FAST PAISA

Banking System

(Technical Documentation)

Group Code: C

Group Members:

Sufiyaan Usmani (21K – 3195)
Qasim Hasan (21K - 3210)
Ahsan Ashraf (21K – 3186)

Project Start Date: February 4, 2022

Link: [sufiyaanusmani/FAST-Paisa (github.com)]

Qr Code:

Tools Used:

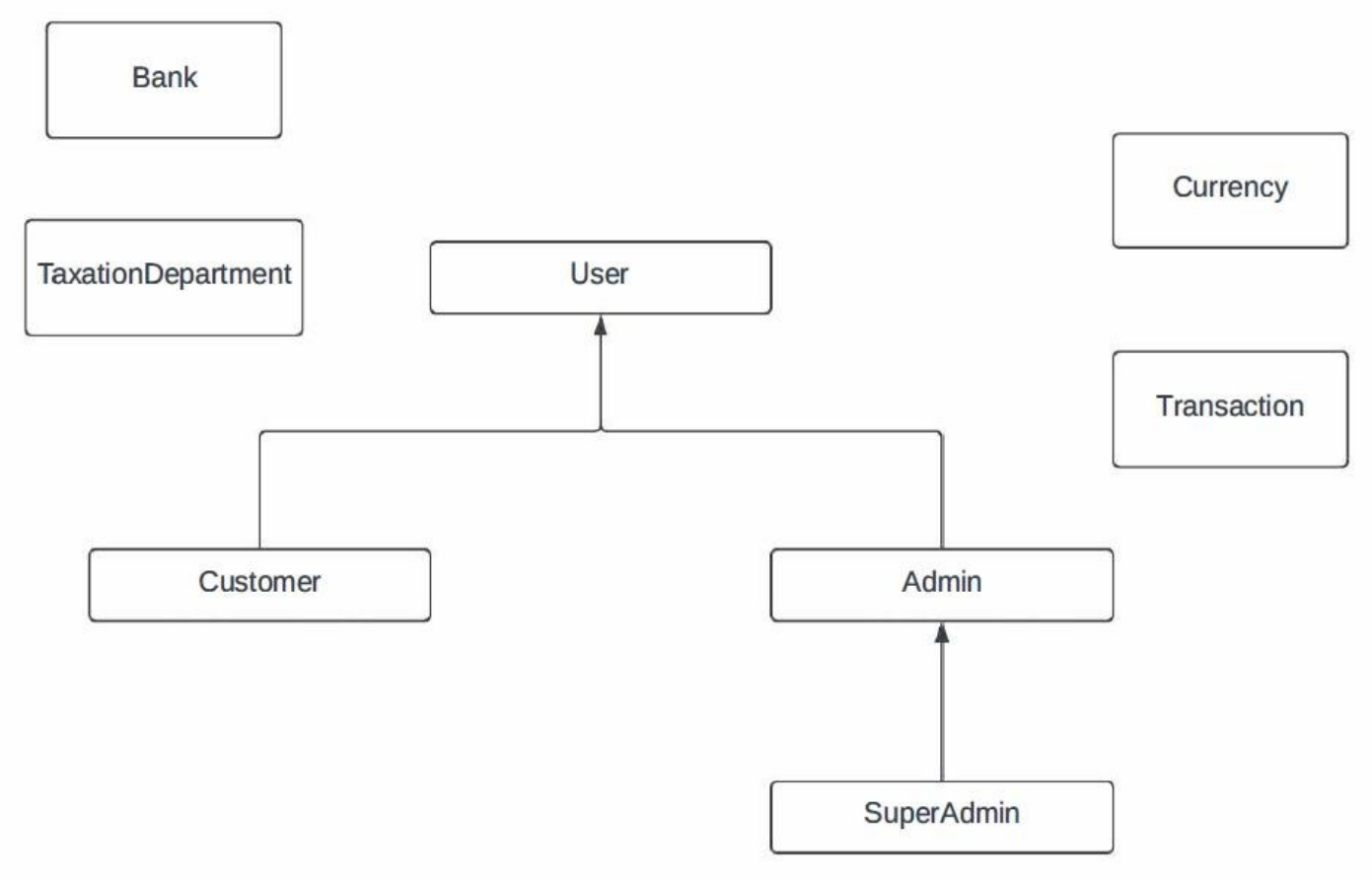





HEADER FILES

#include <string>
#include <iostream>
#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
#include <stdio.h>
#include <fstream>
#include "ui.h"
#include <windows.h>
#include <ctype.h>
#include <iomanip>

