Bank Design

Team: Zeng chuqian, Miao annan, Xu qingyang

1. The project is to make a virtual bank system for a bank manager. The manager can provide services for the customers online. The project is divided into 3 parts: the backstage design of the bank functions, the design of database structure and the design of GUI.
2. Backstage design(Xu qingyang)

1.0 This part implements the basic functions of a bank described in the assignment pdf. I divide the object Bank into several parts: User, including bank customer and bank manager; Account, including accounts for checking, saving, applying loans, and trading stocks; Currency, which is the monetary system of this bank; Collateral, which will be used in the loan system; Transaction, which records every movement of the customers; Time, which is used to record current time; Stock, which represents a stock object; Tool, which is an auxiliary class to robust the system.

1.1 The brief class structure is as following:

User

Abstract class User

Class Customer extends User

Class Manager extends User

The User is designed to have some basic instance variables and methods, including the name, password, id and getter & setter methods.

The Customer class is designed to make the customer object to be able to create any type of account he/she wants – saving or making checks, requesting a loan, and trading stocks and viewing their transactions. It includes the Currency object to show the money of the customer, and the currency is passed to all the accounts of the customer to unite the money streams. The lists of Transaction and Collateral objects are to save the corresponding information of the customer. A customer object can be checked by itself or the manager with the methods in Customer class.

The Manager is designed to have the ability to check-up on a specific customer, all the customers and especially who owe the money. Also, the manager can get a daily report on transactions for that day.

Currency

Class Currency

The Currency is designed to show the money of the specific customer. It initially has three types of currency – dollar, pound and RMB. Due to the loss of the accuracy, the number of the money cannot be set as double, and I use the BigDecimal class to set the number variable. To show all types of the currencies, I use HashMap<String, BigDecimal> to show the type and quantity of currency respectively.

All methods of add and subtraction are included in this class. It can be used in all the accounts of user.

Collateral

Class Collateral

The class is specially designed for the loanAccount, it just shows the item name and price of the collateral, and it can be printed by overriding toString method.

Time

Class Time

The class just use the LocalDate and LocalTime together to get the current time, and it can easily calculate the time between some events by the method of LocalDate and LocalTime.

Tool

Class Tool

This class is designed as a class of helping methods. It does not have any instance variables. It is used for the error checking of the input streams, such as the range of a password.

Transaction

Class Transaction

This class can record every action of a customer. A transaction includes the information of action time, type of currency used, type of account used, customer’s ID, amount of exchanged and remained money.

Stock

Class Stock

This class represents a stock object which can be obtained by the StockAccount class. A stock object owns attributes like name, ID, the buy price, current price and number.

Account

Abstract Class Account

Class SavingAccount extends from Account

Class CheckingAccount extends from Account

Class LoanAccount extends from Account

Class StockAccount extends from Account

The Account is an abstract class representing the account attribute of a customer. It contains an accountType to indicate its kind, a currency passed by the customer, and a customerID to associate with the customer.

The SavingAccount is the basic account. A customer has to first build a saving account to save and withdraw money. Each saving or withdrawing action, no matter successful or not, is recorded as a transaction.

The CheckingAccount let customers make checks. It allows a customer to withdraw money from the bank with a fee. Each checking action, no matter successful or not, is recorded as a transaction.

The LoanAccount let the customer to apply for a loan by trading a collateral to the bank, and check all his/her loans to pay back to get their collaterals back. This account is set to only use dollars. Each apply/payback action is recorded as a transaction.

The StockAccount let rich customers buy and sell stocks in the bank. The stocks are maintained by the bank manager.

Bank

Class Bank

This is the main class of this project. A bank includes a manager and a list of customers. The bank provides the customers with customer menu and the manager with manager menu. The bank also let new customers create their accounts and write the information into database.

1.2 Main functions implementation:

Bank:

getExistUser(): This function imports the data from database to initialize the customers and their information. It creates several loops to obtain information: an customer ID loop to create customers; a currency loop to set the currencies of each customer; an account loop to set each customer’s accounts; a loan loop, a transaction loop and a collateral loop to initialize details of the customers. This function is called at the beginning of the program and every time the customers’ information is updated due to the manager when the program is running.

userMenu(): This function is the first layer of the bank system. It lets the user choose his/her identity from new customer, old customer and manager. As a new customer, the function will let the user input username and password and write the information into database. As an old customer or a manager, the function let the user login with correct name and password in the database.

bankMenu(id): This function passes the id of the customer to indicate which customer is calling this function. It serves the customer by providing a menu for the customer to choose from making actions in checking, saving, loan or stock account. The customer can also choose to check his/her transaction records.

managerMenu(): This function let the bank manager make actions after he/she logins. The manager can choose from viewing basic information of all the customers, checking one customer by his/her ID or name, viewing detailed information of all the debtors, checking the transaction records of the bank and fortune of the bank, and manage stocks.

