CS 1632 – DELIVERABLE 5
Performance Testing Conway's Game of Life
Project under Test: JavaLife

`https://github.com/blester125/SlowLifeGUI`

Brian Lester

March 31, 2016

1 Summary

For deliverable 5 I profiled the Game of Life provided with the Visual VM profiler. In the deliverable description, the code was described as non performent. This meant that the CPU time for each method was most likely a good metric for finding methods that were causing problems. By running the Game of Live, Visual VM, and using the CPU sampler I looked for "Hot Spots". This identified the method convertToInt(int x) in the MainPanel class was a "Hot Spot". This is where I began my investigation.

This method was the highest on the profiler list so I investigated it. Once I looked at it I saw that the method was converting "int"s to "int"s which means that it probably doing extra stuff because the entire method seem pretty pointless. Upon closer inspection I found that function was creating a large string that it would concatenate the "int x" onto then call the Integer. parseInt() method on it. This is clearly extra and pointless operations so I decided that I could refactor this code to fix the problem.

Using exploratory testing I found that the method acted as follows.

$$\begin{cases} NumberFormatException & x < 0 \\ x & x \geq 0 \end{cases}$$

Once I figured out how the method worked I wrote the required pinning test to show that I did not change the functionality of the program. First I changed the method declaration from private to public to make testing easier. I first wrote test for the three equivalence class, negative numbers, zero, and positive numbers. For the negative number equivalence class I created a test to see if it still throw the "NumberFormatException" because the pinning tests are for current behavior not expected behavior. Finally I wrote a property based test for the method. The property that I tested was that when a number greater than 0 is passed in it returns the same number. I tested this by randomly testing the method with 100 integers between 0 and 400. The pinning tests can be found in the "MainPanelTest.java" file.

After fixing the "converToInt()" method I ran the profiler again and examined the profile when I write to the file. This identified the "Cell.toString" method as a "Hot Spot". When I was exploratory testing I had noticed that the file output was really slow so this makes sense as a "Hot Spot".

After examination I found that the method worked like so.

$$\begin{cases} \text{``X''} & \text{text starts with X} \\ \text{``.''} & \text{text doesn't start with X} \end{cases}$$

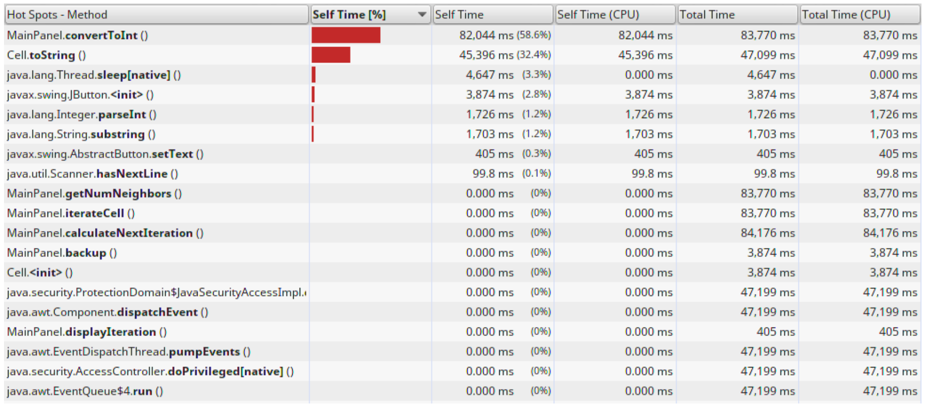
I wrote pinning tests to confirm this behavior after the refactor. The pinning tests can be found in the "CellTest.java" file.

Running the profiler again I found another hotspot. The profiler identified Thread.sleep() as the hotspot but this seemed to used for user input so I kept looking. The first method that I could reasonably change (The first method that was not a jswing function) was the MainPanel.runContinuous() function. Investigating I found a loop that ran a bunch and changed the "_r" value. However, before the loop the value of "_r" is saved into a variable "origR" and after the loop the value of "_r" is overwritten by "origR". This means that the value of "_r" is not changed by the loop so it could be taken out. Pinning tests for this method are manual and are included in this document.

