Public Interface for Transaction class

Task	*Method*	*Return Value*
Constructs transaction	*__init__(tNumber, tType = “”, amount = 0.0)*	*Transaction object*
“==”	*__eq__(other)*	*boolean*
“!=”	*__ne__(other)*	*boolean*
“+”	*__add__(other)*	*sum*
“-”	*__sub__(other)*	*diffrence*
sum( )	*__radd__(other)*	*sum*
Get the amount from the transaction	*getAmount()*	*amount*
Get the date the transaction was made	*getDate( )*	*date*
Get the transaction’s number	*getTNumber( )*	*tNumber*
Get the transaction’s type	*getTType( )*	*tType*
Prints the transaction’s values	*printTransaction( )*	
Converts the transaction into a human readable format	*__str__( )*	*string*
Converts the transaction into a machine readable format	*__repr__( )*	*string*
Calls the system to get the current date and sets the date for the transaction	*_setDate( )*	*None*

Class responsibilities for Transaction class:

- Know
    - Transaction number
    - Transaction type
    - Transaction amount
    - Date of the transaction
        - Day
        - Month
        - Year
- Do
    - Compare if equal or not equal
    - Add or subtract
    - Find sum
    - Get transaction values
        - Amount
        - Date
        - Transaction type
        - Transaction number
    - Print transaction values
    - Create human readable string
    - Create machine readable string
- Notes
    - _typeSet: private list of possible transaction types
    - _setDate(): private helper function

Class Definitions

Class Name: *Transaction*

Data:

tNumber	int – Transaction number
tType	str – Transaction type (“deposit”, “withdrawal”, “interest”, “transfer”, “penalty”)
amount	double – amount of the transaction
day	int – day of the month (1-31)
month	int – month of the year (1-12)
year	int – year (>= 2022)

Queries (Accessor):

getAmount()	returns the transaction’s amount
getDate()	returns the transaction’s date
getTNumber()	returns the transactions’s number
getTType()	returns the transaction’s type
printTransaction()	prints the transaction’s day, month, year, amount, type and number as a single string

Commands (Mutator):

None

Public Interface for BankAccount class

Task	***Method***	***Return Value***
Creates a bank account	*__init__()*	*BankAccount*
Get the current balance	*getBalance():*	*float*
Get the account number	*getAccountNumber()*	*int*
Get the list of transactions	*getTransactions()*	*list<Transaction>*
Get the amount of times overdrawn	*getTimesOverdrawn()*	*int*
Print the account details	*printAccount()*	*None*
Print List of Transactions	*printTransactions()*	*None*
Deposit into an account	*deposit(amount)*	*None*
Add interest into account	*addInterest()*	*None*
Withdraw money from an account	*withdraw(amount)*	*pass*
Transfer money from another into account into this account	*transfer(other, amount)*	*bool*
Create human readable string representation of a bank account	*__str__*	*str*
Create machine readable string representation of a bank account	*__repr__*	*str*
Determine if two bank accounts are equal to each other	*__eq__*	*bool*
Determines if two bank accounts are less than to each other	*__lt__*	*bool*
Determines if a bank account is greater than another based on account number	*__gt__*	*bool*
Determines if a bank account is less than or equal to another bank account	*__le__*	*bool*
Determines if a bank account is greater than or equal to another based on account number	*__ge__*	*bool*

Class responsibilities for BankAccount class:

- Know
  - Class overdraft fee
  - Class interest rate
  - Class next account number
  - Account Number
  - List of transactions
  - Times overdrawn
  - Account type
- Do
  - Accessor Methods for all instance variables
  - Mutator Methods for all mutable instance variables
  - Print the details of an account
  - Print the list of transactions
  - Deposit money into accounts
  - Add interest to accounts
  - Withdraw money from accounts
  - Transfer money between accounts
  - Make string representations for humans and machines
  - Compare by equality, less than, greater than, less than or equal to, and greater than or equal to

Class Definitions

Class Name: *BankAccount*

Data:

Interest Rate: .0075 (Private and immutable)

Overdraft Fee: 20.00 (Private and immutable)

_nextAccountNumber: 1000 (Private)

accountNumber: int – unique number of the account

transactions : list of transactions – list of the transactions the account has

timesOverdrawn: int – amount of times the account has been overdrawn

AccountType: String – either “Checking” or “Savings”

Queries (Accessor):

getBalance(): returns the balance

getAccountNumber(): returns the account number

getTimesOverdrawn(): returns the number of times overdrawn

Commands (Mutator):

_incrementOverdraft(): adds 1 to the timesOverdrawn counter variable

deposit(): Deposits money into the account via creating a deposit transaction

addInterest(): Calculates and adds interest into the account via creating an interest transaction

withdraw(): Withdraws money from the account via creating a withdraw transaction (Abstract)

transfer(): Withdraws money from the other account and deposits it into self (Abstract)

Public Interface for Client class

Task	***Method***	***Return Value***
Creates a client object	*__init__(first, last, phone, address, initialAccountType)*	*Client object*
Open an account for the client	*openAccount(type)*	*None*
Print out the details of a client	*printClient()*	*None*
Withdraw all funds from a client's bank account and remove the account from the client's list of accounts	*closeAccount(number)*	*bool*
Get the first name	*getFirstName()*	*str*
Get the last name	*getLastName()*	*str*
Get the address	*getAddress()*	*Address object*
Get the bank accounts	*getAccounts()*	*List<BankAccount>*
Get the client number	*getClientNumber()*	*int*
Get the client phone number	*getPhoneNum()*	*str*