Customer:

createAccount(str): This function passes a string to indicates the type of the account to be created. The function calls the initial function of each account class to create new account. As creating an account charges a fee, the function owns condition statements to judge whether an account is created successfully.

removeAccount(): This function can let the customer deletes accounts. It sets conditions, for example, a customer cannot delete saving account without withdrawing his/her deposit from the account, or a customer will lose his/her collaterals if he/she cannot pay back the loans and delete the loan account.

Account(Including the specific accounts):

save()&withdraw(): These 2 functions let the customer save and take money from his/her saving account. The functions are overloaded so that the user can indicate whether the money saved cannot be less than some amount or the money taken cannot be more than some amount. These functions actually call the add() and sub() in the Currency class.

initAccount(): This is a function in every extended class. For example, in SavingAccount class, this function lets the customer create a saving account when he/she save enough money for the service fee, then writes the information into database.

Menu(): This function is in every extended class. In SavingAccount class, this function let the customer save and withdraw money in 3 types of currency: Dollar, RMB, Pound. In CheckingAccount, this function lets the customer make a check. In LoanAccount, the customer can choose to view all the loans, apply for a loan or pay for a loan. In StockAccount, the customer can transfer money from saving account to stock account, view all the bought stocks and buy&sell stocks.

createTransaction(): This function is called everytime an action of the customer with respect to money exchange is made. It creates a transaction and writes it into the database.

Currency:

add()&sub(): These 2 functions actually implement the save and withdraw abilities of saving account. But these functions are called frequently during other transactions, for example, when the customer pays a fee view the transaction records.

* 1. Basic settings:
* Opening checking account, saving account or stock account charges 5 dollars, loan account charges 8 dollars.
* Closing each account charges 5 dollars.
* Viewing one’s transaction records charges 10 dollars.
* Withdrawing money charges 2% fee.
* Applying for a loan will get 90% of the price of the collateral.
* The daily interest rate is about 0.01%.
* Only the customers who own over $5000 in their saving account can open stock account. Opening the stock account should at least transfer $1000 from saving account, while maintaining the account should ensure that saving account owns at least $2500.

1. Database and connection to Java program (Zeng, Chuqian)

This part mainly concludes the design of SQL database structure and the database’s connection to our Java project.

2.1 SQL Database

Table related to customer:

CUSTOMER table (ID, NAME, PASSWORD, SAVING\_ID, CHECKING\_ID, LOAN\_ID, STOCK\_ID). It uses ID as primary key, stores the name and password for customer login, and uses the corresponding ID as foreign key to link to the four kinds of account table. If customer has no such kinds of accounts, the corresponding ID is -1.

SAVING\_ACCOUNT table (ID, CUSTOMER\_ID, CURRENCY\_TYPE, MONEY\_AMOUNT). Storing all the existing saving account. It uses ID as primary key, uses CUSTOMER\_ID as foreign key to figure out which customer it belongs to, uses CURRENCY\_TYPE to distinguish what kind of currency for the account (Dollar, RMB and Pound), and MONEY\_AMOUNT to store the amount of money in the account.

CHECKING\_ACCOUNT table (ID, CUSTOMER\_ID, CURRENCY\_TYPE, MONEY\_AMOUNT). Similar to SAVING\_ACCOUNT, but the MONEY\_AMOUNT is zero because checking account has no money, and the money for the check is from saving account.

LOAN\_ACCOUNT table (ID, CUSTOMER\_ID, MONEY\_AMOUNT). MONEY\_AMOUNT is the total loan amount for the corresponding customer.

LOAN table (ID, ACCOUNT\_ID, COLLATERAL\_ID, LOAN\_AMOUNT). The table stores all the existing loan records. ACCOUNT\_ID is a foreign key, linked to ID in LOAN\_ACCOUNT table. COLLATERAL\_ID is a foreign key linked to ID in COLLATERAL table, and each loan record is matched with one item in COLLATERAL table, since one loan needs one collateral. LOAN\_AMOUNT is the money amount that customer borrows in this loan record.

COLLATERAL table (ID, NAME, ACCOUNT\_ID, VALUE). ACCOUNT\_ID is a foreign key, linked to ID in LOAN\_ACCOUNT table. VALUE is the value for the collateral.

STOCK\_ACCOUNT table (ID, CUSTOMER\_ID, MONEY\_AMOUNT). MONEY\_AMOUNT is the amount of money that customer has in the stock account, and the money is transferred from the saving account from the customer.

STOCK\_LIST table (ID, NAME, PRICE). The name and price for all the stocks. Should be maintained by the manager.

