Assignment #5

Design Patterns

CS 4354 Summer II 2015
Instructor: Jill Seaman

Due: before class **Thursday, 7/30/2015** (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);   //0 or more assignments
void addExamScore (double es);         //0 or more exams
double getAverage();                   //the combined average
public int getNumberOfAssignments();       //size
public Double getAssignmentScore(int i);   //score at index i
public int getNumberOfExams();             //size
public Double getExamScore(int i);         //score at index i
```

See the website for an initial version of the Student class, 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 use an existing user interface component to display the Student scores to the screen. This component requires a class that implements the interface java.util.Iterator<Double>. Furthermore, we want to reuse the Student class without changing it.

   Here is the interface that must be implemented for use with the UI component:
   java.util.Iterator<E> {

```
  boolean hasNext(); //Returns true if the iteration has more elems
  E next();          //Returns the next element in the iteration
  void remove();     // (throw UnsupportedOperationException)
}
```

   Your task: Using a specific Design Pattern, develop a design that allows us to reuse Student (without changing it) and to implement Iterator, then implement your design.

   Testing The driver defines a method that takes a java.util.Iterator<Double> as an argument and uses it to display each of the values being iterated over. This method is called from the main method so that it displays all the scores in the Student (simulating the UI component).

3. 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, 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.

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 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 xxxxxx and yyyyyy 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!!