İSTANBUL ŞEHİR UNIVERSITY

ENGR 101

Introduction to Programming

Spring 2019

Mini Project III

SEHIR BANK

In Mini Project 3, you are going to develop the Third version of the text-based simple banking application “Sehir Bank”. The primary goal of this project is to make you practice Classes and inheritance in addition to what have you implemented in the previou mini projects. The detailed explanations about the application that you are going to develop are provided below.

Project Logic:

The application is based on the idea that customers should be able to interact with their bank accounts in the easiest way possible. The main features you will develop in this mini project are as follows:

- withdraw
- deposit money
- transfer money to other accounts
- display the users account information history or report the actions of the user in the bank.

Data Structures that MUST be used:

What is meant by a data structure?

Data structure is any allowable combination of lists, tuples, and dictionaries containing integers, strings or floats.

E.g. [[‘ abc ’ , ’ 0000 ’] , [1, 2, 3]] ;

{‘ 2 ’ : { (2 , 2) : [1 , 1 , 1] , (2 , 3) : [‘ s ’ , ’ d ’] } , { (1 , 1) : [2 , 2 , 2] } } ;

{ [1, 1 ,1] : ” a b c ” } .

Classes Must Be used:

At least three classes should be used in this project:

Requirements :

- One class should be used for storing the data, which means you will need to create a data structure in that class and fill it it when calling the the class, the data in this class will be

used to do the login and the other different services.

- Services should not be implemented on one class, for example deposit and withdraw services can be in class 2, and transfer and account details can be in class 3, while the data is stored in class 1.

Each user needs to have an instance in the data structure that will have a:

- balance which should contain – current balance amount
- user deposits which should contain - amount, timestamp
- user withdrawals which should contain – amount, timestamp
- user transfers which should contain – amount, to who (username), timestamp

Time Stamp should contain date - time (Istanbul) as shown:

```
21/03/2019 19:57:53
```

Important: AFTER EACH CORRESPONDING ACTION, THE BALANCE(S) SHOULD BE UPDATED ACCORDINGLY AND THE INFORMATION OF EACH ACTION WILL BE APPENDED TO A STRUCTURE THAT WILL KEEP A RECORD OF WHAT WAS DONE ! ! !

Important: ALL THE CORRESPONDING DEPOSITS, WITHDRAWALS AND TRANSFERS SHOULD BE ORDERED BY TIME AS SHOWN IN ***Figure 10*** ! ! !

Note: For Data Structure, the exact number and type of data structure are not specified. What matters is that the functionality and data storage capacity should be met within **one** encompassing **structure**.

Text-Based Bank App version 3:

Your bank **MUST** have 3 existing customers by default (when you run your program they should already have been added into your data structure):

- First user's username will be Ahmed and his password should be **1234**
- Second user's username should be Erva and her password should be **4321**
- Third user's username should be Ayse and his password should be **4422**

These three user names and their corresponding passwords should be placed in a proper data structure.

When your program first starts, welcome message and time in Istanbul will be displayed as shown below in ***Figure 1***. Then, users will be asked the service they want. They will have two options: either log in or exit the application.

Valid inputs are 1 or 2. ***Figure 1*** below is a sample of how your welcome page should look like:

```
|-----------------------------------|
     Welcome To Sehir Bank v0.3
|-----------------------------------|
     Istanbul 2019-05-06 17:16:56
|-----------------------------------|
< Choose what do you want to do. >
< 1.Login >
< 2.Exit >
>>>
```

Figure 1. Welcome message

If the user chooses login, your application should ask the user to enter his username and password. If the user enters a wrong input, your application should warn the user and ask for a valid username/password again, until the user provides the correct one. Next, the user will be asked 2 options as shown in ***Figure 2.*** If ***Back*** is entered, the login/exit page should show up.

Your program should keep asking until the user gives the correct credentials. Once your application gets the correct credentials, your program should again ***welcome*** the user but this time with his/her ***name clearly displayed*** (***Figure 2***). Your application should also display the list of the available services (1,2,3,4,5 in ***Figure 2***).

