CS 1632 – DELIVERABLE 4:

Performance Testing Using VisualVM

James Hahn & Ahmed Abdelsalam

Summary

* For the summary, describe how you profiled the application and determined the methods to refactor, and why you changed what you did. The summary should not be more than a few paragraphs - definitely less than a page.

For this project, VisualVM was ran immediately on different sized opinions, such as 100 or 200 so that hotspots would be much easier to see; if 5 monkeys were tested, it wouldn’t give as much of a well-rounded overview of the hotspots and CPU usage.

The methods that were chosen for refactoring were getFirstMonkey() – this was the largest CPU usage initially, then stringifyOutput(), generateId(), monkeyWithBanana(), and finally incrementRounds().

For getFirstMonkey(), it was starting from the end of the monkey list and traversing towards the front, checking the monkey number along the way to see if it was 1; and if it was, other counter variables had to be equal to 0, which required hundreds of thousands of extra operations that were unnecessary. A solution to this was to return the Monkey at index 1 (which always 1 as its monkeyNum because it’s static, starts at 0, and increments by 1 for every Monkey) in constant time.

As for stringifyOutput(), there was a loop that concatenated 50000 @ symbols to an initial return String, then it essentially wiped all of them and began creating the correct output String. The fix to this was to remove that useless for loop.

The method generateId() had 2 nested loops that would iterate a total of 100,000 times simply to add 223492 to the argument n. I removed the 2 loops and refactored the method to simply return the input value added to 223492

In MonkeySim, the method monkeyWithBanana() had a while loop that would iterate 100 times for every monkey encountered in the list of monkeys. The while loop only alters local variables that are discarded after the end of the iteration without any impact on the argument or return values.

Finally, the method incrementRounds() in MonkeyWatcher class declares a local variable (int toReturn) and iterate in a for loop 2.147 billion times then discard the value of toReturn, then increment te value of numRounds by one. We removed the for loop which improved the performance of the method.

Code: <https://github.com/kingsman142/MonkeySim>

Screenshots

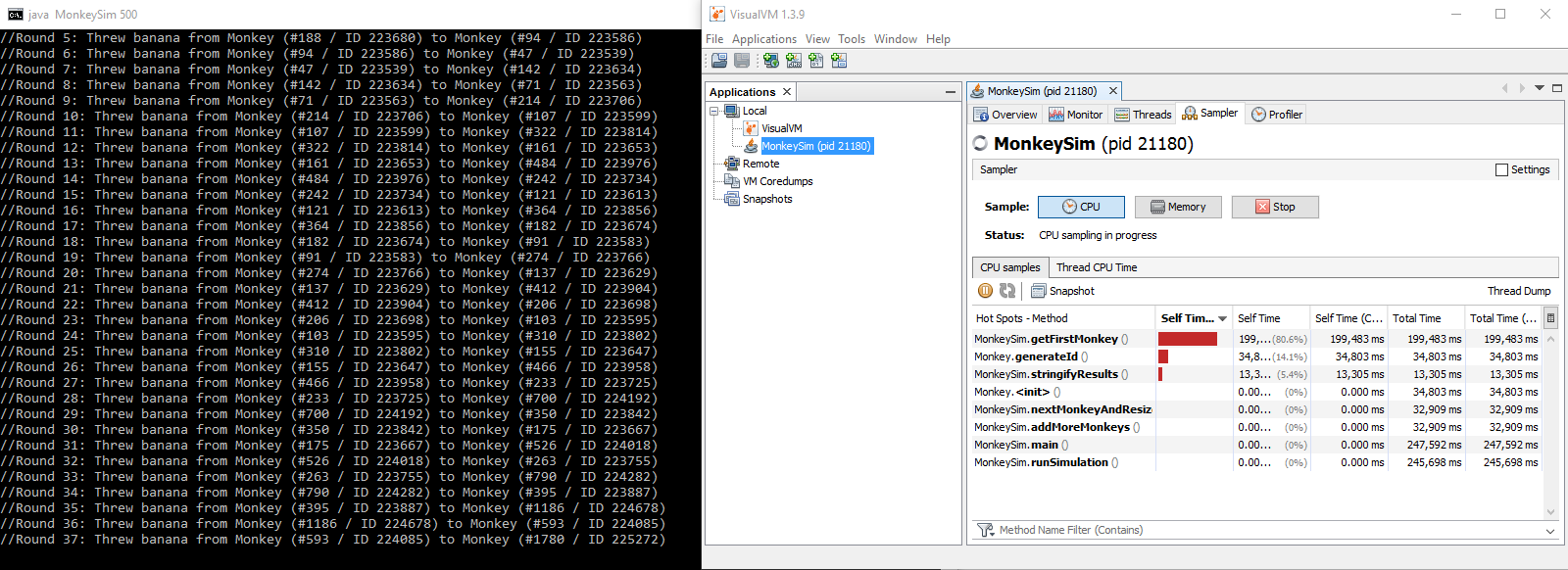
* After this, include screenshots of VisualVM (or another profiler, if you use that) both before and after the refactor. These screenshots should include the relevant sections that let you see what method to refactor.

