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

Nathan Spangler

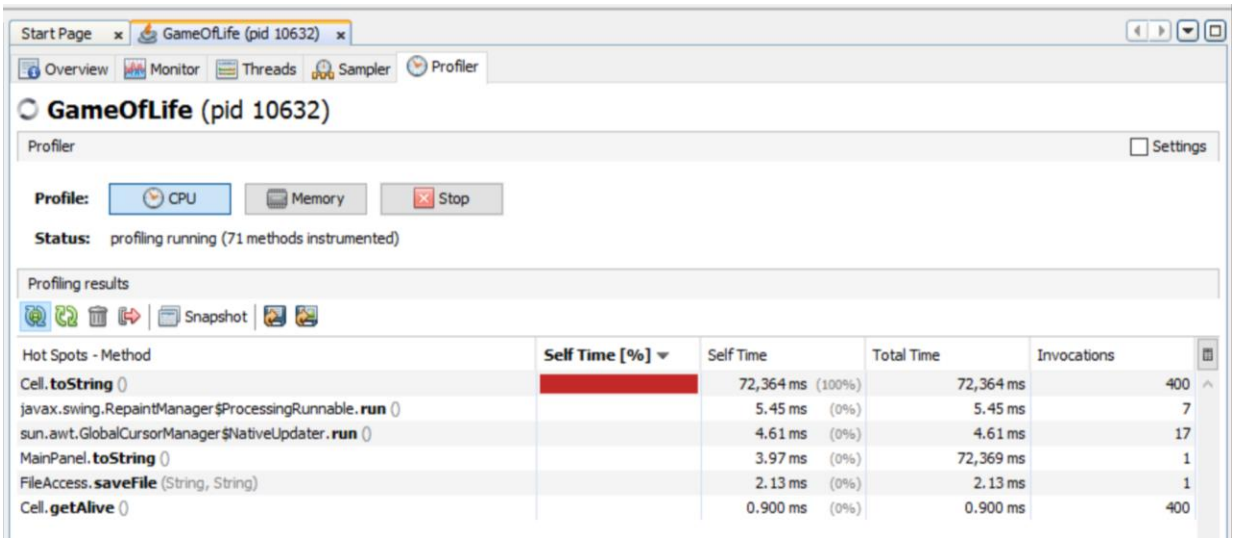
https://github.com/njs55/SlowLifeGUI

Summary

For this deliverable I first started off with just playing with the GameOfLife and seeing if I could see any hot spots from just using it. Right away I noticed that the iterations seemed to be very slow from one to the next. I also noticed that when you try to write to save the state, it would take forever. So I then opened up my visualVM profiler and ran the game continuously and saw that the MainPanel.convertToInt() function was taking most the time. I then also wrote to save the state and saw that Cell.toString() was taking up all the time there. So I knew what 2 functions to start righting pinning tests for.

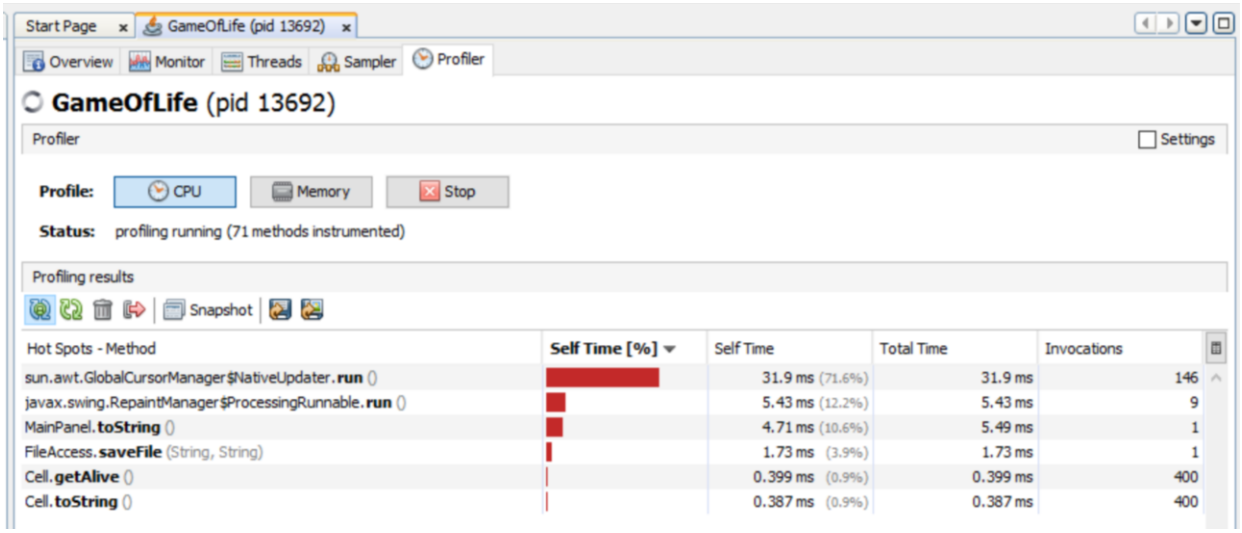
After writing the pinning tests for these 2 methods and making sure they passed I moved on to refactoring these methods to help make the game faster. The only issue I saw with writing these pinning tests, was when I needed to write unit tests for a private method. I needed to look up how to invoke the private method properly. So after refactoring the methods, which was mainly just deleting random loops and timers, I profiled the game again. These 2 methods run times dropped dramatically. Now that these two methods CPU times were reduced I could see other problem areas. One was in the MainPanel.runContinuous() method. This refactor wasn't as dramatic as the others but it still helped speed up the game in terms of CPU time usage. This refactor I tested manually by just looking at the CPU time usage after deleting an unnecessary loop and seeing that the game still operated properly.

