OOP Assignment 2021-2022

Author : Christina Vargka C20737009  
Date Reviewed : 18/12/2021  
Program that simulates an ATM machine where Customers can view and manage their accounts.

**Summary:**

* Classes : Bank -> Customer -> Account == CheckingAccount/SavingAccount
* Main Functions : ATM(), Account\_Choice(), Menu()

**CLASSES:**

* **Bank(object):** Holds any necessary about the Bank. There are default values set.
  + **\_\_init\_\_:** Initializes its Name, Address, Country and Interest Rate.   
     All of them are set as private.
  + Each private attribute has a get method.
  + **get\_monthly\_interest\_rate**: returns interest rate
  + **\_\_str\_\_:** Prints out Bank’s details except for the interest rate. Required output seemed weird otherwise.
* **Customer(object):** Holds information about a Customer

Aggregates Bank class

* + **\_\_init\_\_:** Initializes the Bank object, and the Customer’s Name, Age and address as well as a ID. This ID will be used to access the Customer and their accounts  
    customer\_ID and name are private.
  + Private attribute has get methods
  + **\_\_str\_\_:** Prints out Customer’s details and information.
* **Account(object):** Holds basic account information. Contains most methods and operations.  
   Aggregates Customer Class.
  + **\_\_init\_\_:** Initializes the Customer object, Account\_ID and balance.

Balance is private.

* + Balance get method.
  + **deposit**: Increases balance by a specified amount. Updates Account.txt
  + **withdraw**: Decreases balance by a specified amount. Updates Account.txt
  + **transfer**: withdraws amount from account a, deposits amount to account b
  + **record\_transaction**: passes a float and a string. It saves transactions into the AccountTransactions.txt file in the form of a dictionary. Increments transaction identifier.
  + **update\_balance\_in\_txt**: works together with withdraw and deposit and “updates” balance of account. This is done by first deleting the instance then re-adding it to the list of dictionaries that saves the accounts.
  + **\_\_add\_\_**: allows the balances of two accounts to be added together. Works together with
  + **\_\_radd\_\_**: which allows at least on of them to be printed in case there is an Attribute error.
  + **\_\_str\_\_:** prints out customer’s name, the bank’s name and their account ID.
* **SavingAccount(Account): Child of Account.** One withdrawal per month. Age: 14+
  + **\_\_init\_\_:** Initializes the Account attributes inherited.
  + **get\_monthly\_interest:** Grabs monthly interest from the Bank class then multiplies it with the account balance.
  + **check\_amount**: Ensures that the amount to be withdrawn is not greater than the account balance.
  + **write\_and\_withdraw\_amount**: Checks if amount can be withdrawn then saves the transaction to file. Works with **check\_amount**(), **withdraw**() and **record\_transaction**(). If the returned value from **check\_amount**() is True, it withdraws the amount.
  + **okToWithdraw**: Filters list of dictionaries. Further filters out types of Transactions. Compares today’s date with date of last withdrawal and returns the Boolean value.
  + **\_\_str\_\_**: Prints out savings account information as well as inherited **\_\_str\_\_** from **Account**()and **\_\_str\_\_** from **Customer()**
* **CheckingAccount(Account):** Can have a negative balance up to specified amount.
  + **\_\_init\_\_:** Initializes parent and the credit limit.
  + Get function for credit limit.
  + **check\_amount:** Similar to check\_amount() in SavingAccount().   
    Ensures balance - amount to be withdrawn is not smaller than credit limit
  + **write\_and\_withdraw \_amount:** Similar to write\_and\_withdraw \_amount() in SavingAccount()  
    Has a different output.
  + **\_\_str\_\_():** Prints out checking account information as well as the inherited **\_\_str\_\_** from **Account**() and **\_\_str\_\_** from **Customer()**

**MAIN FUNCTIONS:**

* **ATM():** Main interface for Customers. Allows customers to log in or create their account.  
  **Related functions:**
  + **enter\_customer\_details: for ATM()** Inputs details of a customer then returns a dictionary with their information.
  + **login:** Creates a customer class if details found in the customers.txt file.
  + **sign\_up:** Ensures there are no duplicate PIN numbers.Writes new Customer to text file.
* **Account\_Choice():** Once Customer logs in it displays a menu and they may choose to view their accounts + total balance between them or create/use either a Savings account or a Checking account **Related functions:**
  + **create\_acc: for**Creates either a SavingAccount or a CheckingAccount if user doesn’t have one.
  + **search\_acc:** Loads accounts into the program if they exist.
* **Menu():** Main Function. Allows user to View their balance, withdraw, deposit, transfer funds, view bank’s and their own interest (if a savings account), view transaction history and delete an account.
  + **transfer\_acc:** Creates a temporary Account object so that funds may be transferred.
  + **amount\_verification:** Prompts user to enter an amount and will loop until value is numeric (float)
  + **case\_2\_withdrawal:** If it’s a Saving account it checks if there can be a withdrawal. Withdraws amount
  + **case\_3\_deposit:** Deposits amount
  + **case\_4\_Transfer:** Checks if saving account and if transfer is possible. If yes, transfers amount to another account.
  + **case\_5\_viewInterest:** Displays interest. Monthly interest if is a saving account
  + **case\_6\_transactionHistory:** Displays transaction history of account.
  + **case\_7\_deleteAccount:** If account has a balance > 0, or user has a different account, it will ask if they want to transfer their funds. Deletes account and displays bank information when appropriate.
  + **remove\_from\_accountTrans\_txt:** Removes all instances of account in the accountTransactions.txt file
  + **remove\_from\_accounts\_txt:** Removes account from accounts.txt file

**OTHER FUCNTIONS:**

* **Number\_Verification:** Verifies input is a number and positive
* **openfile:** Opens a file and saves contents into a list. Can create a new file if it doesn’t exist.
* **writefile:** Writes contents of a list into a text file all of which are in separate lines.
* **search\_dic\_list:** Searches through the list of dictionaries read from a txt file.
* **days\_between:** Returns absolute difference between two dates

**USER MANUAL**

Run program and it will call functions and create files as required. To transfer to another account please refer to the accounts.txt file and enter the required Acc\_ID value when prompted.

Each transaction and account have a unique ID

‘ID’ = Customer ID

‘AccID’ = Account ID

‘TransID’ = Transaction ID: Changes depending on the type of transaction and the account its from.

Pressing 0 in any of the menus will exit the program

Pressing 9 will Log user out

**CHALLENGES**

* The time constraint was one of the biggest challenges to overcome.
* I had disproportionately huge issue figuring out how to properly structure the objects. The result worked for what I envisioned.
* Having a comprehensive function structure and what I can or cannot implement as a method.
* I had to abandon plans to improve on the write\_and\_withdraw\_amount() and check\_amount() in a way that they would be an Account method due to time constraints.
* I faced a roadblock when it came to figuring out how to read and write files properly. GeeksforGeeks.org was my savior on that one.
* Creative part was very difficult to figure out and required a lot of research. I went with a very safe option that I would be able to implement in a timely manner.
* I had way too many bugs in the okToWithdraw function, where it had a bug where it wouldn’t allow further withdrawals even if the only transaction was a deposit. Patched it once I figured out a suitable approach and I implemented a second filter to filter out Transactions. Main reason why the TransID has the transaction type in it.
* Testing after structure was changed (I moved functions and made them into methods) was very time consuming.