**Assignment no. 3**

**Note: Use UML notation for designing class diagrams. For better understanding of the problem, the codes for the classes have been provided in a separate zip file. Extra credit will be given for implementing the designs in Java.**

**Question. 1:**

We want to implement a Java class to represent a student and their scores from a class they are enrolled in.

The **Student** class provides at least the following methods:

*Student(String name); //constructor*

*void addAssignmentScore (double as); //add 1 assignment score*

*void addExamScore (double es); //add 1 exam score*

There is also a **Roster** class that stores a list of the Students in a particular class. The Roster class provides at least these methods:

*Roster(String name, String number); //constructor*

*String getCourseName(); //getter for name*

*addStudent (Student student); //add 1 student*

There are two **algorithms for computing the average**:

A. The Assignment average contributes 40%, and the Exam average contributes 60% to the final class average.

B. Use the same percentages as the first algorithm, but first if there are at least 2 Assignments scores, drop the lowest Assignment score.

**Your task:** Using **a specific Design Pattern**, develop a design that satisfies the following requirements.

1. We must be able to select the algorithm to compute the average at runtime.
2. It also must be possible to add new algorithms to compute the average to the program with only minor modifications to the Student class.

**Question 2:**

We want to extend our design with a class GradeTracker that tracks (stores) the current letter grade of a given Student object (>=90=A, >=80=B, etc.). Whenever the Student object is changed, the tracker has to be modified automatically.

**Your task:** Using a specific Design Pattern, develop a design for the tracker.

**Question 3:**

We want to provide access to the averages of the students in the class, without providing access to either the ArrayList of Students or the Student objects. We do not want to provide a collection of the averages to the client code (we don’t want to require the client to store all the values locally).

We also want to be able to allow multiple clients to access the averages simultaneously without interference (so adding a “current element” pointer to the Roster will not suffice).

**Your task:** Using a specific Design Pattern, develop a design that satisfies the above requirements for accessing the averages. Your design must include the provided AvgDispenser interface, which is called from the driver.