Before Refactoring: Cell.toString



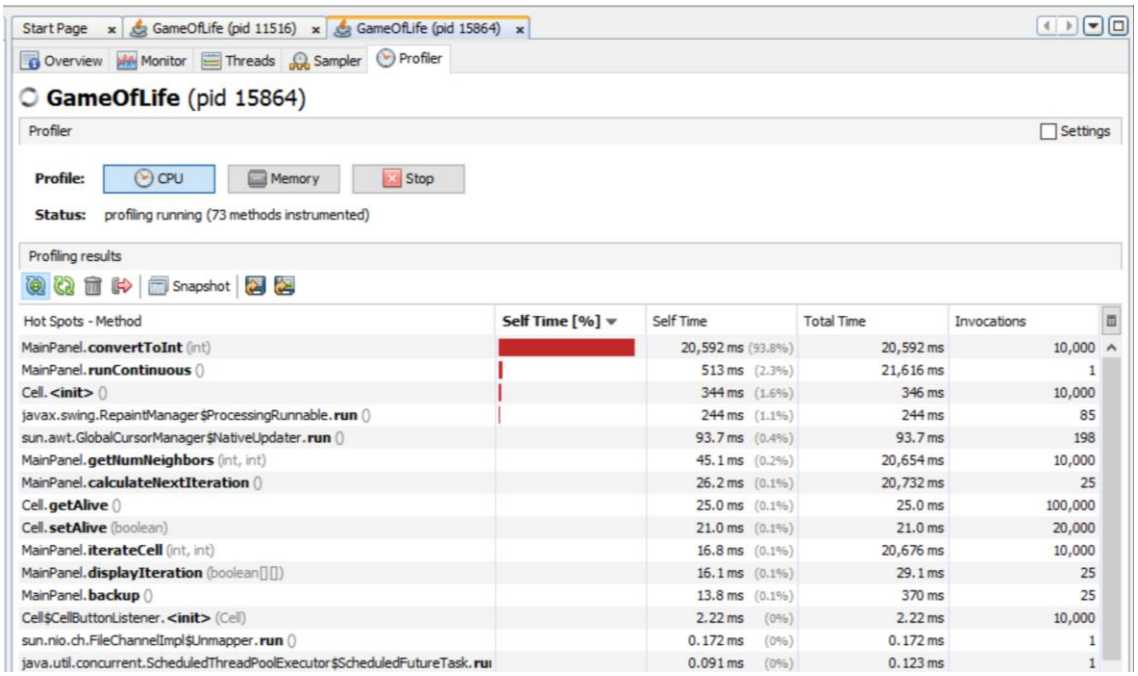
Cell.toString() Self Time when writing 6 vertical alive cells
Time: 72,364ms

