BANKING SYSTEM OPERATION

OBJECT ORIENTED PROGRAMMING PROJECT

K. Surya Teja	- 121cs0021
M.J. Sasi Kanth	- 121cs0030
K. Vineela	- 121cs0037
K. Dhanush Kumar	- 121cs0048

Problem statement:

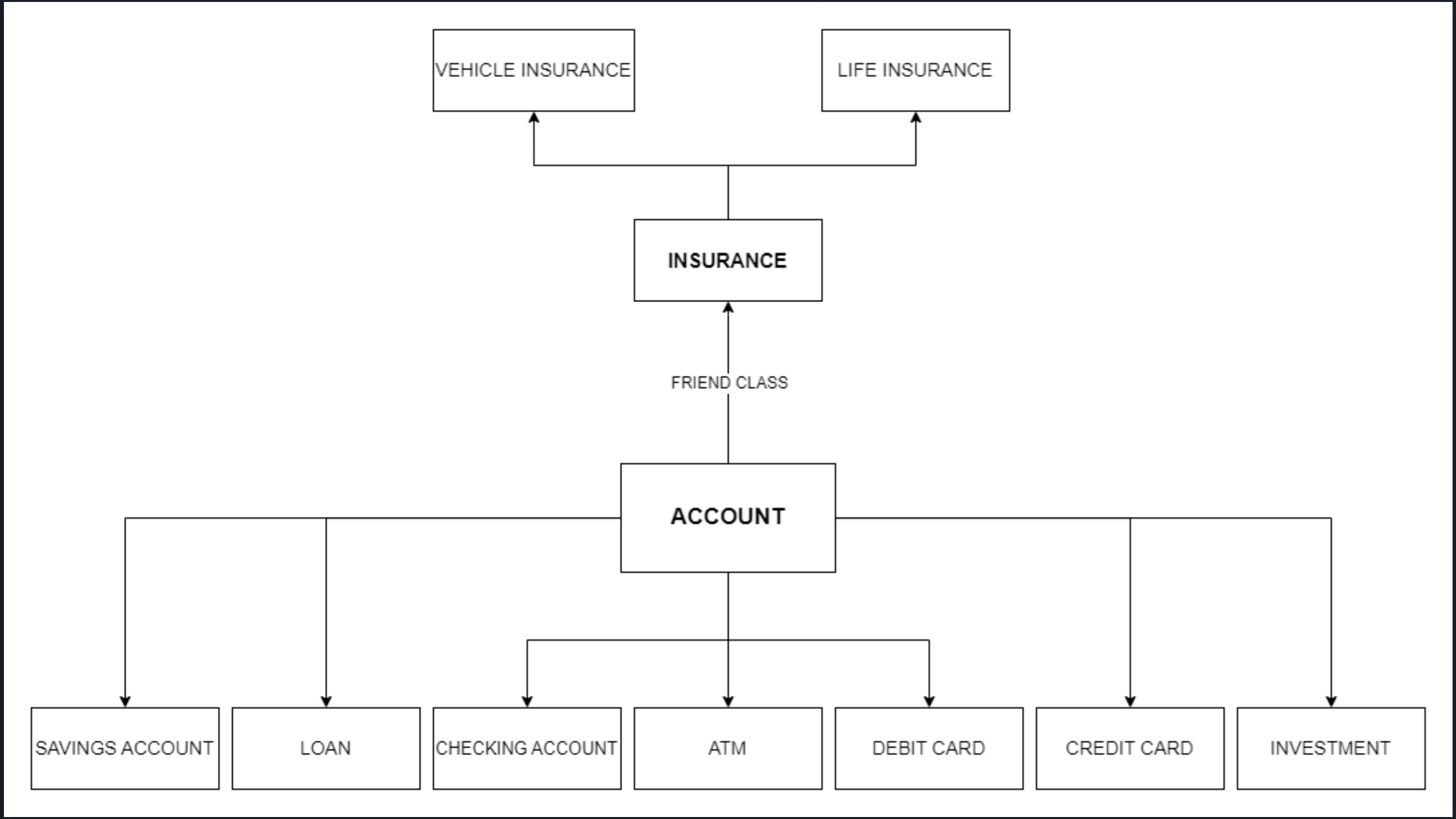
Design a banking system that provides different types of accounts to its customers, including savings, checking, credit card, and debit card accounts. Each account has a unique account number, an account holder name, and a password. The system should allow customers to create accounts, set and change passwords, view their account information, and perform transactions such as deposits, withdrawals, and payments. Each account type has its own unique features, such as interest rates for savings accounts, overdraft limits for checking accounts, credit limits for credit card accounts, and daily withdrawal limits and PINs for debit card accounts. The system should enforce appropriate restrictions on these features to ensure the security and integrity of customers' accounts.

CLASSES

The class used in our project are:

1.Account
2.SavingsAccount
3.CheckingAccount
4.CreditCard
5.DebitCard
6.Loan
7.Investment
8.ATM
9.Insurance
10.LifeInsurance
11.VehicleInsurance

INHERITANCE DIAGRAM



Details of Account Class

1. The class Account represents a bank account and has the following member variables and functions:
2. accountNumber: a string representing the account number, which is randomly generated and cannot be changed once set.
3. accountHolderName: a string representing the name of the account holder.
4. password: a string representing the password of the account holder, which is set by the user.
5. balance: a double representing the current balance in the account.
6. Account(): constructor function which initializes the accountNumber and outputs it to the console.
7. getAccountNumber(): returns the account number as a constant string.
8. getAccountHolderName(): returns the account holder's name as a constant string.

