OOP PROJECT

Bank System

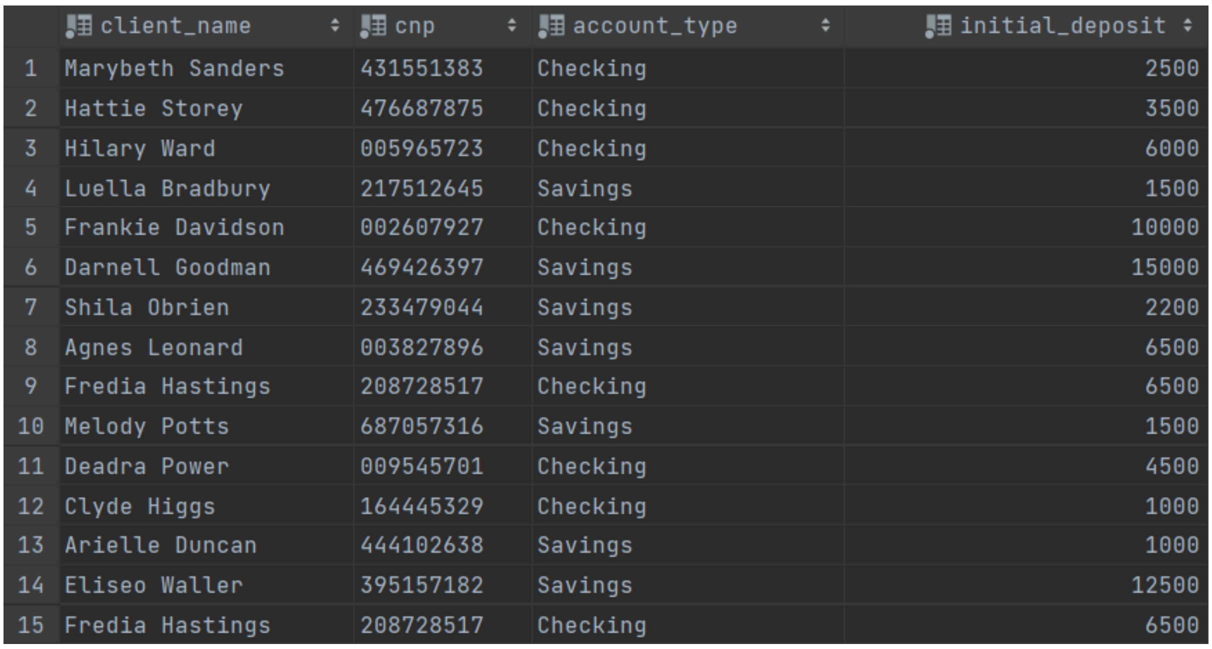
Student: Gătej Nicolae

Technical University Of Cluj-Napoca – Faculty of Automation and Computer Science

Teacher: Aron Baka

1. Short Description

Let’s assume that a couple of people would like to have bank accounts and every one of them completed some type of form in which they specified their full name, their personal numeric code and an initial deposit. They also specified in the form whether they want their account to be a Checking Account or a Savings Account. After they completed the forms and they gave their initial deposit to the bank, we received their data and stored it in a database. The database will look something like this:



	client_name	cnp	account_type	initial_deposit
1	Marybeth Sanders	431551383	Checking	2500
2	Hattie Storey	476687875	Checking	3500
3	Hilary Ward	005965723	Checking	6000
4	Luella Bradbury	217512645	Savings	1500
5	Frankie Davidson	002607927	Checking	10000
6	Darnell Goodman	469426397	Savings	15000
7	Shila Obrien	233479044	Savings	2200
8	Agnes Leonard	003827896	Savings	6500
9	Fredia Hastings	208728517	Checking	6500
10	Melody Potts	687057316	Savings	1500
11	Deadra Power	009545701	Checking	4500
12	Clyde Higgs	164445329	Checking	1000
13	Arielle Duncan	444102638	Savings	1000
14	Eliseo Waller	395157182	Savings	12500
15	Fredia Hastings	208728517	Checking	6500

What our bank system application does is it creates an account for every person in the database, an account with different properties, depending whether it is a Checking Account or a Savings Account. After we have created accounts for the clients, we will allow them to login with their account credentials and let them view their account information and do the following operations: deposit, withdraw and transfer money.

