ENGR 101 - Introduction to Programming

Mini Project 4

(Posted on May 16, 2020 - Due on May 30 , 2020 by 5 pm)

In Mini Project 4, you have developed a console-based Library Management System by employing procedural programming (i.e., a set of functions and function calls). In this mini project, you are going to re-implement the same console-based Library Management System by using object-oriented programming practices! The interface from the user's perspective will stay the same. However, you will have a significantly different implementation in the source code level. The primary goal of this mini project is to practice using classes and objects. Hence, please use every opportunity to employ object-oriented programming concepts in your code. **At minimum**, you should create and use the following classes implemented with the below provided list of attributes and methods. If you see a need, you may add additional classes and/or methods/attributes that are not in the following list.

List of Classes with their Attributes and Methods:

- **MenuItem**
    - Attributes:
        - **text**: Stores the text of the menu item, e.g., "Add a book to my book list."
        - **number**: Stores the number of this menu item, e.g., 2 for the above item in student menu, which will be used when printing the menu and getting user selection
    - Methods:
        - **display()**: This method returns the current MenuItem object's number and text properly when called.
- **Menu**

- Attributes:
  - **header**: header represents the introductory text that usually appears before the menu options, e.g., "Welcome to Admin Menu! "
  - **menuItems**: list of MenuItem objects - Each option in a menu will be represented as a MenuItem object (see the previous class) and stored in a list.

- Methods:
  - **display(display_header)**: prints the menu on the screen, It will display the header attribute before menu items if display_header parameter is True. Otherwise, it will not show the header.
  - **add_menu_item(text, number)**: It should build and add a new MenuItem object to the list of "menuItems" attribute

- **User**

  - Attributes:
    - **menu**: is an object of Menu class with an empty string "" as a parameter for the header. This attribute should be a class level attribute.
    - **menu_items**: is an empty list will be used to store the proper menu items. This attribute should be a class level attribute.
    - **name:** will be used to stores the name of the user
    - **Password:** will be used to stores the password of the user

  - Methods:
    - **display()**: This method returns the current user's object info (name and password).
    - **menu_builder()**: This method will be called from the init() method of Student class and Admin class by using inheritance, it should build the admin menu and student menu with all of its options by using the class level attribute menu which is an instance of Menu class.

- **Admin**

  This class should be inherited from User class

  - Attributes:
    - **menu**: override the menu attribute to update the header parameter to "Welcome to admin menu". This attribute should be a class level attribute.
    - **menu_items**: override the menu_items attribute to update the content of the list. (check the "Required Data Structures" part at the end of the manual to find the list).
    - **Role:** should be always equal to "admin".

- **Student**

  This class should be inherited

  - Attributes:
    - **menu_flag**: will be used to check whether the menu has been created before or not. By default it should be False as a boolean value. This attribute should be a class level attribute.
    - **menu**: override the menu attribute to update the header parameter to "Welcome to student menu". This attribute should be a class level attribute.
    - **menu_items**: override the menu_items attribute to update the content of the list. (check the "Required Data Structures" part at the end of the manual to find the list)
    - **role**: should be always equal to "student".

- **UserDB**

- Attributes:
  - **example_users**: store the information of the users (check the "Required Data Structures" part at the end of the manual to find the dict). This attribute should be a class level attribute.
  - **users_objects**: dict of User objects where key would be user name, value would be the corresponding User object. This attribute should be a class level attribute.
  - **example_users_flag:** this attribute will determine whether to build the example users or not. By default it should be True as a boolean value.

- Methods:
  - **create_example_users()**: This method will be called from the init() method of this class according to the flag. It should build the example users by using the add_user method.
  - **add_user(name, password, role):** The objects of Student class and Admin class should be created in this method and added to users_objects dict.
  - **remove_user()**: It should handle deleting a user.
  - **list_user():** lists all the student users.
  - **validate_user(uid, password):** validate the user credentials by checking the username and the password. If true, It should return the object of the user. Else, it should return False.

- **Book**

  - Attributes:
    - **bid:** stores the book id
    - **name:** stores the name of the book
    - **no_of_copies:** stores the number of the copies of the book
    - **list_of_authors:** stores the a list of authors names of the book

- Methods:
  - **display():** This method returns the current book's object info.

