member\_factory.py

Person Class

This class is an abstract class. This means that it is not actually used to create any instances. It is inherited by children classes that are used to create the instances.

Attributes:

* name

(str) The full name of the person. E.g. “Edward Joshua”. Default is None.

* password

(str) A password for the account. Default is None.

* email

(str) An email address for the account.

Methods:

* save\_to\_database(new\_detail, email)

Parameter new\_detail: (dict) A dictionary that contains the details of a newly

created user that will be saved to the JSON database.

Parameter email:(str) An email address representing the email of the newly

created user.

This method saves a new user to a JSON database. It does nothing if the user already

exists in the database.

* change\_password()

This method allows the user to change the password of his account. It returns True if

successful, else False.

Official Class

This class is used to create bank officials. It inherits from the [Person](#7egh32cr5qrz) class.

Attributes:

* name

(str) The full name of the person. E.g. “Edward Joshua”. Default is None.

* email

(str) An email address for the account. Default is None. It is set to “<name>@bank.com” during the creation of a new instance. Where <name> is the value in the name attribute of the class instance (without any whitespace).

* password

(str) A password for the account. Default is None.

* role

(str) The role of the user. This is set to “official” by default.

* salary

(int) An integer representing the salary of the bank official. Default is

100000.

Methods:

* get\_detail()

This method returns a dictionary containing the details of the instance. E.g.

{

“uchenna@gmail.com”: {

"id": “3”,

"name": “Uchenna Onyenso”,

"email": “uchenna@gmail.com”,

"password": ass,

"role": “official”,

}

}

Customer Class

This class is used to create normal customers. It inherits from the [Person](#7egh32cr5qrz) class.

Attributes:

* name

(str) The full name of the person. E.g. “Edward Joshua”.

* email

(str) An email address for the account.

* password

(str) A password for the account.

* account\_type

(str) The type of account. Can either be set to “current” or “savings”.

* role

(str) The role of the user. This is set to “customer” by default.

* balance

(int) An integer representing the balance of the account. Default is 0.

* account\_number

(int) An integer representing the account number of the user. This is made

up by concatenating the current microsecond, second, minute, hour,

weekday, month, year, and id of the user.

Methods:

* get\_detail()

This method returns a dictionary containing the details of the instance. E.g.

{

“uchenna@gmail.com”: {

"id": “5”,

"name": “Onyenso Uchenna”,

"email": “uchenna@gmail.com”,

"password": “ass”,

"role": “customer”,

"account type": “current”,

"account number": 928242461427920224,

"balance": 550000,

}

}

* check\_balance()

This method prints a string with the current account balance of the user. E.g.

Your current account balance is N1000.😊

* transfer(amount, account\_number)

Parameter amount:(int) The amount to transfer.

Parameter account\_number:(int) The account number to transfer to.

This method is used to “transfer” money from a user’s account to another. It checks that the user has enough funds to transfer and that the inputs are valid. It returns the name of the person (as a string) that was transferred to if the operation was successful, else it returns False if the operation fails.

* transact(operation, amount)

Parameter operation:(str) This string represents the operation to carry out. It can either be “deposit” or “withdraw”.

Parameter amount:(int) The amount to deposit or withdraw.

This method carries out the operation specified. It returns True on success, else False.

* get\_account\_statement()

This method prints the user’s transaction record in a tabular format. E.g.

user time date transaction amount account

0 glory@gmail.com 17:28 2022-09-07 deposit 45000 NaN

2 glory@gmail.com 17:31 2022-09-07 transfer 5000 Ezinne Ihendinihu

utilities.py

Functions:

* start()

This function is the entry point of the program. It asks the user to decide what transaction he/she would like to make using the following prompt:

What would you like to do?

1. Open an account.

2. Do giveaway a.k.a. Transfer money.

3. Deposit money.

4. Withdraw money.

5. Check balance.

6. Account statement.

7. Change password.

8. Admin transactions.

Reply with a number to begin transaction:

* login()

This function is used to authenticate the user. If the user exits, it returns either a

Customer object or an Official object, depending on the role of the user saved in

the database. If authentication is not successful, it returns None.

* record(user\_email, transaction, amount, name)

Parameter user\_email: (str) The email address of the user carrying out a transaction.

Parameter transaction: (str) The type of transaction the user is carrying out. Can be

one of “transfer”, “withdrawal” or “deposit”.

Parameter amount: (int) The amount of funds to transact.

Parameter name: (str) Default is None. But in a transfer transaction, the value is the name of the beneficiary.

This function saves a record of the transaction being carried out, to a csv file named

records.csv.

* open\_account()

This function takes the user through an account opening process, making sure that all

inputs are valid. It also checks that the email does not already exist in the database. On

success, it creates a new Customer object and prints the account number to the

console. E.g.

Success!! Account has been created. The account number is:

276053591127920222.🙂

* transfer()

Within this function, the login() function is called. Upon successful authentication, the

user’s [Customer.transfer(](#jl2669whzgh5))is called.

* deposit()

This function handles deposit transactions for a user. It first carries out authentication, and on success, it calls the user’s [Customer.transact()](#2ige4cxj6unz) method.

* withdraw()

This function handles withdrawal transactions for a user. It first carries out authentication, and on success, it calls the user’s [Customer.transact()](#2ige4cxj6unz) method.

* check\_balance()

This function prints out the user’s account balance. It first carries out authentication, and on success, it calls the user’s [Customer.check\_balance()](#c5cmr3vl2a7o) method.

* get\_account\_statement()

This function handles the retrieval of the user’s account statement. It first carries out authentication, and on success, it calls the user’s [Customer.get\_account\_statement()](#fmbm3tji37d1) method.

* change\_password()

This function handles the process of changing a user’s password. It first carries out authentication, and on success, it calls the user’s [Person.change\_password()](#lydungsrhh5z) method.

* admin\_start()

This is the entry point for transactions that can only be performed by bank officials. It first

calls the login() function for authentication. Upon successful authentication, it checks

to see that the user has access to admin transactions. If all checks are passed, it

prompts the user for which other transaction he/she would like to do. E.g.

Good day Alphadev.🙂

What would you like to do?

1. Open an account.

2. Edit customer details.

3. Get customer details.

4. Change password.

Reply with a number to begin transaction:

* admin\_open\_acount()

This function enables a bank official to create both official and customer accounts. It is

called from within the [admin\_start()](#rvj8qjxhfzbm) function.

* edit\_customer\_details()

This function enables a bank official to edit any customer detail except the password. It

is called from within the [admin\_start()](#rvj8qjxhfzbm) function.

* get\_customer\_details()

This function enables a bank official to view a customer’s account details and account statement. E.g.

-------------Account Details------------------

id 4

name Glory Ijeh

email glory@gmail.com

password qwerty

role customer

account number 928242461427920224

account type savings

balance 139999

Name: glory@gmail.com, dtype: object

-------------Account Statement------------------

user time date transaction amount account

0 glory@gmail.com 17:28 2022-09-07 deposit 45000 NaN

2 glory@gmail.com 17:31 2022-09-07 transfer 5000 Ezinne Ihendinihu

Would you like to perform another transaction? Type 'y' for yes or 'n' for no.

* again()

This prompts the user to state if he/she would like to perform another transaction.