CLASSES




```
Class Person;
Class Bank;
Class SuperAdmin;
Class Admin;
Class Customer;
Class Taxation Department;
Class Transaction;
Class Currency;
```

Explaination of Classes:

1. USER:

User class is useful when presenting common qualities because all types of users, such as admin, super admin, and customer, have the same properties. The user class is the parent class in this program, however, it is also an abstract class because users do exist, but only in the forms of customer and admin. The following data members were deemed necessary for this program: Account Number: This variable in a file can be used to look for any client or admin. Name, Age, Gender, CNIC, Contact Number, Email, and Passwords are all crucial factors to reveal more detail about each customer or admin. User classes have the function to set and get all these data members. Other than that, some virtual functions also exist to easily interact with all other child classes. Login, portal, and portal menus for both customers and administrators to display an interface.

Delete account, viewmyinfo, update any detail of any customer or admin, and most importantly, update password are all virtual functions that demonstrate inheritance, abstract class, polymorphism, encapsulation, and many other oop concepts like making classes friend to each other and giving private data members of different classes excess to each other through the friend function.

2. ADMIN:

Admin is one of the user's child class, with the most access to client data and the entire bank ecosystem. The admin is provided functionality by concepts of use of OOP, which are explained below:

View my Info: A function common in the program helps to see the data of the admin.

Login: Another typical feature of the application was the ability to log in using the admin's account number and the password he created for his account.

Portal/Portal menu: A code involving functions to display an interface, one of which offers the code for display and the other which provides the choice of selecting any admin functionality.

Delete account: An important function that was created to remove any customer's account simply by providing his account number used the same logic as deleting objects from a line by searching that customer.

Database Backup: When a new account register or transaction occurs, this function is called so that if the main system becomes corrupted, a backup of the data in temporary files is available.

Search Customer: A function with more than four sub-functions for searching for a customer's specifics about his behaviour gives admins quick access to the information they need. The admin may easily look at the details by searching by name or account number in both ascending and descending order.

View Customer accounts: a function that reads the binary file of customer data and displays the transaction, transfer or deposit of any client.

Account Settings: A feature that allows an administrator to update his account by giving him access to functions such as changing his password, email, information, and contact information.

3. SUPERADMIN:

A subclass of user and a child class of admin with both user and admin characteristics. The primary goal of this class is to have only one banking system owner who can manage the admins who work for his organization. To manage this system, he has access to all other admins as well. This super admin can view admins' data, and delete admin accounts and bank revenue using the taxation department's calculations which is an independent class. This class completes the concepts of multilevel Inheritance.

4. CUSTOMER:

The most crucial class in the entire banking system without it is nothing. Functions of login, portal, delete the account and updates are all same as The functions in the Admin class but concerning the customer data. With limited access to the service, the primary goal of the customer is to have access to his account's savings, transfers, and withdrawals. The customer class's functionality is described further below:

CreateNewAccount: Give the customer an account number and put it in a file called customer.dat. Aside from that, client information and a password are required to keep the customer account exclusively accessible to him and not the other customers.

View Transaction History: The customer can search his transaction history but not others.

Get Amount: A function which helps to show a customer the amount he currently has in his account.

Email: When a customer creates an account or does any transaction, deposit, or transfer, a code is executed to send an email to the customer.

Transfer/withdraw/deposit: Three functions that interact with the transaction class assist the customer in doing these tasks by allowing him to transfer, deposit, and withdraw amount.

5. CURRENCY:

A separate class was created to allow both admins and customers to view currency rates for major currencies. Admins can also add, amend, and set currency rates here. To make the banking system good, generally, concepts of object filing are utilized, such as updating, writing, and reading a major class.

6. TRANSACTION:

This independent class was created because, as we all know, customers have many transaction operations and properties. To make this system easier to understand, we created the transaction class, which helps to maintain transaction operations such as viewing transaction history for both admin and customer, taking the amount in and out, and assigning a customer with a transaction id.