**getFirstMonkey() :**

**BEFORE (gave up after 200 seconds for 37 rounds because I wasn’t going**

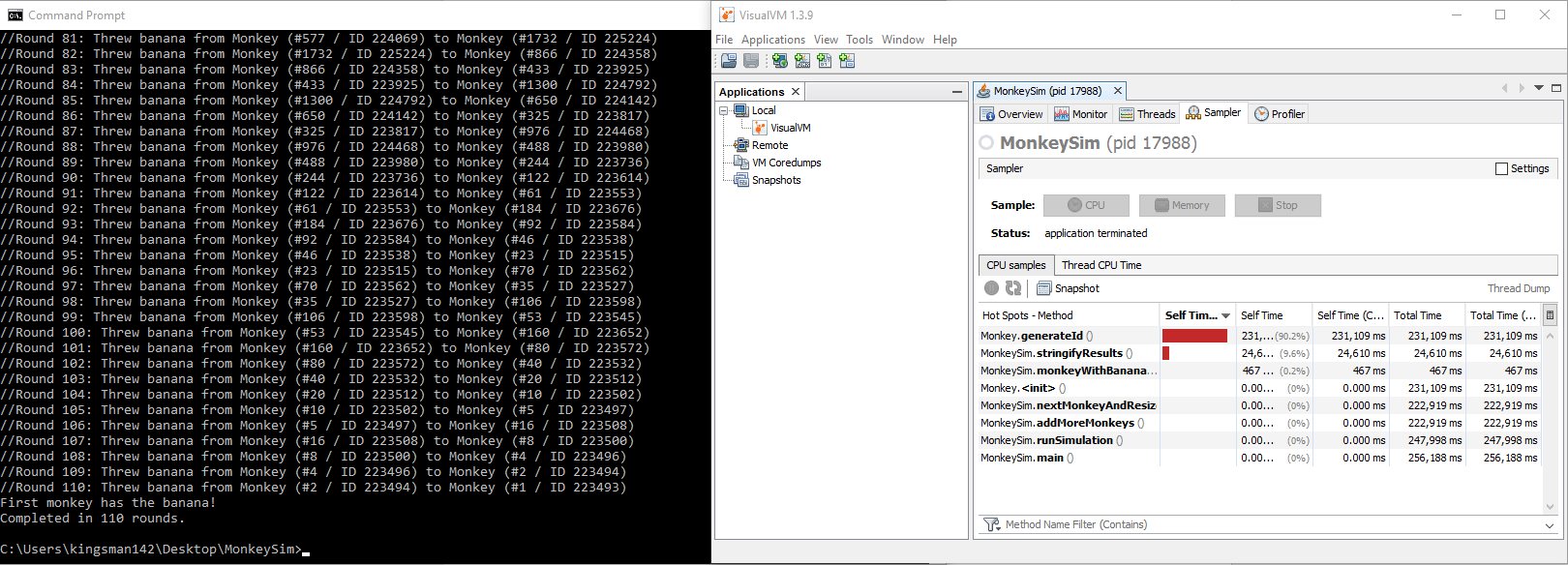
**to waste my entire day waiting for this thing):**

