

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

**Nathan Spangler**

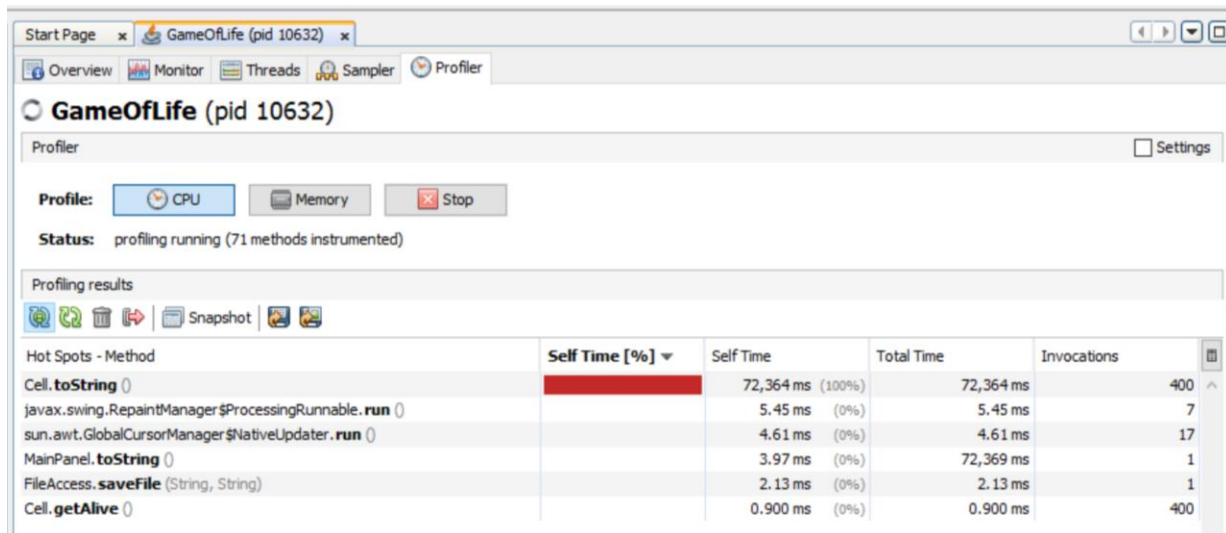
<https://github.com/njs55/SlowLifeGUI>

## 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.

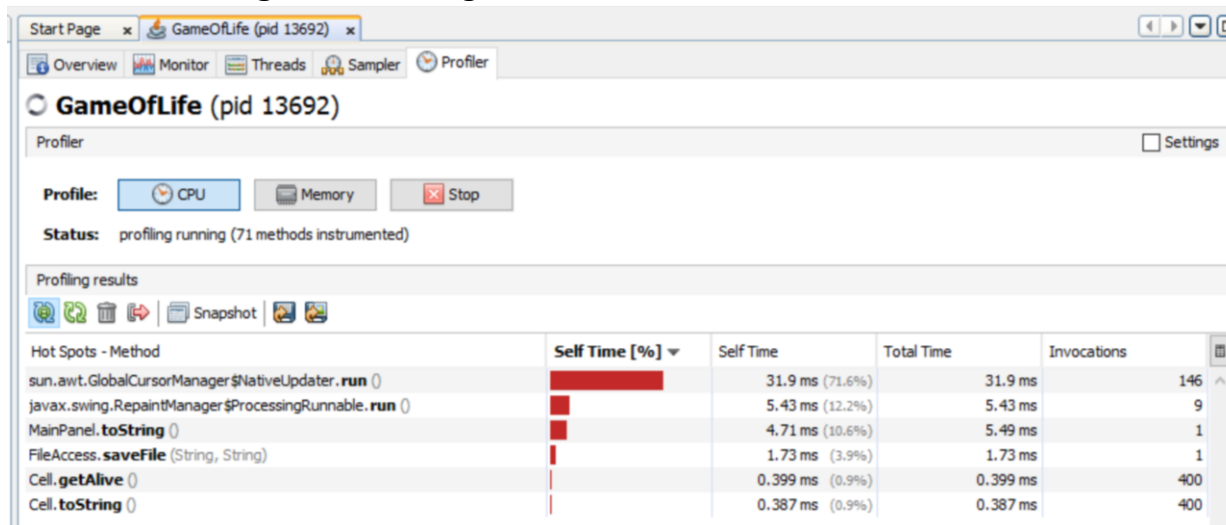
After writing the pinning tests for these 2 methods and making sure they passed I moved on to refactoring these methods to help make the game faster. The only issue I saw with writing these pinning tests, was when I needed to write unit tests for a private method. I needed to look up how to invoke the 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

