Object Orientated Programming Assignment

C20492576 – Finn Maguire

**Classes and Methods**

Customer Class:

The account class is used to create instances of objects that relate to the customer’s account.

Methods:

\_\_init\_\_():

This magic method is used to initialise self-variables that will be used in the methods. These variables are the username, age, filename, and password.

createCustomer():

This method is used to create a new user account given the provided input. The method first checks to see if the inputted username is already inside the file. If so, it returns an error message to the user. It then appends the users’ details into the file. To make this functionality more compact the append uses f-strings which allows the program to run code inside the strings. Its uses functions and methods to create information that the user doesn’t input themselves such as their id number or their calculated credit limit.

findCustomer():

This method is used to take a user inputted username and find its details inside the file. If it locates the line inside the file, it returns a separated list of all the users details using the split() function.

findCreditLimit():

This method is used to assign a credit limit to the users details depending on their age. The flat credit limit is 1000 at 18 years of age which then increases by 100 by year, eg 19 years of age gets a credit limit of 1100.

\_\_Str\_\_():

This magic method is used to display the currently logged in users’ information along with the total balance of their accounts.

Account Class:

This account is used to create instances of objects that are used create and manipulate file contents relating to the user’s accounts.

\_\_init\_\_():

This magic method is used to initialise self-variables that will be used in the methods.

createAccount():

This method is used to create a new account given the provided input. The method takes the selected account type and appends a new account to the file which includes information such as the users ID, the account ID, the account type, the default balance 0, the credit limit associated with the users account and a default date of the last transaction which is used later on.

displayAccounts():

This method is used to display accounts associated with the currently logged in user. It tries to find accounts in the file by comparing the users logged in ID number with the user ID number that is included with the account’s information. If the IDs match the contents are printed in a formatted way to the user. It also includes subsequent validation in the case the user has no accounts.

rewrite():

This method is used to rewrite lines within the file upon either deleting, depositing, withdrawing, or transferring. It takes in certain parameters depending on the user’s choice and input. Before withdrawing or depositing it will first run checks on the chosen account to see if any of the account’s restrictions are being impeded on upon completing the choice such as going into a negative balance on a savings account or exceeding the credit limit on a checking account. After the user’s choice is validated it will search for the account the user chose to delete, deposit, withdraw or transfer to or from. If the account is found the account/s are updated with the new balance. After the transaction has been completed it calls another method to log the details of the transaction.

viewBalance():

This method is used to display the accounts of the passed account type, either checking or savings.

searchBalance():

This method is used to display the account using the user inputted account ID.

calcTotal():

This method is used to calculate the total balance of the all the users accounts and return the total amount.

displayDifferentUser():

This method is used to display the accounts of another user. It excludes information such as the balance and is solely used when transferring money to another user to show what accounts the user can transfer to.

transferValidation():

This methods use is to validate a user choice when deciding to transfer money between accounts. It runs checks on account types, balances, users, and users accounts to see if the transaction can be completed properly. The method returns subsequent messages to the user depending on the check that would not allow the transfer to proceed.

createTransactionNumber():

Thise methods sole use is to create a number when logging transactions after they have occurred. It checks the highest transaction ID number and returns a value higher than that so it will not overlap.

createTransaction():

This method is used to log the transaction that the user has occurred to a file. It includes information such as the sender ID, receiving ID, amount, transaction type, transaction ID number and the date of the transaction.

viewTransactions():

This is used to display transactions that either the user has committed or transactions that have been sent towards the user from a different user.

findName():

This methods use is to find the username of any given ID. This is used when displaying another users accounts or when displaying transactions.

createAccountNumber():

This method is used to create an account ID that will be assigned to any account the user creates. It will count the highest account number and return a larger value to avoid overlapping account IDs.

SavingAccount Class:

The SavingAccount subclass is used to create and manipulate file contents relating to the users savings accounts.

\_\_init\_\_():

This magic method is used to initialise self-variables that will be used in the methods.

checkWithdraw():

This method is used to check the difference between the last withdrawal date and the current date as savings accounts are only allowed to withdraw once every 30 days. It then returns the difference in days.

checkBalance():

This checks to see if there is enough money within the account to withdraw.

CheckingAccount Class:

The CheckingAccount subclass is used to create and manipulate file contents relating to the users checking accounts.

\_\_init\_\_():

This magic method is used to initialise self-variables that will be used in the methods.

checkBal():

This checks to see if there is enough money within the account to withdraw without exceeding the users set credit limit.

**User Manual**

**Main Menu**

* Sign In
* The user will be prompted to enter their username and password to sign into the system. If successful, the user will be signed in and sent to the **Logged In Menu**
* Register New Customer
* The user will be prompted to enter a username, password and age which will create a new customer account.
* Exit
* Exits the program

**Logged In Menu**

* View Balance
* Sends user to **Balance Menu**
* Deposit
* The user will be prompted to enter an amount to deposit and then again to choose which account the amount should be deposited to
* Withdraw
* The user will be prompted to enter an amount to withdraw and then again to choose which account the amount should be withdrawn from
* Transfer
* Sends user to **Transfer Menu**
* Manage Account
* Sends user to **Account Menu**

**Balance Menu**

* View All Balances
* Displays all accounts to the user
* View Savings Balances
* Displays all saving accounts to the user
* View Checking Balances
* Displays all checking accounts to the user
* Search Balance By Account
* Prompt user for an account ID which will then display the account associated with the ID.
* View Total User Balance
* Displays users total balance across all accounts
* Return
* Returns to **Logged In Menu**

**Transfer Menu**

* Transfer Between Your Accounts
* Prompts user for amount they would like to transfer, the account to transfer from and the account to transfer to
* Transfer To Another User
* Prompts user for amount they would like to transfer, the account to transfer from, the user to transfer to and the account to transfer to
* Return
* Returns to **Logged In Menu**

**Account Menu**

* Create New Account
* Sends user to **Create Account Menu**
* Delete Account
* Prompts user to delete an account
* View Accounts
* Displays all users accounts
* View Transactions
* Sends user to **Transactions Menu**
* View User Details
* Displays users’ details
* Return
* Returns to **Logged In Menu**

**Create Account Menu**

* Savings Account
* Creates a savings account for the user
* Checking Account
* Creates a checking account for the user
* Return
* Returns to **Logged In Menu**

**Transaction Menu**

* View Your Transactions
* Displays all transactions the user has made
* Checking Account
* Displays all transactions made towards the user
* Return
* Returns to **Logged In Menu**

**Difficulties and Challenges**

Overall, there wasn’t much that I struggled with creating the system. One of the most challenging parts however was rewriting information back to the files. In some instances of writing back to files it would either no write the information I wanted it to do or delete the contents of the files if it encountered an error during processing. I was also experiencing some trouble with multiple new lines being created just after rewriting to the files but only while using f-strings. Another problem I was encountering was navigating through the menus. In some cases, the menus would be caught inside a loop while there not being and sort of loop, although this was resolved in the end with adding a few breaks at the end of the execution. Even though there was a fair bit of trial and error in trying to overcome these difficulties and challenges most of the program came together quite effortlessly.