẽ hiển thị quay lại giao diện SignIn.

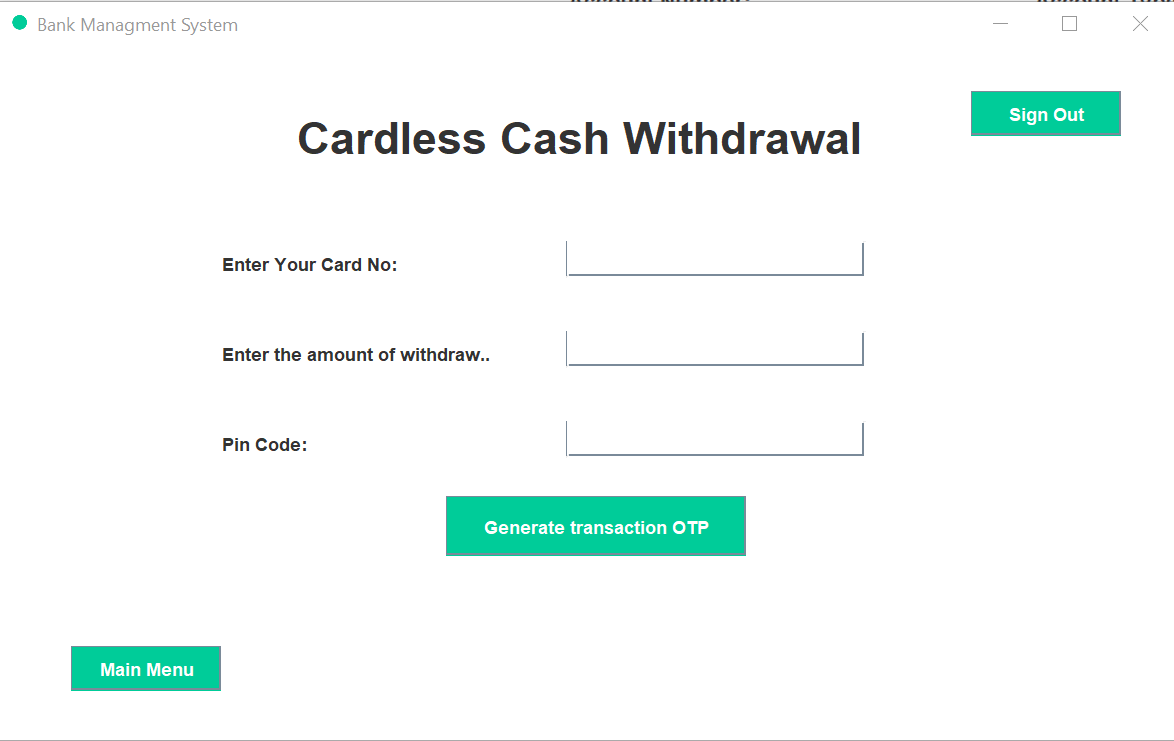


**Button Account Info, Cardless Cash Withdrwal, Transfer Money, View EStatement, Change Password:** Bắt sự kiện khi click vào button nào sẽ gọi đến phương thức tương ứng hiển thị các giao diện tương ứng với nó.

**11.6 open\_info\_page(JFrame frame, Client client, Bank\_Account account):** Phương thức này sẽ hiển thị thông tin tài khoản khi client click vào button **Account InFo:**

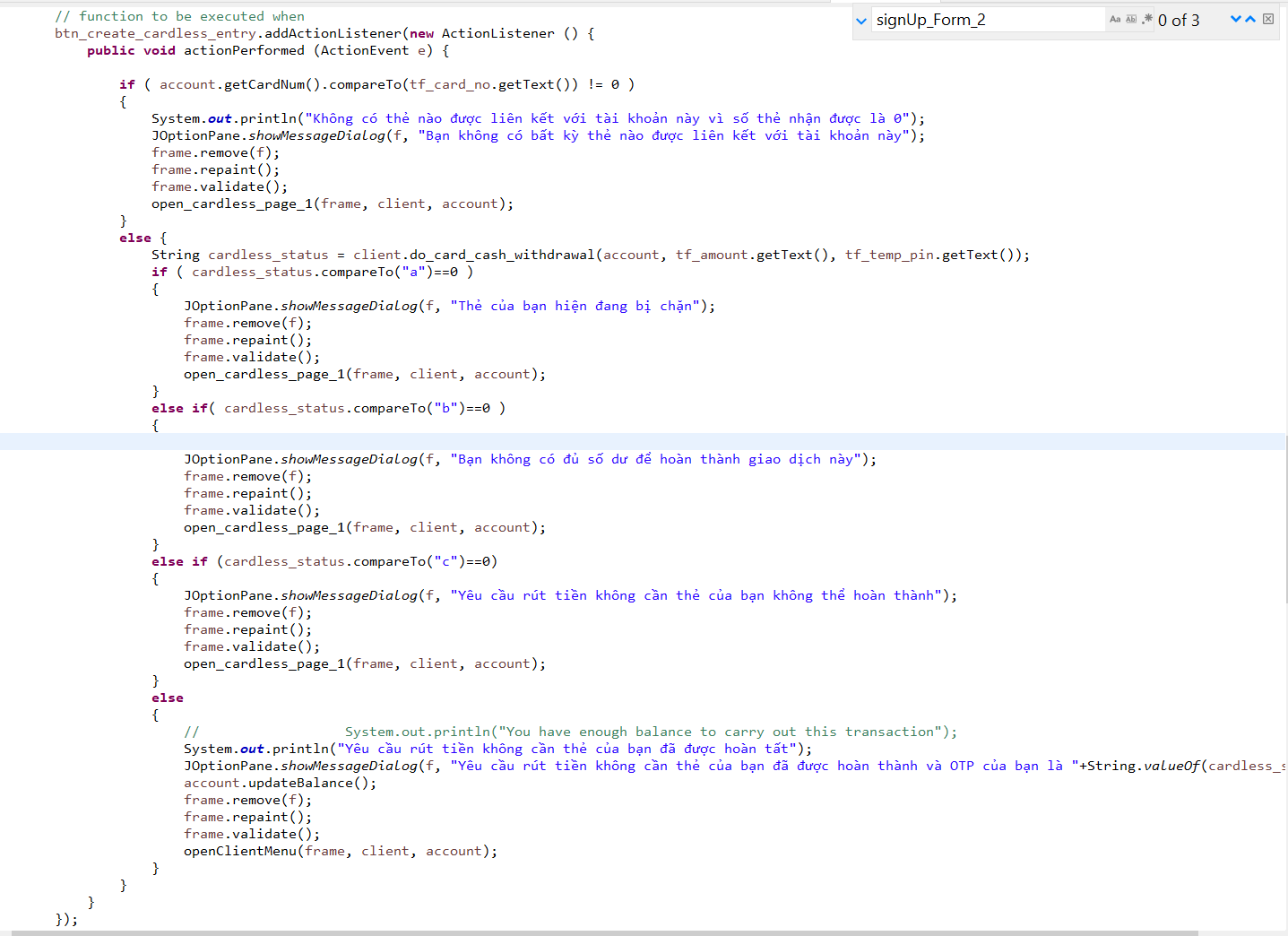


**11.7 open\_cardless\_page\_1(JFrame frame, Client client, Bank\_Account account)**: Phương thức này sẽ hiển thị ra giao diện Rút tiền không cần thẻ