average time: 11.671s

**** longest time: 11.851s

**AFTER (simulated with 500 monkeys, hotspot is no longer existent):**

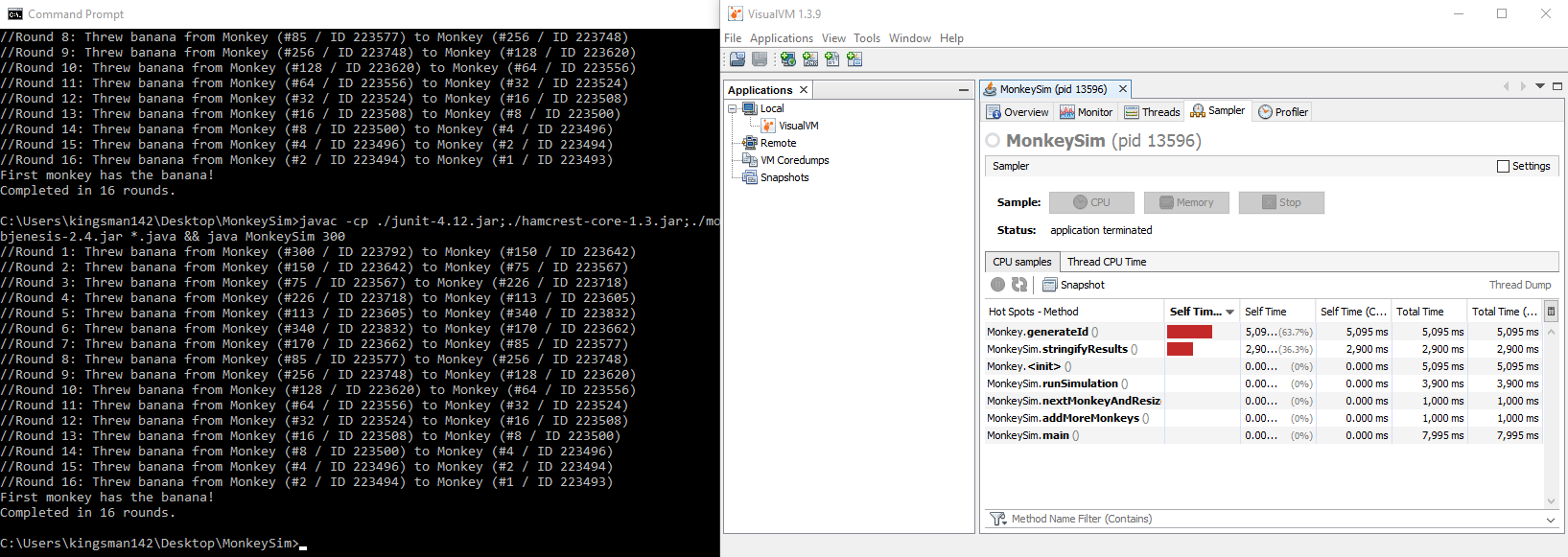
average time: 10.324s

**** longest time: 10.465s

**stringifyResults():**

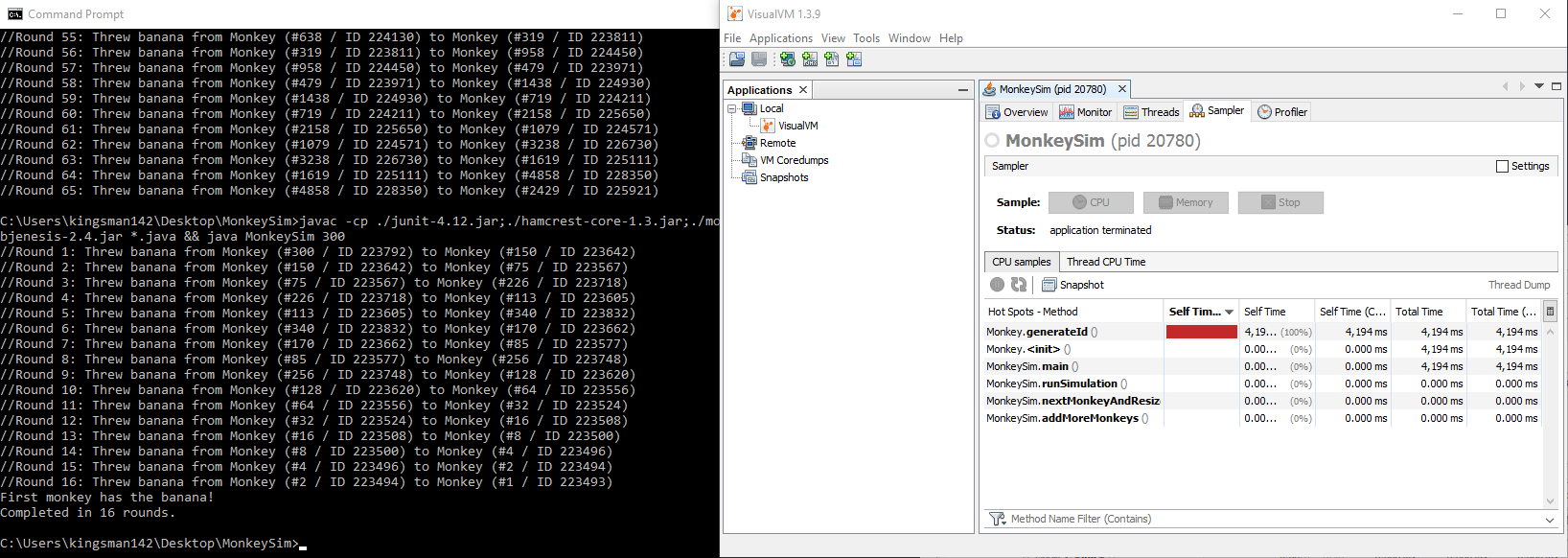
