Project 1: Average Sentence Length (Deliverable 2)

In this first project, you and your team will develop a program that counts the average number of words per sentence in a file. To get the set of requirements for the projects, see the videos in P1L5, paying particular attention to the interview between Lauren and Alvin and the subsequent video, which summarizes some of the details discussed in the interview. **When watching the videos, make sure to also read the instructor notes below both videos**.

You should construct your program using a waterfall-like process in which we focus mainly on the requirements gathering phase. It is your team's job to specify the exact process by defining appropriate process activities. In particular, you must include activities responsible for producing the following deliverables (templates for the documents are linked off the corresponding items--make sure to use these templates and not the ones available on the Udacity site):

- Project plan
- Requirements document
- User documentation (textual man page)
- Code
- Test cases

For this deliverable, you must:

(as usual, we refer to directory Project1 in the root directory of your local team repo as `<dir>`)

1. If needed, revise your requirements document based on Lauren's answers. If this leads to changes to the project plan, feel free to update that document as well.
   Important note on the requirements: As discussed during the "Questions to Lauren" office hours, Lauren changed her mind on the fact that the average sentence length should be rounded down to the nearest integer. Instead, she would like to see, as a result, a real number rounded to two decimal places (e.g., 9.6845 would be rounded to 9.68, and 9.6854 would be rounded to 9.69). This is now also mentioned in the Instructor Notes for the second video.
2. Write a concise user manual in MD format, call the file `manual.md`, save it in `<dir>`, and commit it. For the format, you can use Unix man pages as an example, but no specific format is really required. You can assume that the users will receive the program already compiled and ready to be run (i.e., do not add to the manual compilation and installation instructions).
3. Write the code that satisfies the requirements you identified. Make sure to have, in your code, (1) a class that contains the `main` method of the program and that will be invoked to run it and (2) a class `edu.gatech.seclass.project1.Core` that implements the core functionality of the program. It is fine to have additional classes. All classes should

be saved under `<dir>/src/edu/gatech/seclass/project1`. When you are done, commit your code.

4. Develop a set of JUnit 4.0 test cases for class `Core`. Specifically, **you should create four or more test cases for each relevant method in the class** (i.e., each method that performs a non-trivial computation). Similar to what you did for Assignment 3, make sure that every test method has a suitable oracle (e.g., an assertion or an expected except ion), and that the tests are not trivial. In other words, each test should (1) exercise a specific piece of functionality, (2) check that such piece of functionality behaves as expected, and (3) not be a trivial variation of another test. And also in this case, **make sure to add a concise comment to each test that you implement to clarify its rationale**. Make also sure that your test methods have meaningful names. The JUnit test cases should be implemented in a class called `CoreTest` that should also be part of package `edu.gatech.seclass.project1` and should be saved under directory `<dir>/test/edu/gatech/seclass/project1`. When you are done, commit your code.
5. Make sure that all files are committed and push your local changes to the remote team GitHub repository. (Coordinate with your team members so that everybody's changes are committed before pushing and all local repos are updated.) The project manager will then get the commit ID for the version your team pushed by running, as usual, "git log -1" and submit it to T-Square. **Only the project manager should submit the commit ID, which means that the other team members will not submit anything on T-Square.**

Additional Notes:

- All team members must contribute to the project and should be committing changes.
- Use only GitHub to share code and documents and collaborate on them. If you find this to be too limiting, check with us.
- Try to also create test cases that exercise interesting scenarios (e.g., corner cases).