7. TAXATIONDEPARTMENT:

This independent class was created to receive money and profit from customers' accounts to generate revenue for the bank's administration and maintenance. It just has two functions: one to calculate and display tax from client accounts at a rate of 0.15 per cent.

8. BANK CLASS:

Bank classes were formed to compute the total number of accounts and the total amount because they assist the taxation department in imposing taxes and generating income for the bank.

FUNCTIONS OF THE PROGRAM:

int mainMenu();
void loginAsCustomer();
void init();
void fillData(char[256], char[30], char[256]);
int sendMail(int);
void bankPolicy();

POINTERS USED:

```
FILE *locBit;
FILE *locBit1;
FILE *locBit2;
FILE *locBit3;
FILE *locBit4;
FILE *MainCRET;
```

MAIN

The main has classes objects of (Customer, Admin, Currency, Taxation Department), and we may send a command to act on a choice on the main menu that is predefined in the mainMenu() method previously, in which the interface is already established, by using the objects of classes using association. Following the execution of any case, the switch case can have a function redirect to its relative working location, enabling access to a class function or classes functions. The case will keep running until any correct case is not selected with the help of the while loop.

USER CLASS

FUNCTION AND THEIR WORKING

setName():

Explaination

First, we'll make a backspace count integer. The customer will provide his or her entire name. We've set circumstances so that alphabets can only be entered by incrementing that integer; nevertheless, if the user tries to enter a number, the backspace count will be decremented. This process will continue until the entire name has been entered.

setAge():

Explanation

First, we'll make a backspace count integer. The customer will provide his or her age. We've set circumstances so that alphabets cannot be entered as same as we did in the set age function. This process will continue until the age has been entered.

setGender():

Explanation

We will now enter/set the data and the same working will be applied as it did setname/setage.

setCNIC():

Explanation

We will now enter/set the data and the same working will be applied as it did setname/setage.

setContactNumber():

Explanation

We will now enter/set the data and the same working will be applied as it did setname/setage.

readData():

Explaination

Simply said, the function will open the file, read the data, and then close the file after displaying it.

inputPassword():

Explaination

When this function is called the password limit through the condition and the use of (*) to hide the password we are entering.

setEmail():

Explaination

We will now enter/set the data and the same working will be applied as it did setname/setage.

setPassword():

Explaination

We will now enter/set the data and the same working will be applied as it did setname/setage.

getAccountNumber():

Explaination

This function will get the account number of that current active user.

getname():

Explaination

This function will get the name of that current active user.

CUSTOMER CLASS:

FUNCTION AND THEIR WORKING

login():

Explaination

To log in to an account with this function, we must verify two things: password and id. After entering the password and id, the file will be opened to compare the data, and if the data matches, the user will be granted access to the account's features. When the user enters a password, the function input password() is called to specify the requirements and hide the password with (*) when entering the password. When the function is invoked during runtime, an animated function will be called to display a bar using the gotoxy function, which can also be used to display an interactive interface.

CreateNewAccount():

Explanation

When this function is called, the bank policy is displayed by calling bank policy(); then a random account number is assigned to the user so that he can begin filling in his details (name, gender, age, email, contact number, password, CNIC); finally, the account amount is initialized to 0 and the data entered by the user is displayed again so that he is satisfied; finally, he is asked whether he wants to create an account or not. An email will be sent to the new customer as soon as the account is created. All this will be saved in storeData().

GenerateAccountNumber():

Explanation

Before assigning a new account number, we must first verify that the number created does not already exist. To do so, we will open the file in which customer data is saved, compare the account number and its size, and then use a bool to find the result, after which the new account number will be saved in the file.

storeData():

Explanation

In this method, we will first open the fill, write the data, and then close it. We will also use fout to check for errors.

getAmount():

Explaination

This function will return the amount of a user active at that time.

viewMyInfo():

Explaination

Display all the data relating to the customer including his current balance.

portal():

Explaination

After logging in, the customer portal will run, and we will notice that the client has options to choose from (deposit, withdraw, transfer, transaction history, settings), and a function named portualMenu() will be called, with the main aim of displaying the customer portal's interface.

portalMenu():

Explaination

When the portal function is called, the function portalMenu is called, which displays the customer's options and returns a choice to the function portal.

depositAmount():

