**CS 1632 – DELIVERABLE 5: Performance Testing Conway’s Game of Life**

JavaLife – https://github.com/marcelo-dalmeida/CS-1632-DELIVERABLE-5-Performance-Testing-Conway-s-Game-of-Life

Marcelo d’Almeida

1. **Introduction**

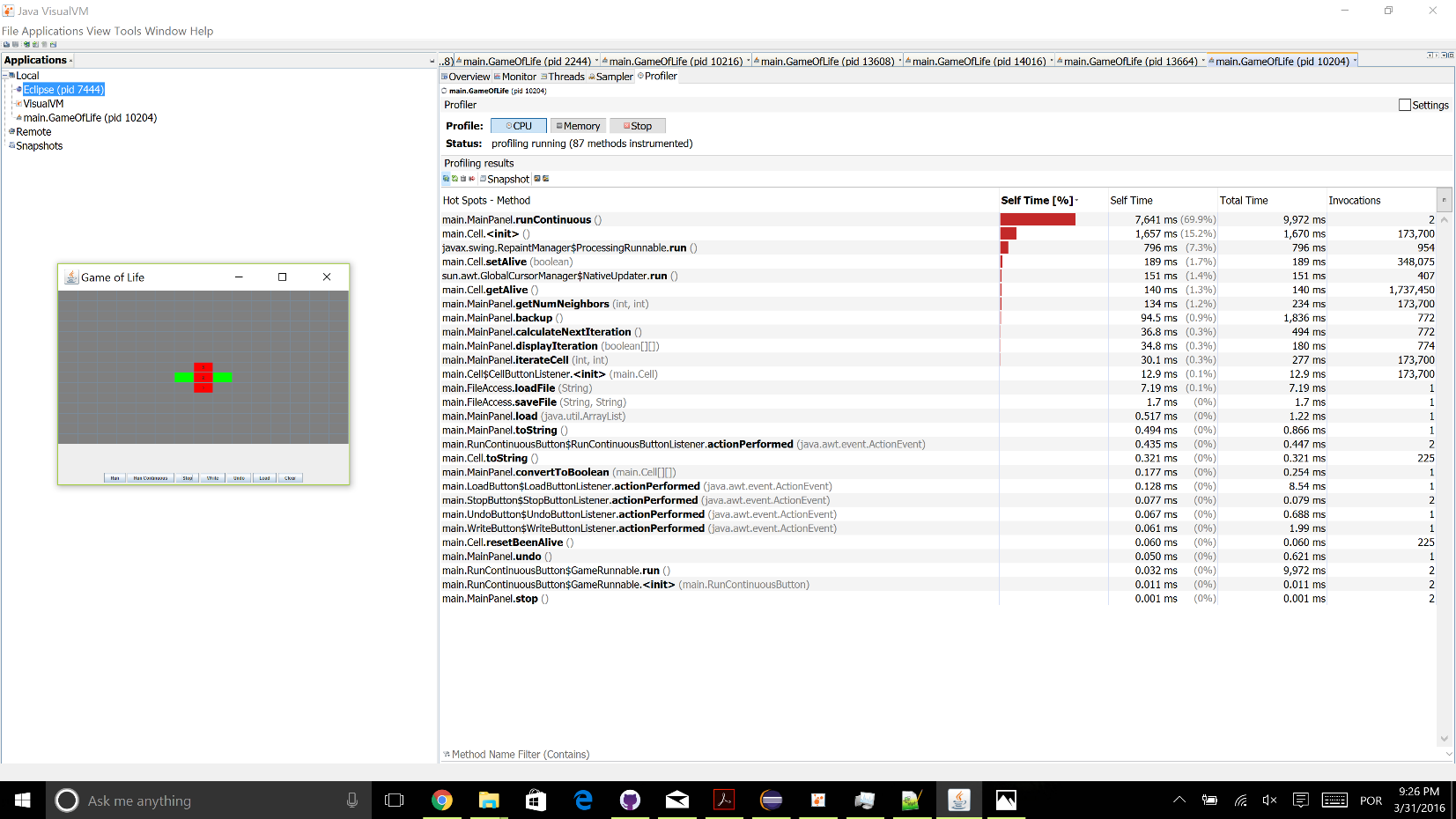
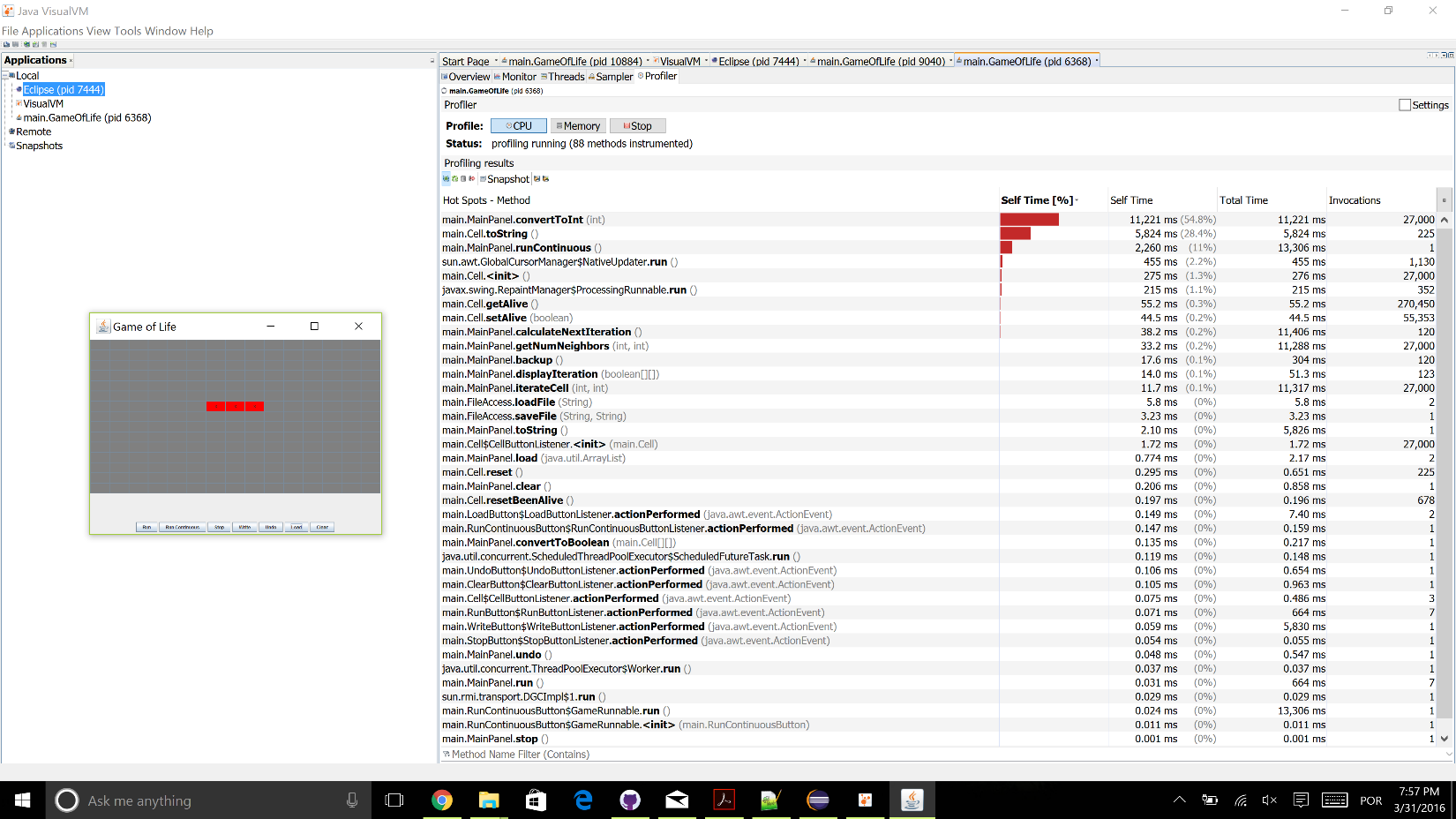
After installing and configuring VisualVM on eclipse IDE, it waas simply as just run the application with the VisualVM launcher and select profiling.