Details of Account Class

1. getPassword(): returns the account holder's password as a constant string.
2. getBalance(): returns the current balance as a double.
3. setPassword(): sets the password for the account by taking input from the user.
4. setAccountHolderName(): sets the name of the account holder by taking input from the user.
5. displayAccountInfo(): displays the account number, account holder name and the balance of the account.
6. deposit(): deposits the given amount to the account and updates the balance accordingly. Outputs a message indicating the success of the deposit.
7. withdraw(): withdraws the given amount from the account and updates the balance accordingly. Outputs a message indicating the success of the withdrawal, or a message indicating insufficient funds if the withdrawal amount exceeds the balance. Returns true if the withdrawal was successful, false otherwise. This function is virtual, meaning it can be overridden in child classes to provide different behavior.

Details of ATM Class

1. ATM() is the constructor of the class that prints a message when an object of the class is created.
2. addAccount(Account* account) function adds an account pointer to the vector of accounts in the ATM class.
3. displayAccounts() function displays the account number and balance of each account in the vector of accounts.
4. withdrawFromAccount(string accountNumber, double amount) function withdraws the given amount from the account with the specified account number and displays a success message if the withdrawal is successful.
5. depositIntoAccount(string accountNumber, double amount) function deposits the given amount into the account with the specified account number and displays a success message if the deposit is successful.

Details of Insurance Class

1. Insurance(): Constructor function that prints a message when an instance of the Insurance class is created.
2. getPolicyNumber(): Returns the policy number of the insurance policy.
3. getPolicyHolderName(): Returns the name of the policy holder of the insurance policy.
4. getPolicyAmount(): Returns the amount of the insurance policy.
5. setPolicyAmount(double amount): Sets the amount of the insurance policy.
6. displayPolicyInfo(): Virtual function that displays information about the insurance policy, including the policy number, policy holder name, and policy amount.
7. calculatePremium(): Virtual function that calculates the premium for the insurance policy. By default, it returns 0.0.

Details of Loan Class

1. Loan(): Constructor for the Loan class, which is called when an object of this class is created. It displays a message "Loan Created" on the console.
2. getInterestRate() const: Returns the interest rate of the loan account.
3. displayAccountInfo() const override: Overrides the virtual function displayAccountInfo() of the base class Account and displays the account information for a loan account, including the account type and interest rate.
4. deposit(double amount) override: Overrides the virtual function deposit() of the base class Account and subtracts the deposited amount from the account balance, as it represents a loan payment.
5. withdraw(double amount) override: Overrides the virtual function withdraw() of the base class Account and prevents any withdrawal from a loan account, as it does not make sense to withdraw money from a borrowed loan amount.

Details of CreditCardClass

1. CreditCard(): Constructor of CreditCard class that sets initial values and prints a message.
2. getCreditLimit(): Returns the credit limit of the Credit Card object.
3. getInterestRate(): Returns the interest rate of the Credit Card object.
4. displayAccountInfo(): Displays the account information of the Credit Card object, including the credit limit and interest rate.
5. deposit(double amount): Decreases the balance of the Credit Card object by the amount specified and prints a message.
6. withdraw(double amount): Checks if the amount to be withdrawn is less than or equal to the balance plus credit limit of the Credit Card object, and if so, decreases the balance by the amount specified and prints a message. Otherwise, it prints a message indicating that the payment limit has been exceeded and returns false.

Details of Investment Class

1. Investment() is the constructor of the Investment class which is called when an object of the class is created.
2. getInterestRate() returns the interest rate of the investment account.
3. displayAccountInfo() overrides the virtual function of the Account class and displays the investment account information including its balance and interest rate.
4. deposit() and withdraw() are the overridden virtual functions of the Account class which perform deposit and withdrawal operations for the investment account, respectively.

ROUGH IMPLEMENTATION

- The system has a main class, Accountant others for services like insurance, loan, and investment. The Account class stores the account holder's details such as name, account number, password, and balance.
- The program starts by displaying a menu with three options: Login, Create a New Account, and Exit. If the user chooses Login, they will be prompted to enter their account number and password. If the login is successful, the user will be taken to the main menu, where they can perform transactions like deposit, withdrawal, and checking their balance. They can also access additional services like ATM, insurance, loan, and investment. If the user enters the wrong password three times, their account will be locked.
- If the user chooses to create a new account, they will be prompted to enter their details such as name, date of birth, and phone number. An account number and password will be generated for them, and the details will be stored in the Account object. The user can then log in using the account number and password generated.
- The program uses a vector to store pointers to Account objects. The vector is initialized with 100 elements, and each element is initialized to a null pointer. When a new account is created, a new Account object is created dynamically using the 'new' operator, and the pointer to the object is stored in the vector.

CONCLUSION

- In conclusion, this project implements a basic banking system using object-oriented programming principles in C++. It includes several classes such as Account, ATM, Insurance, Loan, and Investment, each representing a specific aspect of the banking system. The main function acts as the user interface, allowing users to create new accounts, log in to existing ones, and access various banking services such as depositing, withdrawing, checking balances, and more.
- Overall, this project provides a good starting point for building more complex banking systems and demonstrates the power of object-oriented programming in developing real-world applications. With further improvements, such as adding more services, enhancing security features, and improving the user interface, this project could be a useful tool for managing banking transactions and services..