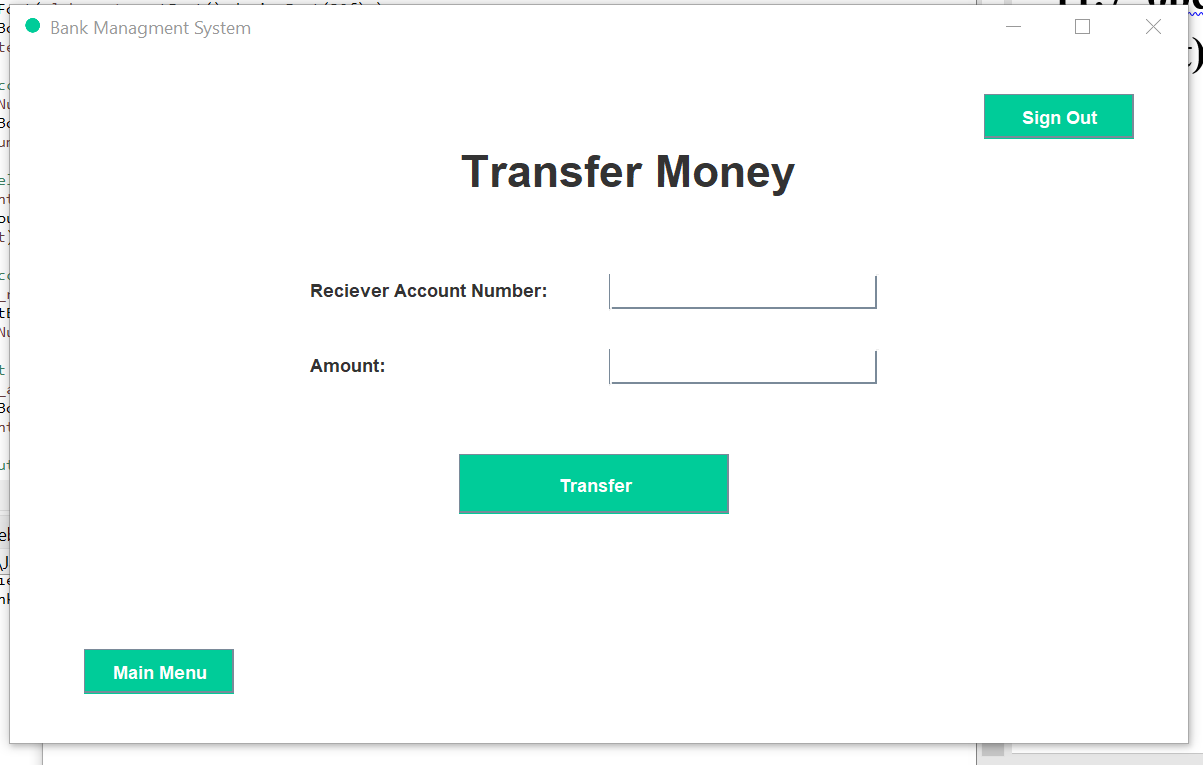


Khi click vào **Button Generate transaction OTP** sẽ bắt sự kiện và thực hiện :

Kiểm tra thẻ có được liên kết với tài khoản không.Nếu có sẽ kiểm tra trạng thái của thẻ: “a” : Thẻ bị chặn, “b” : Số dư không đủ để giao dịch, “c” : Đã có lỗi xảy ra trong hệ thống, trường hợp còn lại: Rút tiền thành công và hiển thị mã OTP, cập nhật lại số dư.



**11.7 openTransferMoneyForm(JFrame frame, Client client, Bank\_Account account)**: Phương thức này hiển thị ra giao diện chuyển tiền như hình dưới



Nhập số tài khoản muốn chuyển và số tiền.

Khi click vào **Button Transfer** sẽ thực hiện các bước kiểm tra:

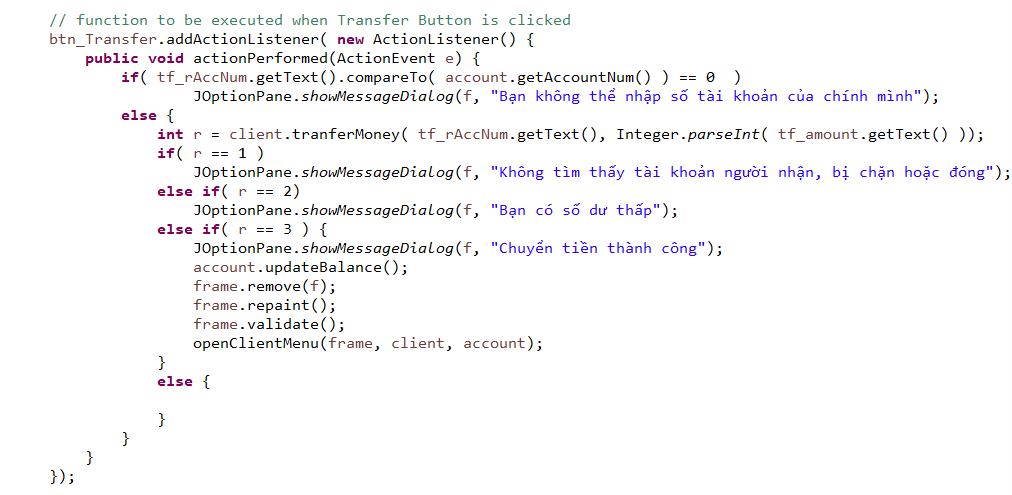
- Nếu số tài khoản muốn chuyển trùng với số tài khoản của mình sẽ hiển thị thông báo “Bạn không thể nhập số tài khoản của chính mình”.

- Gọi đến phương thức transferMoney() của đối tượng client trong Client để trả về một giá trị tương ứng với thông báo lưu vào biến r.

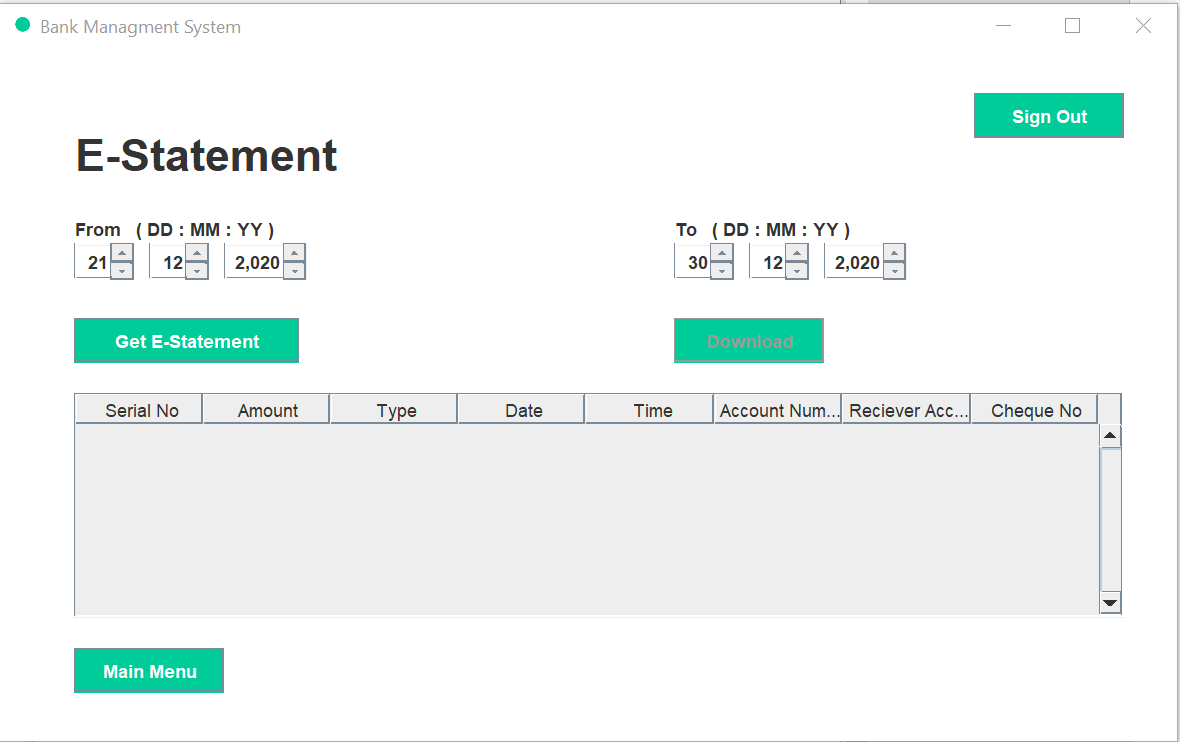
- Nếu r=1 hiển thị thông báo “Không tìm thấy tài khoản người nhân, bị chặn hoặc đóng”

- Nếu r=2 hiển thị thông báo “Số dư thấp”

