**Data Structures and Concurrency**

**Continuous Assessment 1**

**Due Monday 12th November 2024**

**You will be submitting this document when you have it completed.**

**N.B. Please keep your submission brief and to the point.**

**Before submitting, you should rename this document to include your TNumber in the document name – for example T00123456 CA 1.docx.**

Submit your completed document to Canvas->Data Structures and Concurrency. (Ouriginal will be used)

Student Name \_Alperen Akcakaya\_

Submission Date \_\_12/11/2024\_\_

**SpellCheck Application (updated)**

For the SpellCheck application, various Collection classes can be used to store the dictionary of words – the words are read in from ‘words.txt’. For the CA, you may use another dictionary if you prefer – see ‘Sources of Data.docx’.

It counts the number of misspelt words found in the text you are spell checking (war-and-peace.txt). For the CA, please use **ANOTHER TEXT FILE** – other text files are available in ‘Sources of Data.docx’ or whatever source you prefer.

Use IntelliJ Profiler to generate % of time and actual time (in ms) for contains() method of your chosen Collection classes – code as given uses a LinkedList.

1. Complete the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Structure (Collection class) used to store dictionary** | **% of Time for contains() method** | **Time in ms for contains() method** | **Big Oh for contains() method**  **n – size of the dictionary** |
| LinkedList | 198 ms 13.2% | 149 ms | O(n) 98342 |
| ArrayList | 336 ms 8.2% | 293 ms | O(n) 98342 |
| HashSet | 0 ms 0% | 0 ms | O(1) 98342 |
| TreeSet | 0 ms 0% | 0 ms | O(log n) 98342 |
|  |  |  |  |

(Use as many rows as you need in above table)

1. Obtained with Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz 2.21 GHz, Java Version jdk20, Windows 10

When doing the above, try different Collection classes and see the different values you will get for the contains() method. The text file you use, to do the spell check on, must be large enough to allow you to discriminate between the different Data Structures (Collection classes) used to store the dictionary.

The Collection classes that you should use are the ones that we covered in:

OneDrive -> Data Structures and Concurrency 2023\_2024 -> “2. Java Collections Framework”

**that would be suitable to store the dictionary.**

When using ArrayList, you can do better than using the contains() method.

We will assume that the contents of the dictionary are in sorted order, so instead use the Collections binarySearch() method:

public static <T> int binarySearch([List](https://docs.oracle.com/en/java/javase/18/docs/api/java.base/java/util/List.html)<? extends [Comparable](https://docs.oracle.com/en/java/javase/18/docs/api/java.base/java/lang/Comparable.html)<? super T>> list, T key)

Change the SpellCheck.java code to handle this.

