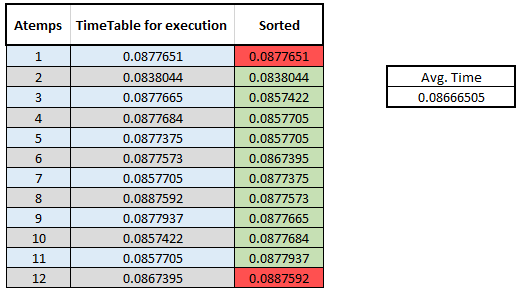
**Perfromance**

**Project**: *Code-Performance\_letter-frequencies*

**Why my performance measurements are free of errors:**

1. Task manager were examined for any programming running in the background which may affect the performance (CPU performance).
   1. Any program running is terminated.
2. Print\_tally has commented out, since the assignment state that we shouldn’t take that into considerations
3. We also want to look at the specific method and se how much this improve throughout this experiment.
4. MOST important is the MAIN method run time, which is what we’re going to conclude our result from, since that’s our program

**Documentation of the current performance:**

When all the above has been executed, we run the program and see how fast it is in its current state – we run 12 test I total. Then we sorted them from highest to lowest and remove the smallest and the largest of the numbers. Then we calculated the average time.   


Figur – Read method

Figur – Total run time

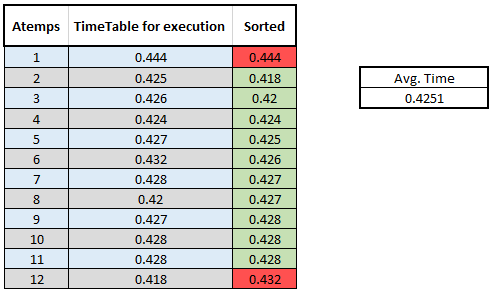
Figur 3 - Main method

**An explanation of the bottlenecks in the program.**

* Not quite sure. But I know that a ***Scanner*** is reading async, which therefor by its nature will be faster than the reader which is not reading async, but line by line (really fast).
* You could also use a ***Bufferedreader***, which has a larger (8KB) buffer compared to the Scanner (1KB) but not ASYNC! *(which is why we don’t want to use it as our first implementation)*
* Have the method being async might also give a huge performance boost

**A hypothesis of what is causing the problem, and a changed program which is improved to solve the problem.**

We’ll try to implement the **Scanner**, then do 12 more test. Afterwards we then make the method itsself async and test 12 times more.





**Total run time:**  
So we have gotten from 0.5308 🡪 0.4251 which is equivalent to (19,91%) **~20%**.  
Is was accomplished just by changing the **Reader** to an **Scanner**, which took 5min in total spent work time.

**Main Method:**  
So we have gotten from 0.08666505 🡪 0.01794816 which is equivalent to (79,29) **~80%**.

**ReadMethod:**So we have gotten from 0.07220703 🡪 0.00179899 which is equivalent to (39,13) **~40%**.

BUT!

**Main Method with BufferedReader:**  
it turned out that the more KB *(8 vs 1)* was wort it at the end, we went from 0. 08666505 🡪 0.01567679 which is equivalent to (81,91) **~82%**. Which is a little bit better than the scanner with **~80%**.

**Conclussion**

We improve the program with *20%* but the main method is up with to **82%** which we would say is the most important, since the total run time, also includes building the project *(which takes up the majority of the time)*

The other data is just for fun and could have been that we could see a relation or a pattern*. (which we didn’t..)*