After Refactoring: Cell.toString



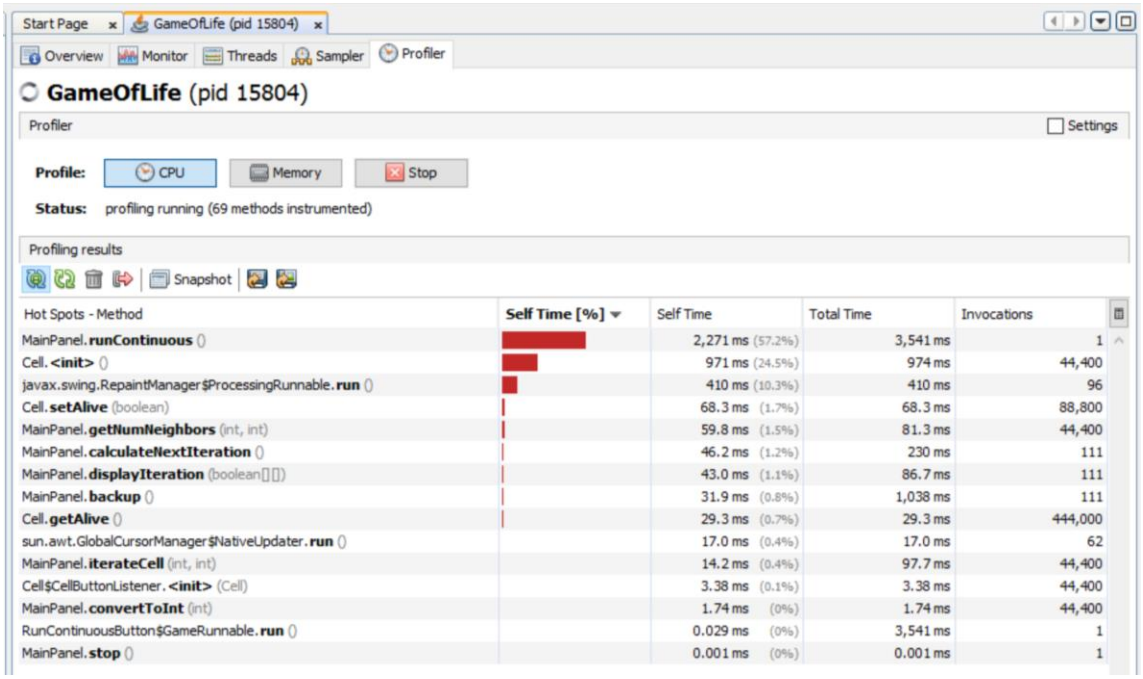
Cell.toString() Self Time when writing 6 vertical alive cells
Time: 0.387

Before Refactoring: MainPanel.convertToInt



MainPanel.convertToInt() Self Time when running continuous starting with 8 vertical cells.
Time: 20,592ms

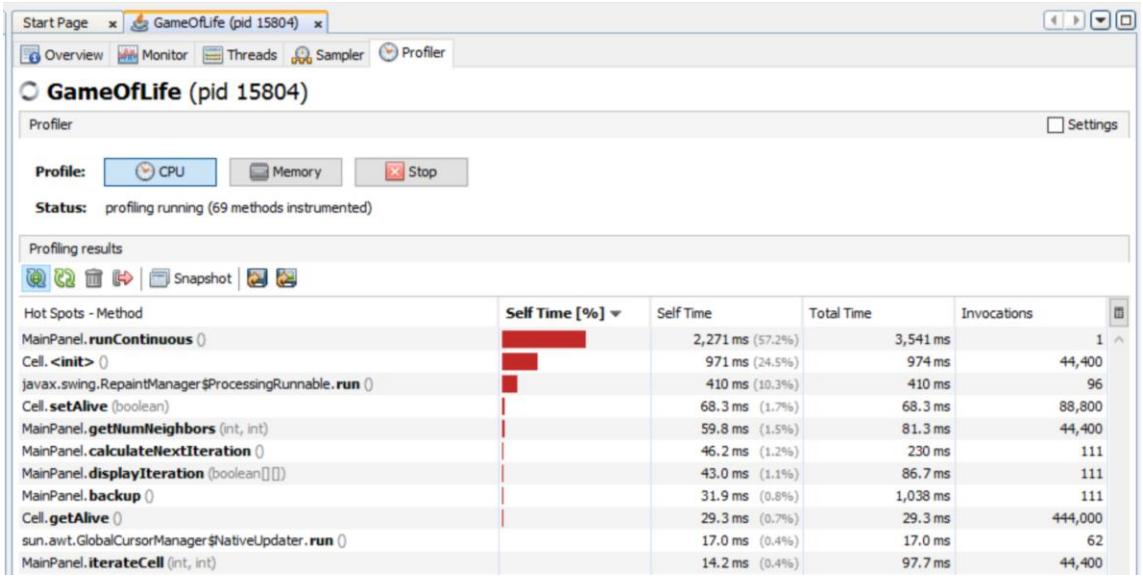
After Refactoring: MainPanel.convertToInt



MainPanel.convertToInt() Self Time when running continuous starting with 8 vertical cells.
Time: 20,592ms