- **Library**

  - Attributes:
    - **example_books**: store the information of the example books (check the "Required Data Structures" part at the end of the manual to find the dict). This attribute should be a class level attribute.
    - **author_to_books**: class level attribute dict of author names where key would be the author name, value would be a list including the corresponding book object.
    - **book_objects**: class level attribute dict of Book objects where key would be the id of the book, value would be the corresponding Book object.
    - **example_books_flag:** this attribute will determine whether to build the example books or not. By default it should be True as a boolean value.

  - Methods:
    - **create_example_books()**: This method will be called from the init() method of this class according to the flag. It should build the example users by using the add_book method.
    - **add_book(bid, name, copies, authors)**: The objects of Book class should be created in this method and added to book_objects dict and author_to_books dict.
    - **remove_book()**: It should handle deleting a book, it should delete the given book and remove it from authors_to_books dict.
    - **list_book()**: list all the books in the library
    - **search_book()**: This method searches the library for a specific book by the book name or the author name.

        - **update_book_copies()**: This method updates the number of copies of a specific book after checking the number of the students holding the book

- **Main**

    - Attributes:
        - **library**: a Library object with flag True
        - **userDB**: a UserDB object with flag True
        - **current_user**: will hold the current_user object. By default is None

    - Methods:
        - **login()**: This is the central method that runs the primary logic of the system. It will call methods of other objects whenever required and handle user credentials.
        - **show_admin_menu()**: This method will control the main flow of the admin menu when the user signs in as admin.
        - **show_student_menu()**: This method will control the main flow of the student menu when the user signs in as a student.

Required Data Structures:

- **menu_items for Admin class**
    - menu_items = ["List books", "Create a book", "Delete a book","Search for a book", "Change number of a copies of a book", "Show students borrowed a book by ""ID", "List Users", "Create User","Delete User", "Exit"]
- **menu_items for Student class**
    - menu_items = ["Search for a book", "Add a book to my book list", "delete a book from my book list","show my borrowed books", "Exit"]
- **example_users for UserDB class**
    - example_users = {'Ahmet': ['1234', 'student'], 'Ayse': ['4321', 'student'], "admin": ["0000", 'admin']}
- **example_books for Library class**
    - example_books = {"001": ["Biology", 2, ["Alice", "Bob"]], "002": ["Chemistry", 3, ["Alice"]]}

Implementation Notes:

- You must use the given template.
- You shoıuld fill the attributes and the methods of the g□ ven classes.
- "TODO" statements in the template will help you for implementation.
- To have a clear understanding please use the attached executable file to test the project and check how it works
- **Outside of class definitions, only 2 lines of code should exist** that do the following:
    - In one line an object of the Main class will be created.
    - In the other line, a proper method of the Main object will be called to start the program.
    - **Everything else should be under a class.**
- Please note that although __init__ methods are not specified, for every class in the above list, you should have an __init__ method in each class.
- In each method of a class, self should be the first input parameter. Self parameter is not

included in the above method specifications for brevity.

- You should actively use all the methods listed under each class at least once or more. Make sure that you eliminate repeated or similar codes by defining new methods for them.

How to run the executable file of the project;

- **For Windows;**
  - Download the corresponding file "mp4_win.exe".
  - Right click on the file and run it as administrator to start using the project.
- **For Mac;**
  - Download the corresponding file "mp4_mac".
  - Open a terminal in the same directory with the file.
  - Write "chmod +x mp4_mac" and press enter.
  - Double click on the file itself "mp4_mac" and start using the project.
- **For Linux distros;**
  - Download the corresponding file "mp4_linux".
  - Open a terminal in the same directory with the file.
  - Write "chmod +x mp4_linux" on the terminal and press enter.
  - Write "./mp4_linux" and press enter to start using the project.

Warnings:

- **Do not** talk to your classmates on project topics when you are implementing your projects (This is serious). **Do not** show or email your code to others (This is even more serious). If you need help, talk to your TAs 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**", "**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

resources 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 Can Barı and Ahmet Çalı kan, then name your code file as deniz_can_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 by **May 30, 2020 by 5 pm.**

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 up-to 24 hour late.
- -50%: Submissions which are up-to 48 hours late.
- Submissions more than 48 hours late will not be accepted.

Grading Criteria? :

Code Organization			**Functionality**			
Not using meaningful variable names (-10% cut off from your overall grade)	Not using or improper use of classes and objects (Deduct up to **80**%)	Insufficient commentin g (-10% cut off from your overall grade)	Student functionali ty with all of its submenus (35 pts)	Admin functionalit y with all of its submenus (40 pts)	Proper use of data structures to implement the student and admin menu items (35 pts)	Others (10 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!