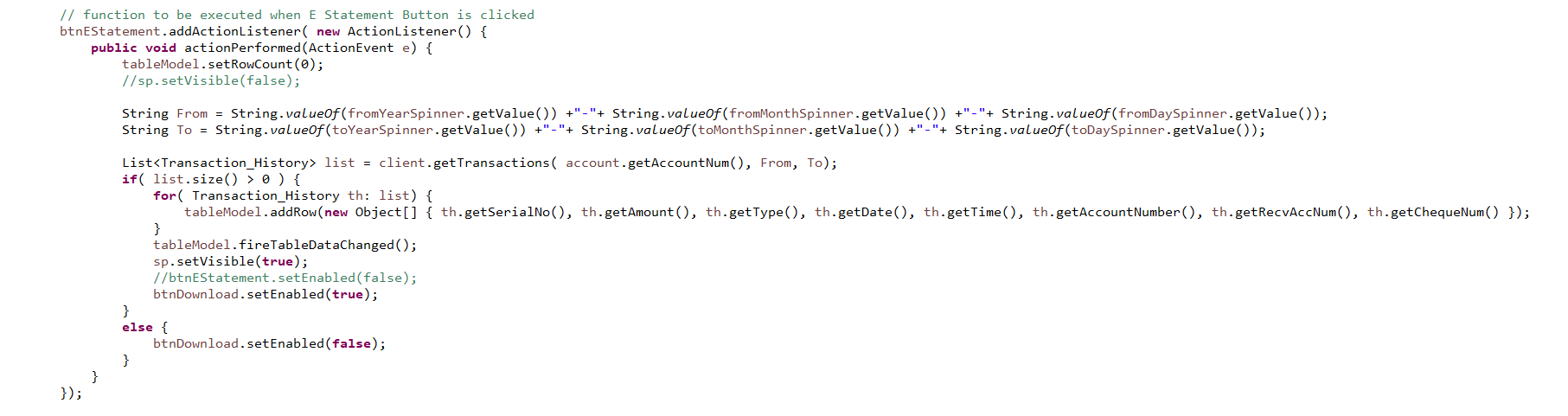
- Nếu r=3 hiển thị thông báo “Chuyển tiền thành công”



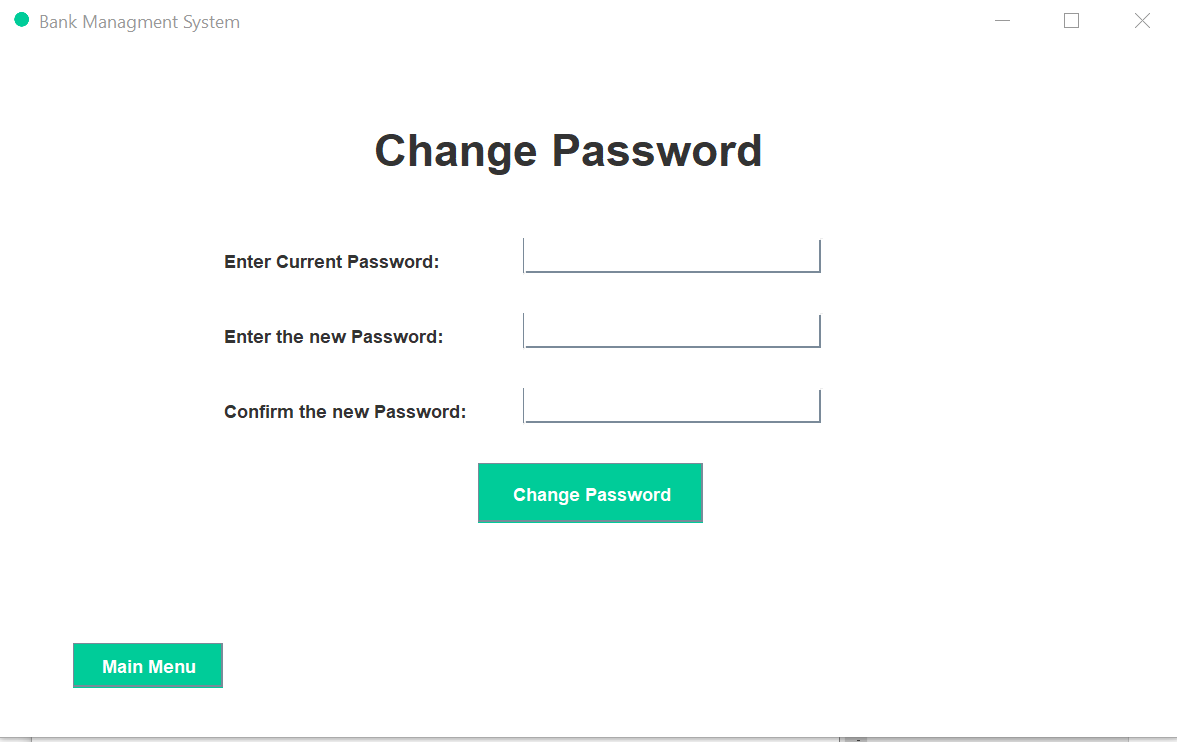
**11.8 eStatement2(JFrame frame, Client client, Bank\_Account account):** Phương thức này hiển thị giao diện báo cáo điện tử của tài khoản client



Khi chọn khoảng thời gian phù hợp và click vào button Get E-Statement bắt sự kiện và thực hiện tạo list in ra danh sách lịch sử các giao dịch tương ứng với khoảng thời gian đó



**11.9 open\_change\_password\_form(JFrame frame, Client client, Bank\_Account account)** : Phương thức hiển thị giao diện đổi mật khẩu của tài khoản client



Khi đã nhập các thông tin và click vào **Button Change Password** sẽ bắt sự kiện và thực hiện:

- Gọi đến phương phức change\_password của đối tượng client trong Client và trả về trạng thái đổi mật khẩu lưu vào biến change\_pwd\_status.

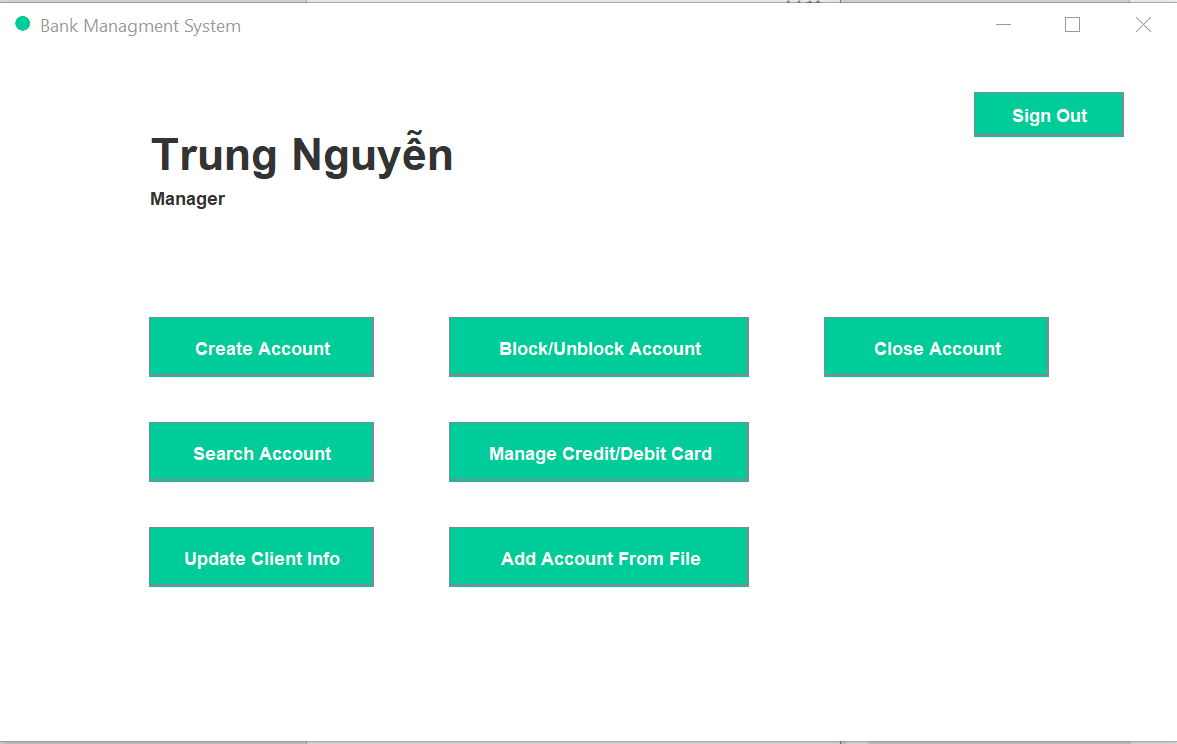
- Nếu change\_pwd\_status = -1 thì hiện thông báo “Hệ thống không tìm thấy tài khoản đăng nhập”

- Nếu change\_pwd\_status = -2 thì hiện thông báo “Mật khẩu hiện tại không khớp với mật khẩu trong hệ thống”

- Nếu change\_pwd\_status = -1 thì hiện thông báo “Các mật khẩu mới không khớp nhau”

- Trường hợp còn lại thì hiển thị thông báo “Bạn đã thay đổi thành công. Bạn phải đăng nhập lại để tiếp tục”. Quay về giao diện SignIn.

