ENGR 101 - Introduction to Programming
Mini Project 3

(Posted on December 19, 2019 - Due on January 2 , 2020 by 5 pm)

In Mini Project 2, you have developed a console-based Course 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 Course 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:

- **User**
  - Attributes:
    - id
    - password
    - budget
    - registered_courses (i.e., a list of Course objects that the student is taking)
  - Methods:
    - display: This method prints the current user's object info (Id, budget..).Also, it should handle when to show the budget and when not to.
    - can_take_course: This boolean method will determine whether the current user's budget is enough to register the selected course.
    - course_exists: this boolean method checks whether a student has already registered to a given course or not.
- **Course**
  - Attributes:
    - name
    - credits
    - registered_users (i.e., a list of User objects registered to this course)
  - Methods:
    - display: It should show current object's details.
    - delete_course: It should handle deleting courses by admin or by student, it should delete the given course.
    - course_price: It should return a given course's price.

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

            Displays the menu on the screen, prompts the user for his/her selection, gets the valid user selection, and returns it to the caller. 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 items attribute.
- **MenuItem**
    - Attributes:
        - text
            - Stores the text of the menu item, e.g., "Display questions for the next competition."
        - number
            - Stores the number of this menu item, e.g., 2 for the above item in admin menu, which will be used when printing the menu and getting user selection.
    - Methods:
        - display()
            - This method displays the current MenuItem object's number and text properly when called.

- **CourseManagementSystem**

    - Attributes:
        - courses: list of offered courses
        - users: dict of User objects where key would be user id, value would be the corresponding User object.
        - admin_menu (a Menu object)
        - student_menu(a Menu object)
        - admin_money_sub_menu (a Menu object)
        - current_user
    - 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.

        - build_menus( )

            This will be called from init(), it should build the admin menu, student menu and admin sub menu as a Menu class object with all of its options as listed in MP2 manual.

        - show_admin_menu( )

            This method will control the main flow of the admin menu when user signs in as admin.

        - show_student_menu( )

            This method will control the main flow of the student menu when user signs in as a student.

Implementation Notes:

- **Outside of class definitions, only 2 lines of code should exist** that do the following:
    - In one line you will create an object of CourseManagementSystem class.

- In the other line, you will call a proper method of the CourseManagementSystem object that you created above 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.

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 Barış and Ahmet Çalışkan, 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 **January 2, 2019 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.
- Submission 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 specified classes and objects (Deduct up-to **80**% of the total grade)	Insufficient commenting (-10% cut off from your overall grade)	Student functionality with all of its submenus (20 pts)	Admin functionality with all of its submenus (30 pts)	Proper use of data structures to implement the student and admin menu items (40 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!