Project (Part 2)

Your customer decided to assign the development of the user interface for the *GradesCalculator* to Rufus, a professional UI developer. This means that you won't have to worry about developing that part of Story Card #1. It also means that you will now have to make sure that your code integrates nicely with Rufus's. In addition, the customer provided you with two additional story cards, and identified, with the help of Rufus, an initial set of corresponding task cards and related test cases. You will have to write code that makes these test cases pass, which means that it should integrate nicely with Rufus's code. You will also have to write some additional task cards and test cases (but not the application code that makes these latter tests pass). See the instructions below for details.

Story Card #2

"The instructor/TA launches the *GradesCalculator* tool to get information about the students and the class. In this scenario, the application allows the user to output, for a given student, its average assignments grade, average projects grade, and overall grade based on a user-provided formula."

Story Card #3

"The instructor/TA launches the *GradesCalculator* tool in editing mode to enter information about the class. In particular, the application allows its users to add students, assignment, and projects, and to add grades and individual contribution grades."

Task Card #2.1

Extend the `GradesDB` class so that it provides a way to compute the average assignments grade.

Task Card #2.2

Extend the `GradesDB` class so that it provides a way to compute the average projects grade, taking into account the individual contribution grades.

Task Card #3.1

Extend the `GradesDB` class so that it provides a way to add an assignments.

Task Card #3.2

Extend the `GradesDB` class so that it provides a way to add grades for an assignment.

Task Card #3.3

Extend the `GradesDB` class so that it provides a way to add individual contribution grades.

Instructions:

- Together with this description you will find a zip file containing

    - An updated `GradesDBTest` class that contains the test cases created by Rufus. You will have to write code that makes them pass, as described below.
    - A golden version of the database that is used by the new tests (`GradesDatabase-goldenversion.xlsx`).

  Just to clarify, you will not be provided with Rufus's code. so you should not worry about it. All you have to do is make sure that the code you write makes the provided test cases pass, as they encode the expectations of Rufus's user interface code.
- Keeping in mind the story cards and their corresponding task cards, write code that makes the newly provided test cases pass. Doing so may require you to refactor your existing code, which is expected, as the tests assume that the system behaves in a given way. *Note*: When writing and (possibly) refactoring this code, you don't have to worry about the parts of the story cards for which there are no corresponding task cards.
- Write task cards to realize the parts of Story Card #3 that refer to the ability of (1) adding students (2) adding projects, and (3) adding grades for a project.
- Add to class `GradesDBTest` test cases for the task cards that you just produced (but do not write the code that makes them pass).

Notes:

- **Important**: you cannot modify the provided test cases; that is, you can add to class `GradesDBTest` new test methods, but you cannot modify the existing ones (i.e., you cannot modify anything above the "NO CHANGES ABOVE THIS LINE" comment in the class). Nor you can modify file `GradesDatabase-goldenversion.xlsx`, the golden version of the database. Note that, however, the test cases will modify the "regular" database file, `GradesDatabase.xlsx`, as the new functionality of the `GradesCalculator` involves database updates.
- When converting real numbers to integers, you should suitably round them up or down based on their decimal part (<.5 round down, >= .5 round up).