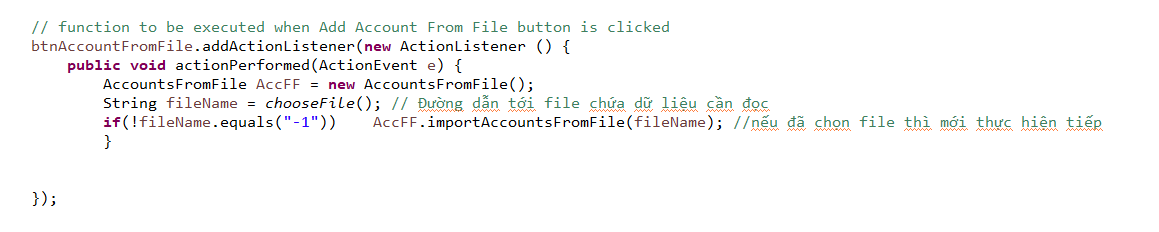
**11.10 openManagerMenu(JFrame frame, Manager manager)**: Phương thức hiển thị giao diện khi đăng nhập tài khoản Manager



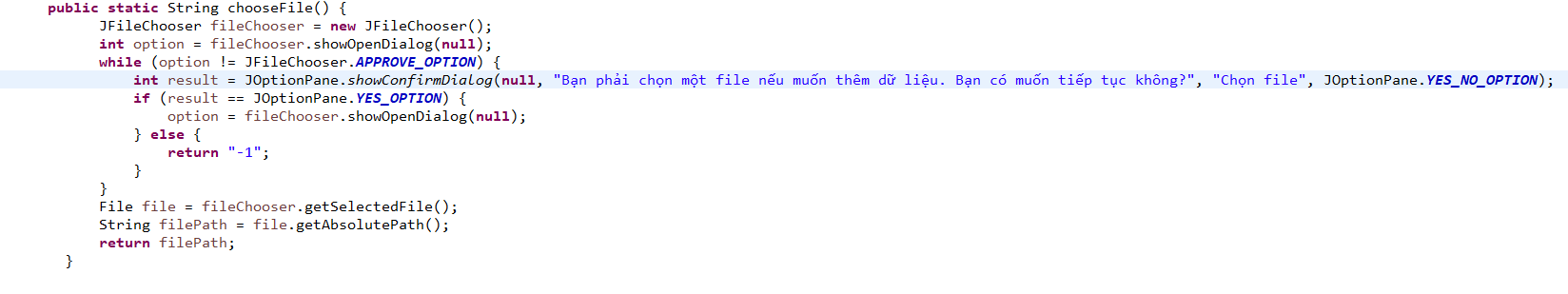
Khi click vào các button tương ứng như trên hình sẽ bắt sự kiện và chuyển sang giao diện tương ứng với từng button.

Riêng button **Add Account From File** khi click vào sẽ bắt sự kiện và thực hiện:

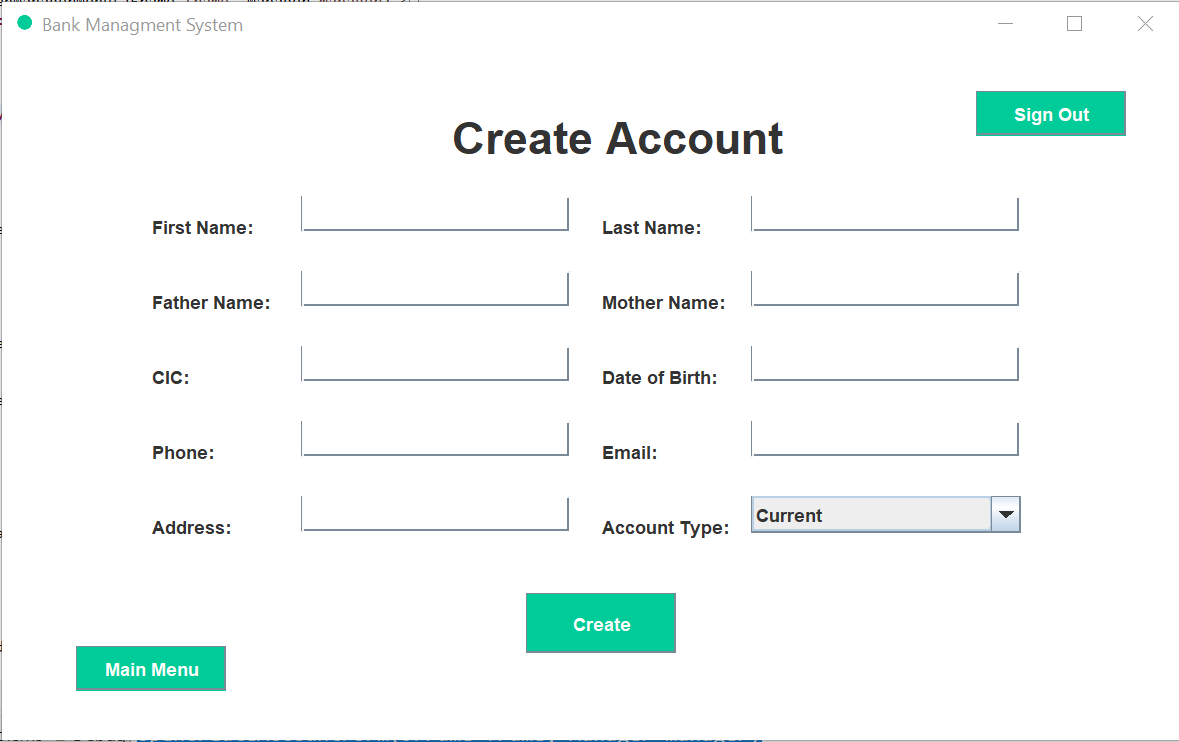
- Tạo đối tượng AccountsFromFile mới đặt tên là AccFF. Tạo biến fileName lưu đường dẫn file mà quản lí đã chọn. Kiểm tra nếu có đường dẫn thì gọi đến phương thức importAccountsFromFile(filename) để thêm các thông tin Account vào hệ thống.



**11.11 Phương thức tĩnh chooserFile()** : Trả về đường dẫn file tương ứng, nếu không chọn đường dẫn thì trả về -1.



**11.12 openCreateAccountForm(JFrame frame, Manager manager)**: Phương thức hiển thị giao diện thêm Account của tài khoản manager.



Khi đã nhập thông tin và click vào button **Create** thì sẽ bắt sự kiện và thực hiện các thao tác:

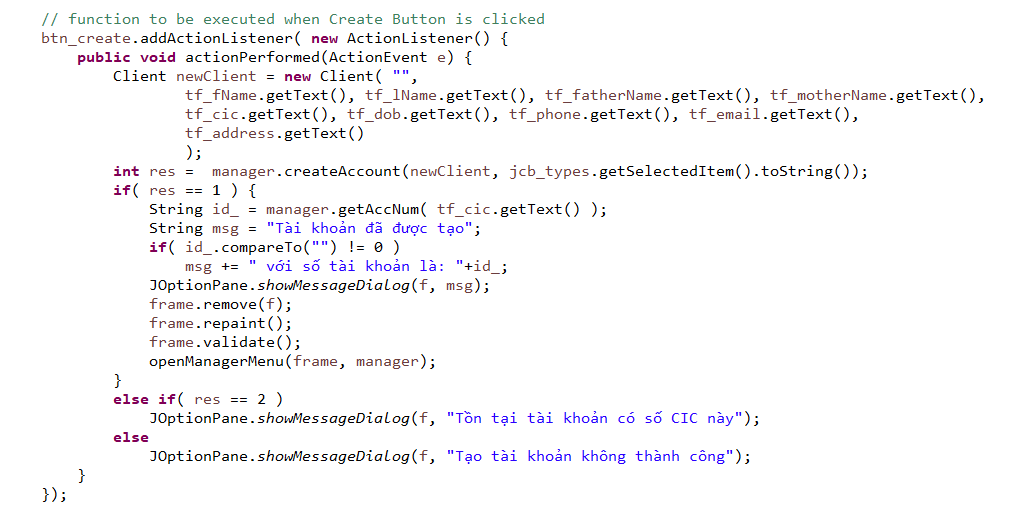
- Tạo đối tượng client mới với các thông tin vừa nhập.