2. Architecture and Implementation

We will design a parent class called Account with the following attributes: Name, CNP, balance, account number and rate. The account number will be generated by a given pattern(depending on the bank). Let’s assume the pattern is the following:

- each account number has 11 digits
- first digit will be 1 if the account is a Savings one and 2 otherwise
- the next two digits will be the last 2 digits of the customer’s CNP
- the next five digits will represent an unique number
- the last three digits will be a random generated number

The rate will be given by some national economic standard meaning that both accounts will use an interface that determines their base interest rate. Let's assume the following rule:

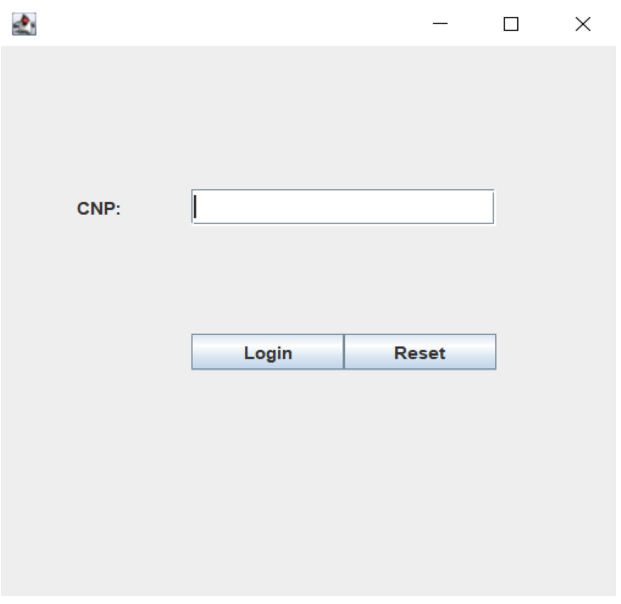
- Savings accounts will use .25 points less than the base rate
- Checking accounts will use 15% of the base rate

Now, both checking and savings accounts will have some specific attributes. We will use Inheritance here meaning that these two account types will have the same attributes as the parent class Account + some specific ones. The Checking Account will have a debit card ID/Number and a PIN Code and the Savings Account will have a Safety Deposit ID and a Safety Deposit Key.

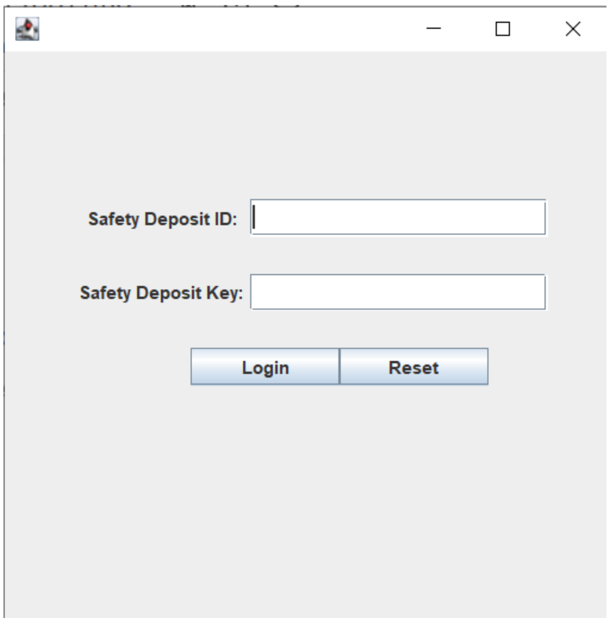
We will store the accounts we've created in a HashMap<String, Account>, where each CNP(String) will have mapped an account. We will store the information of each account in a .txt file so we can see that it properly works. Obviously, we will print the credentials encrypted in the file, so that no one can see the private information. The encryption algorithm is a very solid one and was implemented in a separate class called SecretCrypt. For example for a user the information will be printed like this:

```
Account Type: Checking
Name: Marybeth Sanders
CNP: 431551383
Balance: 2500.0
Account Number: 28310001990
Rate: 0.375
Debit Card Number: $2a$10$QlRiCYwJYQLm7azsa2USZO6ircuDHNQxuLXT.Jb4CbB1TzT3cSs2W
Debit Card PIN: $2a$10$iCerao0qoMwq30VXtdignOeLfQWETYTu4k7AYP1rNSxhWc4t5x.Ty
```

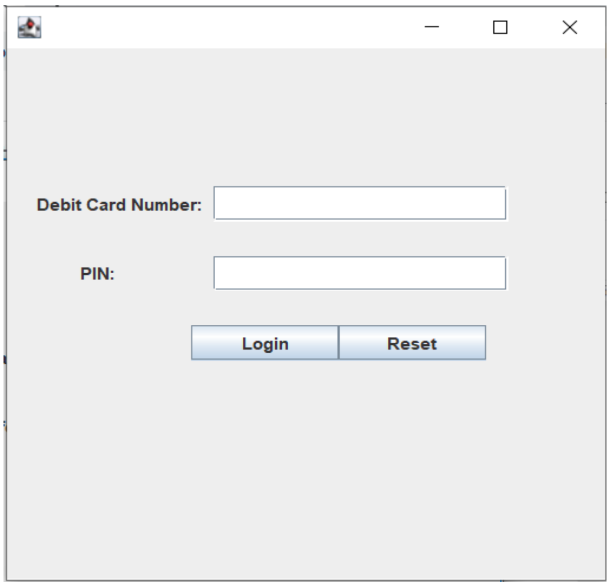
Let's see an example of how a client who received their credentials could use the app. Firstly, the client will open the app and the following window will pop:



The client will introduce his/her CNP and if it is not a valid one, meaning that it is not stored in the bank's system a message will appear, otherwise after pressing the login button, the following window will pop out. Window 1 will appear for Saving Accounts and 2 for Checking Accounts.

