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

Project: Improving the Performance of the Game of Life (JavaLife)

Github Page: https://github.com/nkowdley/SlowLifeGUI

Neel Kowdley

Summary:

For this deliverable, I had to take an existing implementation of Conway's Game of Life and profile it, looking for places in the code where the code was slow, or resource intensive and change the code in order to improve the performance of the program.

I started this project by doing exploratory testing on the program, and getting familiar with how to run the program. I tested all of the available buttons and made some preliminary observations. Through my exploratory testing, I determined that the write operation appeared to be slow.

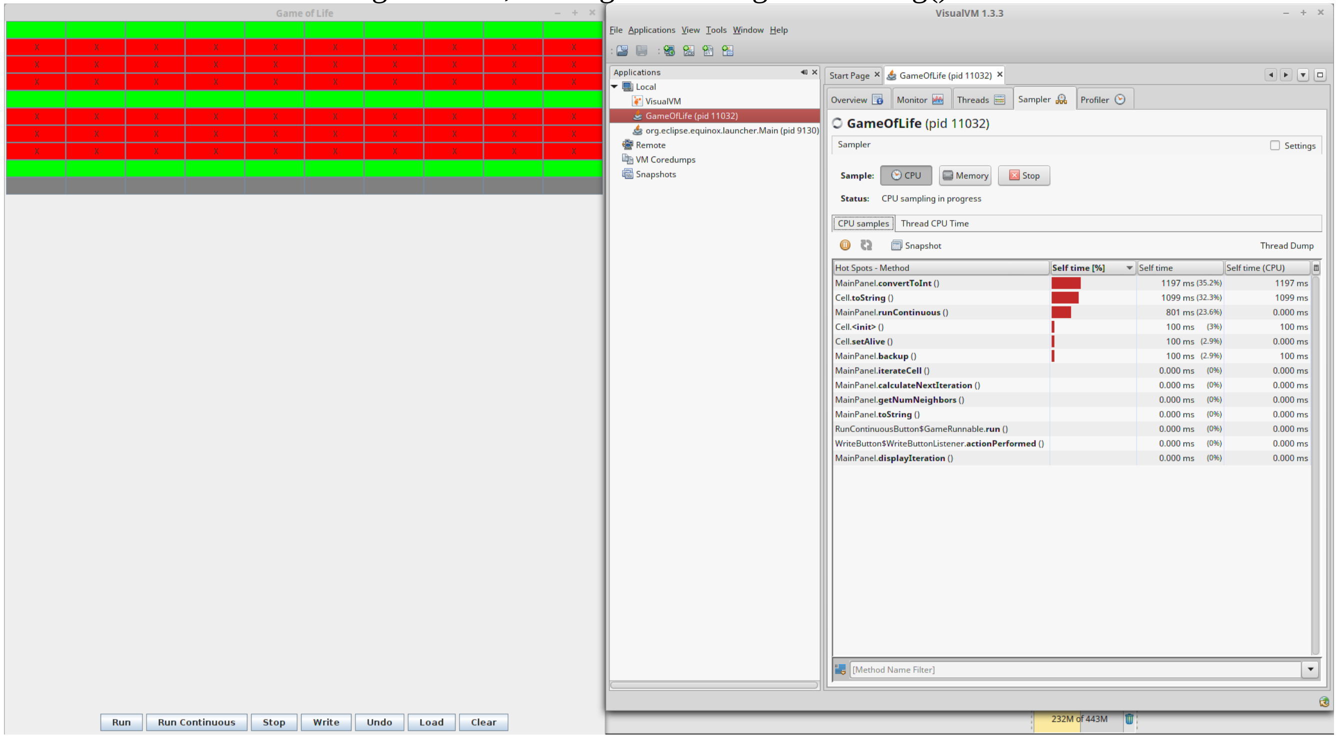
After running through some exploratory testing and becoming familiar with the program, I used a profiler to determine what other methods in the program were acting slow. I used VisualVM for this, specifically the CPU Sampler. This gave me a lot of insight into what methods were taking up a lot of CPU Time. From this, I was able to confirm that write was resource intensive, as the profiler showed that toString in Cell.java. I also determined that a method in MainPanel.java called convertToInt() was also taking a lot of CPU power. The last method I found, was also in MainPanel, and occurred in the runContinuous() method.