1. Explain why you would use binarySearch() method instead of contains() method for ArrayList?

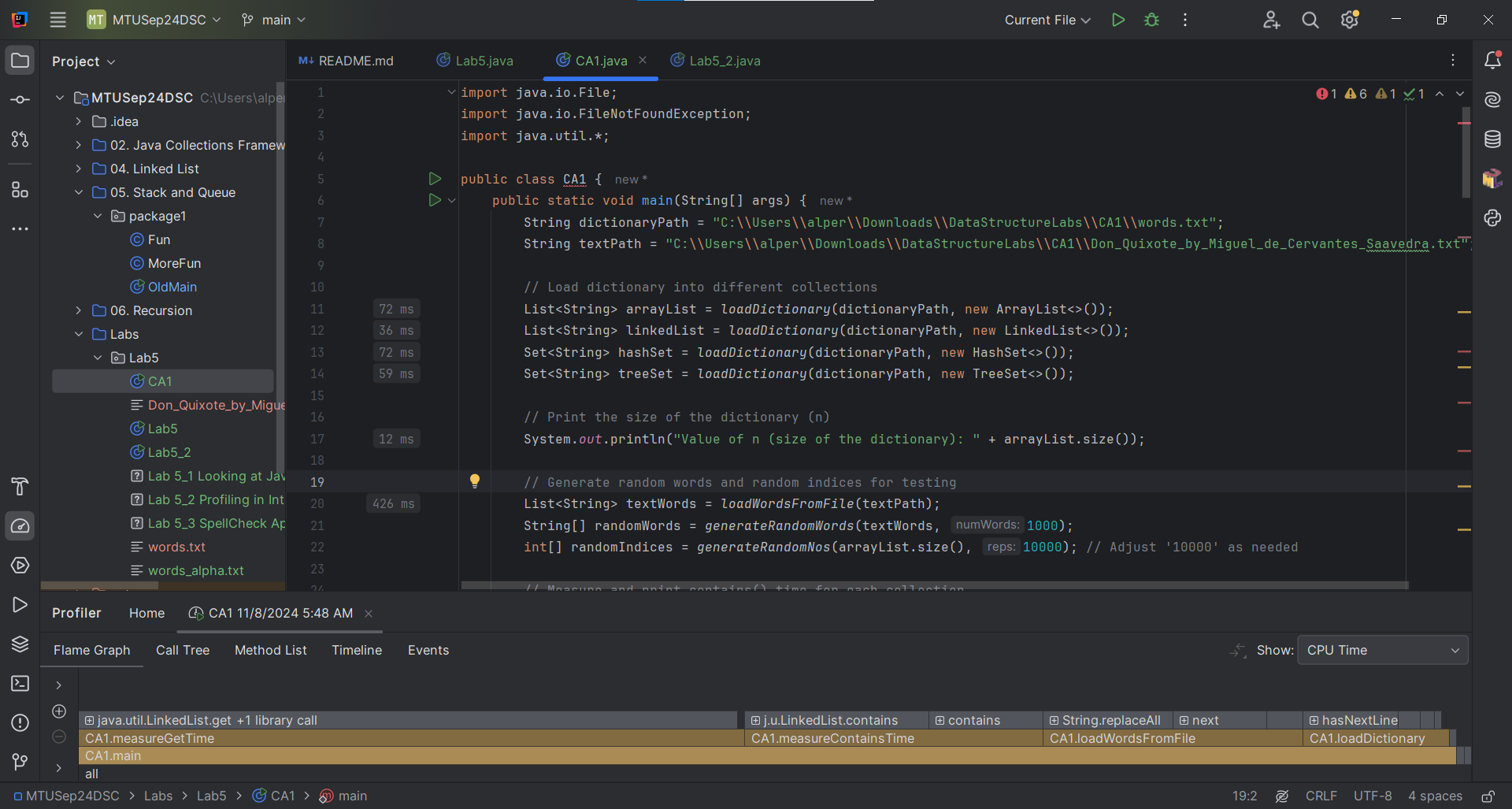
\_\_ It makes the binarySearch() method preferable when compared to the contains() of ArrayList, considering the huge difference in their time complexities-the binarySearch() having a complexity of O(logn), while it works by dividing the list and searching in halves; it is considerably fast in instances where a huge dataset is dealt with. On the other hand, the time complexity of the contains() method is O(n), which needs to iterate linearily, which could be more time-consuming. Thus, keeping in mind those applications which need to optimize the search operations-for example, spell checker projects-it will be much more effective in execution time and resource usage to apply binarySearch().\_\_

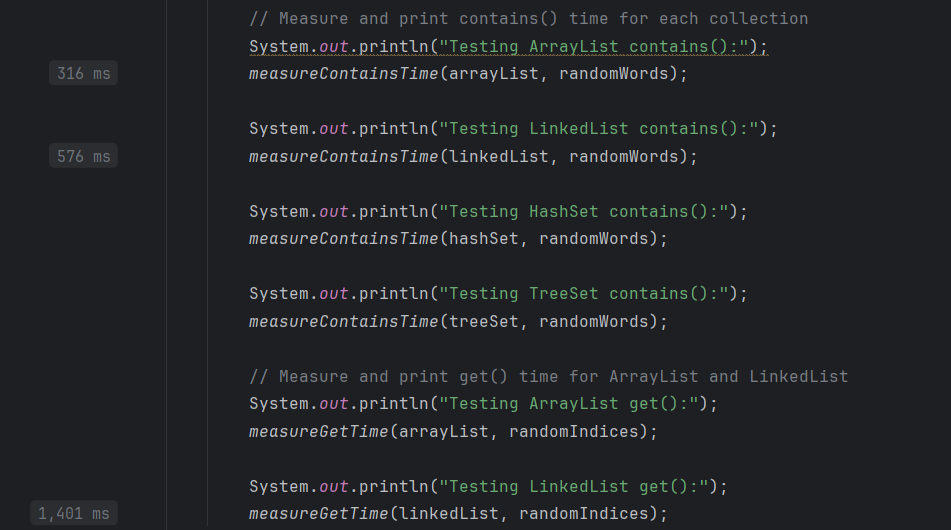
1. Specify the changes that you made to SpellCheck.java for ArrayList version