The included screen shots show that the refactored methods are no longer a "Hot Spots" and the JUint screen shots show that the behavior of the methods were not changed.

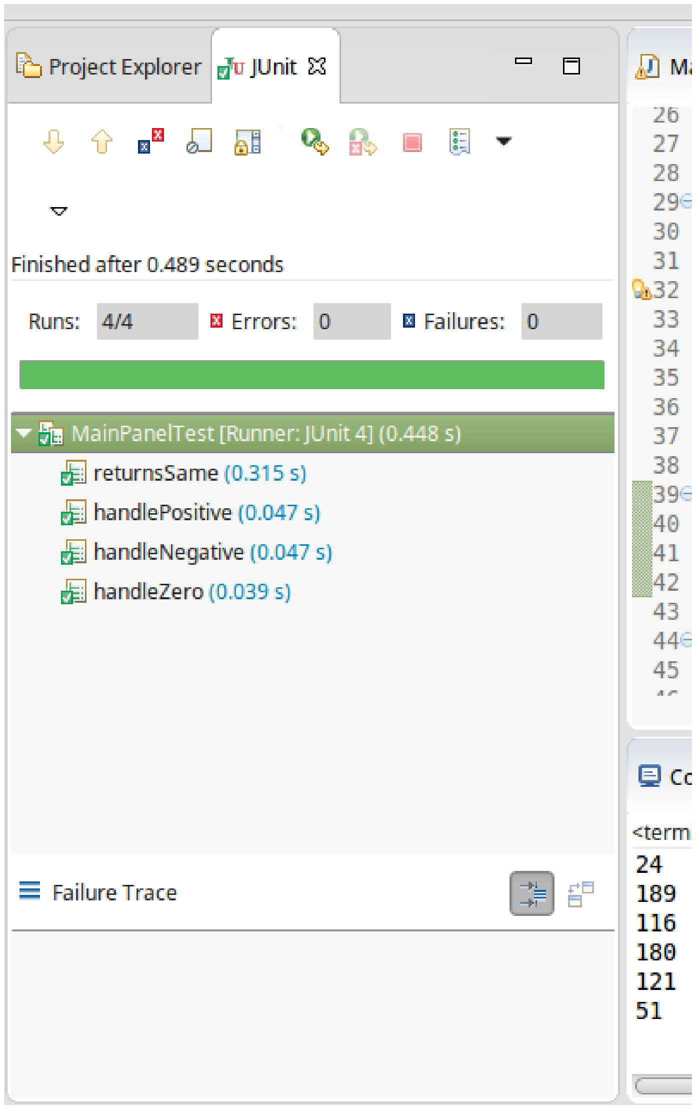
2 Before Refactoring of MainPanel.convertToInt()

2.1 Profiler ScreenShot



Hot Spots - Method	Self Time [%]	Self Time	Self Time (CPU)	Total Time	Total Time (CPU)
MainPanel.**convertToInt** ()		82,044 ms (58.6%)	82,044 ms	83,770 ms	83,770 ms
Cell.**toString** ()		45,396 ms (32.4%)	45,396 ms	47,099 ms	47,099 ms
java.lang.Thread.**sleep[native]** ()		4,647 ms (3.3%)	0.000 ms	4,647 ms	0.000 ms
javax.swing.JButton.**<init>** ()		3,874 ms (2.8%)	3,874 ms	3,874 ms	3,874 ms
java.lang.Integer.**parseInt** ()		1,726 ms (1.2%)	1,726 ms	1,726 ms	1,726 ms
java.lang.String.**substring** ()		1,703 ms (1.2%)	1,703 ms	1,703 ms	1,703 ms
javax.swing.AbstractButton.**setText** ()		405 ms (0.3%)	405 ms	405 ms	405 ms
java.util.Scanner.**hasNextLine** ()		99.8 ms (0.1%)	99.8 ms	99.8 ms	99.8 ms
MainPanel.**getNumNeighbors** ()		0.000 ms (0%)	0.000 ms	83,770 ms	83,770 ms
MainPanel.**iterateCell** ()		0.000 ms (0%)	0.000 ms	83,770 ms	83,770 ms
MainPanel.**calculateNextIteration** ()		0.000 ms (0%)	0.000 ms	84,176 ms	84,176 ms
MainPanel.**backup** ()		0.000 ms (0%)	0.000 ms	3,874 ms	3,874 ms
Cell.**<init>** ()		0.000 ms (0%)	0.000 ms	3,874 ms	3,874 ms
java.security.ProtectionDomain$JavaSecurityAccessImpl.		0.000 ms (0%)	0.000 ms	47,199 ms	47,199 ms
java.awt.Component.**dispatchEvent** ()		0.000 ms (0%)	0.000 ms	47,199 ms	47,199 ms
MainPanel.**displayIteration** ()		0.000 ms (0%)	0.000 ms	405 ms	405 ms
java.awt.EventDispatchThread.**pumpEvents** ()		0.000 ms (0%)	0.000 ms	47,199 ms	47,199 ms
java.security.AccessController.**doPrivileged[native]** ()		0.000 ms (0%)	0.000 ms	47,199 ms	47,199 ms
java.awt.EventQueue$4.**run** ()		0.000 ms (0%)	0.000 ms	47,199 ms	47,199 ms

