University: UNITBV
Faculty: IESC
Study program: IEC

PROJECT
CREDIT MANAGEMENT

Moraru Anastasia, 1st year, group 4LF522
30.05.2023

1. Specification defining the problem

1.1. Description of the theme

Achieve an application for managing credits granted by a bank. The app will keep track of creditors, will calculate rates, will record payments, refinance loan, payment in advance of a loan and will display various statistics.

1.2. Description of requirements

Develop a program which reads and retains information about clients and their bank accounts and use that data in functions for the calculation of credit rates, recording payments, refinancing loans, displaying various statistics etc.

1.3. Functional specification

The program is run from the main.cpp file which includes the files "Bank.h" and "Client.h".
The program has two classes: one for the accounts and one for the clients. The class "Bank" is a friend class of the class "Client". As a result, in the file "Client.h" the file "Bank.h" is included.
The program asks for a name, checks if it exists in the client file. If it exits it then displays the menu, otherwise it first saves the client to the file and then continues. The menu has 9 possible options ranging from the creating an account, checking the credit rates, making a payment etc. (they are mentioned in more detail in 1.4 User Interface). Using a switch function you can access each option which will call their specific functions. The functions are either member functions of the "Client" class or member functions of the class "Bank". Each function gets the parameter values from the account file in which are retained informations such as the number of the account, the account holder, the password, the credit amount, the credit period and the paid amount.

1.4. User Interface

The interface will ask for a name and then it will display the menu which has different colors for each possible option. The options are:

1. Create Account
2. Delete Account
3. Credit Rates
4. Make A Payment
5. Check Payment History
6. View Account
7. Refinance Loan
8. Statistics
9. Exit

2. Project Development
2.1. General objectives

Manage multiple clients with multiple credit accounts.

2.2. Description of data processed by the program

The class "Bank" is responsible for the parameters: account holder, account number, paid amount, credit amount, credit period, the payment structure p, password.
The class "Client" is responsible for the parameters: number of accounts, name and a parameter called bank which is of type Bank.
The program works with multiple files: "clients. txt" in which the name of the clients are retained; and multiple account files which retain information about the account such as account number, password, credit amount, credit period, account holder, paid amount; the names for these files are composed by the account number and they have the termination ".txt".

2.3. Description of program modules

The program has two classes.
The first class called "Bank" has the following parameters in its definition: account number, credit amount, credit period, payments, amount paid, account holder and password.
It has the next functions:

1. an initializer;
2. authenticateAccount which reads an account number and a password and checks to see if the match the information in accountFile
3. creditRates which gets the creditAmount and creditPeriod parameters from the file accountFile and displays the payment plan for the client, month by month;
4. makePayment which authenticates the account and then asks for the amount the client wishes to pay and then retains that information in a vector p.history;
5. checkPaymentHistory displays information from the vector p.history regarding the amount paid and the payment number;
6. refinanceLoan which authenticates the account and then gives 3 options: change the credit period, change the credit amount or exit; it overwrites the variable creditAmount or creditPeriod, depending on the choice;
7. Statistics which gets informations about the total credit amount given by the bank, the total credit period, the number of clients, the number of accounts etc and then displays averages for credit amount per client, credit amount per account, credit period per client, credit period per account and so on;
8. generateAccountNumber generates a random account number to ensure each account number is different;
9. get functions that return the private parameters in order to be used when working in the other class;
10. set functions that are used to change the parameters when working in the other class

The other class is called "Client" and contains a variable of type Bank used for the accounts of the client, the name of the client and the number of accounts which is initialized with 0. It contains the following functions:

1. an initializer;
2. saveClientToFile which saves the name of the client in a file named "clients.txt";
3. checkClientExists which gets the name of the client, checks the file "clients.txt" to see if the client exists and if it doesn't it calls the function saveClientToFile;
4. createAccount opens up an account and retains the information in the accountFile which has the same name as the accountNumber and increments the number of accounts of the client;
5. deleteAccount athenticates the account and then deletes the respective accountFile and decrements the number of account;
6. viewAccount authenticates the account and displays information from the accountFile including the name of the account holder, the number of the account, the credit amount, the credit period, the paid amount and the balance;
7. displayMenu includes the menu used to call all functions;
8. runProgramm calls the functions checkClientExists, saveClientToFile (if the client does not exist) and the displayMenu function;

The main source code includes both of these classes. In the main function we define a variable of type client and then we call the function runProgramm.
On the interface we will first be asked to enter our name and then the menu will appear. We can choose between the following options:

1. Create Account which calls the function createAccount;
2. Delete Account which calls the function deleteAccount;
3. Credit Rates which calls the function creditRates;
4. Make A Payment which calls the function makePayment;
5. Check Payment History which calls the function checkPaymentHistory;
6. View Account which calls the function viewAccount;
7. Refinance Loan which calls the function refinanceLoan;
8. Statistics which calls the function Statistics;
9. Exit which ends the program

3. Conclusions

I believe working with a class hierarchy would've simplified the project significantly since there wouldn't be a need for the get functions. I also would've liked to retain after the clients name the number of accounts so I could keep track of that. For now, that information is only memorized during the running time of the project, not in a file.
For future projects I would first make a plan of how I would like the project to run and only afterwards start programming because I found that without a plan of some sort the program can get messy and cluttered very easily.

4. References

https://www.geeksforgeeks.org/
https://www.w3schools.com/
https://cplusplus.com/reference/stl/
https://stackoverflow.com/