ENGR 211 - Introduction to Programming

Mini Project 2

December 19, 2016

(Due: by 5 pm on Jan. 2, 2017)

In this mini project, you are going to develop the same software that was assigned for Mini Project 1, but this time, you are going to use the object-oriented programming paradigm that we covered in the class. That is, the primary goal of this mini project is to make you practice development with classes and objects. Hence, please try to take very opportunity to use classes and objects in your code. At various points, you will be forced to use a particular class as explained below.

Text-based User Interface:

The interface will be the same as the one in Mini Project 1, except that "Quit" menu items (in every place that it appears) will be replaced by "Sign Out" menu item. When a user chooses this new menu item, s/he will sign out of the program, and the initial login interface will be provided to the user. The user here may sign in with a different account. Any operation that was done before the user signs out should be kept in your corresponding data structures.

Implementation Notes:

- You need to create a class to represent a **User.** It should have type of user (may be one of "regular", "admin"), user name, and password as its attributes.
- You need to create a class to represent an **Account**. It should have user, balance, and transactions as its attributes. User attribute should be an instance of the User class. Transactions attribute should be a list which keeps the past user transactions where each transaction should be an instance of the Transaction class. Whenever an operation is performed by the user in the below-listed class methods, a new transaction should be added properly. It should provide the following methods:
    - **deposit**
    - **withdraw**
    - **transfer**
    - **get_latestN_transactions**
- You need to create a class to represent a **Transaction.** It should have type, amount, additional_info as attributes. additional_info attribute should be dictionary where the key should be type/name of some additional info (e.g., 'recipient' in transfer transactions) and the value would be the value of that information field (e.g., the recipient's name in transfer transactions). It is up to you to add methods into this class.
- You need to create a class which will represent a **Bank**. It will have a dictionary of accounts as an attribute where dictionary will store all accounts as objects (i.e., key would be account holder's user name and value would be an Account instance). It should provide the following methods:
    - **create_account**
    - **close_account**
    - **compute_and_print_statistics**
    - **load_accounts_from_file**
    - **search_for_an_account**

- You need to create a class which will represent a **MenuEntry**. Each option on any menu that you provide to the user will be an instance of MenuEntry class. It will have option_number and text as its attributes.
- You need to create another class which will represent a **Menu**. It will have a list of MenuEntry objects. In your interface, you have four different menus (regular main menu, admin menu, the small menu that you show after each operation for regular users, and the small menu that you show after each operation for admin user). Each of these menus should be represented as separate Menu objects. Menu class should provide the following methods:
    - **add_menu_entry** – this method should take a text for the new menu entry as input, and create a new instance of MenuEntry object with the provided text. Option_number of the new menu item object should be assigned automatically based on the current size of the Menu. For instance, if the menu currently contains 1 entry, the new MenuEntry will have option_number 2.
    - **print_menu_and_get_user_selection** – this method should print the menu as in previous project, ask the user to make a selection, handle any invalid selections that the user may do, and return the selection option number to the caller.

Warnings:

- **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 TAs or myself, 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**", "**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. You may be interviewed about any part of your 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 pts off** of their grade.

- Submit it online on LMS (Go to the Assignments Tab) by **5 pm on Monday, Jan. 2, 2017.**

  Late Submission Policy:

  - -10%: Submissions between 17:01 – 18:00 on the due date
  - -20%: Submissions between 18:01 – midnight (00:00) on the due date
  - -30%: Submissions which are 24 hour late.
  - -50%: Submissions which are 48 hours late.
  - Submission more than 48 hours late will not be accepted.

Grading Criteria? :

Code Organization			Functionality
Meaningful names (4 pts)	**Proper use of classes and objects (22 pts)**	**Sufficient commenting (4 pts)**	**There are 14 menu items in total. Proper implementation of each of these menu items will be 5 points.**

- Interview evaluation
  - Your grade from interview will be between 0 and 1, and it will be used as a coefficient to compute your final grade. For instance, if your initial grade was 80 before the interview, and your interview grade is 0.5, then your final grade will be 80*0.5 = 40. Not showing up for the interview appointment will **result in** grade 0.

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!