JavaLife

Casey Nispel

CS 1632 - DELIVERABLE 5: Performance Testing Conway's Game of Life

Summary:

In order to profile JavaLife, I used visualvm to view the performance of the individual methods. I first ran ‘java com/laboon/JavaLife 100 100 50 30000’ and ran the profiler in visualvm to find that the method getNumNeighbors within the World class was taking up about 95% of the CPU Time. In order to be sure this was the ineffective method and not just an error, I proceeded to run ‘java com/laboon/JavaLife’ 4 more times with the parameters ‘100 100 10 30000’, ‘100 100 90 30000’, ‘5 100 50 30000’, and ‘500 100 50 30000’. For all of these runs, the method getNumNeighbors also took up over 90% of the CPU Time, so I determined this was the extremely ineffective method.

In looking at getNumNeighbors, I noticed that it was running through a for loop 10000 times that only had to execute once. It ran through the loop 10000 times, but only ever made any changes in the first iteration. To fix this, I took out the for loop and executed the body of it just once. This had the same effect as the original program, but it is more efficient and quicker to execute.

I then ran the command ‘time java com/laboon/JavaLife 100 100 50 1000’ on both the original code and my altered code to compare their real execution times and prove that my altered code performs more effectively. While the original code ran in

“real 1m45.942s, user 1m39.523s, sys 0m0.669s”

my altered code ran in

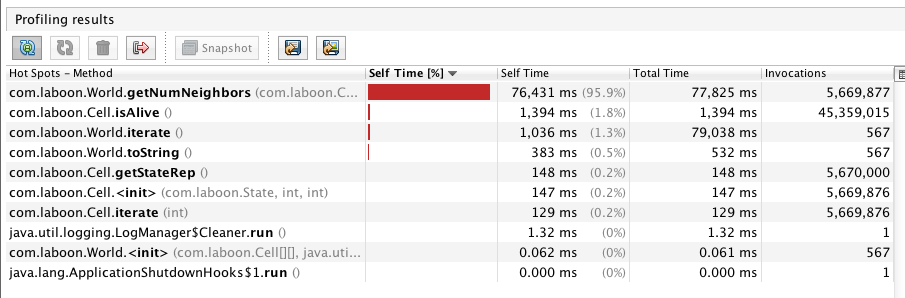
“real 0m2.033s, user 0m1.177s, sys 0m0.287s”.

I also ran visualvm to profile the altered code like I did to the original code. I compared the profile of the altered code to the profile of the original code to ensure that it is more effective.

My altered code is located at https://github.com/Cln42/JavaLife

Screenshots:

Profiling Results of Original Code:



Profiling Results of Altered Code:

