An Python application for bank transactions
Technical Report

Ionescu Iulian Ştefan

January 10, 2017

Student Name: Ionescu Iulian Ştefan
Group: CE2.1
Year: II
Section: Computers and Information Technology - English

1 Problem Statement

The task of the project is to realise an application that allows an user from a certain bank to make different operations on his accounts.

This module will provide the following operations:

- Login operation
- Create an account
- Close an account
- Deposit money into an account
- Withdraw monet from an account
- Money transfer from an account to another
- Display raport of accounts

2 Specification of the input-output

2.1 Input

The input consists in values given by an user, generally names and passwords. He can choose to use a menu or to call a function that goues through all operations once and that prove the correctness .

2.2 Output

The output consists in the results of several bank operations (account reports, display of accounts' balance to prove that some operations worked).

3 The modules and their description

- main.py
- clase.py
- functions.py

i) **main.py**

In the python file **main.py** the user can choose two ways: use a menu to make any accounts' operations he wants or launch a function that test each operation once.

ii) **clase.py**

In the python file **clase.py** there are provided the classes used to create the user profile and the accounts.

iii) **functions.py**

In the python file **functions.py** there are provided the functions that help us to simulate the bank operations and the test functions.

3.1 The classes and their description

- **Profil**
  This class contains the name and password used to create a profile of the bank custome.

- **Cont**
  This class contains the balance of the account and ID of the account. Every account has an unique ID, just like the IBAN code. This class provides methods that display the balance and the ID. We also have getter and setter functions needed for another operations.

- **Client**
  This class contains as parameter an object of class Profile used for the customer profile and a list of accounts, where the elements of the list are objects of class Cont. The list is not a constructor parameter, because we want every customer to have his list of accounts, so the list is unique for every instantiated object. The class provides methods used to add an account to the list, to close an specified account (the account is closed by popping the element of list with the specified position), to search an account using the given ID (it returns his position from list or it returns -1 if there are no accounts with that ID), to display the details of an account (name of customer, balance and ID of every account he has). The class also has getter and setter functions.

3.2 The functions and their description

The header of functions:

- **Autentificare_Cont(** *banca***)**
- **Creare_Profil(***banca***)**
- **Deschidere_Cont(***banca***,***sold***,***exista***)**
- **Inchidere_Cont(***banca***,***exista***)**
- **Depunere_Bani(***banca***)**
- **Extragere_Bani(***banca***,***exista***)**
- **Tranfer_Bani(***banca***,***exista***)**
- **Afisare_Raport(***banca***,***exista***)**
- **functie_test_user()**

- **functie_test_automata()**
- **id_generator(***size*,*chars*)

The description of functions:

- **Autentificare_Cont(** *banca*)
  In this function is simulated the login operation. The user is asked to type the name and the password. If the values correspond with those from bank data base (the list 'banca' in our program) the function returns the position of list where these elements are stored, else it returns -1.
- **Creare_Profil(***banca*)
  In this function the user must give a name and a password used to create a profile. With the given values we create an object of class Profil that it will be used to create an object of class Client. Finally, this object is added to list 'banca'.
- **Deschidere_Cont(***banca*,*sold*,*exista*)
  Using the method of class Client, *adauga_cont*, we will add an account to the customer with the position *exista* in our list and display all the accounts in order to prove the existence of the new account.
- **Inchidere_Cont(***banca*,*exista*)
  In this function the user give the ID of the account he want to close. The account is found using the method of class Client, *cauta_cont*. The returned value is the position of the account that it will be closed or, if there is not an account with that ID, we will display an error message. When the account is found, this will be popped from accounts list using the method *inchidere_cont*.
- **Depunere_Bani(***banca*)
  In this function the user must give the ID code of the account he want to deposit money. When the account and the customer is found we can make the changes. The initial balance of the account is stored, and then, using a setter functions, we set the new balance by adding the given sum to the old balance. Finally, we will display the account details.
- **Extragere_Bani(***banca*,*exista*)
  The user must give the ID of one of his accounts to withdraw money. If he has no opened accounts, an error message will be displayed. Else, we will search the position of the account. When the account is found, the user must give the sum he want to withdraw. If the sum is bigger than the daily limit or bigger than account balance, an error message will occur, else, the balance will be changed by substracting from the initial sum.
- **Tranfer_Bani(***banca*,*exista*)
  The user must provide the ID of his account and the ID of the account in which he wants to put money. When the positions of the accounts are

found, the money can be transferred. We save the initial balances of the accounts in order to make the changes. Using the setter functions, the given sum will be substracted from one account and added to the other one. If the accounts are not found, error messages will occur.