\_\_In order to use the binarySearch() method with ArrayList, we have to sort the ArrayList - binarySearch() works properly only on sorted lists. We replaced contains() by the following condition : if (Collections.binarySearch(arrayList, word) >= 0) { } The above check is for the existence of the word in the ArrayList. If binarySearch() finds the word, then would return an index greater than or equal to zero. This refinement takes advantage of the O(logn) efficiency of binarySearch() for faster searches than that provided by contains().\_\_\_

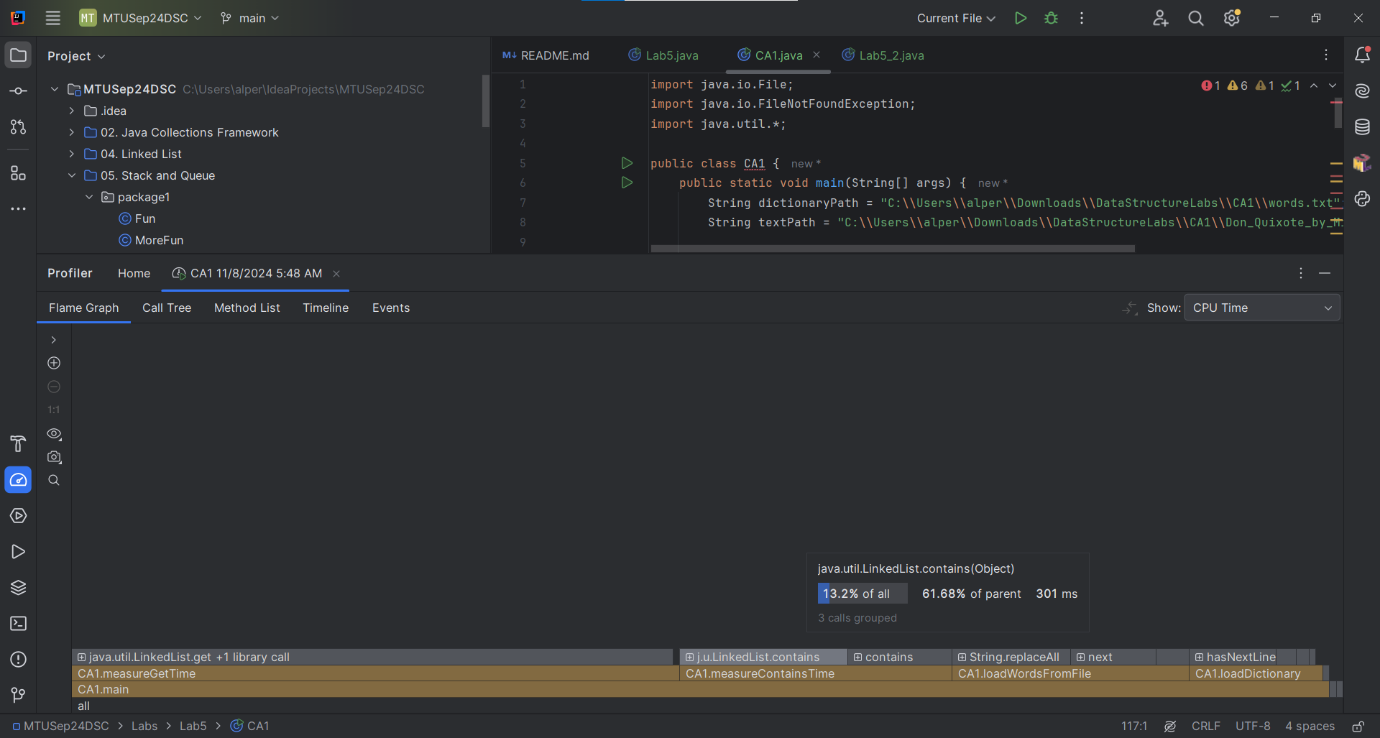
Big Oh values for the methods of Collection classes are available in the java api.

