Problem Description

You will implement a Student class that represents a student and his or her scores from a class in which he or she has enrolled. The Student class has at least the following methods:

- __init__(self, studentName)
- addHWScore(self, score)
- addExamScore(self, score)

Note that you will need to determine what private attributes and public properties are needed for this class.

There is also a Course class that stores a list of the students in a particular class. The Course class has at least these methods or properties:

- __init__(self, courseTitle, courseNo)
- addStudent(self, student)

Note that you will need to determine what private attributes and public properties are needed for this class.

Other requirements:

1. You must add a new method to your Student class to calculate and return a student's final average. The class also allows new algorithms to be added to calculate the average without modification of the Student class. Therefore, you should use a specific Design pattern, develop a design for the Student class to meet the requirements.
   a. Use the following two algorithms for computing the average in your implementation:
      i. The homework average contributes 40%, and the Exam average contributes 60% to the final class average.
      ii. Use the same percentages as the first algorithm, but first, if there are more than 5 homework scores, drop the lowest homework score.
2. Create a main method to create a Student object, add at least six homework scores and two exam scores, and compute their average first using method(i) then using method (ii), then using method (i) again.
3. Enhance the Course class to have a new method called getStudentAverages that will iterate the Students list and return a dictionary of the student's name and the final average pairs.
4. Enhance the main method by creating a Course object which will store some Student objects. Next, test the getStudentAverages() method of the Course object.
5. Add a new class called RecordOffice that should have a list to store the current letter grade of Student objects (>=90=> A, >= 80=> B, >= 70=> C, >=60=>D, <=59=>F) for each Course object. Therefore, if there are two courses, then the list should have two courses of student letter grades. Whenever the

test the getStudentAverages() method of the Course object.
5. Add a new class called RecordOffice that should have a list to store the current letter grade of Student objects (>=90=> A, >= 80=> B, >= 70=> C, >=60=>D, <=59=>F) for each Course object. Therefore, if there are two courses, then the list should have two courses of student letter grades. Whenever the

final average of a Student object is changed, the records office has to be informed and the letter grade will be updated automatically. Again, you have to use a proper design pattern, develop a design for the RecordOffice, and then implement your design.
   a. Enhance the main method to create a RecordOffice object to track the student and output the letter grade of the student. You may need to create another course object and enroll students in it. You should then test by adding a homework score or an exam score to one Student object, it will then trigger the change in the letter grade of the student and output the letter grade again to show it was updated automatically.

Deliverables

Please submit the following files individually on the OSC website:

1. A PDF contains the class diagram showing how your classes implement the Design Pattern and label it with the name of the design pattern you used.
2. A ZIP file holds all your Python source files, including the data files used in your software if any.