2.2 Pinning Tests



Finished after 0.489 seconds

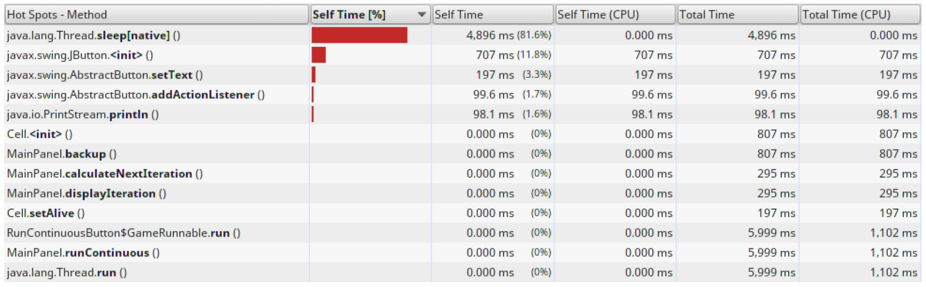
Runs: 4/4 Errors: 0 Failures: 0

```
MainPanelTest [Runner: JUnit 4] (0.448 s)
    returnsSame (0.315 s)
    handlePositive (0.047 s)
    handleNegative (0.047 s)
    handleZero (0.039 s)
```

```
<term
24
189
116
180
121
51
```

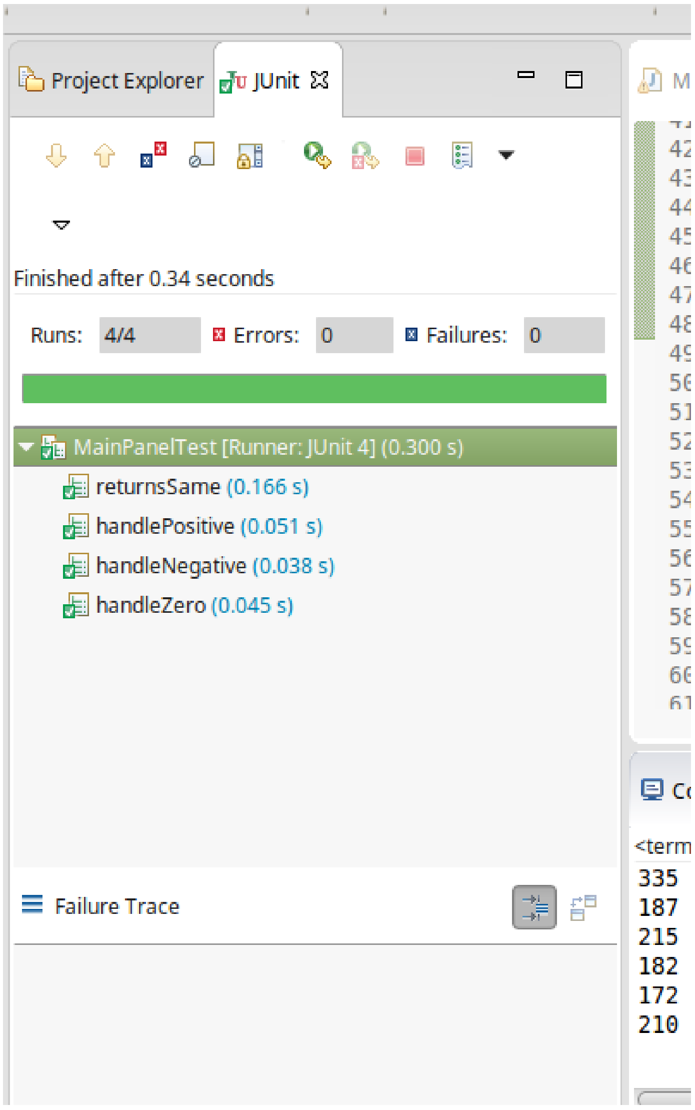
3 After Refactoring of MainPanel.convertToInt()