**BEFORE (300 monkeys):**

average time: 10.581s

longest time:**** 10.767s

**AFTER (300 monkeys, hotspot no longer existent):**

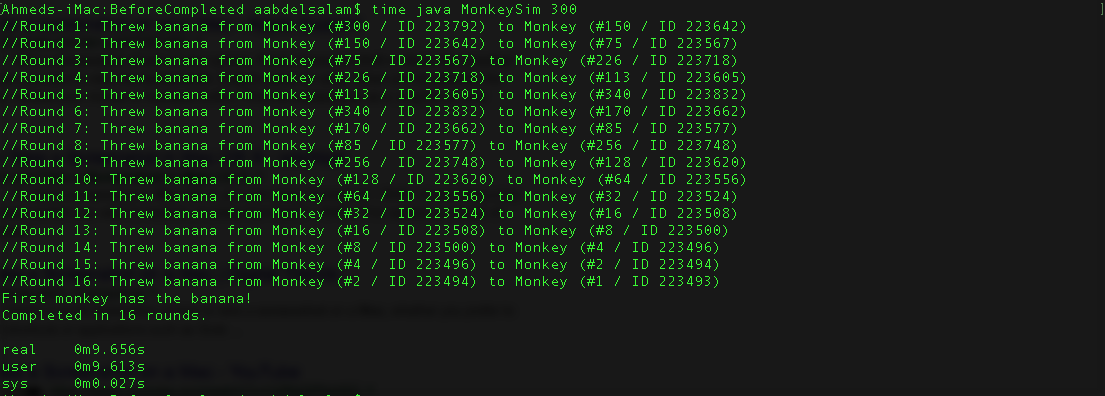
average time: 4.30s

 longest time: 4.502s

**generateId() :**

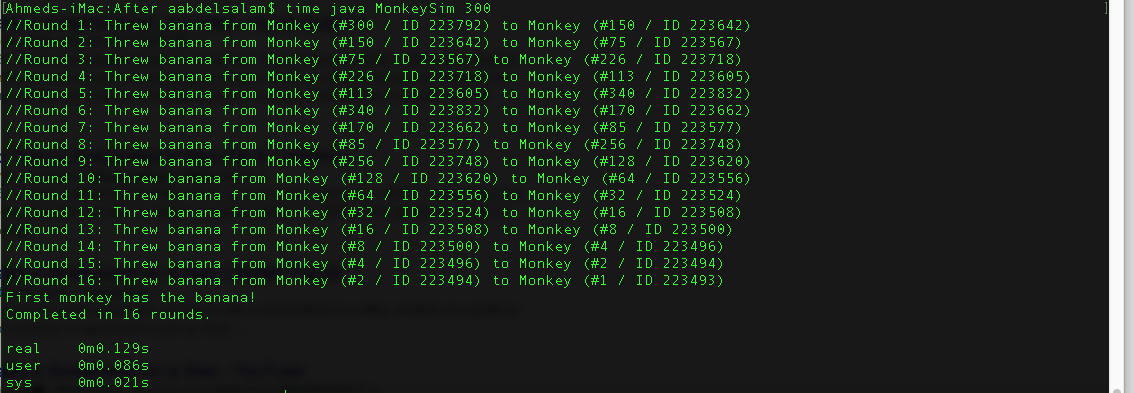
**BEFORE (300 monkeys):**

average time: 9.656s