- **Afisare_Raport(***banca*,*exista***)**
  In this function we will display the details of every account one customer has.

- **functie_test_user()**
  This function create a list of accounts with random balances and generated IDs, simulating a data base. Then we will create a user profile used to make the bank operations. We create a menu in order to choose what operaions we want to make. When an option is chosen, a certain function will be called.

- **functie_test_automata()**
  In this function is created the 'data base' and an user profile. Then there is called once every function in order to prove the correctness of the operations.

- **id_generator(***size*,*chars***)**
  This function generate an ID with 6 characters which are uppercase letters and digits.

4 Results

As we can see, the program is mostly designed to be used by an user. Using the function $functie\_test\_automata$ we can go through every function and test its correctness. Firstly, we will create a data base where there are introduced a number of customers, with theirs names and passwords. These accounts will be displayed. Then it will be created one profile that will help us to make the accounts' operations. After the user's autentification is valide, his accounts are opened and displayed. Next, it will be called the functions for withdraw, deposit and tranfer money. In these calls of functions, the user must provide the accounts' IDs to complete the operations.
Numarul de clienti introdusi in baza de date:
3
$---Creare\ Profil--$
Setati un nume :
Popescu
Setati o parola de acces:
1234
$---Creare\ Profil--$
Setati un nume :
George
Setati o parola de acces:
3344
---Creare Profil---
Setati un nume :
Mihai
Setati o parola de acces:
5566
==========
Popescu
$---------CONT----------$
ID: A9QIIH
Sold: 4669.2 lei
==========
George
$---------CONT----------$
ID: RQ5I0V
Sold: 2376.67 lei
==========
Mihai
$---------CONT----------$
ID: DDZRFL
Sold: 3082.15 lei
$---Creare\ Profil----$
Setati un nume :
Iulian

Setati o parola de acces:
8899
$---Autentificare--$
Nume:
Iulian
Parola:
8800
Autentificare esuata!
$---Autentificare--$
Nume:
Iulian
Parola:
8899
Autentificare reusita!

$--Deschidere\ conturi--$

$---Contul\ a\ fost\ creat--$
Detalii
==========
Iulian
$---------CONT----------$
ID: 46AAY3
Sold: 3000.0 lei
$---Contul\ a\ fost\ creat--$
Detalii
==========
Iulian
$---------CONT----------$
ID: 46AAY3
Sold: 3000.0 lei
$---------CONT----------$
ID: RKBHUS
Sold: 2500.0 lei

$--Extragere\ bani--$

Dati codul contului dumneavoastra din care se vor extrage banii
46AAY3
Dati suma:
1500

$--Depunere\ bani--$

Dati codul contului in care doriti sa depuneti:
DDZRFL

Dati suma:
400
Extrasul contului in care s-au depus banii
ID: DDZRFL
Sold: 3482.15 lei

$--Transfer\ bani--$

Dati codul contului dumneavoastra din care vor fi transferati banii:
RKBHUS
Dati codul contului in care vor fi transferati banii:
RQ5I0V
Dati suma pe care doriti s-o transferati:
3000
Fonduri Insuficiente!
Dati suma pe care doriti s-o transferati:
1000
Extrasul contului din care s-au tranferat banii:
ID: RKBHUS
Sold: 1500.0 lei
Extrasul contului in care s-au tranferat banii:
ID: RQ5I0V
Sold: 3376.67 lei

$--Afisarea\ detaliilor\ conturilor\ unui\ client--$

==========
Iulian
$----------CONT----------$
ID: 46AAY3
Sold: 1500.0 lei
$----------CONT----------$
ID: RKBHUS
Sold: 1500.0 lei

$--Inchidere\ cont--$

Dati codul contului de inchis:
46AAY3
$---Contul\ cu\ ID-ul\ 46AAY3\ a\ fost\ inchis!---$
==========
Iulian
$----------CONT----------$
ID: RKBHUS
Sold: 1500.0 lei

References

[1] Python tutorials http://www.tutorialspoint.com/python/

[2] Python 3.5.2 Documentation https://docs.python.org/3/

[3] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994.

[4] LaTeXproject site, http://latex-project.org/