University of Pittsburgh

JavaLife

Dan Mahoney

CS 1699 - DELIVERABLE 4: Performance Testing Conway's Game of Life

**Summary:**

For deliverable four I choose to use “YourKit” performance and memory java profiler. I had errors inside the visualVM program when I launched the JavaLife applicaition so I choose to use YourKit.

In an effort to introduce myself to the JavaLife application I ran the program in degub mode and set a break point in the JavaLife class at line 73. I stepped into each method call to see how the applicaition worked.

After I had an idea how the application flowed I began profiling it. I set the command line arguments in eclipse to values that were testable. In particular I set the number of simulation iterations to 30000 to identify the “hotspots” of the applicaition more easily.

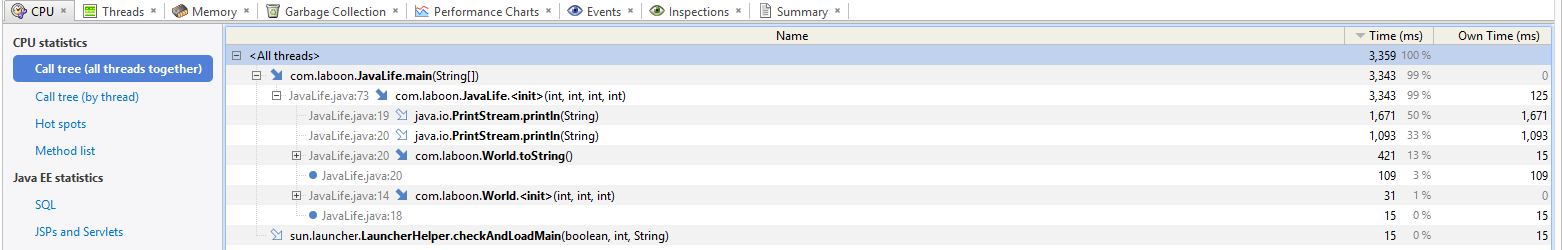
Once I had profiled the applicaition serveral times I began to analyze the results. The YourKit profile allows the user to take “snapshots” of profiling result. This made comparing different runs extremely easy. I was able to identify high memory usage and resource intensive methods in the JavaLife applicaition. The most resource intensive area was the “toString” method, which made sense because it had a doubly nested for loop and string concatenation code inside it.

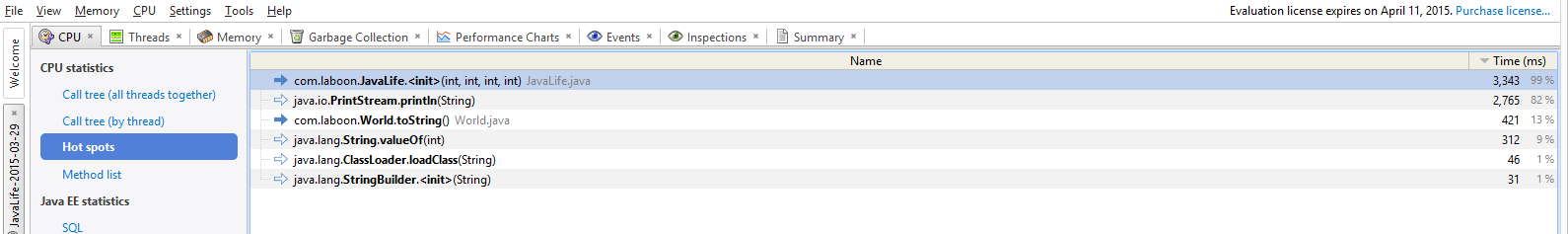
Before I began to refactor the toString method I created Junit tests so I could test my new code with the same Junit tests to ensure it produces the same results. The Junit tests consisted of edge, corned, and boundary cases. For example I created tests where every cell in the “World” was alive or dead. After my Junit tests passed I began to refactor the toString method.

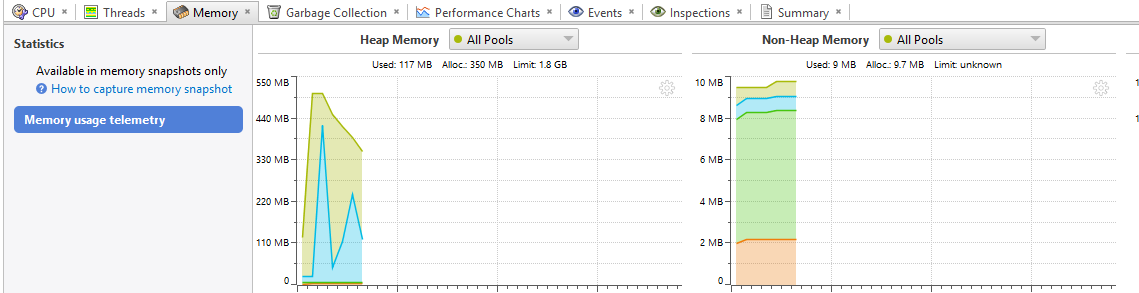
The biggest issue with the toString method was the string concatenation used throughout the method. I created an instance of StringBuilder at the beginning of the method, and appended the remaining strings to that instance. I choose to use StringBuilder because it optimized by the JVM, and would preform better with higher number of iterations.

The biggest challenge of deliverable four was analyzing the profiling results and determining the code “hotspots”. I found that once I had a deeper knowledge of the applicaition’s code it was easy to determine that the toString method was troublesome. Also discovering and utilizing the YourKit features took some time.

**Before Refactoring:** Notice the Time column time column is in the 3000 ms, and the “toString” method is included in the Hots Spots category in the second screen shot. Also the heap memory usage is high.







**After Refactoring:** Notice that the “toString” is no longer included in the Hots Spots category, and the heap memory usage is down considerably.

