BANK MANAGEMENT SYSTEM

CA PROJECT | Object Oriented Program 20/21

Abstract

This report reflects the development of the Bank Management System I have created for this CA project. It will contain explanation of the codes, user manual and any difficulties or challenges that I faced through the development process of the application.

C20305696

L Fernandez

Contents

About

Project Name

Bank System

Project Description

Develop an application to manage bank services including balance, deposit, transfer, and withdrawal methods which is located within the Account Class.

There are two possible account types to choose from:

A current account in which allows all services, in additional to a 500-credit limit. There is an age restriction, user must be 18 years old or above.

A savings account in which also has all services but there is a limit of one withdrawal and transfer per month and user can be 14 years old or above.

All data is saved on three different files:

Customers.txt will contain any information about the user. This includes the user's name (first, last), date of birth, address, number, accountID and customerID. The accountID is used to connect to their account as customerID is unique to all users for identification.

Accounts.txt will contain any data about the user's account which are AccountID, Account Type and balance. This will help identify the type of Account a user has and saves user's balance.

AccountsTransactions.txt will record all made transactions from all accounts – withdrawal, transfer, and deposit transactions.

A simple menu display is provided which allows the user to navigate through the bank services:

Create account where the users will be asked their information as mentioned in customers.txt and is saved within the text file. They will be provided an account ID and for any further verification on the system, they will just need to enter their account ID twice and phone number.

View of transaction history and balance will also be an option for the user. It will display all transactions made of the account and includes their balance.

Account Services will provide another menu option for the methods – Withdrawal, Deposit and Transfer – for the user to choose from. This will link back to the Account Class where these functions are located in.

A customer is also given an option to delete their account. This will remove all data that is related to the account except for the transaction history as, in my opinion, in a real bank system, they would have past data of transactions saved for a certain amount of time for legal purposes despite the user had deactivated their account.

Author(s)

L Fernandez – C20305696 (N/A Partner)

Project Classes and Methods

Classes

Customer

This class contains the attributes that creates the structure of all instances made under this class. This includes, first name, last name, full name, date of birth, address, number, customer ID and accountID – this information will be saved in the same order within the customers text file (customers.txt). It also contains the string method which allows to print the details of each customer.

Account

This class contains two attributes, the account ID and account type. It also contains the 3 methods of withdrawal, transfer and lastly, deposit functions. This class is the parent for the following subclasses:

Savings Account

This class inherits the attributes of Account and in additional, also has a balance attribute. This class differentiates from the Current Account, hence, allowing customed restrictions to only affect this account type.

Current Account

This class inherits the attributes of Account and in additional, also has a balance attribute. This class differentiates from the Savings Account, hence, allowing customed restrictions to only affect this account type.

Methods

All class methods are accessed through an external function called *accServices().* In this function, we prompt the user to log in through the *logIn()* function and verify their information as correct. Once that is done, we store the user information, and ask for their choice. All the methods below are called from an instance of the class representing the user.

Deposit()

Amount will be requested and will be added onto the current balance of the user. It will also date in which does these transactions as all transactions will be dated to restrict any savings account from their monthly usage of certain features.

Data will be saved as a string in the accountsTransactions.txt.

Withdrawal()

Amount will be requested and decreased from the current of the user. All transactions will be dated to restrict any savings account from their monthly usage of certain features.

If a Savings account tries to exceed their balance below 0 – transaction will fail. If a Current account tries to exceed their allowed credit limit of 500 – this will also fail. Both will return to Main menu.

Data will be saved as a string in the accountsTransactions.txt.

Transfer()

User will be asked for an account to which the amount is to be transferred to. It will iterate through the accounts text file to find the account – if it exists, amount is asked from the user. Balance will decrease and the account chosen will have an increase in balance. This transaction is saved in accountsTransactions.txt.

Monthly usage restriction is also applied in this function for Savings account.

User Manual

The Menu should present;

[1] Create Account

[2] View Balance and Transaction History

[3] Account Services

[4] Delete Account

[1] Create Account

You will be asked to enter your details and what account type you would prefer, Savings or Current Account. Please keep in mind that you must be 14 years old or above to create a Savings Account and 18 years or above to create a Current account.

Details will include your: first and last name, date of birth, home address, number most importantly, customer identification number. Afterwards, a summary of your information will be displayed and you will be asked to confirm information.

When account has been created successfully, your name and your account number will be displayed.

[2] View Balance and Transaction History

This option will require the user to verify their account by inputting their account number twice and their phone number. After the verification, the user will be able to see their following transaction history and balance.

[3] Account Services

User is asked to verify account similarly in [2] option and opens services available to user;

[1] – Deposit

[2] – Withdrawal

[3] - Transfer

[1] – Deposit

You can enter amount to be lodged into your account and your balance will update if successful.

[2] - Withdrawal

Enter the amount you would like to withdraw, and this transaction will be updated into your history as well as balance.

[3] - Transfer

Enter the account you request to transfer to, if the account exists, user will be asked for the amount and money is decreased from their balance and transferred to the chosen account.

[4] Delete Account

When this option is chosen, all related information to the account number will be deleted.

For security purposes, verification steps will be done in the same way mentioned in

[2] View Balance and Transaction History. In additional, a last prompt message to inform you that the deletion of your account will be erase all data and will be unretrievable.

Once service is done, system will refer to the main menu.

Difficulties and Challenges

I had a few difficulties throughout the code, but I was able to fix them through research and trial and error.

logIn() Function:

The iteration on my former For loop failed due to the if and else statements. As if the IF statement was false, it will divert to the ELSE statement which broke the loop as I included a return to the menu() function. This was removed and improved.

This had also broken the Savings Account transactions as it was not reading the account ID properly, this was due to the 2 For loops using [i]. Once the first loop ends, it keeps the value of i and becomes the starting point of the next For loop which should not be the case. This was amended to [i] for the first loop and [j] for the second loop.

menu_operations = [createAcc, viewBalTrans, accServices, deleteAcc] (line591)

Link can't be found in my history anymore but a youtube video is used to reference for this code. I had a difficulty in making a menu as Python is yet to have a switch case operator, but this video showed an alternative way of making a menu similar to a switch case.

Place the functions in an array and call them by their index locations. This method is also used to call the methods under Account Class for deposit, withdraw and transfer.

Code:

```
def __init__(self, accountID, type_, balance):
        super().__init__(accountID, type_)  # Inherits instances from parent: Account
        self.balance = balance
```

Ref: https://www.youtube.com/watch?v=RSl87lqOXDE

I did not understand or know how to reference the balance between Savings/Current and Account Classes, but this video had a clear explanation which helped.