Your program should keep asking until the user gives the correct credentials. Once your application gets the correct credentials, your program should again ***welcome*** the user but this time with his/her ***name clearly displayed*** (***Figure 2***). Your application should also display the list of the

available services (1,2,3,4,5 in ***Figure 2***).

```
< Choose what do you want to do. >
< 1.Login >
< 2.Exit >
>>>1
Enter your username.Ayse
Enter your password.1122

 Login Success!
 Transferring...

Hello,  Ayse
<====================== Bank Services Available ==============================>
Choose Service
1. Withdraw Money
2. Deposit Money
3. Transfer Money
4. My Account Information
5. Logout
<============================================================================>
>>>
```

Figure 2. Login screen and available services after a proper login.

The user should have 4 services to choose from and a logout option (***Figure 2***). Your program should print the service numbered and the users should be able to enter the number corresponding to the service they want to use.

Important: Wrong user inputs and ***Back*** should be handled properly, if the user inputed **Back** at anytime after choosing a service, the user should be directed to the services menu, also at any time the user gives the wrong inputs should be asked again to correct it.

If **option 1** (***Figure 2***) is picked, your program should ask the user to provide an amount of money to ***withdraw*** from their account as shown in ***Figure 3*** below. Your program should check whether or not the user's account has ***enough money*** to do that transaction. If not, your program should warn the user that he/she doesn't have enough money and go back to

the main menu. If the user has enough money to do the transaction required, then your program should inform the user that the amount he/she requested has been withdrawn from his/her account.

```
How much do you wish to withdraw ? 100
100 tl Has been withdrawed from your account
```

Figure 3. The user been asked the amount to withdraw.

The change in amount of money after withdrawal should be reflected on the balance and recorded in data structure. A time stamp and the amount should also be appended to the respected ***transactions*** structure.

In this project you will keep track of each service a user has done and store it in a data structure so that it can be shown as a part of the user account information service.

If **option 2** (***Figure 2***) is picked, your program should ask the user to provide an amount of money to ***deposit*** to their account (***Figure 4***). After the deposit is successful, your program should inform the user that the amount he/she entered has been deposited into his/her account. This change should again be reflected on the balance and be recorded in data structure. A time stamp and the amount should also be appended to the respected transaction structure.

Your program should go back to the main menu after the deposit is complete.

```
>>>2
How much do you wish to deposite ? 200
200 tl has been deposited to your account
You Have deposited 200
```

Figure 4. The user been asked the amount to deposit.

If **option 3** (***Figure 2***) is picked, your program should ask the user to enter a name of the user to whom the money will be ***transferred*** to (***Figure 5***).

```
>>>3
For who do you wish to transfer to ?Ayse
How much would you like to transfer ?500
Transferering  500  tl   to  Ayse  Succeeded
```

Figure 5

If the user enters his/her own name, the program should print that the user does not exist . Or, if a name of a non-existing user is entered, the program should print that the user does not exist and give another chance to the user to enter another username.

If the user exists, then you should give the chance to enter the money for transfer. If the user has enough money in their account for the amount of transfer, the transfer should be allowed (***Figure 5***). Otherwise, your program should not allow the user to transfer money and print that the user does not have enough money and give the choice to transfer again or go back to services menu .

If the transfer is possible and successful you should reflect this to the balance of BOTH users and update the necessary information (amount, to/from who (username), timestamp) in BOTH users' transfer structures as part of data structure 3.

If **option 4** (***Figure 2***) is picked, ***My Account Information***, your application should display the name of the bank, the current date and time. The USER NAME, PASSWORD, BALANCE AND ALL THE TRANSACTIONS, DEPOSITS, WITHDRAWALS BEING ORDERED BY TIME should be displayed as shown in ***Figure 10***.