STOCK\_OWNERSHIP table (ID, ACCOUNT\_ID, STOCK\_ID, PURCHASE\_PRICE, HOLDINGS). The table store all the existing owning record of stock for all the customers. ACCOUNT\_ID is a foreign key, linked to ID in STOCK\_ACCOUNT table. STOCK\_ID is a foreign key linked to ID in STOCK\_LIST table, specifying which stock is bought in this record. PURCHASE\_PRICE is the price of the stock when customer buys the stock. HOLDINGS is how much share the customer buys in this record.

TRANSACTION table. Record all the transactions of the customers.

Table related to manager:

MANAGER table (ID, NAME, PASSWORD, LAST\_UPDATE\_DATE). Record the information for the only one manager. LAST\_UPDATE\_DATE is the latest manager login date.

MANAGER\_ACCOUNT table (ID, CURRENCY\_TYPE, MONEY\_AMOUNT). Store all the money manager has for three kinds of currency. Manager has 100000000 for each currency at first, and use the money to lend money to customers.

2.2 Database’s connection to Java project

We have several java classes in package “connect\_database” to add, delete, alter and search information in database.

Connector.java:

Get the connection to database. We can change the user name and password according to local database if we move the program to run on another computer.

CustomerAddingFunction.java:

Database adding function related to customer.

addCustomer: add a new customer into the database.

addSavingAccount: add a new saving account for a customer.

addCheckingAccount, addLoanAccount, addStockAccount: similar to above.

addLoan: add a loan record of dollar for a customer into the database.

addStockOwnership: add a stock owning record of dollar for a customer into the database.

addTransaction: add a transaction record for a customer into the database.

CustomerDeletingFunction.java

Database deleting function related to customer.

deleteSavingAccount: delete the saving account for customer.

deleteCheckingAccount: similar to above.

deleteLoanAccountWithoutPayback: delete the loan account without paying back the loan of customer, and all the collaterals remain in the bank.

deleteOneLoan: delete the loan record with paying back the loan of customer, and the corresponding collateral is removed from the bank.

deleteOneStock: delete one stock record for customer, and the corresponding money is added to the stock account.

CustomerAlteringFunction.java:

Database altering function related to customer.

alterSavingAccount: alter the balance with the money amount changed in saving account according to currency type.

alterStockAccount: alter the balance with the money amount changed in stock account according to currency type.

CustomerSearchingFunction.java:

Database searching function related to customer.

customerLogin: login for old customer with name and password.

searchSavingMoneyAmount: get the balance in saving account according to currency type.

searchLoanMoneyAmount: get the total loan amount in loan account.

searchLoanList: get the total loan list for a specific customer.

searchStockMoneyAmount: get the total balance in stock account.

searchStockList: get the total stock list owned by a specific customer.

searchTransaction: get all the transaction list for a specific customer

ManagerFunction.java:

Database function for manager.

managerLogin: login for manager.

searchManagerAccount: get the money amount in manager account according to the currency type.

alterManagerAccount: Alter the balance with the amount of money changed in manager account according to currency type.

searchAllCustomer: get all the customer list for manager.

searchAllCustomerAccount: get all the accounts of customers for manager.

searchAllLoanCustomer: get all the customer list with loan account for manager.

searchTransactionToday: get all the transaction list for today for manager.

searchAllStockList: get all the stock list for manager.

addStock: insert a new stock into the stock list.

alterStockPrice: alter a stock price according to stock\_ID.

updateDailyInterest: update daily 0.01% the interest paid to high-balance customers and 0.1% interest charged to loan accounts. It works each time the manager logins to the system, and counts automatically the interest changed since last time login.

2.3 Benefit:

The usage of SQL database allows the data created to be persistent.

It also eases our work on selecting and altering the data.

The connection between Java and SQL database is mature, which adds to the robustness and portability of our system.

SQL database is widely-used in real-life development of program, for which our program is more like a real system.

1. UI design (Annan Miao)

This part introduces the implementation of the UI design of our Java program using Java Swing in Netbeans.

3.1 Overall Design

The UI version transplants the functionality of the already implemented command line version of the Bank System program. There are several UI interfaces provided:

1. Login/Signup for customers and manager (2) Manager screen to check customers, debtors and transaction records (3) Manager screen to add and change stocks (4) Customer screen that direct user to open account, make a check/loan, save/withdraw money (5) Customer screens to enable creating saving/stock accounts (6) Customer screen to make checks (7) Customer screen to make loans (8) Customer screen to buy/sell stocks.

3.2 Implementation

The implementation uses Java Swing GUI in Netbeans IDE. Netbeans provided visualized JFrame creation and drag&drop items will automatically generate codes.

3.3 Benefit

The GUI design make user interaction much more easier. Also users can with several different screens, which means that we can use different functionalities simultaneous (e.g. buy stocks and make a loan in 2 different screens at the same time).