Once I had identified these methods, I determined their functionality by tracing through the code. After doing this, I created manual and JUnit pinning tests based on this functionality. After I did this, I renamed the original methods to contain the word “old” before the method. I then re-factored the methods to be faster and use less memory. I then confirmed that my changes improved performace with the VisualVM Sampler, and then used my pinning tests to confirm that my refactored methods and my new methods had the same functionality.

After I did this, I ran through some extra exploratory testing in order to determine that I did not unintentionally break any other parts of the code base. My changes to the Game of Life code did not break anything and everything appeared to work as intended.

VisualVM Screenshots:

Screenshot of original code, and high CPU usage of toString() and convertToInt:



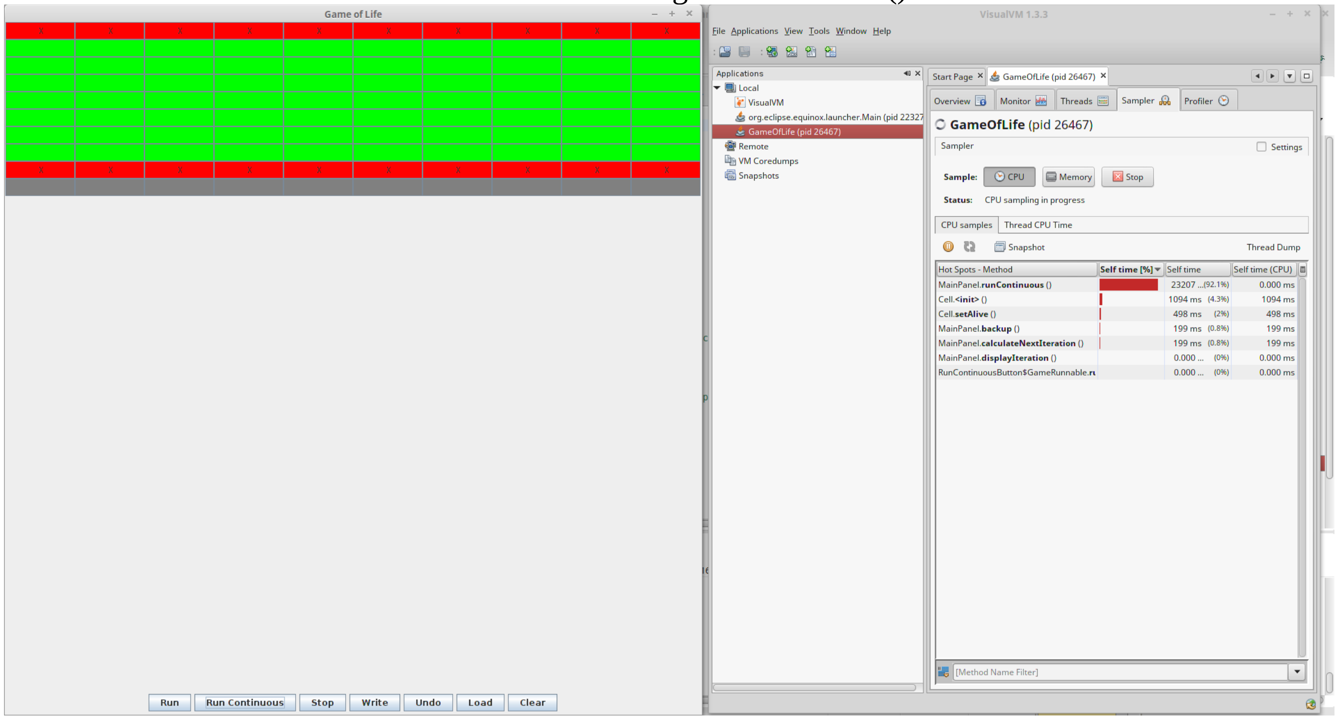
GameOfLife (pid 11032)

