# Differences in Java Methods: Concatenation Effectiveness

## Specifications

The specifications for the machine used are as follows:

**Processor** 2.6 GHz Intel Core i5
**Memory** 8GB 1600 MHz DDR3
**System** Mac OS 10.10.5
**Java** SE build 1.8.0_31-b13

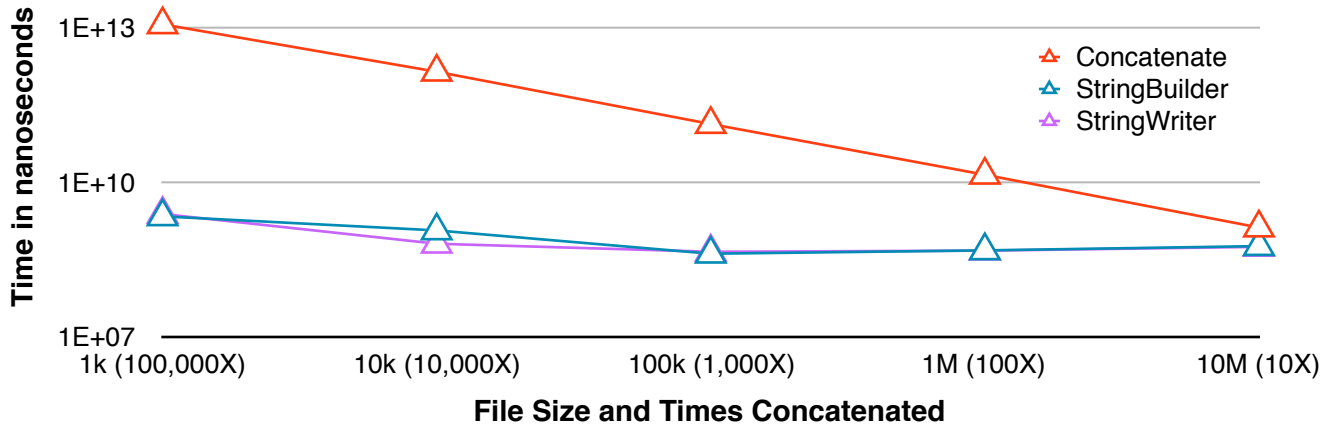
## Description

We tested three different methods of concatenation. The first, referred to solely as "concatenation", used the concatenation operon ('+') to append a string to itself a specified number of times. The second and third methods used the 'StringBuilder' and 'StringWriter' classes' append functions respectively to concatenate. The test involved inputting files of various sizes, appending them to themselves a certain number of times and measuring the time that it takes to complete the task.

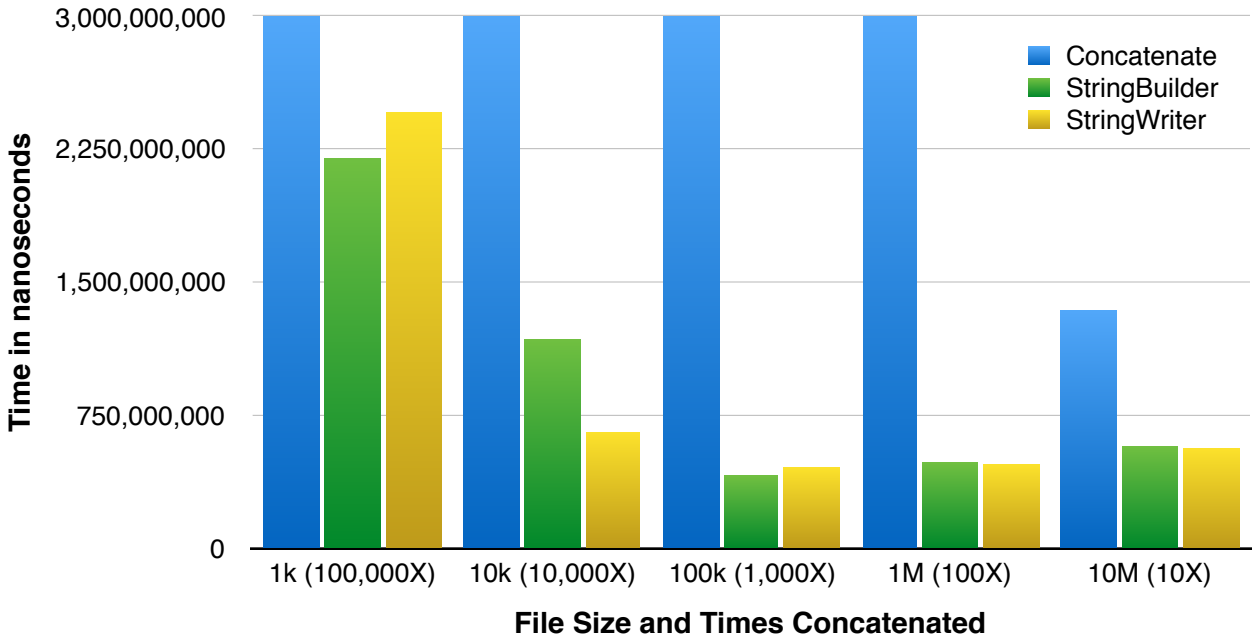
**Table 1** This table contains the time each method took in nanoseconds (*ns*) to complete each task. Tasks are described in the first column (For example, the first row refers to the time taken to concatenate a 1kb file 100,000X)

| File Size (Times Concatenated) | Concatenate | StringBuilder | StringWriter |
|---|---|---|---|
| 1k (100,000X) | 11,833,764,286,700 | 2,206,443,592 | 2,452,282,481 |
| 10k (10,000X) | 1,421,663,064,313 | 1,175,722,050 | 648,113,309 |
| 100k (1,000X) | 137,021,355,929 | 414,788,005 | 452,363,696 |
| 1M (100X) | 14,179,629,047 | 481,165,674 | 475,941,987 |
| 10M (10X) | 1,336,577,151 | 582,730,056 | 563,134,264 |

**Graph 1** This graph displays the relationship between concatenation tasks and the time taken to perform them.



**Graph 2** This graph further enhances the comparison in effectiveness between StringBuilder and StringWriter in particular. The general "Concatenate" method's measured times are larger than the y-axis for the first four tasks.




## *Discussion*

In Graph 1 and Table 1 we can see that the general "Concatenation" method is highly ineffective in terms of the time taken to complete the tasks in comparison to StringBuilder and StringWriter for all but the 10M(10X) task. This is because the general "Concatenation" method uses String objects which are immutable. Immutable objects are unchangeable, meaning a String had to be created during every iteration of the loop. The StringBuilder and StringWriter classes use mutable objects, meaning they can be changed. This meant the objects did not have to be recreated, and the same object could be appended to. This resulted in less memory waste and a faster processing time.

When comparing StringBuilder to StringWriter in Graph 2, we can see little difference in performance. StringBuilder performed more effectively than StringWriter at 2 of the 5 tests: 1k(100,000X) and 100k(1,000X). It is worth mentioned that StringWriter completed 10k(10,000X) task in almost half the time as StringBuilder.

In conclusion, both StringWriter and StringBuffer outperformed the general "Concatenation" method. StringWriter and StringBuffer differed in performance only marginally. In terms of best out of five tasks, StringWriter outperformed StringBuffer (3/5).