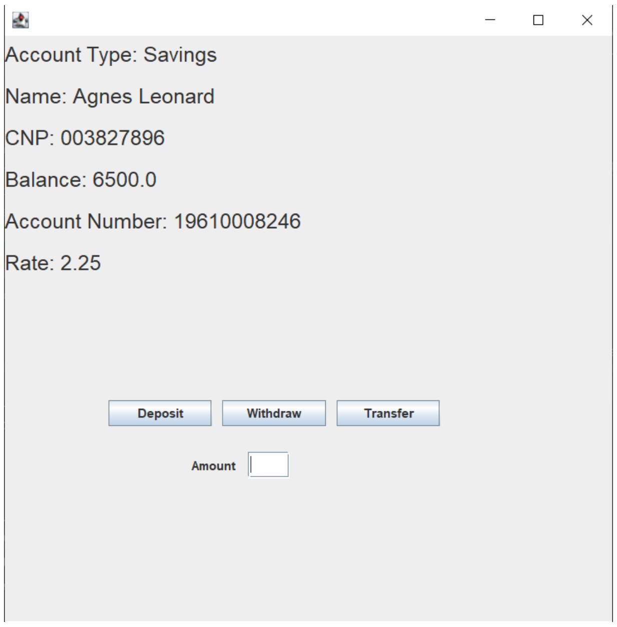


1)



2)

After logging in, the client will see something like this:


Account Type: Savings

Name: Agnes Leonard

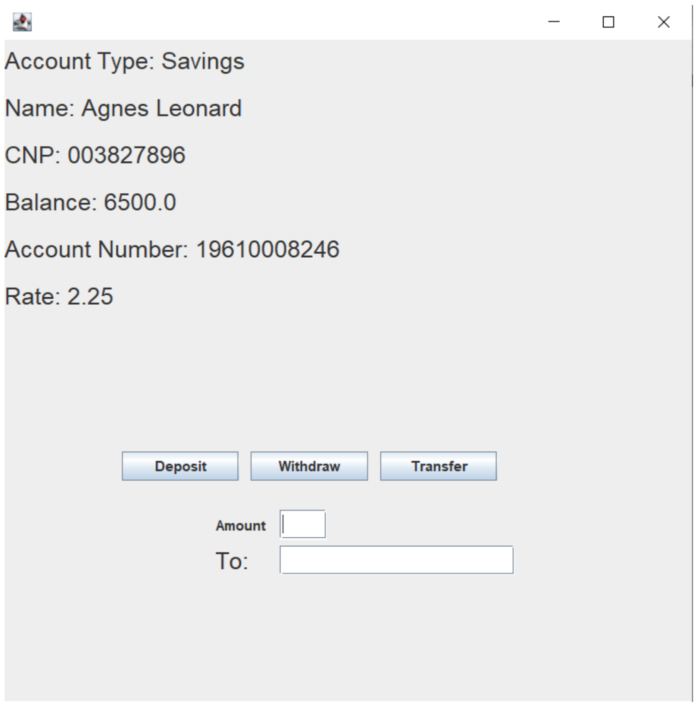
CNP: 003827896

Balance: 6500.0

Account Number: 19610008246

Rate: 2.25

Now, the client can see his personal information and can do different operations. To deposit or withdraw money, the client must introduce the amount in the amount box and then press the corresponding button(deposit/withdraw). After pressing it, the window will refresh and will show the new information. This change(in terms of money) can also be seen in the bankAccounts.txt file we discussed earlier. The transfer operation is a little bit more interesting. After pressing the transfer button, a box will appear on the window like in the following picture:


Account Type: Savings

Name: Agnes Leonard

CNP: 003827896

Balance: 6500.0

Account Number: 19610008246

Rate: 2.25

The client now must introduce the amount and the CNP's of the person to which he/she wants to send money. If the person introduces a wrong CNP or a CNP that is not in the system of our bank a message will pop and the transfer obviously will not be processed. If the person introduces the right CNP the transfer will be processed and the change will be seen in the window that will refresh but also in the bankAccounts.txt file.

3. Further Improvements

Some minor improvements that could be made would be to make the user experience better by designing the user interface to be more friendly. Another improvements that could be made would be to allow clients to have more than one type of an account. For example, because of the way we store the accounts, each CNP has mapped an account meaning that a person who possesses a Savings Account can not have a Checking Account and vice versa. Also, a nice improvement that could be done would be to allow transfers between clients registered at different banks.

4. Conclusion

In conclusion, the bank system app has some nice features and we covered a lot of OOP concepts and principles in the process of creating it, but it could also be improved by making the adjustments discussed before.