1. Give screenshots of the output from the Profiler showing the results that you have used to populate the 3rd column in the table above. You should have one screenshot for each row in the table and please give them in the same order. The screenshots should clearly show the Collection class used and the time for the **contains()** method or **binarySearch()** in the case of ArrayList.

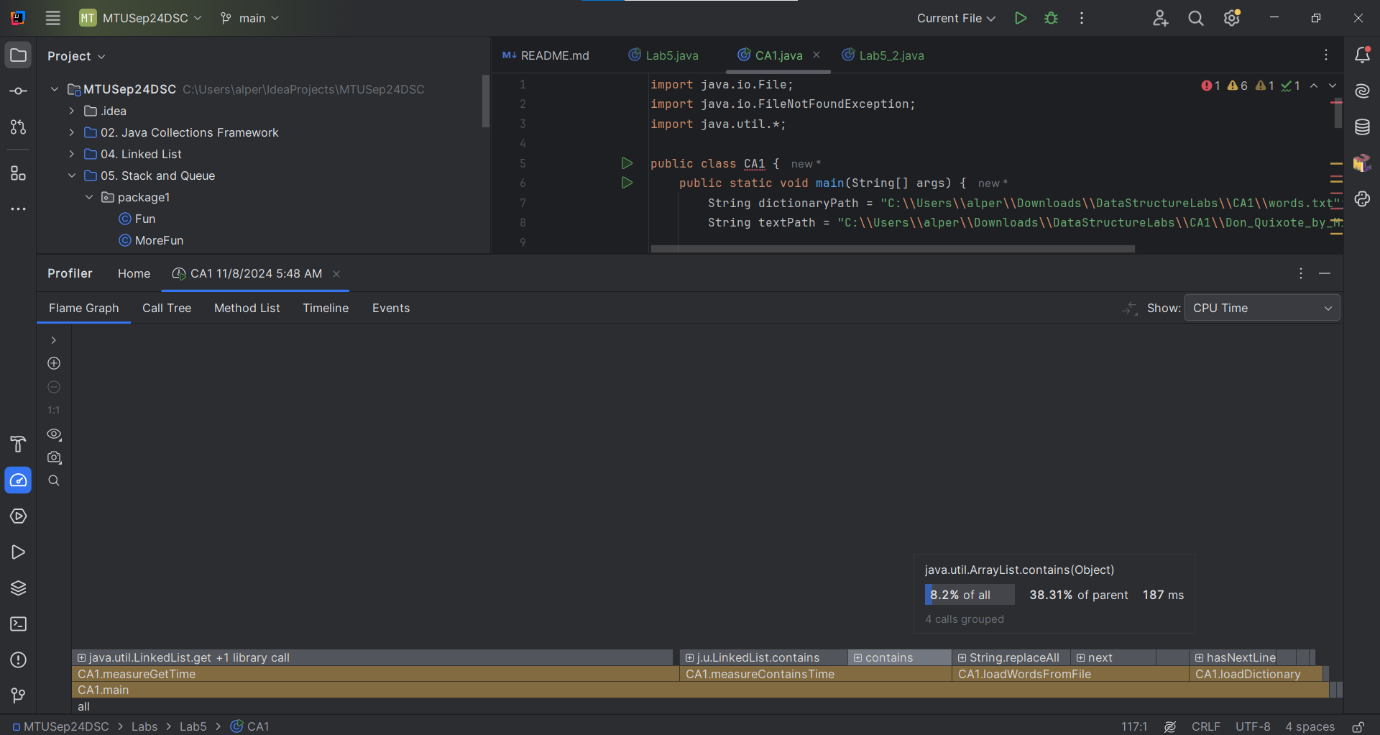
All Code



LinkedList.Contains

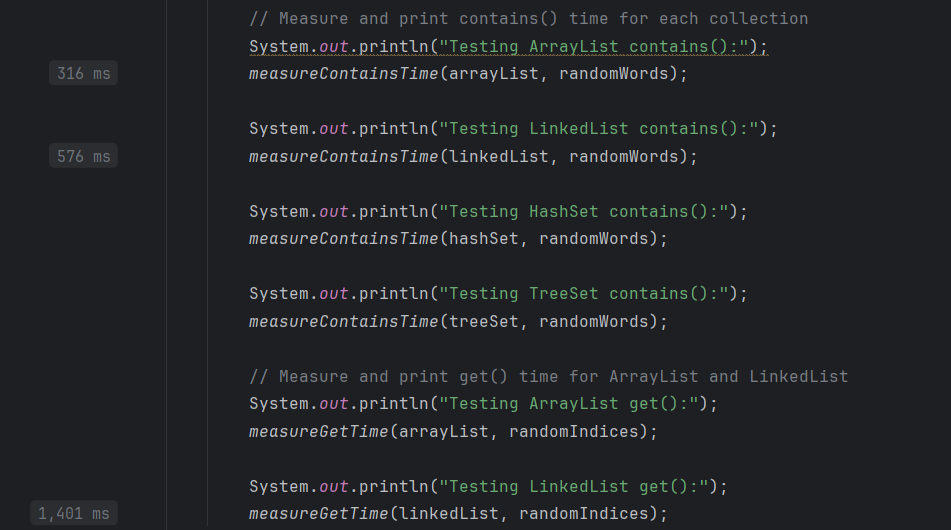


ArrayList.Contains



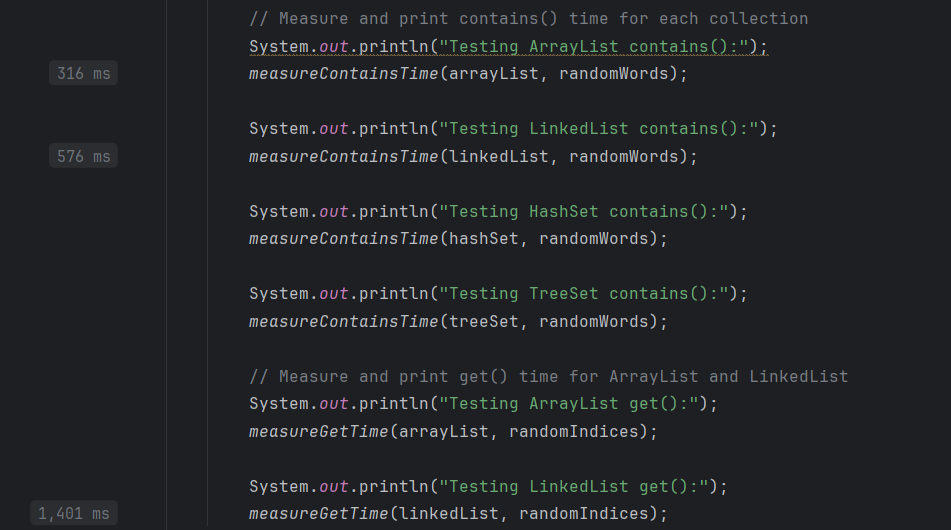
HashSet.Contains

It uses 0 ms thats why we dont see it on profiler menu but we can see it on code.