3.1 Profiler ScreenShot



Hot Spots - Method	Self Time [%]	Self Time	Self Time (CPU)	Total Time	Total Time (CPU)
java.lang.Thread.**sleep[native]** ()		4,896 ms (81.6%)	0.000 ms	4,896 ms	0.000 ms
javax.swing.JButton.**<init>** ()		707 ms (11.8%)	707 ms	707 ms	707 ms
javax.swing.AbstractButton.**setText** ()		197 ms (3.3%)	197 ms	197 ms	197 ms
javax.swing.AbstractButton.**addActionListener** ()		99.6 ms (1.7%)	99.6 ms	99.6 ms	99.6 ms
java.io.PrintStream.**println** ()		98.1 ms (1.6%)	98.1 ms	98.1 ms	98.1 ms
Cell.**<init>** ()		0.000 ms (0%)	0.000 ms	807 ms	807 ms
MainPanel.**backup** ()		0.000 ms (0%)	0.000 ms	807 ms	807 ms
MainPanel.**calculateNextIteration** ()		0.000 ms (0%)	0.000 ms	295 ms	295 ms
MainPanel.**displayIteration** ()		0.000 ms (0%)	0.000 ms	295 ms	295 ms
Cell.**setAlive** ()		0.000 ms (0%)	0.000 ms	197 ms	197 ms
RunContinuousButton$GameRunnable.**run** ()		0.000 ms (0%)	0.000 ms	5,999 ms	1,102 ms
MainPanel.**runContinuous** ()		0.000 ms (0%)	0.000 ms	5,999 ms	1,102 ms
java.lang.Thread.**run** ()		0.000 ms (0%)	0.000 ms	5,999 ms	1,102 ms

3.2 Pinning Tests



Finished after 0.34 seconds

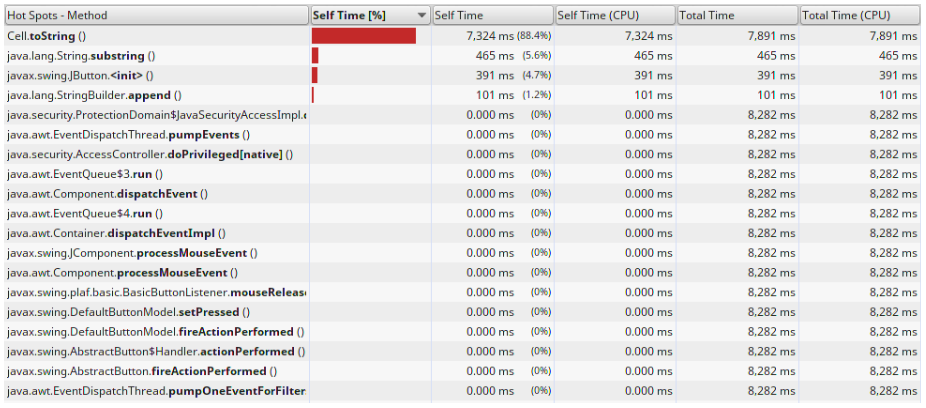
Runs: 4/4 Errors: 0 Failures: 0

```
MainPanelTest [Runner: JUnit 4] (0.300 s)
    returnsSame (0.166 s)
    handlePositive (0.051 s)
    handleNegative (0.038 s)
    handleZero (0.045 s)
```

```
<term
335
187
215
182
172
210
```

