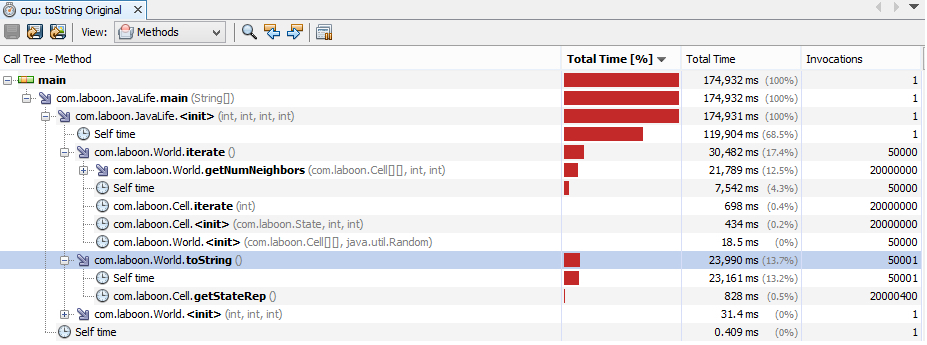
JavaLife

Michael Kudlaty

CS 1699 – DELIVERABLE 4: Performance Testing Conway’s Game of Life

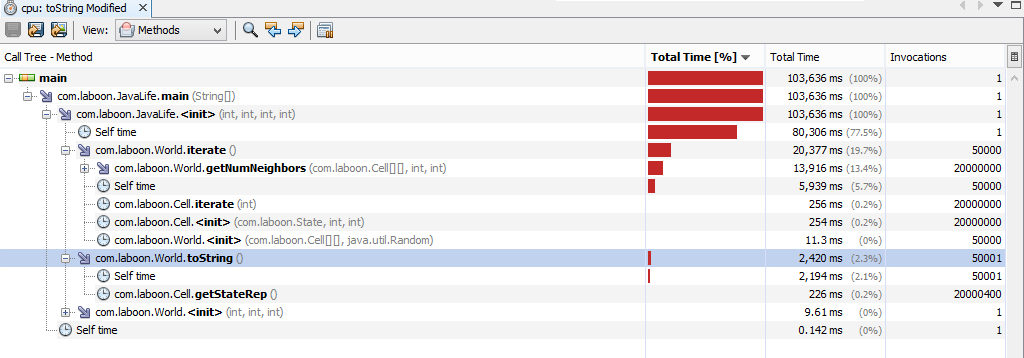
After spending sometime tinkering with the application and examining the source code, but nothing seemed explicitly inefficient. Using Netbean’s VisualVM, a profiler tool that can track CPU and Memory usage during executions, I was able to see that the program spent most of its time in the iterator and toString methods. Reexamining the source code closer, the program appeared not to be taking any extra steps in either method. The iterator method and the functions called within are expense but necessary. The toString method iterated through each of the cells in order to create a string representative of the world. However, the way the string was being constructed failed in taking advantage of tools in the java libraries meant for editing strings. The String class is a set of none mutable characters whereas the String Builder class is a mutable sequence of characters. Which means when you appending to a string, the JVM has to create a new string object each time, and this is costly. The String Builder class is able to append characters to an existing string without create a new string matching it. By switching from the String class to the String Builder the toString method went from 13.7% usage of CPU to 2.3%, a drop of about 10% with the inputs of a world of size 20, random seed number of 4, 30% of the map initially alive, and 50000 iteration (20 4 30 50000).

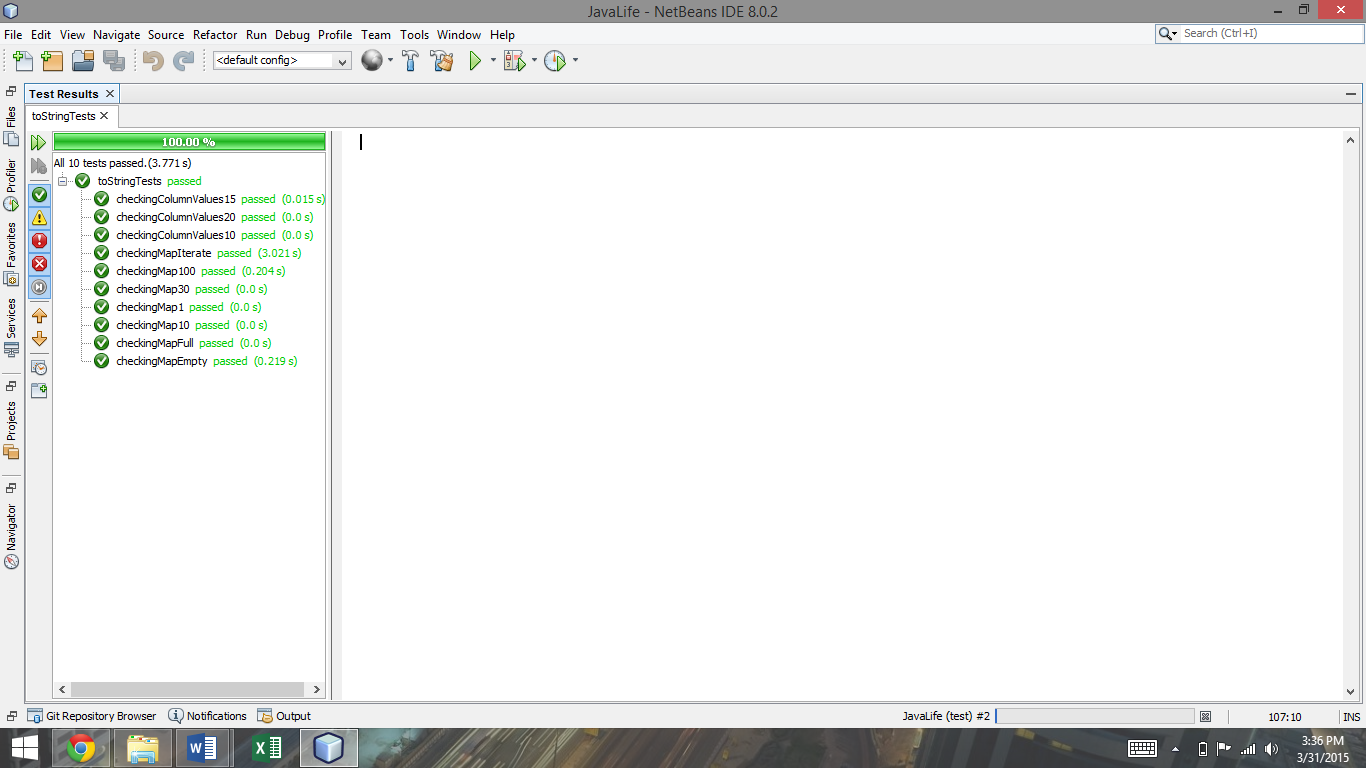
Since changes were made to the toString method, it is important to ensure that modified method produces the same results as the original. The first 3 test assert that column number were properly labeled. The next 4 tests make sure the returned string from the original toString method, now name toStringOriginal, is equal to the modified toString method using the String Builder Class. These test cases cover world with the size of 1x1, 10x10, 30x30, and 100x100. Test Cases 8 and 9 assert that the new and original toString methods are equal with an empty world and a full world. The last test case checks that the new and original toString methods are equal through a thousand iteration of the world.



toString Original

toString Modified





Junit Test Results