**CS 1699 - Software Testing**

Spring Semester 2015

DUE 31 MAR 2015

**Deliverable 4**

For this assignment, you (NOT a group) will profile a Conway's Game of Life simulation, and improve its performance by refactoring a single method (to be determined by the results of the profiling). This will consist of three parts:

1. Profiling (before and after) to determine which method is the most resource-intensive, and showing that your rewrite helped make your method more performant
2. Writing unit tests for that method
3. Refactoring the method to be more performant

Test code will be on Github( <https://github.com/laboon/JavaLife> ), and should be issued **as a PULL REQUEST (PR)**against the original repo.

There are no partners for this deliverable.

**Format**

Every assignment should have a title page with:

* The name of the project under test (JavaLife)
* The name of the student
* The title "CS 1699 - DELIVERABLE 4: Performance Testing Conway's Game of Life"

There is no need to print out the code. It should be issued as a PR against the original repo BY THE BEGINNING OF CLASS.

In order to determine the "hot spots" of the application, you will need to run a profiler such as VisualVM (included with the JDK). Using a profiler, determine a method you can use to measurably increase the speed of the application without modifying behavior. You may wish to use "time" or a similar command to ensure that you have in fact reduced the amount of time necessary to execute n iterations of the World.

The application accepts four command line arguments. All of these should be positive integers. They are:

1. The size of the world (n x n)
2. The seed for the random number generator
3. The percentage of cells alive initially
4. The number of iterations to run the simulation

Note that iterations can go by pretty quickly, so for your debugging, you may want to set the number of iterations very high (e.g. 30000) and just kill the application when you are done profiling.

At a minimum, there should be three (3) unit tests (note that this is a MINIMUM). For a good grade, there should be various edge cases and equivalence classes considered. Remember that since you are refactoring the method, the tests should pass both before and after. Whatever method you choose to refactor, it should produce the same RESULTS as before.

For the summary, describe how you profiled the application and determined the method to refactor, and explain how you did so. Additionally, explain how you refactored the method and how you chose what to unit test. Finally, explain any challenges that arose in the process.

After this, include screenshots of VisualVM (or another profiler, if you use that) both before and after the refactor. These screenshots should include the relevant sections that let you see what method to refactor.

Finally, issue the changes that you made as a PR against my local repo. Remember that you will need to do all of your work in a separate branch for this to work correctly! You shouldn't be doing work on the master branch, anyways.

**Grading**

* Summary - 25%
* VisualVM (or other profile) screenshots (before and after) - 25%
* Method refactoring - 25%
* JUnit Tests - 25%

Reminder that you NEED TO ISSUE THIS AS A PR. SIMPLY LINKING TO A SEPARATE GITHUB REPO IS NOT ACCEPTABLE.

Please feel free to email me at [bill@billlaboon.com](mailto:bill@billlaboon.com) or come to office hours to discuss any problems you have.