

Assignment #5

Design Patterns

CS 4354 Summer II 2016

Instructor: Jill Seaman

Due: before class **Thursday, 8/4/2016** (upload electronic copy by 9:30am, bring paper copy of the UML diagrams to class).

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
double getAverage();             //the combined average
```

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
```

See the website for initial versions of the Student and Roster classes, as well as an incomplete version of the Driver used below. Do not change these files except as specified in the notes below.

1. The algorithm to compute the average must be able to be selected at runtime. It also must be possible to add new algorithms to compute the average to the program without modifying the Student class.

Your task: Using a specific Design Pattern, develop a design for Student that satisfies the above requirements, then implement your design.

Use the following two algorithms for computing the average in your implementation:

- 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.

Testing The driver creates a Student, adds three assignment scores and two exam scores, and computes their average first using method A, then using method B.

2. We want to extend our design with a class `GradeTracker` that tracks (stores) the current letter grade of a given `Student` object ($\geq 90=A$, $\geq 80=B$, 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, then implement your design.

Testing The driver defines a tracker object to track the student and outputs the letter grade of the student. It then adds an exam score that will change the letter grade of the student, and outputs the tracker's letter grade again to show it was updated automatically.

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, then implement your design. (Coding hint: return an anonymous class, See Java-Inheritance slide 22).

Testing The driver creates two students and adds them to a `Roster` object. It has a while loop framework for accessing the averages and outputting them to the screen. You must add code to initialize the traversal mechanism, to test if all the elements have been accessed, and to access the next element.

NOTES:

- This assignment is to be done with your partner (in groups of 2).
- For each problem, draw the **class diagram** showing how your classes implement the Design Pattern, and **label it with the name of the design pattern** you used.
- Use the package "assign5" for your classes and put your files in the appropriate directory structure.
- **Allowed Changes to Student** You may add a field, a setter for it, and initialize it in the constructor. You may make it extend a subclass or implement an interface. You may add code to the methods.

- **Allowed Changes to Roster** You may add an accessor method.
- **Allowed Changes to Driver** These are stated in comments in Driver.java
- Follow the style guidelines from the class website. **Use javadoc comments for all of your public elements.**

Submit:

- A. Please combine your *.java files into a single zip file (assign5_XXXXXX_YYYYYY.zip). The XXXXX and YYYYY are your TX State NetIDs (mine is js236, you have two, one for each partner). Submit an **electronic copy**, using the Assignments tool on the TRACS website for this class.
- B. Submit the **UML diagrams** showing the design of each of the three problems (on paper, bring to class). Put the name of the design pattern you chose for each problem!!