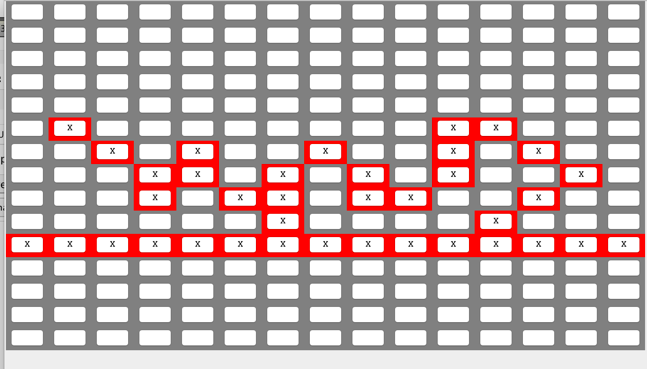
IS2545 - DELIVERABLE 5 - Performance Testing Conway's Game of Life

Name: Xin Wen

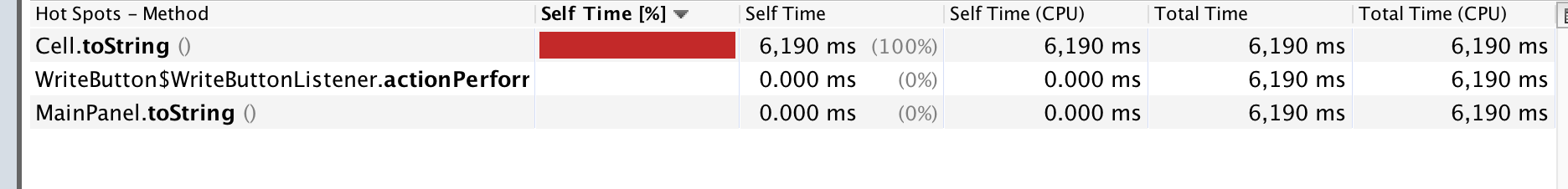
Git:

**Summary:**

In this deliverable I profiled the application with a solid pattern to control the environment. The pattern is as followed:



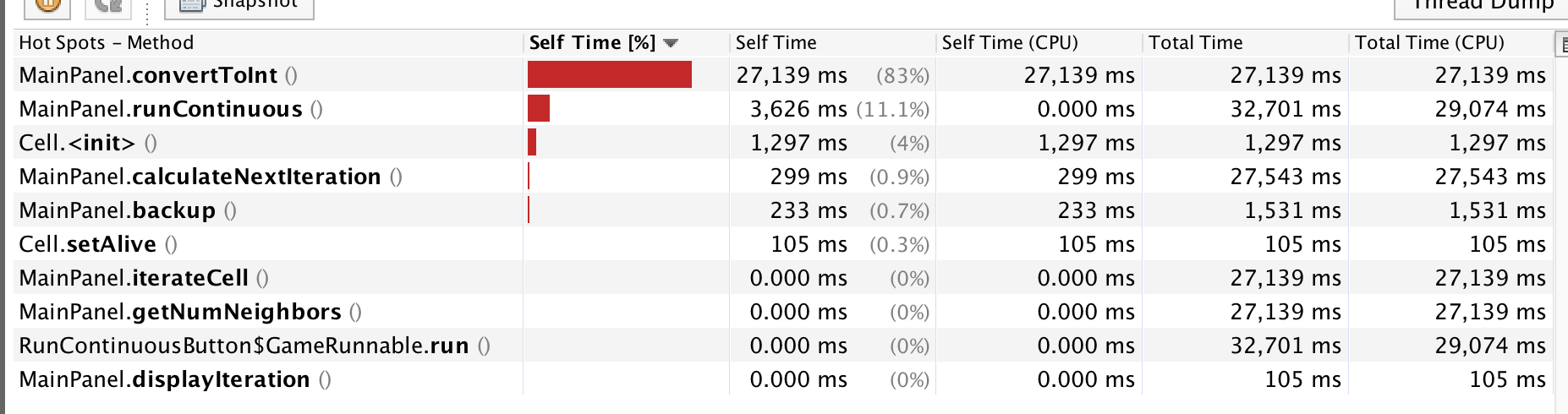
When this pattern of neighors prepared I clicked the “Write” while VM was running, then I noticed that the toString() method in Cell class costs relatively much time. So this is the thing I decided to have a look.



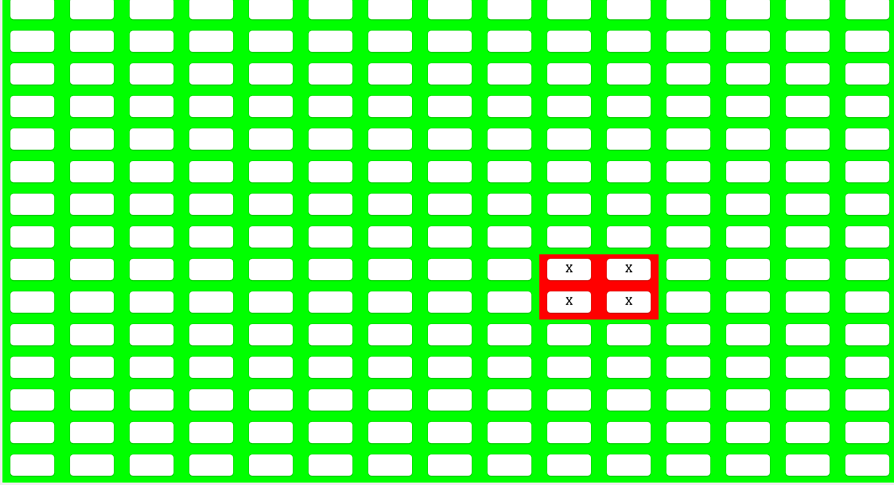
With the modified toString() method nothing appears on the VM profiler I assume that time to operate that method is low enough. In addition to that, I got the real time to operate “write” with timer and checked the result by “load” the written pattern, the function after modification runs very quick and result is not changed.