****

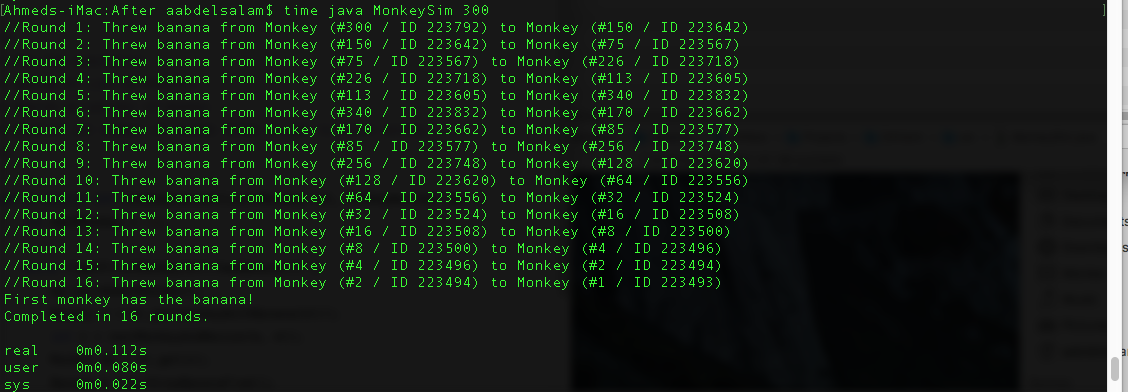
**After (300 monkeys):**

average time: 0.129sd



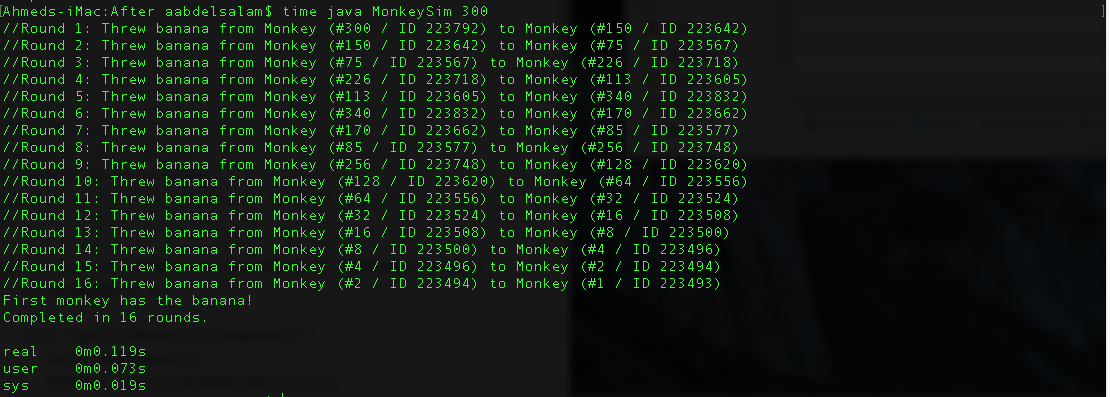
**monkeyWithBanana() :**

**BEFORE (300 monkeys):**

**** average time: 0.112s

**After (300 monkeys):**

average time: 0.110s



All tests passed.