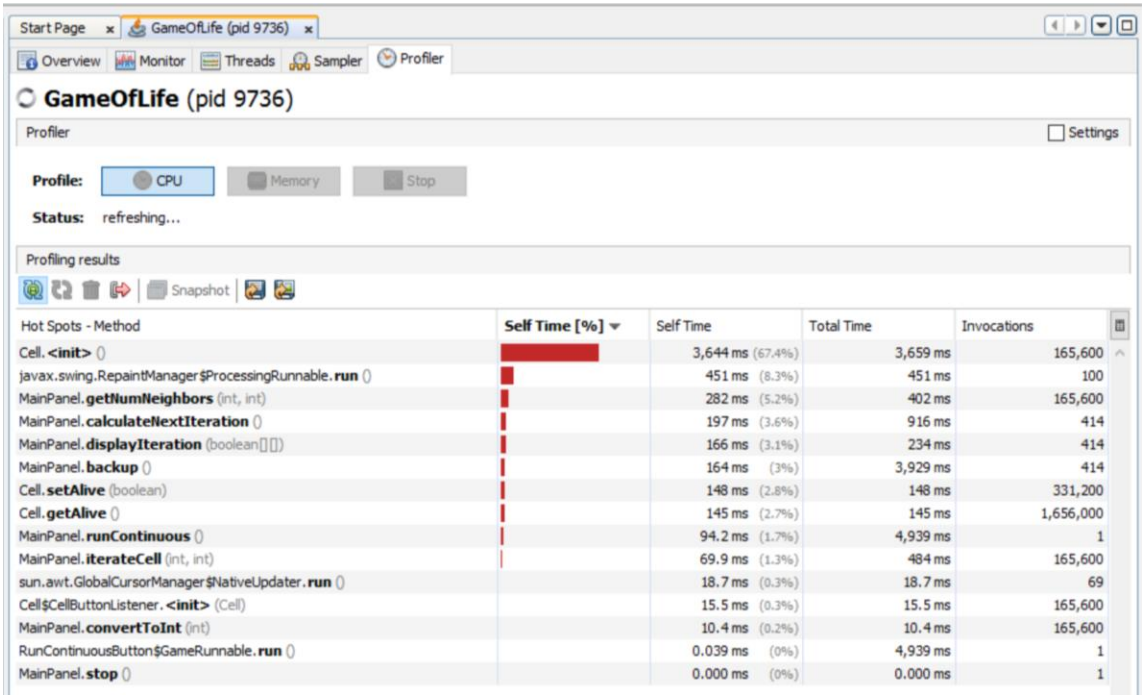
After refactoring MainPanel.convertToInt, I noticed MainPanel.runContinuous could be improved.

Before Refactoring: MainPanel.runContinuous

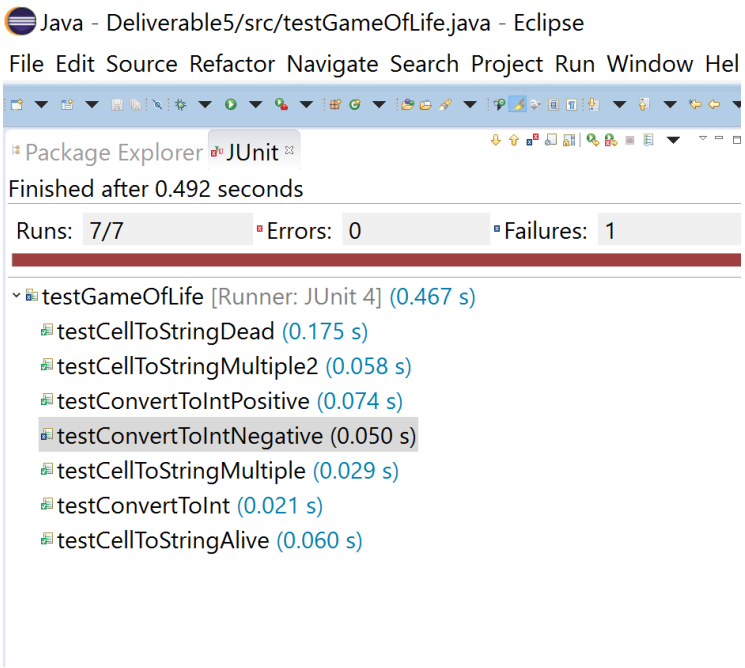


MainPanel.runContinous() Self Time when running continuous starting with 8 vertical cells.
Time: 2,271ms

After Refactoring: MainPanel.runContinuous



MainPanel.runContinous() Self Time when running continuous starting with 8 vertical cells.
Time: 94.2ms



These are the pinning tests that were ran before and after refactoring to ensure the functionality was not altered.

1 fail is on purpose. This was when I was testing inserting a negative into the MainPanel.converToInt method to see if it was capable of handling this error.