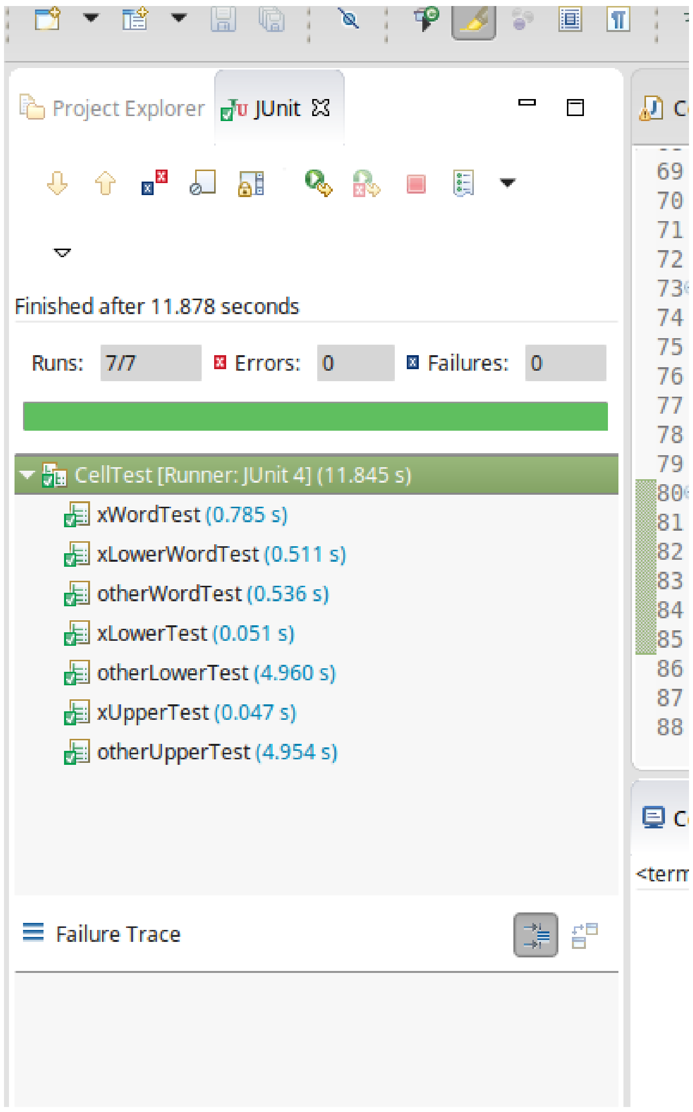
4 Before Refactoring of Cell.toString()

4.1 Profiler ScreenShot



Hot Spots - Method	Self Time [%]	Self Time	Self Time (CPU)	Total Time	Total Time (CPU)
Cell.**toString** ()	🟥	7,324 ms (88.4%)	7,324 ms	7,891 ms	7,891 ms
java.lang.String.**substring** ()	🟥	465 ms (5.6%)	465 ms	465 ms	465 ms
javax.swing.JButton.**<init>** ()	🟥	391 ms (4.7%)	391 ms	391 ms	391 ms
java.lang.StringBuilder.**append** ()	🟥	101 ms (1.2%)	101 ms	101 ms	101 ms
java.security.ProtectionDomain$JavaSecurityAccessImpl.		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.awt.EventDispatchThread.**pumpEvents** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.security.AccessController.**doPrivileged[native]** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.awt.EventQueue$3.**run** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.awt.Component.**dispatchEvent** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.awt.EventQueue$4.**run** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.awt.Container.**dispatchEventImpl** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
javax.swing.JComponent.**processMouseEvent** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.awt.Component.**processMouseEvent** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
javax.swing.plaf.basic.BasicButtonListener.**mouseReleas**		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
javax.swing.DefaultButtonModel.**setPressed** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
javax.swing.DefaultButtonModel.**fireActionPerformed** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
javax.swing.AbstractButton$Handler.**actionPerformed** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
javax.swing.AbstractButton.**fireActionPerformed** ()		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms
java.awt.EventDispatchThread.**pumpOneEventForFilter**		0.000 ms (0%)	0.000 ms	8,282 ms	8,282 ms