Hot Spots - Method	Self Time [%]	Self Time	Total Time	Invocations
MainPanel.convertToInt (int)	93.8%	20,592 ms	20,592 ms	10,000
MainPanel.runContinuous ()	2.3%	513 ms	21,616 ms	1
Cell.<init> ()	1.6%	344 ms	346 ms	10,000
javax.swing.RepaintManager\$ProcessingRunnable.run ()	1.1%	244 ms	244 ms	85
sun.awt.GlobalCursorManager\$NativeUpdater.run ()	0.4%	93.7 ms	93.7 ms	198
MainPanel.getNeighbours (int, int)	0.2%	45.1 ms	20,654 ms	10,000
MainPanel.calculateNextIteration ()	0.1%	26.2 ms	20,732 ms	25
Cell.getAlive ()	0.1%	25.0 ms	25.0 ms	100,000
Cell.setAlive (boolean)	0.1%	21.0 ms	21.0 ms	20,000
MainPanel.iterateCell (int, int)	0.1%	16.8 ms	20,676 ms	10,000
MainPanel.displayIteration (boolean[][])	0.1%	16.1 ms	29.1 ms	25
MainPanel.backup ()	0.1%	13.8 ms	370 ms	25
Cell\$CellButtonListener.<init> (Cell)	0%	2.22 ms	2.22 ms	10,000
sun.nio.ch.FileChannelImpl\$Unmapper.run ()	0%	0.172 ms	0.172 ms	1
java.util.concurrent.ScheduledThreadPoolExecutor\$ScheduledFutureTask.run ()	0%	0.091 ms	0.123 ms	1

MainPanel.convertToInt() Self Time when running continuous starting with 8 vertical cells.

Time: 20,592ms

## After Refactoring: MainPanel.convertToInt

Hot Spots - Method	Self Time [%]	Self Time	Total Time	Invocations
MainPanel.runContinuous ()	57.2%	2,271 ms	3,541 ms	1
Cell.<init> ()	24.5%	971 ms	974 ms	44,400
javax.swing.RepaintManager\$ProcessingRunnable.run ()	10.3%	410 ms	410 ms	96
Cell.setAlive (boolean)	1.7%	68.3 ms	68.3 ms	88,800
MainPanel.getNeighbours (int, int)	1.5%	59.8 ms	81.3 ms	44,400
MainPanel.calculateNextIteration ()	1.2%	46.2 ms	230 ms	111
MainPanel.displayIteration (boolean[][])	1.1%	43.0 ms	86.7 ms	111
MainPanel.backup ()	0.8%	31.9 ms	1,038 ms	111
Cell.getAlive ()	0.7%	29.3 ms	29.3 ms	444,000
sun.awt.GlobalCursorManager\$NativeUpdater.run ()	0.4%	17.0 ms	17.0 ms	62
MainPanel.iterateCell (int, int)	0.4%	14.2 ms	97.7 ms	44,400
Cell\$CellButtonListener.<init> (Cell)	0.1%	3.38 ms	3.38 ms	44,400
MainPanel.convertToInt (int)	0%	1.74 ms	1.74 ms	44,400
RunContinuousButton\$GameRunnable.run ()	0%	0.029 ms	3,541 ms	1
MainPanel.stop ()	0%	0.001 ms	0.001 ms	1