Class responsibilities for Client class:

- Know
    - Private instance variables:
    - First name
    - Last name
    - Phone number
    - Valid address: street, city, state abbreviation
    - List of bank accounts
    - Client number
- Do
    - Create a client object
    - Display the details of the client object to the user
    - Create a new account associated with the client
    - Remove an account associated with the client

Class Definitions

Class Name: *Client*

Data:

First name: max of 25 characters, no special characters

Last name: max of 40 characters, no special characters

10 digit phone number, all numeric digits, length of 10, cannot start with 0, 1, or 2

Valid address:

Street, cannot be empty, max of 30 characters, no special characters

City, cannot be empty, max of 30 characters, no special characters

State abbreviation: 2 characters, cannot be empty, must be one of VA, MD, NJ, PA, DE, NC, WV, DC

List of client's bank accounts

Client number, monotonically increasing integer starting at 100

Queries (Accessor):

getFirstName()

getLastName()

getAddress()

getAccounts()

getClientNumber()

printClient()

getPhoneNumber()

Commands (Mutator):
openAccount()

closeAccount()

__init__()

Public Interface for CheckingAccount class

Task	***Method***	***Return Value***
Creates a checking account	*__init__(number)*	*eckingAccount*
Withdraw money from an account	*withdraw(amount)*	*bool*
Print all checking transactions to standard output	*printTransactions()*	*None*
Encrypt and write all checking transactions to "checking.txt"	*writeTransactions()*	*None*
Read, decrypt, and print checking transactions from "checking.txt"	*readTransactions()*	*None*
Read, decrypt, and return checking transactions from checking.txt	*getTransactionData()*	*str*

Class responsibilities for CheckingAccount class:

- Know
  - Everything inherited from bank account class
  - Class interest rate
- Do
  - Withdraw money from accounts
  - Transfer money between accounts
  - Print all checking account transactions
  - Write all checking account transactions
  - Read all checking account transactions

Class Definitions

Class Name: *CheckingAccount*

Data:

Interest Rate: 0.015 (Private and Immutable)

Queries (Accessor):

readTransactions(): Read the account transactions from the file "checking.txt", and print to console (Decrypted before printing)

GetTransactionData(): *Read the account transactions from the file "checking.txt", and return the decrypted text*

Commands (Mutator):

withdraw(): Withdraws money from the account via creating a withdraw transaction (Implemented from abstract Bank Account class)

printTransactions(): Prints all transactions from account, ordered by date

writeTransactions(): Write all transactions from account, ordered by date, into the file "checking.txt" (Encrypted before writing)

Public Interface for SavingsAccount class

Task	***Method***	***Return Value***
Creates a savings account	*__init__(number)*	*vingsAccount*
Withdraw money from an account	*withdraw(amount)*	*bool*
Print all checking transactions to standard output	*printTransactions()*	*None*
Write all checking transactions to "savings.txt"	*writeTransactions()*	*None*
Read checking transactions from "savings.txt"	*readTransactions()*	*None*
Reads the transaction data from the file, decrypts it, and returns it	*getTransactionData()*	*str*
Deposits money into the account via creating a deposit transaction	*deposit()*	*None*

Class responsibilities for SavingsAccount class:

- Know
    - Everything inherited from bank account class
    - Class interest rate
- Do
    - Withdraw money from accounts
    - Transfer money between accounts
    - Print all savings account transactions
    - Write all savings account transactions
    - Read all savings account transactions

Class Definitions

Class Name: *SavingsAccount*

Data:

Interest Rate: 0.04 (Private and Immutable)

Queries (Accessor):

None

Commands (Mutator):

withdraw(): Withdraws money from the account via creating a withdraw transaction. Can only be overdrawn 3 times. (Implemented from abstract Bank Account class)

deposit(): deposits money into the account via creating a deposit transaction. Contains special logic to update the times overdrawn counter, so it overrides BankAccount's deposit method

printTransactions(): Prints all transactions from account, ordered by date

writeTransactions(): Write all transactions from account, ordered by date, into the file "savings.txt" (Encrypted before writing)

readTransactions(): Read the account transactions from the file "savings.txt", and print to console (Decrypted before printing)

Public Interface for Address class

Task	***Method***	***Return Value***
Creates an Address	*__init__(street, city, state)*	*Address*
Compares Addresses, returns True if equal, False otherwise	*__eq__(other)*	*bool*
Allows an Address to be printed	*__str__()*	*str*
Returns the street name of the Address	*getStreet()*	*str*
Returns the city name of the Address	*getCity()*	*str*
Returns the state abbreviation of the Address	*getState()*	*str*

Class responsibilities for Address class:

- Know
    - The street name
    - The city name
    - The state name
- Do
    - Accessor methods for all instance variables
    - Support equality testing
    - Support printing address information
    - Assert that all data in valid

Class Definitions

Class Name: *Address*

Data:

Street name – length must be 1 <= len <= 30, characters must be alphanumeric

City name – length must be 1 <= len <= 30, characters must be alphabetical

State abbreviation – 2 characters (only VA, MD, NJ, PA, DE, NC, WV, DC)

Queries (Accessor):

getStreet – returns the name of the street as a string

getCity – returns the name of the city as a string

getState – returns the abbreviation of the state as a string

Commands (Mutator):

None