4.2 Pinning Tests



Project Explorer JUnit

Finished after 11.878 seconds

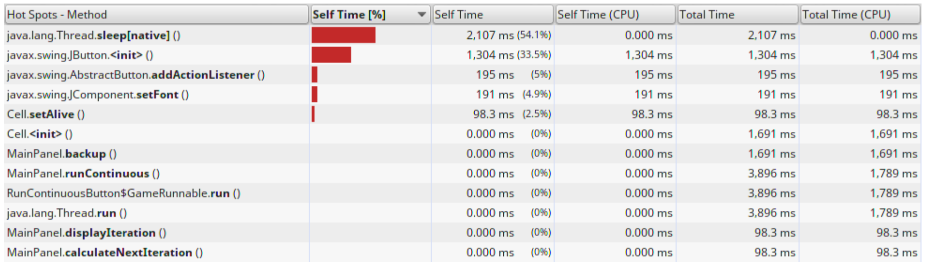
Runs: 7/7 Errors: 0 Failures: 0

CellTest [Runner: JUnit 4] (11.845 s)
xWordTest (0.785 s)
xLowerWordTest (0.511 s)
otherWordTest (0.536 s)
xLowerTest (0.051 s)
otherLowerTest (4.960 s)
xUpperTest (0.047 s)
otherUpperTest (4.954 s)

Failure Trace

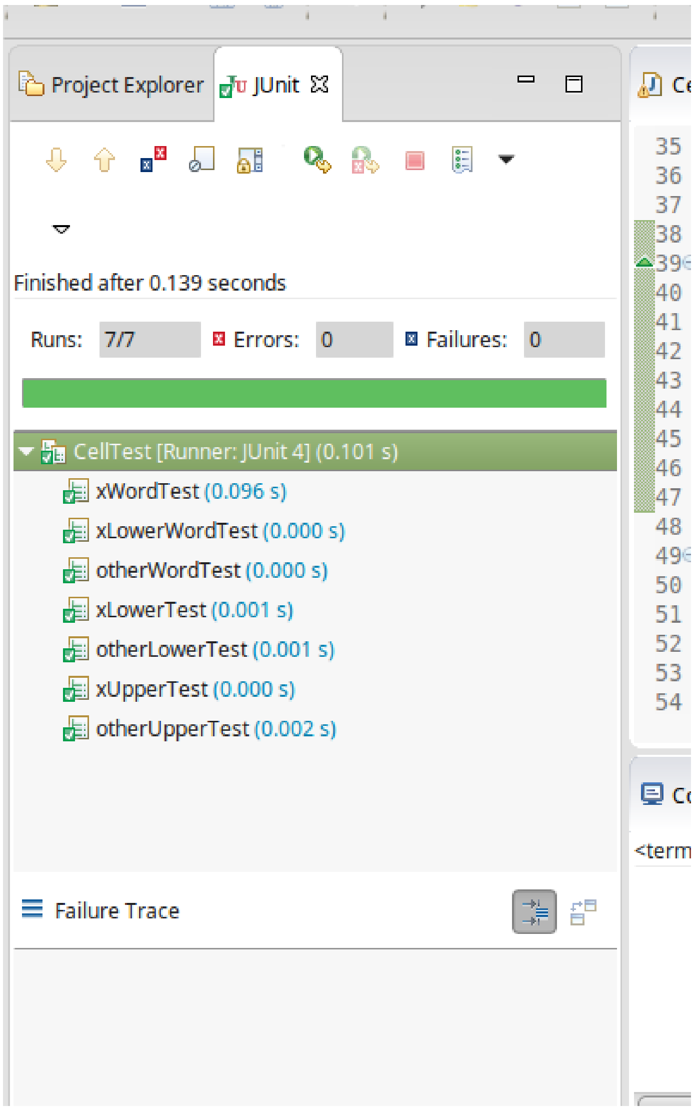
5 After Refactoring of Cell.toString()