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

Hot Spots - Method	Self Time [%] ▼	Self Time	Total Time	Invocations
MainPanel.runContinuous ()	57.2%	2,271 ms (57.2%)	3,541 ms	1
Cell.<init> ()		971 ms (24.5%)	974 ms	44,400
javax.swing.RepaintManager\$ProcessingRunnable.run ()		410 ms (10.3%)	410 ms	96
Cell.setAlive (boolean)		68.3 ms (1.7%)	68.3 ms	88,800
MainPanel.getNiumNeighbors (int, int)		59.8 ms (1.5%)	81.3 ms	44,400
MainPanel.calculateNextIteration ()		46.2 ms (1.2%)	230 ms	111
MainPanel.displayIteration (boolean[])		43.0 ms (1.1%)	86.7 ms	111
MainPanel.backup ()		31.9 ms (0.8%)	1,038 ms	111
Cell.getAlive ()		29.3 ms (0.7%)	29.3 ms	444,000
sun.awt.GlobalCursorManager\$NativeUpdater.run ()		17.0 ms (0.4%)	17.0 ms	62
MainPanel.iterateCell (int, int)		14.2 ms (0.4%)	97.7 ms	44,400

MainPanel.runContinuous() Self Time when running continuous starting with 8 vertical cells.

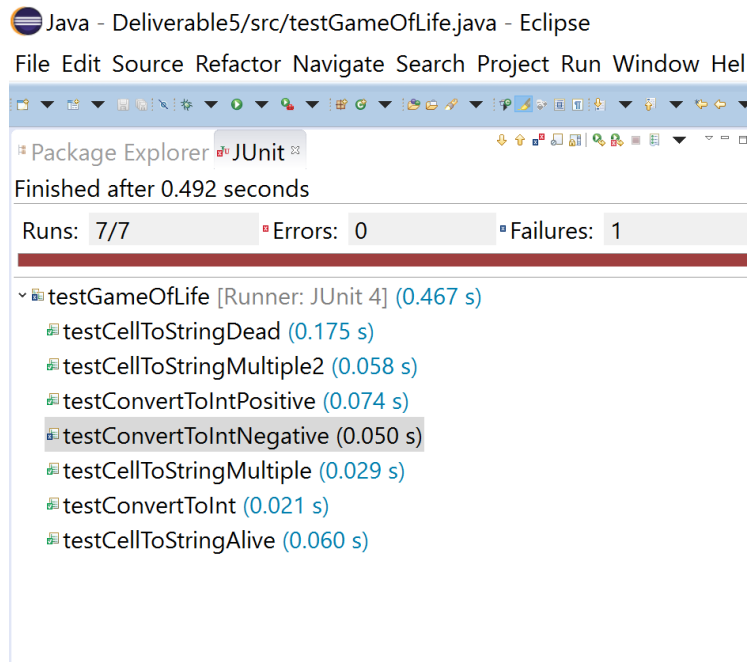
Time: 2,271ms

## After Refactoring: MainPanel.runContinuous

Hot Spots - Method	Self Time [%] ▼	Self Time	Total Time	Invocations
Cell.<init> ()	67.4%	3,644 ms (67.4%)	3,659 ms	165,600
javax.swing.RepaintManager\$ProcessingRunnable.run ()		451 ms (8.3%)	451 ms	100
MainPanel.getNiumNeighbors (int, int)		282 ms (5.2%)	402 ms	165,600
MainPanel.calculateNextIteration ()		197 ms (3.6%)	916 ms	414
MainPanel.displayIteration (boolean[])		166 ms (3.1%)	234 ms	414
MainPanel.backup ()		164 ms (3%)	3,929 ms	414
Cell.setAlive (boolean)		148 ms (2.8%)	148 ms	331,200
Cell.getAlive ()		145 ms (2.7%)	145 ms	1,656,000
MainPanel.runContinuous ()		94.2 ms (1.7%)	4,939 ms	1
MainPanel.iterateCell (int, int)		69.9 ms (1.3%)	484 ms	165,600
sun.awt.GlobalCursorManager\$NativeUpdater.run ()		18.7 ms (0.3%)	18.7 ms	69
Cell\$CellButtonListener.<init> (Cell)		15.5 ms (0.3%)	15.5 ms	165,600
MainPanel.convertToInt (int)		10.4 ms (0.2%)	10.4 ms	165,600
RunContinuousButton\$GameRunnable.run ()		0.039 ms (0%)	4,939 ms	1
MainPanel.stop ()		0.000 ms (0%)	0.000 ms	1

MainPanel.runContinuous() 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.convertToInt method to see if it was capable of handling this error.