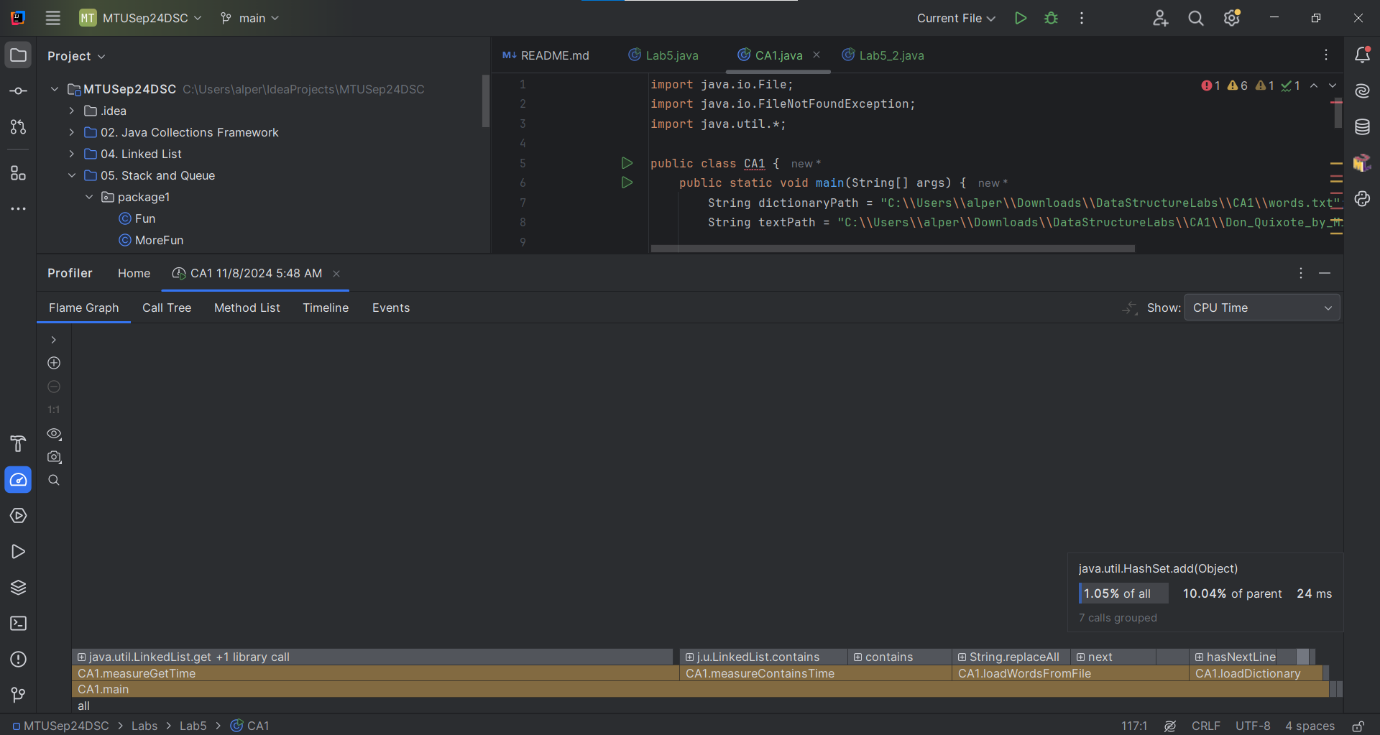


TreeSet.Contains

Same as HashSet.contains it uses 0 ms thats why we dont see it on profiler menu but we can see it on code.

