**Individual Project: *Grades Calculator***

**Background:**

For this new project, you must develop a simple tool called *GradesCalculator*. Given a database with various data about the students in a class, *GradesCalculator* allows the instructors to get various information about the students and compute their final grade. You will develop the *GradesCalculator* application by following a test-driven development approach such as the one we saw in P4L4, in which development happens in iterations and according to the following cycle: (1) select story cards to implement, (2) define task cards for the selected story cards, (3) write test cases for the classes in the task cards, (4) write code that makes the test cases pass. Specifically, this project will consist of three or four deliverables. (Being in an agile world, we will make some decisions on the fly based on how the project is progressing.)

**Goals:**

* Develop a Java application for assessing students information and compute students’ final grades.
* Get experience with an agile, test-driven process.

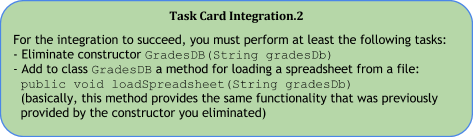
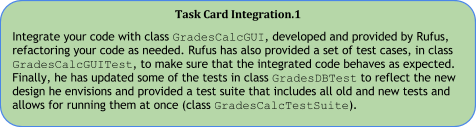
**Deliverables:**

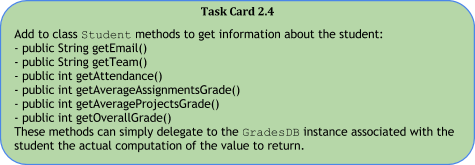
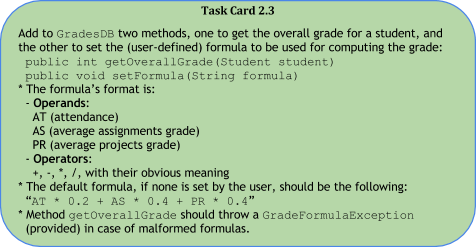
**DELIVERABLE 3**

* **provided:**
  + Task cards
  + JUnit test cases
  + Code to be integrated
* **expected:**
  + Code that makes the test cases pass

**Deliverable 3: Details**

Several things happened during the last week. First, your customer decided that she would like to see a working prototype of the *GradesCalculator*, even if it has only limited functionality. (So far, Rufus has only implemented the part of the GUI for visualizing, rather than updating, students’ information.) You will therefore put on hold the development of the code that makes the test cases that you wrote for Deliverable 2 pass **and focus instead on integrating your code with the current version of Rufus’s GUI code, which may require some further refactoring of your code.** In addition, Rufus would like to improve the earlier design of the GradesDB class to eliminate some issues that were reported to him by other developers. In particular, he would like to slightly modify the GradesDB’s API so that a new (or updated) database can be loaded without any need to create a new instance of GradesDB (which may introduce inconsistencies in some implementations). More precisely, here are the detailed task cards for the third and final deliverable of this project:





**Instructions:**

* Download file [GradesCalculatorUpdateD3.tar.gz](https://dl.dropboxusercontent.com/u/12485366/GradesCalculatorUpdateD3.tar.gz), which should contain all you need for your third deliverable:  
  GradesCalculator/src/edu/gatech/seclass/gradescalc/GradesCalcGUI.java  
  (GradesCalcGUI class, which you will have to integrate with your code)  
  GradesCalculator/test/edu/gatech/seclass/gradescalc/GradesCalcGUITest.java  
  (test cases for the GradesCalcGUI class)  
  GradesCalculator/test/edu/gatech/seclass/gradescalc/GradesDBTest.java  
  (updated GradesDBTest class)  
  GradesCalculator/test/edu/gatech/seclass/gradescalc/GradesCalcTestSuite.java  
  (test suite that runs all tests in both GradesCalcGUITest and GradesDBTest)  
  GradesCalculator/src/edu/gatech/seclass/gradescalc/GradeFormulaException.java  
  (exception to be thrown in case of malformed grade formulas, provided  
   for your convenience)
* **Keeping in mind the task cards and discussion provided above, as well as the code provided by Rufus, write code that makes all test cases in test classes GradesCalcGUITest and GradesDBTest pass.**
* As discussed above, doing so **may require you to refactor** your existing code, which is expected.
* Commit and push your changes.
* Paste the commit ID for your submission on T-Square.

**Notes:**

* **Important**: Also for this deliverable, you cannot modify the provided test cases. Nor you can modify file GradesDatabase6300-goldenversion.xlsx, the golden version of the database.
* As for previous deliverables, 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).
* **One possible way to implement the getOverallGrade method in GradesDB is to use Java’s built-in script engine (javax.script.ScriptEngine). You should be able to find documentation and plenty of examples for it online. Feel totally free to implement the method directly or using a different library, if you prefer.**
* As usual, **if the code you write makes the provided test cases pass, you are done.**
* After you are done integrating your code, and all test cases pass, you should also be able to run the Grades Calculator by running the GradesCalcGUI class as a normal Java application. Here is a screenshot of what you should see if you do so:  
  