5.1 Profiler ScreenShot



Hot Spots - Method	**Self Time [%]**	Self Time	Self Time (CPU)	Total Time	Total Time (CPU)
java.lang.Thread.**sleep[native]** ()	🟥	2,107 ms (54.1%)	0.000 ms	2,107 ms	0.000 ms
javax.swing.JButton.**<init>** ()	🟥	1,304 ms (33.5%)	1,304 ms	1,304 ms	1,304 ms
javax.swing.AbstractButton.**addActionListener** ()	🟥	195 ms (5%)	195 ms	195 ms	195 ms
javax.swing.JComponent.**setFont** ()	🟥	191 ms (4.9%)	191 ms	191 ms	191 ms
Cell.**setAlive** ()	🟥	98.3 ms (2.5%)	98.3 ms	98.3 ms	98.3 ms
Cell.**<init>** ()		0.000 ms (0%)	0.000 ms	1,691 ms	1,691 ms
MainPanel.**backup** ()		0.000 ms (0%)	0.000 ms	1,691 ms	1,691 ms
MainPanel.**runContinuous** ()		0.000 ms (0%)	0.000 ms	3,896 ms	1,789 ms
RunContinuousButton$GameRunnable.**run** ()		0.000 ms (0%)	0.000 ms	3,896 ms	1,789 ms
java.lang.Thread.**run** ()		0.000 ms (0%)	0.000 ms	3,896 ms	1,789 ms
MainPanel.**displayIteration** ()		0.000 ms (0%)	0.000 ms	98.3 ms	98.3 ms
MainPanel.**calculateNextIteration** ()		0.000 ms (0%)	0.000 ms	98.3 ms	98.3 ms

5.2 Pinning Tests



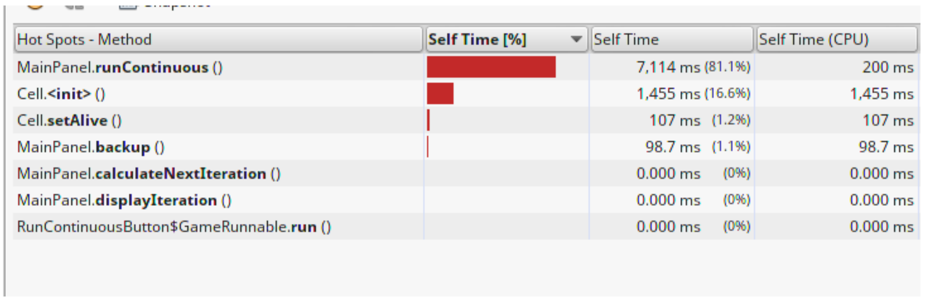
Finished after 0.139 seconds

Runs: 7/7 Errors: 0 Failures: 0

```
CellTest [Runner: JUnit 4] (0.101 s)
    xWordTest (0.096 s)
    xLowerWordTest (0.000 s)
    otherWordTest (0.000 s)
    xLowerTest (0.001 s)
    otherLowerTest (0.001 s)
    xUpperTest (0.000 s)
    otherUpperTest (0.002 s)
```

6 Before Refactoring of MainPanel.runCountinous()

6.1 Profiler ScreenShot



Hot Spots - Method	Self Time [%]	Self Time	Self Time (CPU)
MainPanel.**runContinuous** ()		7,114 ms (81.1%)	200 ms
Cell.**<init>** ()		1,455 ms (16.6%)	1,455 ms
Cell.**setAlive** ()		107 ms (1.2%)	107 ms
MainPanel.**backup** ()		98.7 ms (1.1%)	98.7 ms
MainPanel.**calculateNextIteration** ()		0.000 ms (0%)	0.000 ms
MainPanel.**displayIteration** ()		0.000 ms (0%)	0.000 ms
RunContinuousButton$GameRunnable.**run** ()		0.000 ms (0%)	0.000 ms

6.2 Pinning Tests

This method would be very difficult to test with automatic tests so we will need to use manual testing.