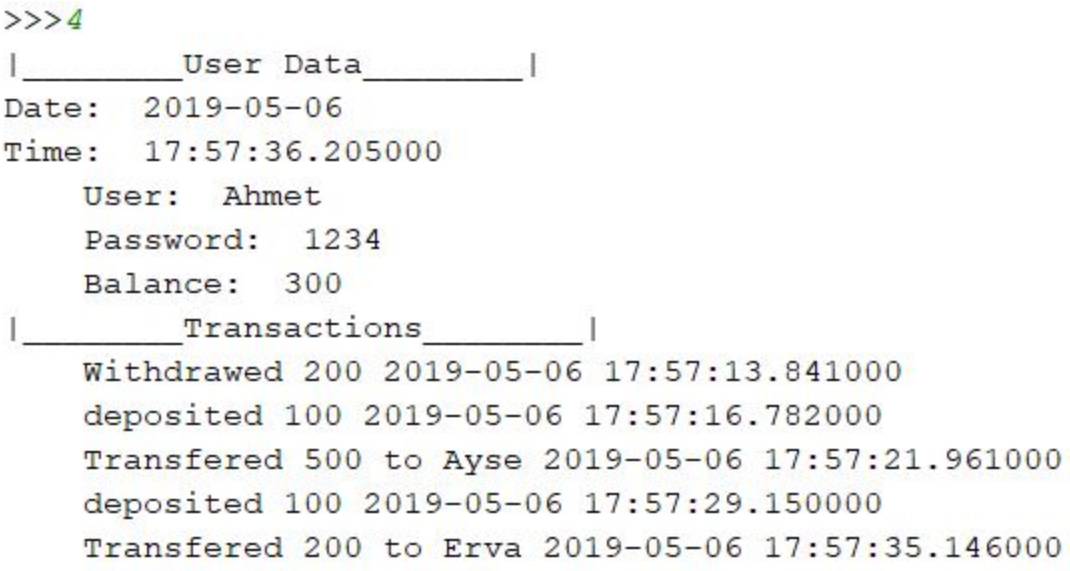

Figure 6. My Account Information for different users: Ahmet at different date-times.

When you log out from the system and then log in with ***another user account*** and go to ***My Account Information***, the right information for the right user should be displayed.

Each user should be able to transfer to all the existing users in the bank!!!

If **option 5** (***Figure 2***) is picked, your program should ***logout*** but keep the users' account balance and all the information in the data structure such that if the user logs in again, his/her account balance is correct and reflects the most recent update. For example, if the user with 0 balance logs in, deposits 60 TL and then logs out; when he/she logs in again and checks his/her account information, the balance should display 60 TL (providing that he did not withdraw or deposited any other money.)

Implementation Notes:

- You may check this link to get further information about the time stamps with the time module. (https://docs.python.org/3/library/time.html)
- You ***MUST*** use functions. You are strongly advised to use ***a function for each service***. However, you have to use ***at least 3 functions***. (10% reduction if you do not create 3 functions)
- YOU ***MUST*** USE 3 DATA STRUCTURES (including the one for bonus part), NO MORE AND NO LESS.
- Your code should be efficient, easy to follow, and track.

Warnings:

- If you don't follow the implementation notes, you are very likely to lose points.
- **Do not** talk to your classmates on project topics when you are implementing your projects. **Do not** show or email your code to others. If you need help, talk to your TA's or the instructor, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have **serious consequences** for both parties.
- Carefully read the project document, and pay special attention to sentences that involve **"should", "must", "should not", "do not",** and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resource at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so **may result in** plagiarism investigation.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is **important** to understand all the details in your submitted code.

How and when do I submit my project? :

- Projects may be done individually or as a small group of two students (doing it individually

is recommended for best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for details).

- Submit your own code in a **single** Python file. Name your code file with your and your partner's first and last names (see below for naming):
    - ❖ If your team members are Deniz Barış and Ahmet Çalışkan, then name your code file as deniz_baris_ahmet_caliskan.py (**Do not** use any Turkish characters in file name). If you are doing the project alone, then name it with your name and last name similar to the above naming scheme. Those who **do not** follow the above naming conventions will get **5 points off** of their grade.
- Submit it online on LMS by **Monday, May 20^{h}, 2019, 23:59**.

Late Submission Policy:

- 10%: Submissions which are 1 hour late
- 20%: Submissions which are 2-23 hour late
- 30%: Submissions which are 24 hours late.
- 50%: Submissions which are 48 hours late.
- Submission more than 48 hours late will not be accepted

Grading Criteria:

Meaningful variable names and following the text interface as given in this document **(2 pts)** and sufficient commenting **(4 pts)** **(6 pts)**	Proper use of functions, compact code with no unnecessary repetitions **(4 pts)**	Proper use of data structures as explained in this document **(15 pts)**	Login Function **(5 pts)**	Depositing (**10 pts**) and Withdrawing (**10 pts**) and Transferring (**15 pts**) **(25 pts)**	Display My Information **(10 pts)**	Logout **(5 pts)**	Using 3 classes **(30 pts)**

Have further questions?

Please contact your TAs if you have further questions. If you need help with anything, please use the office hours of your TAs and the instructor to get help. Do not walk in randomly (especially on the last day) into your TAs' or the instructor's offices. Make an appointment first. This is important. Your TAs have other responsibilities. Please respect their personal schedules!

Good Luck!!