Then I tried clear, load, undo. Nothing happened in the VM, so I believe they are fine.

Then I clicked “run continuous” with this pattern and 30 seconds (time control) after it was stared I stopped the running. The profiling showed as below:

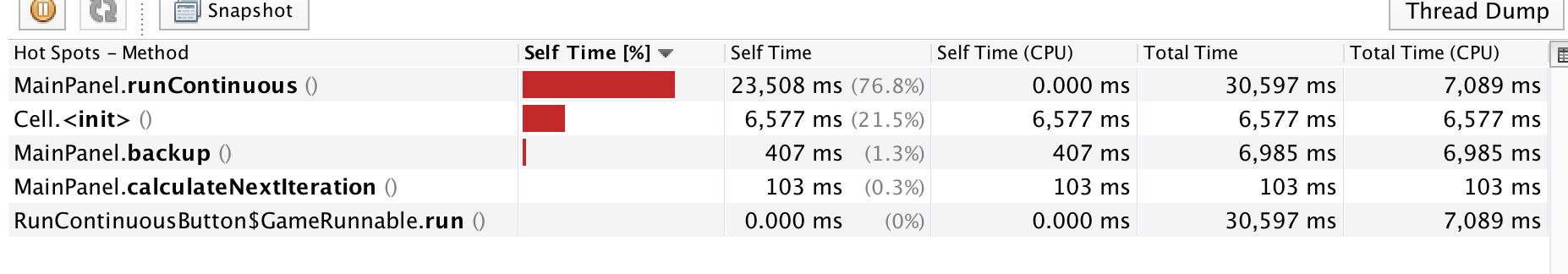


and the result shows like this:



From the profiling result, I noticed that convertToInt() and runContinuous() calculateNextIteration() and backup() are the methods I decided to look at.

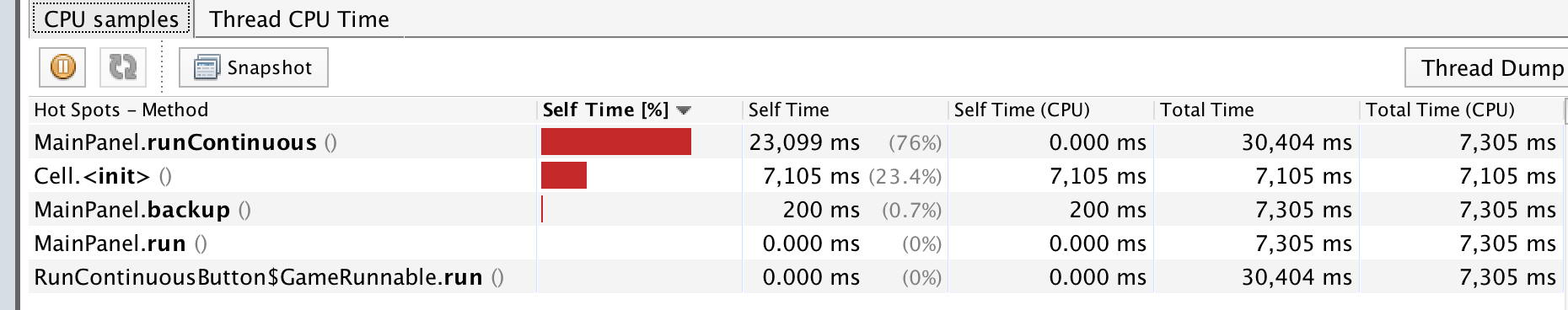
After convertToInt is modified() with deleting meaningless while loop, it turns out time for running convertToInt() method dose not make it being accounted.



Then I turned to runContinuous() method. Although it does not take much CPU time, the running time of this method could be reduced by taking the “for” loop away from the code.

I used manual test for this method since it has no return value which makes it hard to be unit-tested. After I modified the runContinuous() method time for the pattern finally becomes to that result above (picture does not change any more) comes to 9 seconds from 12 seconds.

The profiling result after runContinuous() modified:



Next I move to backup() method. However, I met a problem when I tried to use one line command to copy the two-dimensional array “\_cell” to “\_backupCell” with the for loops. It seems like the copy using System.arraycopy() is still a shallow copy, the state of alive cannot be copied to the new array. So that is the problem I met … and I did not modify backup().

When I explored the code I found toString() method in MainPanel.java also can be modified since a “if” statement is unnecessary. But it does not affect performance.

To conclude, the methods I modified are:

* toString() in Cell.java
* toString() in MainPanel.java
* convertToInt() in MainPanel.java
* runContinuous() in MainPanel.java