IDENTIFIER: 1. FUN-BUTTON-STARTS-THE-GAME
TEST CASE: Testing that pressing the run continuous button runs the game
PRECONDITIONS:
The Board is 8x8.
The cells at row 0 column 1, row 1 column 2, row 2 column 0,1,2 are alive.
INPUT VALUES: N/A
EXECUTION STEPS: Click the "Run Continuous" button
OUTPUT VALUES: N/A
POSTCONDITIONS: The game should begin and cells should start changing

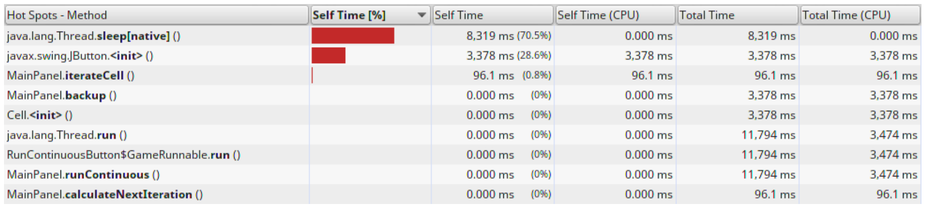
IDENTIFIER: 2. FUN-MOVE-2-ITERATIONS
TEST CASE: Move multiple iterations
PRECONDITIONS:
The Board is 8x8.
The cells at row 0 column 1, row 1 column 2, row 2 column 0,1,2 are alive.
INPUT VALUES: N/A
EXECUTION STEPS: Press the "Run Continuous" button.
Let the game run for iterations.
Press the "Stop" button.
OUTPUT VALUES: N/A
POSTCONDITIONS: The cells at row 1 column 2, row 2 column 0 and 2, row 3 column 1 and 2 are alive.

IDENTIFIER: 3. FUN-UNDO-AFTER-RUN
TEST CASE: Try to undo after running for a bit.
PRECONDITIONS:
The Board is 8x8.

The cells at row 0 column 1, row 1 column 2, row 2 column 0,1,2 are alive.
INPUT VALUES: N/A
EXECUTION STEPS: Press the “Run Continuous” button.
Let the game run for iterations.
Press the “Stop” button.
Press the “Undo” button.
OUTPUT VALUES: N/A
POSTCONDITIONS: The cells at row 1 column 0 and 2, row 2 column 1 and 2, row 3 column 1 are alive.

7 After Refactoring of MainPanel.runContinous()

7.1 Profiler ScreenShot



Hot Spots - Method	Self Time [%]	Self Time	Self Time (CPU)	Total Time	Total Time (CPU)
java.lang.Thread.**sleep[native]** ()		8,319 ms (70.5%)	0.000 ms	8,319 ms	0.000 ms
javax.swing.JButton.**<init>** ()		3,378 ms (28.6%)	3,378 ms	3,378 ms	3,378 ms
MainPanel.**iterateCell** ()		96.1 ms (0.8%)	96.1 ms	96.1 ms	96.1 ms
MainPanel.**backup** ()		0.000 ms (0%)	0.000 ms	3,378 ms	3,378 ms
Cell.**<init>** ()		0.000 ms (0%)	0.000 ms	3,378 ms	3,378 ms
java.lang.Thread.**run** ()		0.000 ms (0%)	0.000 ms	11,794 ms	3,474 ms
RunContinuousButton$GameRunnable.**run** ()		0.000 ms (0%)	0.000 ms	11,794 ms	3,474 ms
MainPanel.**runContinuous** ()		0.000 ms (0%)	0.000 ms	11,794 ms	3,474 ms
MainPanel.**calculateNextIteration** ()		0.000 ms (0%)	0.000 ms	96.1 ms	96.1 ms

7.2 Pinning Tests

All the manual test should still pass.