- Gọi đến phương thức createAccount của đối tượng manager trong Manager để trả về giá trị kiểm tra lưu vào biến res.

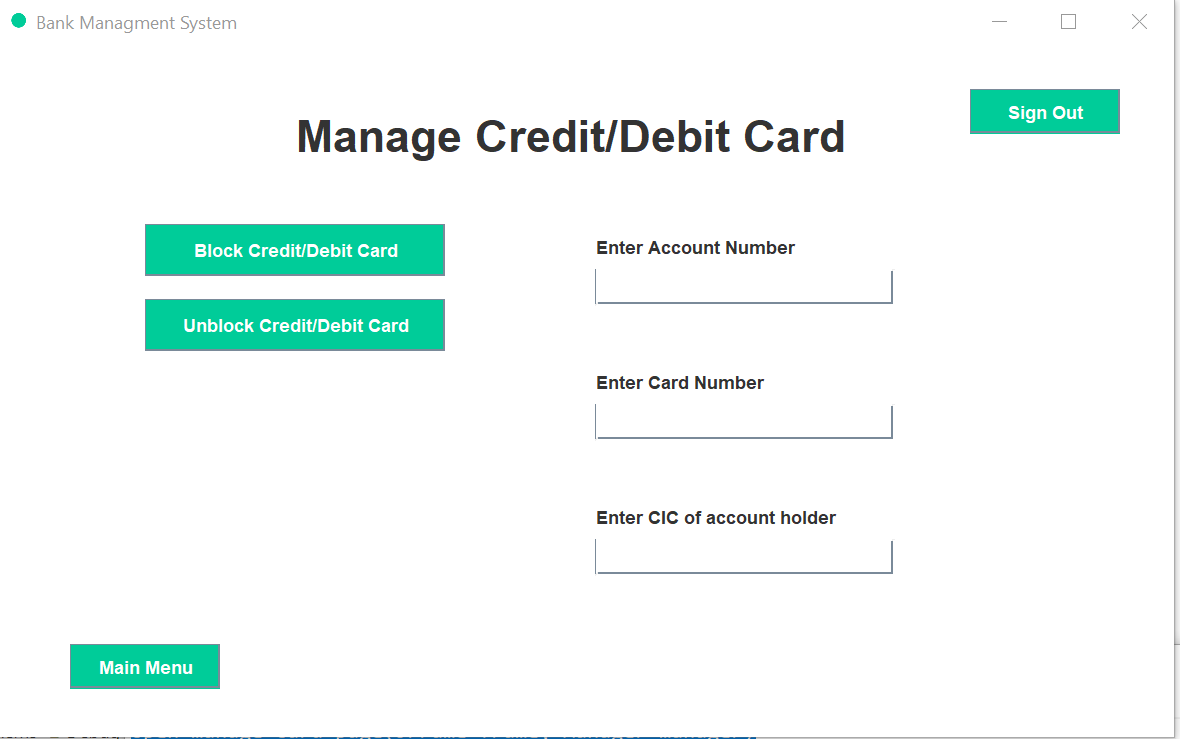
- Nếu res = 1: Có nghĩa là thành công. Gọi đến hàm getAccNum trong Manager để lấy số tài khoản và hiển thị thông báo “Tài khoản được tạo” + số tài khoản.

- Nếu res = 2: Tồn tại tài khoản có CIC này. Hiện thị thông báo tương ứng.

- Trường hợp còn lại: Hiển thị thông báo “Tạo Tài khoản không thành công”.



**11.13 open\_manage\_card\_page(JFrame frame, Manager manager)** :Phương thức này hiển thị ra giao diện quản lí khóa/Mở khóa thẻ.



Nhập thông tin xong.

Khi click vào button **Block Credit/Debit Card** sẽ bắt sự kiện và thực hiện các thao tác:

- Gọi đến phương thức block\_card(…) trong Manager để lấy về giá trị trạng thái lưu vào biến block\_status.

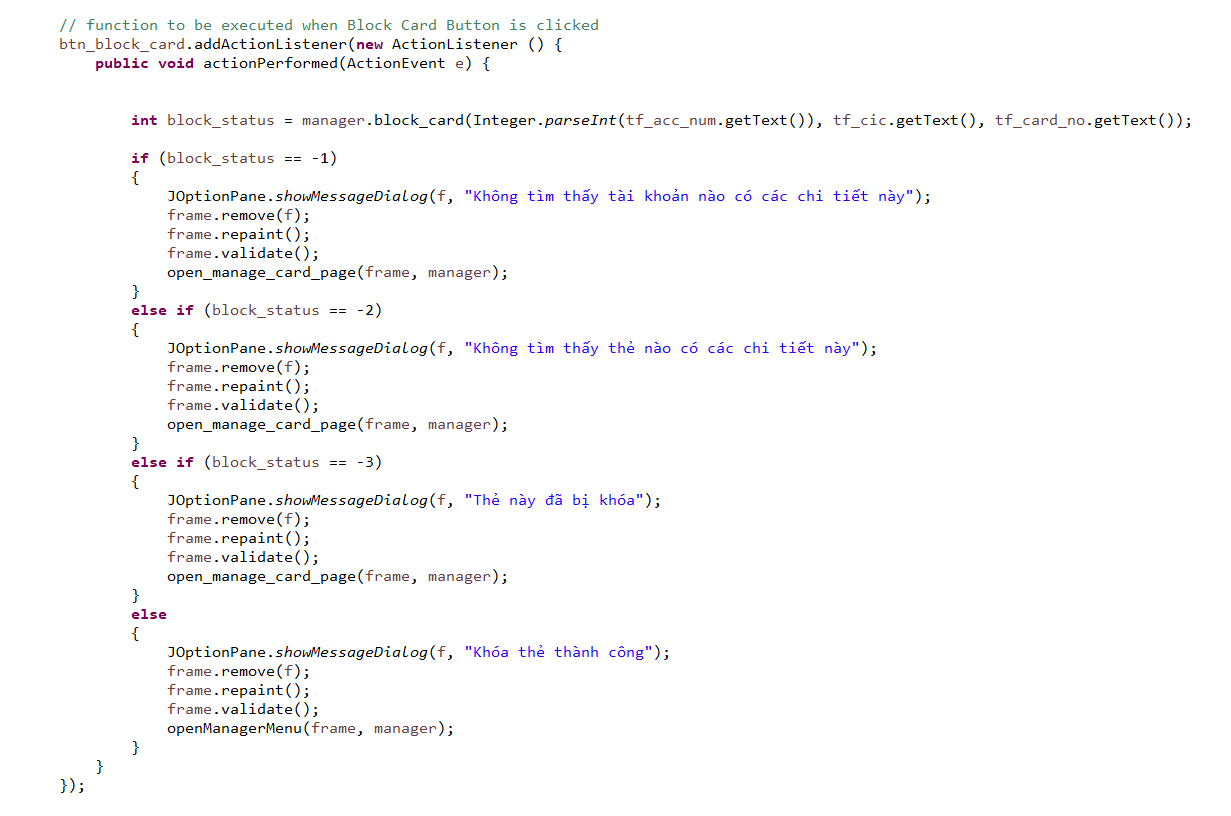
- Nếu block\_status = -1: Không tìm thấy tài khoản với các thông tin trên.

- Nếu block\_status = -2: Không tìm thấy thẻ với các thông tin trên.

- Nếu block\_status = -3: Thẻ đã bị khóa.