Then the execution of all common functionalities of the Conway’s game of life application, like choosing a pattern (e.g. 3-vertical shape) and hitting ‘Run Continuously”, “Stop”, “Save”, etc. with the intention of simulate an actual user execution of the program.

After that, it was easy to verify which functions were using a lot of CPU and investigate them. After creating the pinning tests (automated and manual testing) and refactoring the actual code, running the profiler again showed that the CPU usage behavior has changed.

https://github.com/marcelo-dalmeida/CS-1632-DELIVERABLE-5-Performance-Testing-Conway-s-Game-of-Life

1. **Profiler Run**



1. **Manual Testing**

**IDENTIFIER**: 1a

**TEST** **CASE**: Base case; running continuously with 3-vertical shape.

**PRECONDITIONS**: The following grid positions: (1, 2); (2, 2); (3, 2) are marked as alive cells.

**INPUT VALUES**: Press ‘Run Continuous’ button.

**EXECUTION STEPS**: After filling the grid with the 3-vertical shape, press ‘Run Continuous’ button when the window is ready.

**OUTPUT VALUES**: The game starts to run indefinitely.

**POSTCONDITIONS**: The cells form an infinity pattern changing between 3-vertical shape and 3-horizontal shape.

**IDENTIFIER**: 2a

**TEST** **CASE**: Base case; running continuously with 4-block shape.

**PRECONDITIONS**: The following grid positions: (2, 2); (2, 3); (3, 2); (3, 3) are marked as alive cells.

**INPUT VALUES**: Press ‘Run Continuous’ button.

**EXECUTION STEPS**: After filling the grid with the 4-block shape, press ‘Run Continuous’ button when the window is ready.

**OUTPUT VALUES**: The game starts to run indefinitely.

**POSTCONDITIONS**: The cells stand still in a 4-block shape formation.

**IDENTIFIER**: 3a

**TEST** **CASE**: Base case; running continuously with 7-horizontal shape.

**PRECONDITIONS**: The following grid positions: (2, 2); (2, 3); (2, 4); (2, 5); (2, 6); (2, 7); (2, 8) are marked as alive cells.

**INPUT VALUES**: Press ‘Run Continuous’ button.

**EXECUTION STEPS**: After filling the grid with the 7-horizontal shape, press ‘Run Continuous’ button when the window is ready.

**OUTPUT VALUES**: The game starts to run indefinitely.

**POSTCONDITIONS**: The cells expand for 19 iterations and then stand still.

Those are to exemplify that neighborhood is correctly computed and therefore ‘convertToInt()’ is not necessary.

**IDENTIFIER**: 1b

**TEST** **CASE**: Base case; running once with 3-vertical shape

**PRECONDITIONS**: The following grid positions: (1, 2); (2, 2); (3, 2) are marked as alive cells.

**INPUT VALUES**: Press ‘Run’ button.

**EXECUTION STEPS**: After filling the grid with the 3-vertical shape, press ‘Run’ button when the window is ready.

**OUTPUT VALUES**: The game run 1 iteration.

**POSTCONDITIONS**: The cells changes to a 3-horizontal shape.

**IDENTIFIER**: 2b

**TEST** **CASE**: Base case; running once with 4-block shape

**PRECONDITIONS**: The following grid positions: (2, 2); (2, 3); (3, 2); (3, 3) are marked as alive cells.

**INPUT VALUES**: Press ‘Run’ button

**EXECUTION STEPS**: After filling the grid with the 4-block shape, press ‘Run’ button when the window is ready.

**OUTPUT VALUES**: The game run 1 iteration.

**POSTCONDITIONS**: The cells stand still in a 4-block shape formation.

**IDENTIFIER**: 3b

**TEST** **CASE**: Base case; running one with 7-horizontal shape

**PRECONDITIONS**: The following grid positions: (2, 2); (2, 3); (2, 4); (2, 5); (2, 6); (2, 7); (2, 8) are marked as alive cells.

**INPUT VALUES**: Press ‘Run’ button

**EXECUTION STEPS**: After filling the grid with the 7-horizontal shape, press ‘Run’ button when the window is ready.

**OUTPUT VALUES**: The game run 1 iteration.

**POSTCONDITIONS**: The cells expand for 1 iteration.