Sample: CPU Memory Stop

Status: CPU sampling in progress

Hot Spots - Method	Self time [%]	Self time	Self time (CPU)
MainPanel.**convertToInt** ()		1197 ms (35.2%)	1197 ms
Cell.**toString** ()		1099 ms (32.3%)	1099 ms
MainPanel.**runContinuous** ()		801 ms (23.6%)	0.000 ms
Cell.**<init>** ()		100 ms (3%)	100 ms
Cell.**setAlive** ()		100 ms (2.9%)	0.000 ms
MainPanel.**backup** ()		100 ms (2.9%)	100 ms
MainPanel.**iterateCell** ()		0.000 ms (0%)	0.000 ms
MainPanel.**calculateNextIteration** ()		0.000 ms (0%)	0.000 ms
MainPanel.**getNumNeighbors** ()		0.000 ms (0%)	0.000 ms
MainPanel.**toString** ()		0.000 ms (0%)	0.000 ms
RunContinuousButton$GameRunnable.**run** ()		0.000 ms (0%)	0.000 ms
WriteButton$WriteButtonListener.**actionPerformed** ()		0.000 ms (0%)	0.000 ms
MainPanel.**displayIteration** ()		0.000 ms (0%)	0.000 ms

Screenshot showing runContinuous()



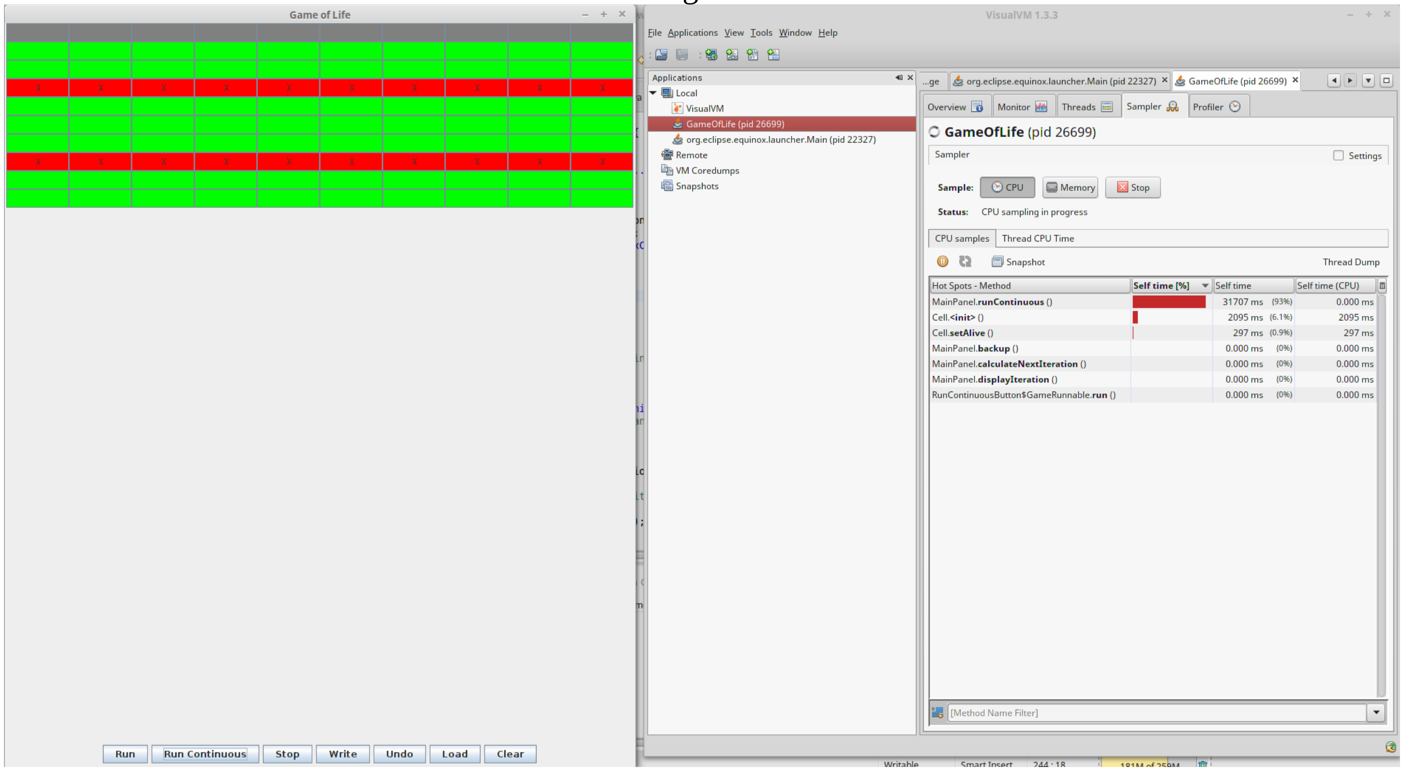
GameOfLife (pid 26467)