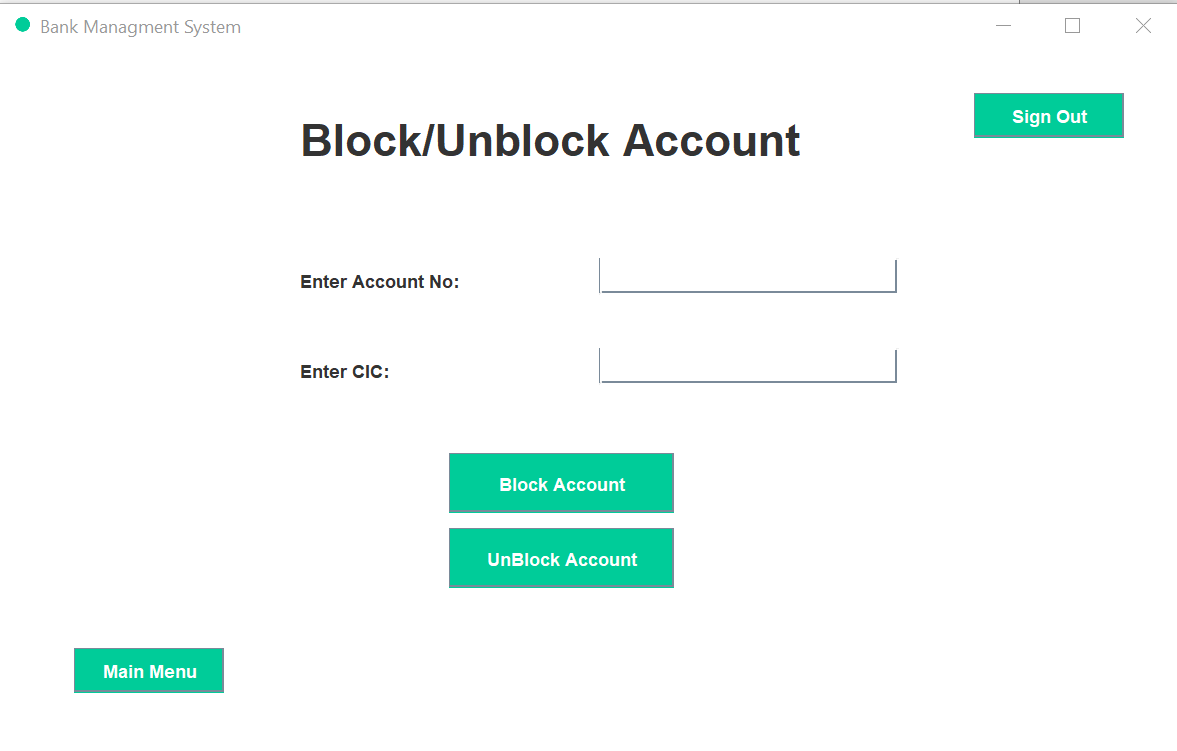
- Trường hợp còn lại: Khóa thẻ thành công. Trở lại giao diện MenuManager.

Khi click vào button **Unblock Credit/Debit Card** cũng tương tự như vậy.

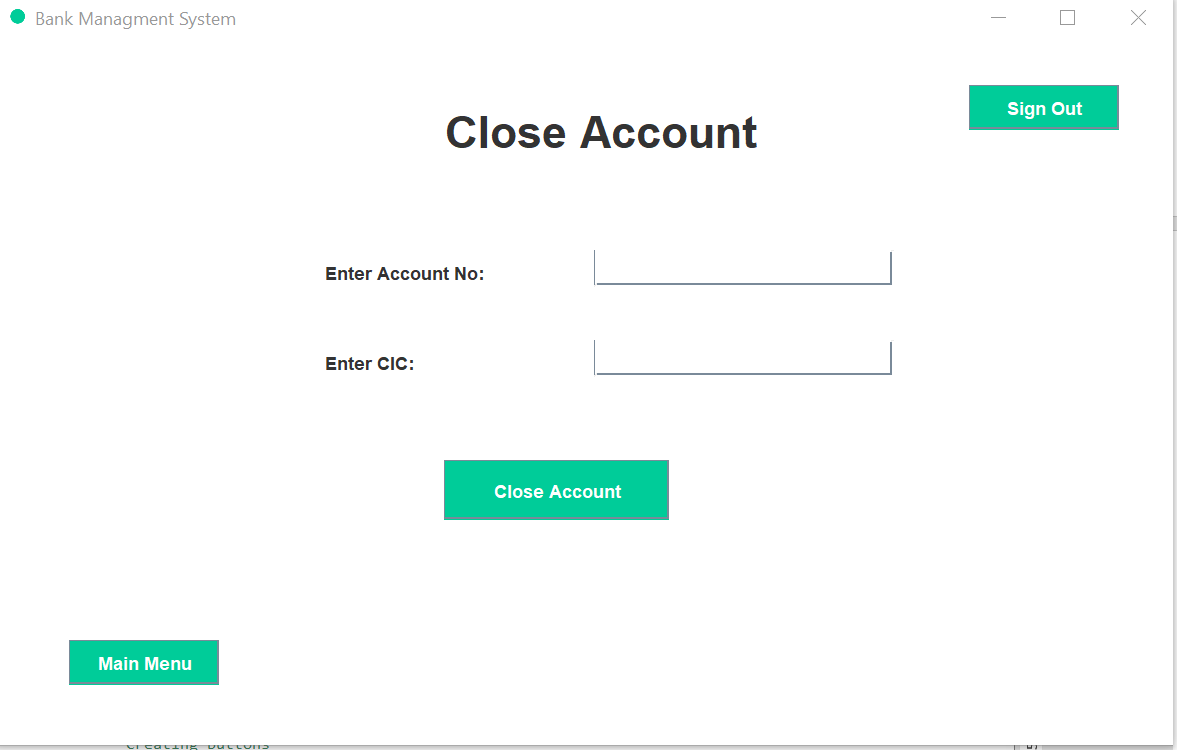


**11.14 open\_block\_unblock\_account\_page(JFrame frame, Manager manager)** : phương thức này hiển thị giao diện Mở/Khóa tài khoản.

Khi click vào button **Block Account/Unblock Account**  các thao tác cũng tương tự như khi Block/Unblock Credit/Card.



**11.15 open\_close\_account\_page(JFrame frame, Manager manager)**: Phương thức hiện ra giao diện đóng tài khoản (vĩnh viễn)

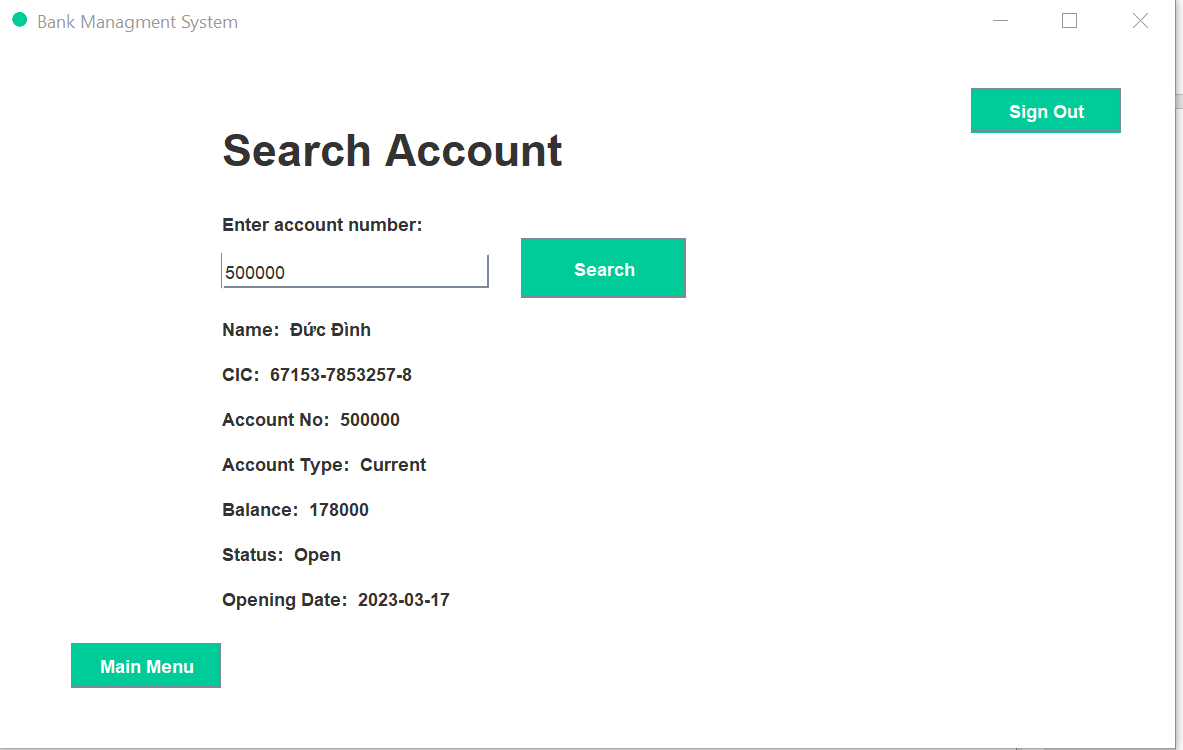


Khi click vào button **Close Account**: sẽ bắt sự kiện và gọi đến phương thức close\_account và trả về giá trị trạng thái lưu vào biến close\_status.

- Nếu close\_status = 0: Hiển thị thông báo đóng tài khoản thành công.

- Ngược lại: Hiển thị thông báo không có tài khoản nào khớp với thông tin này.

**11.16 searchAccount(JFrame frame, Manager manager)**: Phương thức này hiển thị giao diện tìm kiếm Account và chỉ cần thông tin Account Num.



Khi click vào button **search**  sẽ thực hiện các thao tác:

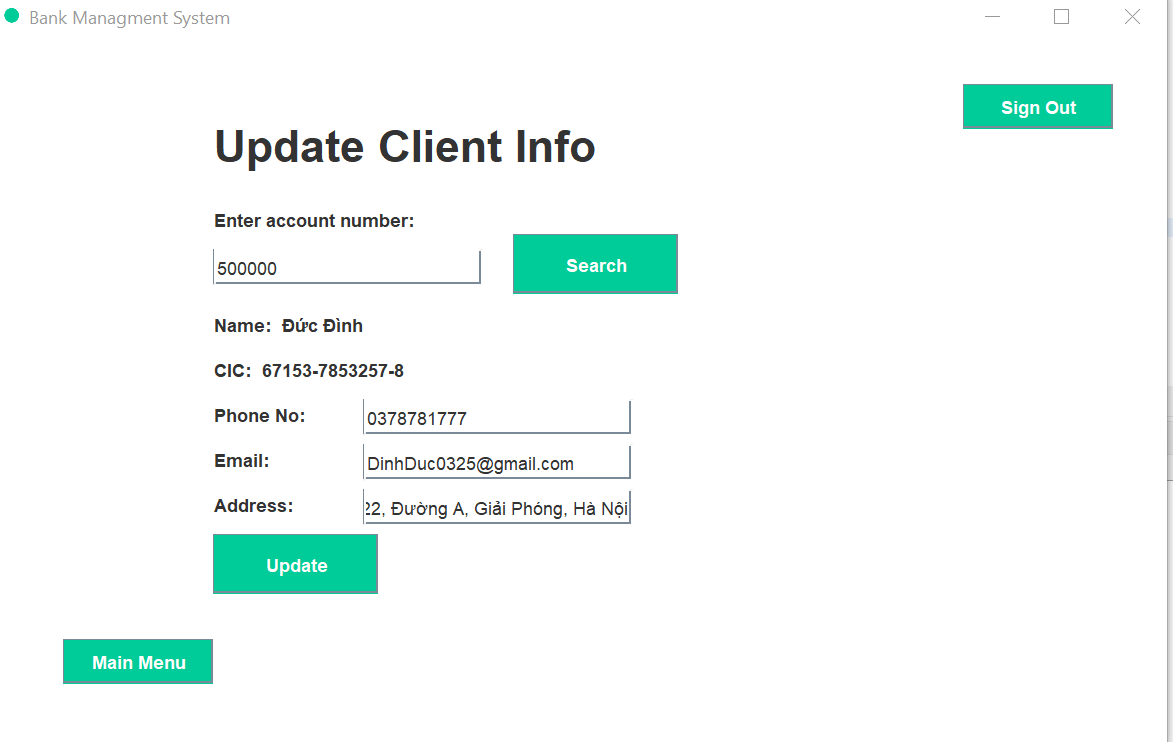
- Gọi phương thức getAccountInfo trong Manager để lấy thông tin Account tương úng với Account Number lưu vào đối tượng account của Bank\_Account

- Gọi phương thức getClientInfo trong Manager để lấy thông tin Client tương ứng với Account Number lưu vào đối tượng client của Client.

- Kiểm tra nếu AccountNum trong account bằng rỗng hoặc CIC trong client bằng rỗng thì hiển thị ra thông báo “Không tìm thấy Account” hoặc “Không tìm thấy Client”.

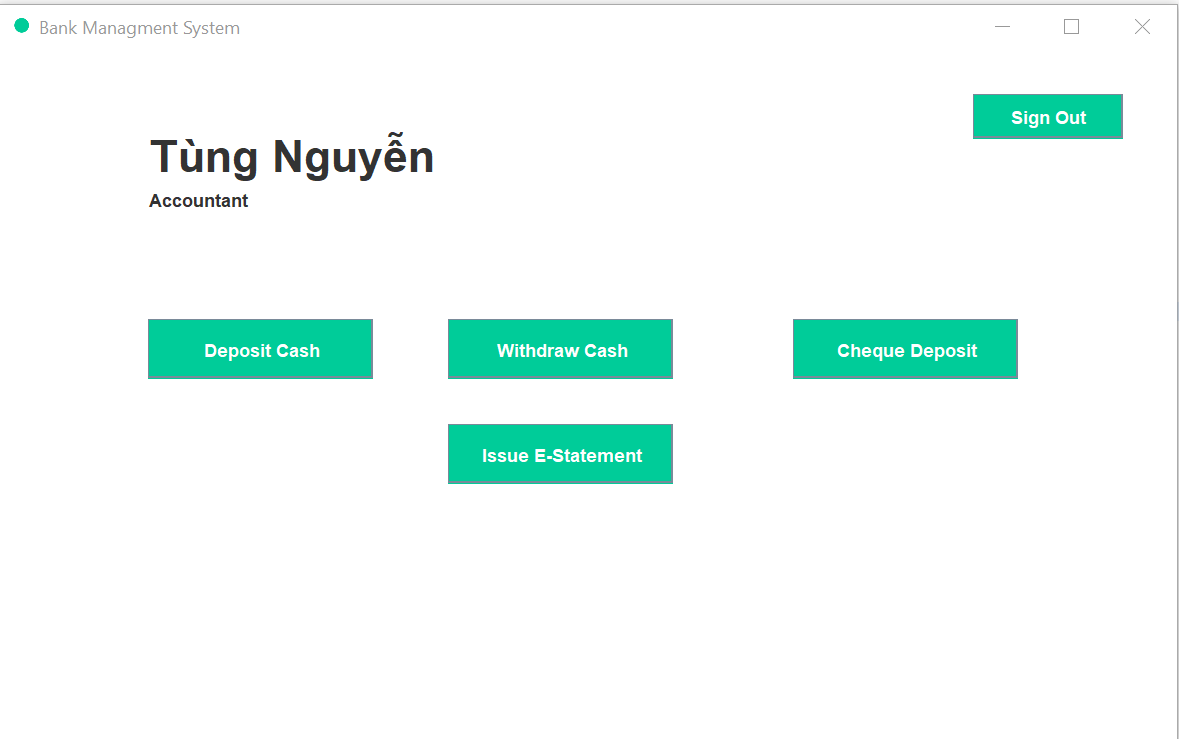
- Ngược lại thì hiển thị ra màn hàn các thông tin của account đó.

