Summary

Before profiling this application, I used some exploratory testing to find any obvious defects. Immediately I noticed that the write button took forever to complete. Everything else seemed acceptable.

To profile I used the Visual VM profiler. As seen in the screenshot below, the most obvious source performance slowdown was the convertToInt() function. When looking at the code, I noticed that this function was taking in an integer and returning that same number. Therefore I was able to eliminate this function entirely, and remove the call from the getNumNeighbors function, correcting this issue.

My next task was to correct the write function problem. When tracing through the code for this function, I noticed an unnecessary loop in the Cell.toString() function. By eliminating this loop, I was able to reduce the time to write the file to nearly instantaneous.

Running the profiler again with these changes in place, I saw that the runContinuous() method was taking up a lot of time. While this is to be expected, I decided to explore a little more. The first thing I discovered was that most of this time was spent completing the backup() method, specifically cell.<init>. The backup function was creating new array of cells each time it was called. This killed performance. I chose to declare and fill this cell array at the beginning of the program instead. In my refactor the backup method now resets each cell in the existing array instead of creating new ones.

Finally there was one more glaring error I found. This was present in the runContinous() method itself. This method contained a sleep command as well as a loop which modified two variables used nowhere else in any of the classes. These variables were not used in the run() method either, which means they are not needed to run the game. To speed up performance, I removed this sleep command as well as the loop and unused variable declarations.

For this last refactor I was forced to create manual unit tests. I was unable to write automated tests as I could not stop the runContinuous() function with a simple call to stop(). It was also much easier to visually see what cells were alive or dead than try and program to check the cells individually.

As for the automated tests, please note that some private methods and variables were changed to public in the MainPanel class. This was done so that the JUnit tests could access these items during unit test runs.

NOTE: “+2 Consolation Prize” for in class Performance Test challenge