CENG 104
Computer Programming 2
HOMEWORK 3

In this homework, you will create an application for assigning advisors to undergraduate students. In this application, you will create classes for the **Student** and **Advisor** objects. In order to test your application and assign advisors to students, you will create another class, named **Assignment**.

Student class should keep information about the following:

- **firstName** that contains the first name of the student.
- **lastName** that contains the last name of the student.
- **studentId** that contains the student number of the student.
- **year** that keeps which year the student is studying. **year** can take integer values between 1 and 4.
- **GPA** of the student that can take any double value between 0 and 4, 0 and 4 included.
- **advisor** of the student keeps which advisor is assigned to the student. It will be updated when a student is assigned to an advisor by the `assignStudentToAdvisor()`method of the Assignment class.
- **assignedFlag** that contains **0**(not assigned) or **1**(assigned) values.
- You can keep extra data members (attributes) in your class if you need. But they should be necessary for the application, so please do not add unnecessary attributes. You will determine datatypes, return types of functions and etc.

Advisor class should keep information about the following:

- **firstName** that contains the first name of the advisor.
- **lastName** that contains the last name of the advisor.
- **advisorId** that keeps the ID of the advisor.
- **numberOfStudentsAssigned** that keeps track of how many students are assigned to the advisor,
- **allowedStudentCount** for each advisor is max 3.
- You can keep extra data members (attributes) in your class if you need. But they should be necessary for the application, so please do not add unnecessary attributes. You will determine datatypes, return types of functions and etc.

Assignment class is like a driver class with some utility functions defined in it, details of which are described in the following paragraphs.

Application Requirements

1. It should be possible from the application to create an advisor with a method named "`createAdvisor()`" in the advisor class, the function initializes data members with entered values (hint: you may need initialization of some attributes with default values)

```
**************
CREATE ADVISOR
**************
```
Please enter the first name of the advisor: XXX
Please enter the last name of the advisor: XXX
Please enter the ID of the advisor: XXX

A new advisor is created.

2. It should be possible from the application to create a student with a method named "`createStudent()`" in the student class with similar output of "`createAdvisor()`", the function initializes data members with entered values (hint: you may need initialization of some attributes with default values)

3. Attributes of the classes not be seen and modified from other classes directly. But these classes can access and **set** some of these, **return** via proper method calls. (make research on set-get functions). You must decide what you need.

4. It should not be possible to set year attribute a value greater than 4 or less than 1. If a wrong value like 5 is tried to be set in main function, your application should not set that value and print the following message:

```
************************************
ERROR: Wrong year value for student: 5
************************************
```
"ERROR: Wrong year value for student:" part of the output is constant, and then you should print the wrong value that is tried to be set to the year field.

5. It should not be possible to set GPA attribute a negative value or a value greater than 4. If a negative value like -1 is tried to be set, your application should not set that value and print the following message:

```
************************************
ERROR: Wrong GPA value for student: -1.0
************************************
```
"ERROR: Wrong GPA value for student:" part of the output is constant, and then you should print the wrong value that is tried to be set to GPA.

6. There should be a method named "`showStudentDetails()`" in the student class that prints student information. The output of this method should be as follows:

```
************************************
STUDENT NAME: XXX
STUDENT LASTNAME: XXX
STUDENT ID: XXX
YEAR: 1
GPA: 0.0
ADVISOR: XXX (hint: you have to check assignedFlag here)
************************************
```

7. There should be a method named "`showAdvisorDetails()`" in the advisor class that prints advisor information. The output of this method is similar to the output of "`showStudentDetails()`".

8. There should be a method named "`isSuccessful()`" which will return true if the student's GPA is greater than or equal to 2.5. Otherwise it should return false.

9. There should be a method named "`currentStatus()`" which will return the students status depending on the GPA. This method should use "`isSuccessful()`" method in order to find out whether the student is successful or not. It should print the name and lastname of the student together with its success status in the following format;

```
*********************************
STUDENT: XXX XXX NOT SUCCESSFUL
*********************************
```

If the student's GPA is less than 2.5 than it prints NOT SUCCESSFUL after student name. Otherwise, it prints SUCCESSFUL after student's name.

10. **numberOfStudentsAssigned** can only be modified by `assignStudentToAdvisor()` method during the assignment of an advisor to a student.

11. Assignment class performs assigning students to advisors. This is the main class of your application. There should be a method named `assignStudentToAdvisor()` in the class that assigns a student to an advisor. This method should set the advisor field of the student object. One important requirement of your application is, **advisors can be assigned to at most 3 students.** For instance, if 3 students are assigned to an advisor and one more student is tried to be assigned, your application should not permit it, and print an error message as follows;

```
ERROR: Cannot assign more students to advisor ADVISOR ID: XXX
```

Here in this output, like errors for year and GPA, "`ERROR: Cannot assign more students to advisor`" part is fixed. Then, you should print the advisor information.

12. "`recordStudent()`" in the assignment class calls `createStudent()` function of Student object and stores student information in a **file**. User can add records serially based on an entered sentinel value to exit.

13. "`recordAdvisor()`" in the assignment class calls `createAdvisor()` function of Advisor object and stores advisor information in a **file**. User can add records serially based on an entered sentinel value to exit.

14. "`showRecordAdvisor(parameter)`" in the assignment class calls `showAdvisorDetails()`function of given advisortID as parameter, that is entered in main function and passed to here. You must check existence of the object advisortID in records.

15. (bonus) "`deleteRecordStudent()`" in the assignment class, deletes student information from **file** for an entered **studentId**. You must check existence of the entered ID (by user) in records.

16. (bonus) "`deleteRecordAdvisor()`" in the assignment class, deletes entered advisor information from **file** for an entered **advisorId**. You must check existence of the entered ID (by user) in records.

17. (bonus) Assignment class should have a method `whichSemester()` for calculating which semester we are in. If the current month is one of {9,10,11,12 and 1} it should return the string "FALL". If the current month is one of {2,3,4,5,6} it should return the string "SPRING". Otherwise, it should return "SUMMER".
    Hint: You can use `Calendar and Date` classes. Many examples exist on the internet.

18. In assignment class, write a main function and test your application for all possible function calls and object instantiation mechanisms.

19. For the test, be sure to write code for the following in your main function:

- Create and record 4 students, 3 advisors
- Print student and advisor details
- Assign student to advisors
- Test whether more than 3 students can be assigned to an advisor
- Test whether a wrong GPA value can be assigned to a student
- Test whether a wrong year value can be assigned to a student
- Show a specific advisor information
- Print student status

You should create objects and test all the functionalities required from your application.

IMPORTANT:

- This homework should be done **individually**.

- There is no one correct solution, but all requirements should be met. The code should be properly written such that unnecessary information and accesses to fields violating encapsulation should be prevented.
- Do it on your own, you can read tutorials and forums, many examples exist on the internet.