**11.17 updateClientInfo(JFrame frame, Manager manager)**: Phương thức hiện ra giao diện cập nhật thông tin Client.



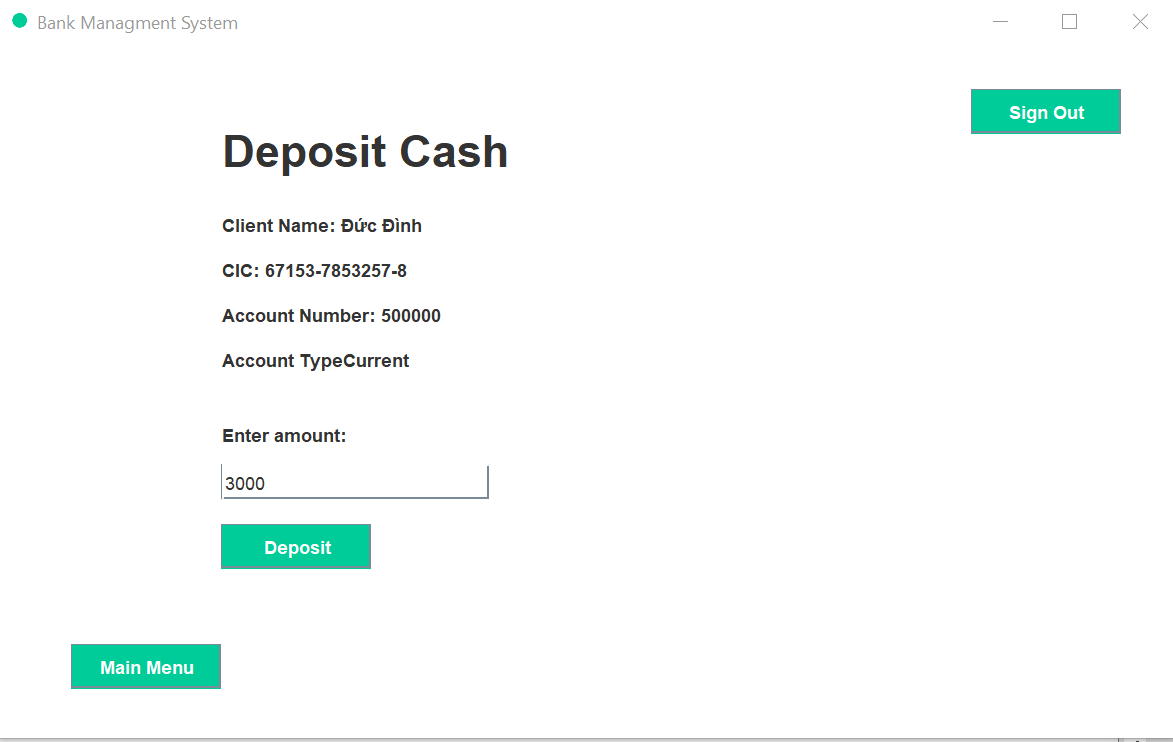
Tương tự như trên ghi click vào button **Searcn** nếu tìm thấy sẽ hiện ra thông tin Client và nhập và thông tin cần sửa. Click vào button **Update** sẽ bắt sự kiện và gọi đến phương thức updateClientInfo trong Manager để cập nhật lại thông tin.

**11.18 openAccountantMenu(JFrame frame, Accountant accountant):** Phương thức hiển thị ra các chức năng của Accountant.



Khi click vào các button tương ứng sẽ hiển thị ra các giao diện chức năng tương ứng.

**11.19 depositCash(JFrame frame, Accountant accountant, Client client, Bank\_Account account)** : Phương thức hiển thị ra giao diện Deposit Cash



Khi nhập đúng thông tin và click vào button **Search** sẽ hiện ra textField để nhập số tiền cần gửi và một button Deposit. Click vào button **Deposit**  sẽ bắt sự kiện và

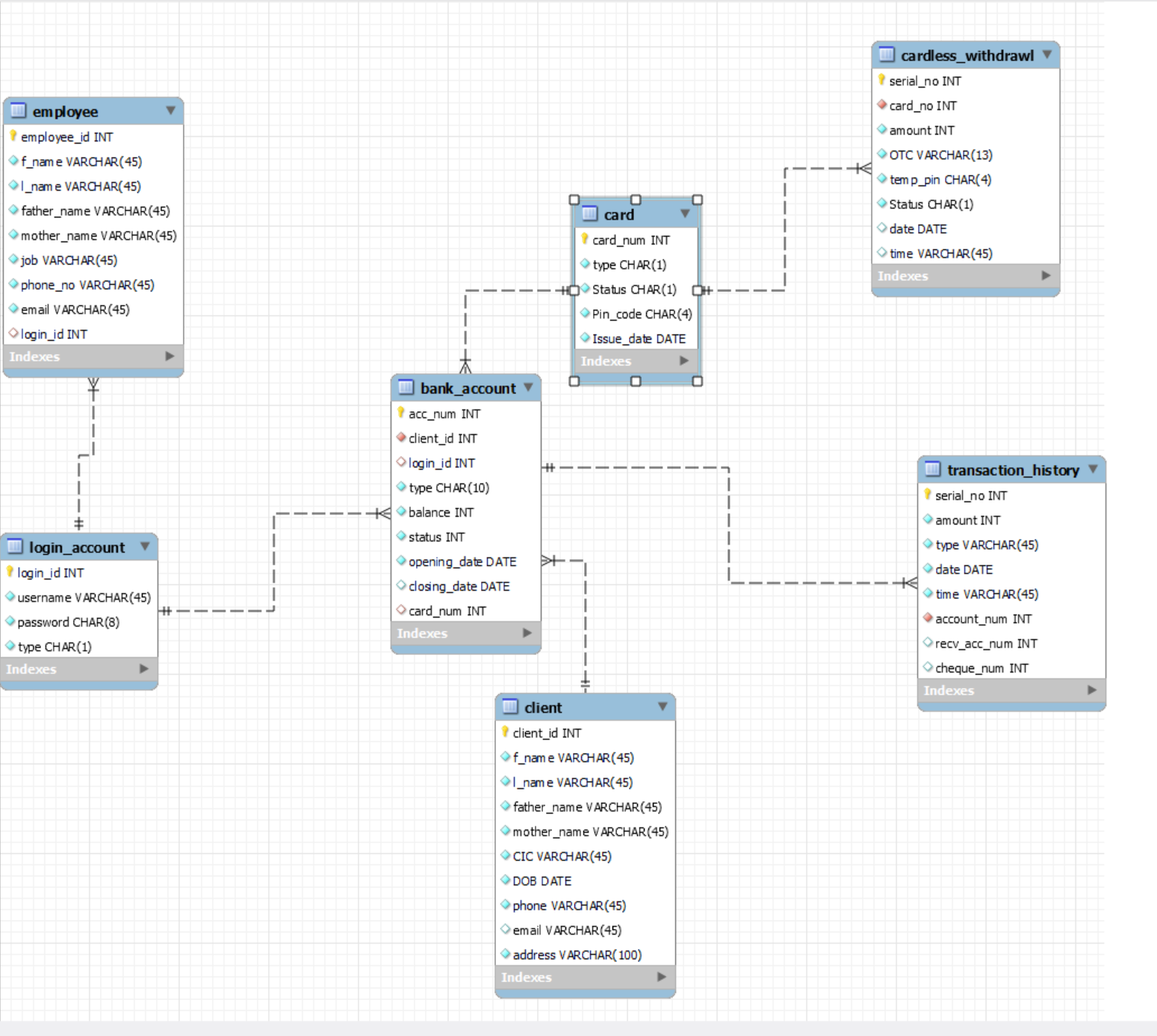
- Kiểm tra số tiền nếu hợp lệ: Gọi đến phương thức addAmount trong Accountant để thêm tiền vào tài khoản trả về giá trị trạng thái lưu vào biến r.Nếu r=1: Gửi tiền thành công. Ngược lại : Gửi tiền thất bại.

- Nếu số tiền không hợp lệ thì hiện ra thông báo.

**11.20 withdrawCash(JFrame frame, Accountant accountant, Client client, Bank\_Account account):** Phương thức này hiển thị ra giao diện rút tiền. Và khi click vào các button cũng tương tự như depositCash.

**11.21 eStatement(JFrame frame, Accountant accountant, Client client, Bank\_Account account)**: Phương thức tương tự eStatement1 ở giao diện client. Chỉ khác ở giao diện Accountant có thêm 1 bước tìm kiếm accountNumber và số CIC tương ứng trước rồi mới đến giao diện E-Statement.

# IV. Mô hình Cơ sở dữ liệu

****

# V. Tài liệu tham khảo

https://www.youtube.com/watch?v=pMR\_48AFA0&list=PL\_6klLfS1WqG8mRCW5a-bIViq1DbzQkp9.

https://github.com/topics/bank-management-system

Các bài giảng của thầy.