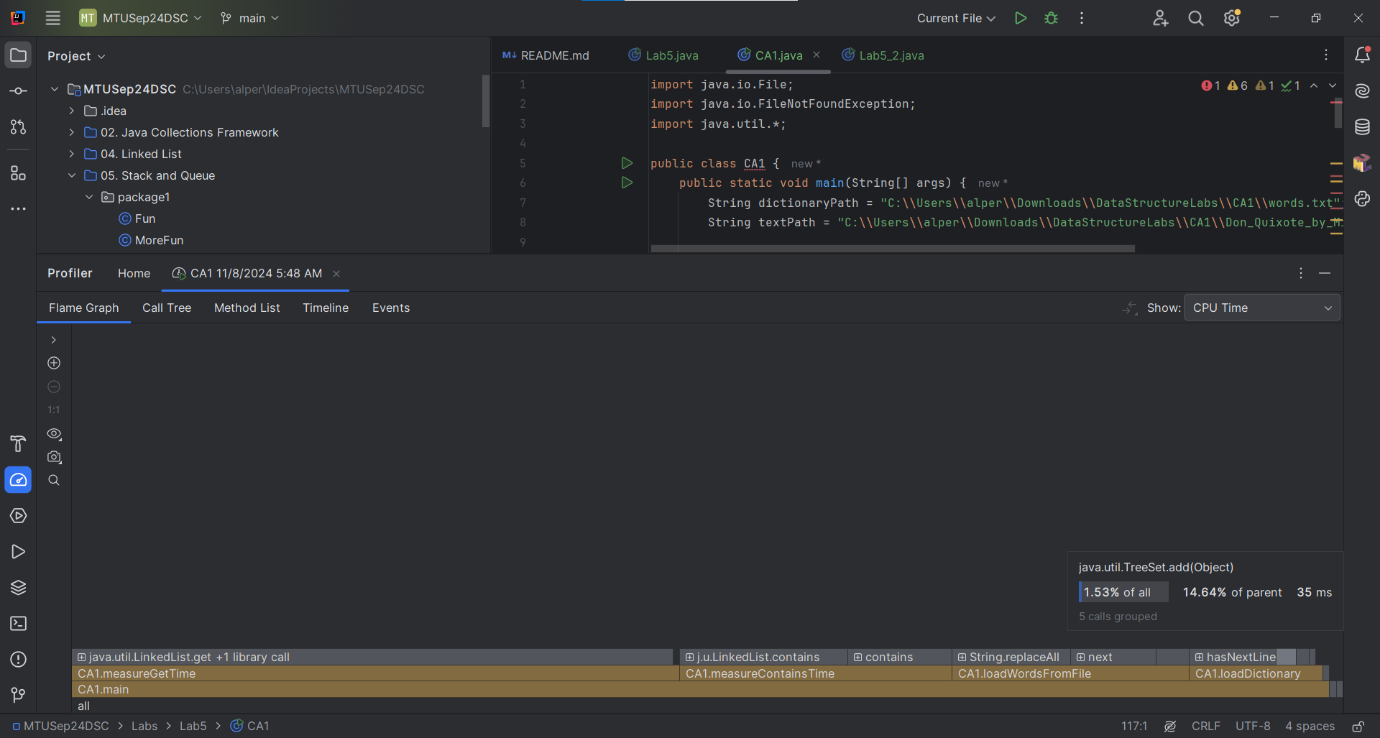


HashSet.Add  
Also we can see it from here there is no HashSet.contains only HashSet.add we have.