Sample: CPU Memory Stop

Status: CPU sampling in progress

Hot Spots - Method	Self time [%]	Self time	Self time (CPU)
MainPanel.**runContinuous** ()		23207 ...(92.1%)	0.000 ms
Cell.**<init>** ()		1094 ms (4.3%)	1094 ms
Cell.**setAlive** ()		498 ms (2%)	498 ms
MainPanel.**backup** ()		199 ms (0.8%)	199 ms
MainPanel.**calculateNextIteration** ()		199 ms (0.8%)	199 ms
MainPanel.**displayIteration** ()		0.000 ... (0%)	0.000 ms
RunContinuousButton$GameRunnable.**ru**		0.000 ... (0%)	0.000 ms

Screenshot showing the fixed code:



GameOfLife (pid 26699)

Sampler

Sample: CPU Memory Stop

Status: CPU sampling in progress

Hot Spots - Method	Self time [%]	Self time		Self time (CPU)
MainPanel.**runContinuous** ()		31707 ms	(93%)	0.000 ms
Cell.**<init>** ()		2095 ms	(6.1%)	2095 ms
Cell.**setAlive** ()		297 ms	(0.9%)	297 ms
MainPanel.**backup** ()		0.000 ms	(0%)	0.000 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

Methods and Testing:

Through my exploratory testing and use of VisualVM, I identified three methods that needed to be refactored:
1) Class: Cell Method: toString()
2) Class: MainPanel Method: runContinuous()
3) Class: MainPanel Method: convertToInt()

All of these methods contained unnecessary loops, that I was able to eliminate. For onvertToInt(), I fixed a bug that would cause negative numbers to crash inside the loop. This did not change the functionality of the program, because the numbers passed to convertToInt() are always positive. If not, the program crashes before it reaches this method.

I then focused on testing by coming up with different scenarios to test. Here are the scenarios I came up with:

MainPanel convertToInt():
1) Test 100 random numbers from 0-Integer.MAX_VALUE
2) Test the number 0
3) Test the number Integer.MAX_VALUE

Cell toString()
1) Test a dead cells string
2) Test an alive cells string
3) Test a been alive cells string

MainPanel runContinuous()
1) (Unit Test) Test a configuration that takes multiple iterations to reach a stable state.
-I did this via a unit test, where I checked all of the background colors of the cells. This involved creating a thread to run the runContinuous() method. I ended up switching to manual pinning tests, as this test is highly dependent on the testers machine running through the appropriate steps in a reasonable amount of time.
2)(Manual Test) Test a spinner
3)(Manual Test) Test a configuration that takes 1 iteration to reach a stable state.
4)(Manual Test) Test the default configuration, where no cells are alive.

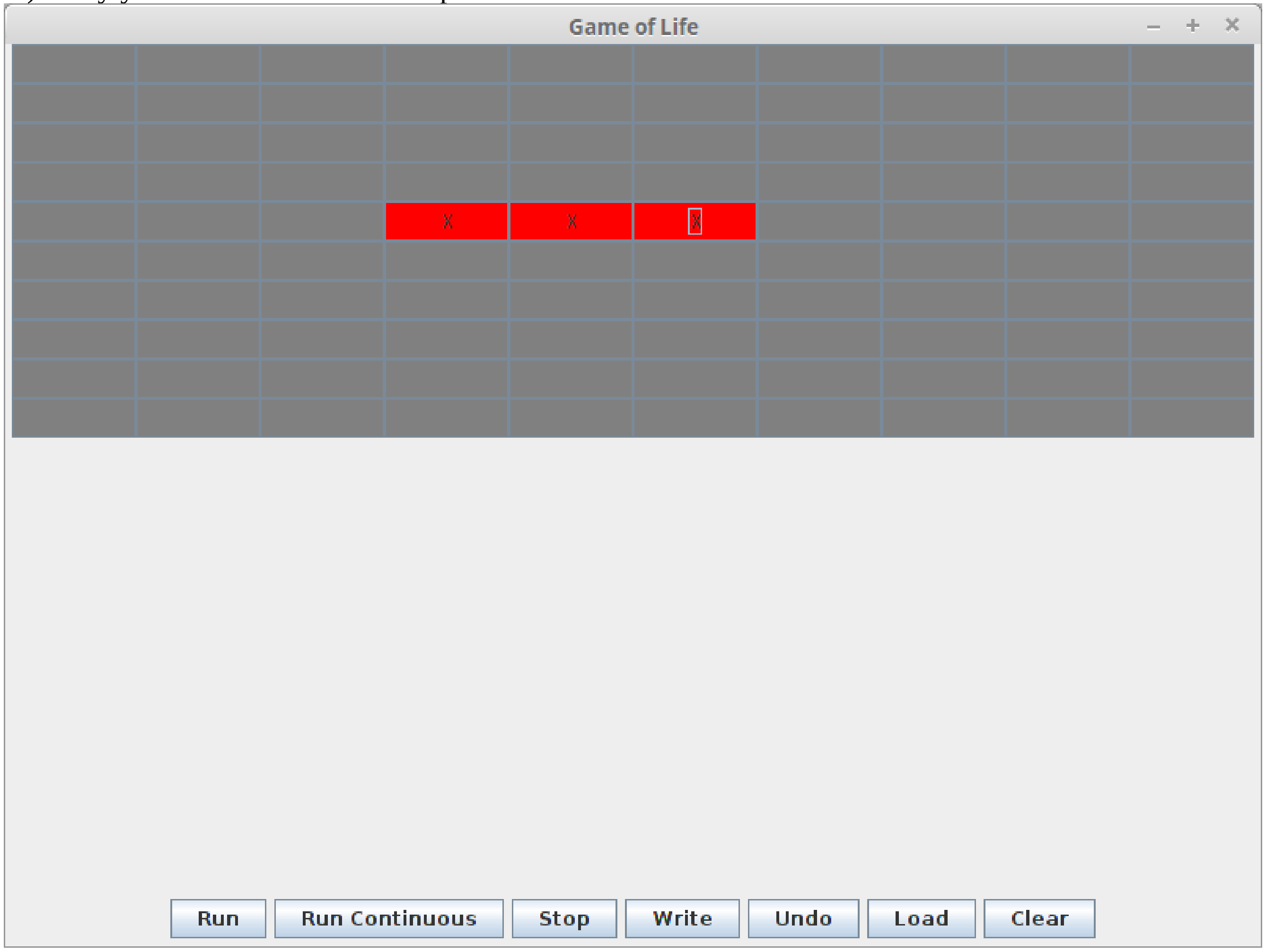
Manual Unit Tests:

Test Name: Run-Continuous-1

Description: Test to see if the run continuous function in main panel is equivalent to the old version of run continuous

Test Setup:
1) Start the Game of Life with an argument of 10
2) Verify the window that pops up has a button that says “Run Continuous”
3) Verify the window has a 10x10 grid of gray cells

Test Case:
1) Go 5 rows from the top, and 4 cols from the left.
2)Click the 4th, 5th and 6th cell from the left in the 5th row from the top.
3)Verify your window matches the picture below:

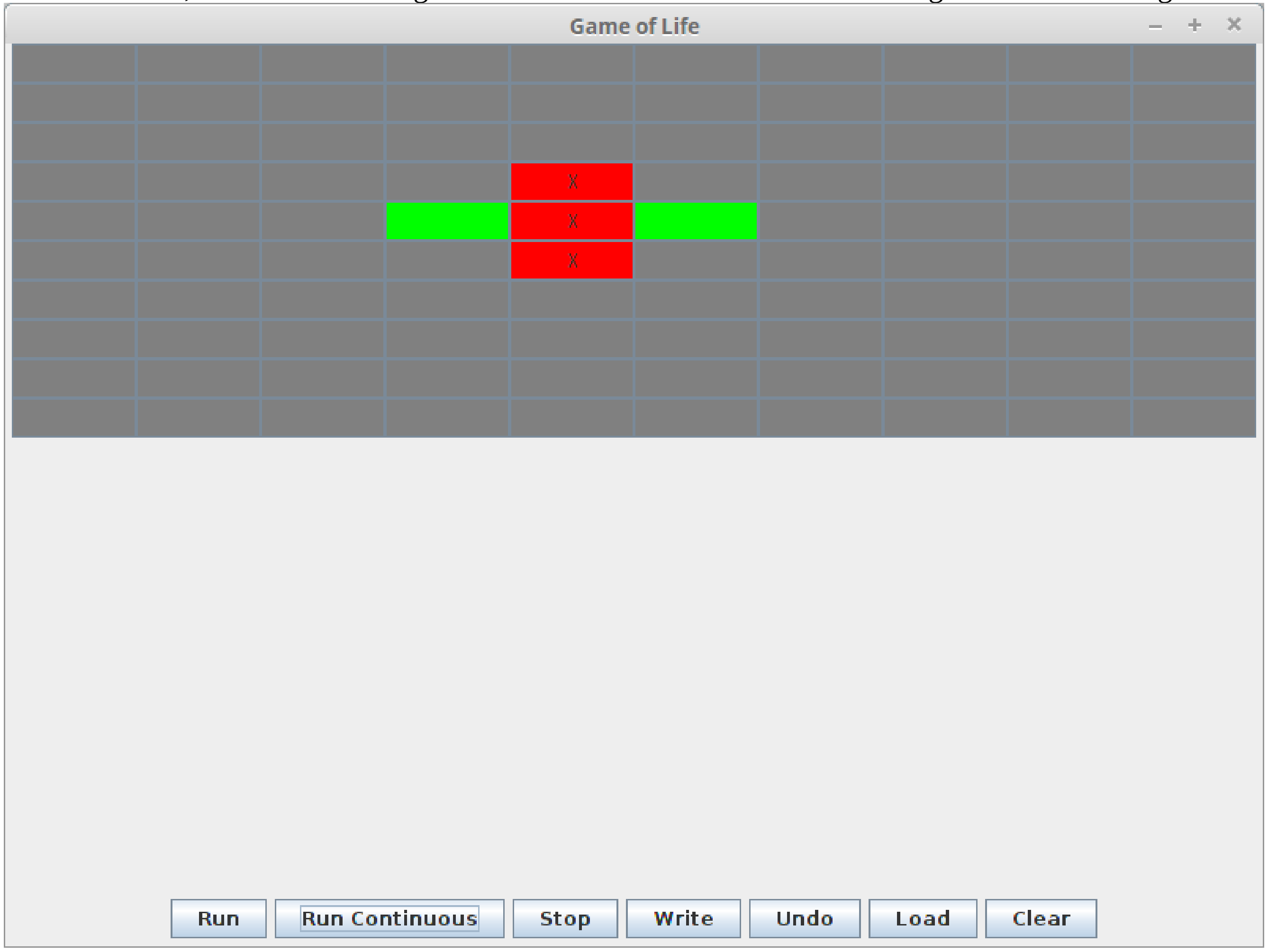


4) Press the run continuous button

Results/Validation
1) Verify the console output is continuously displaying the following text

```
“Running
      Calculating..
      Displaying...”
```

2)Verify the the 10x10 grid shows a “spinner” that looks like the below window. Note that the Middle Cell should be a red cell with an “X”. Verify the cell on top of the middle cell, the cell on the bottom of the middle cell, and the left and right cell of the middle cell are all alternating between red and green



Test Name: Run-Continuous-2

Description: Test to see if the run continuous function in main panel is equivalent to the old version of run continuous, by testing a system that results in a stable configuration after one iteration

Test Setup:
1) Start the Game of Life with an argument of 10
2) Verify the window that pops up has a button that says “Run Continuous”
3) Verify the window has a 10x10 grid of gray cells

Test Case:
1) Click any cell, and verify it is red with an 'X'.
2)Verify that there is only one red cell with an 'X'
3) Press the run continuous button
Results/Validation
1) Verify the console output is continuously displaying the following text
```
“Running
      Calculating..
      Displaying...”
```

2)Verify the cell that was once red is now green, and not changing colors. Verify no other cells are changing colors.

Test Name: Run-Continuous-3

Description: Test to see if the run continuous function in main panel is equivalent to the old version of run continuous, by testing a system that has no alive cells

Test Setup:
1) Start the Game of Life with an argument of 10
2) Verify the window that pops up has a button that says “Run Continuous”
3) Verify the window has a 10x10 grid of gray cells

Test Case:
1) Press the run continuous button
Results/Validation
1) Verify the console output is continuously displaying the following text
```
“Running
      Calculating..
      Displaying...”
```

2)Verify that no cells are changing colors.