Explanation

In this function, the user adds money to his account balance, and a transaction object is created as a result. It is not possible to input a negative or zero value. If the answer is affirmative, the file will be opened, the deposited amounts will be credited to the customer's account and the file will be closed. When the money is placed, an email will be sent with the identical code that was used to create the new account () function.

withdrawAmount():

Explanation

The user withdraws money from his account using this method, and the amount can be negative or zero. Another criterion is that we will verify whether the amount the customer had before withdrawing is sufficient.

transferAmount():

Explanation

This function allows a user to send money to another customer account by simply entering the receiver account number and the amount he wants to send. We will then check the file to see if the receiver account exists, and if he does, the sender will send him the

desired amount. It will also ensure that the sender sending money has enough credit to send the money to that other customer.

deleteAcount():

Explanation

If a customer wants to shut his account with that bank, he can simply delete it. After that, he will be given the option of yes or no. If he chooses yes, the account will be opened in a file, and the data for that account will be destroyed.

ADMIN CLASS:

FUNCTION IN ADMIN CLASS AND THEIR WORKING

storeData():

Explanation

In this method, we will first open the fill, write the data, and then close it. We will also use fout to check for errors.

login():

Explanation

To log in to an account with this function, we must verify two things: password and id. This time around a super admin also exist if we enter his detail we will log in to his account otherwise normal admins account. if the account logging is After entering the password and id, the file will be opened to compare the data, and if the data matches, the user will be granted access to the account's features. When the user enters a password, the function input password() is called to specify the requirements and hide the password with (*) when entering the password. When the function is invoked during runtime, an animated function will be called to display a bar using the cursor position function, which can also be used to display an interactive interface.

createNewAccount():

Explanation

When this function is called a random account number is assigned to the user so that he can begin filling in his details (name, gender, age, email, contact number, password, CNIC); and the data entered by the user is displayed again so that he is satisfied; finally, he is asked whether he wants to create an account or not. An email will be sent to the new admin as soon as the account is created. All this will be saved in storeData().

portal():

Explanation

After logging in, the Admin portal will run, and we will notice that the client has options to choose from (customer account, viewmyinfo, view transaction history Admin, delete Account, add currency, search

customer, update currency rates, accounting settings, generate a report, create customer database backup), and a function named portalMenu() will be called, with the main aim of displaying the Admin portal's interface.

portalMenu():

Explaination

When the portal function is called, the function portalMenu() is called, which displays the admin options and returns a choice to the function portal.

ViewMyInfo():

Explaination

Display all the data relating to the admin.

viewCustomerAccounts():

Explaination

In this function, the Admin has access to any client's whole information simply by accessing the customer. The relevant customer information will be sent to the admin by the bank.

generateTransactiomID():

Explaination

In this function, we will simply create a random id and saved the details in the transaction.bank file.

storeTransaction():

Explanation

This function will save the transaction history using transaction ID and account number. We will save each transaction as well as the money, and then the file will be closed.

viewTransactionHistoryAdmin():

Explanation

In this function, the admin will view the transaction history of every customer

viewTransactionHistoryCustomer()

Explanation

In this function, the customer can only view his transaction history.

deleteAccount():

Explanation

If the admin wants to shut the customer's account with that bank a customer object will be used, he can simply delete it. After that, he will be given the option of yes or no. If he chooses yes, the account will be opened in that file, and the data for that customer account will be destroyed.

CreateCustomerDatabaseBackup Animation():

Explaination

This animation function is run in this function when the function createCustomerDatabaseBackup() is executed within the function to demonstrate that the backup has been produced using cursor location to make the interface seem good..

searchCustomer():

Explaination

This feature allows the administrator to search for a customer using their name and account number in both ascending and descending order. Once the administrator selects his selection, he has access to the search algorithm.

CURRENCY CLASS:

FUNCTION IN CURRENCY CLASS AND THEIR WORKING

generateCurrencyCode():

Explaination

Every time a new currency is added to the banking system,
a code is generated.

addcurrency():

Explaination

Only the admin can add a currency and change its rate, therefore generateCurrencyCode is used to assign the new currency a code that the admin can easily access.

viewCurrencyRates():

Explaination

The currency.bank file will be opened in this function, allowing both customers and administrators to view currency rates.

UpdateCurrencyRates():

Explaination

We can adjust the value of any currency code in this method by inputting it again.

setRate():

Explaination

This function will set the rate to 0.0 until the user enters a rate.