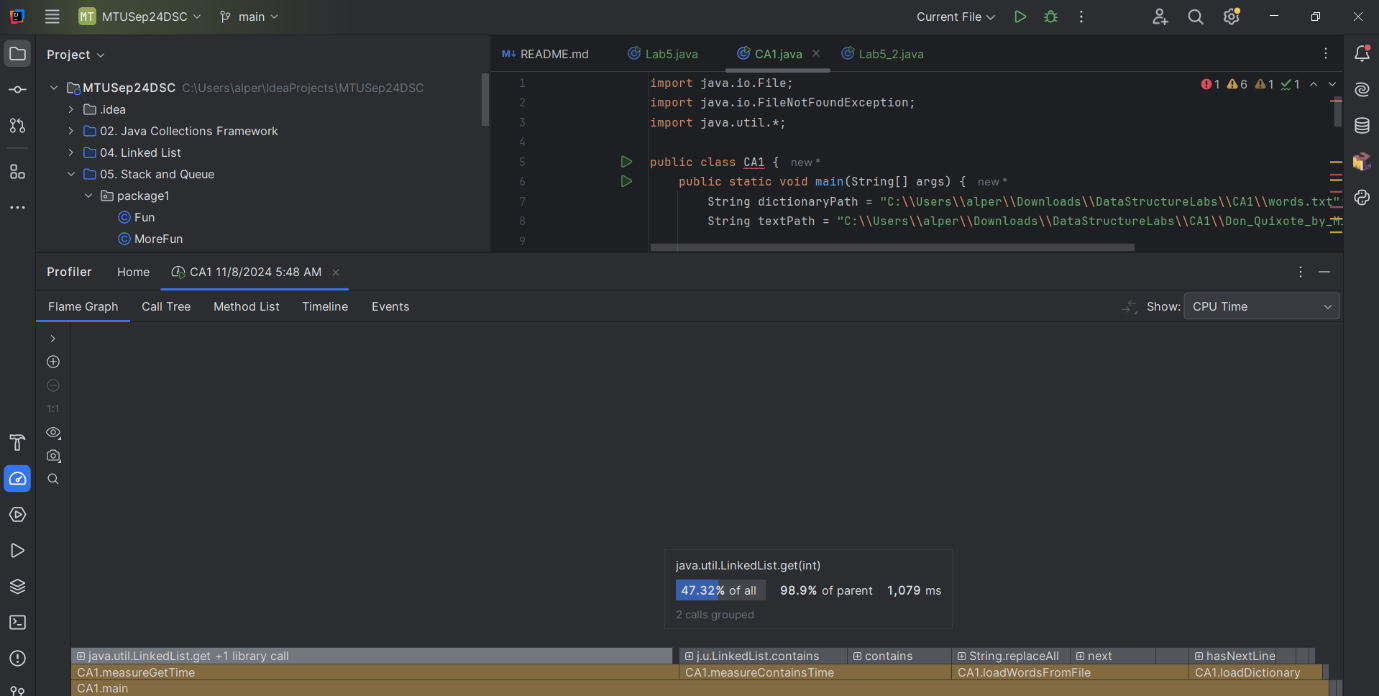


TreeSet.Add

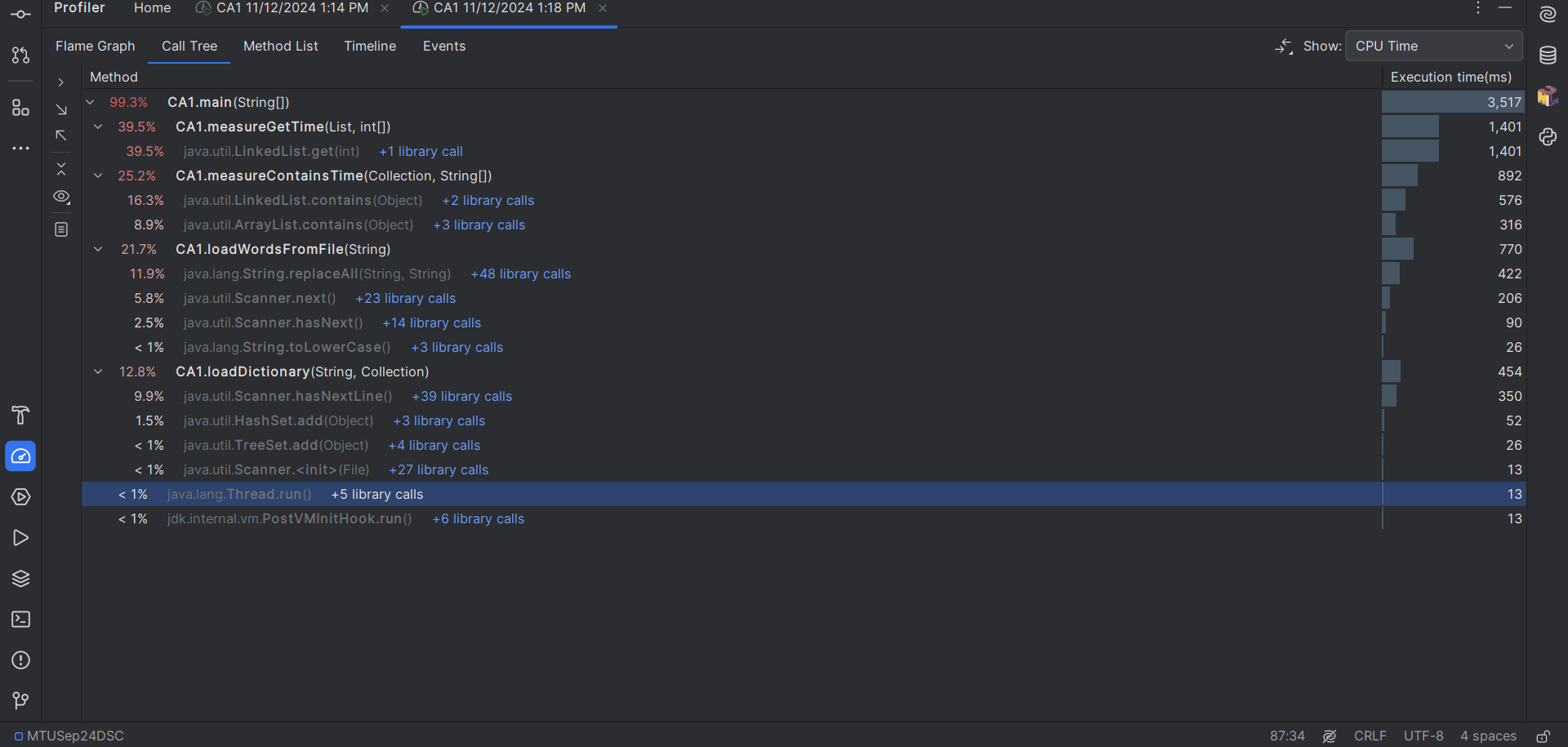
As we can see from the profiller menu there is no TreeSet.contains but only TreeSet.Add.



LinkedList.Get



CallTree



1. File used for dictionary (if you used a different one): \_\_\_\_\_ words.txt\_\_\_\_\_

(b) Source of this file (give URL): https://mtukerry.instructure.com/courses/4769/files/683620?module\_item\_id=223313

(c) Value of n (size of the dictionary) \_\_ 98342\_\_\_

(d) File on which spell checking is done: Don\_Quixote\_by\_Miguel\_de\_Cervantes\_Saavedra.txt

(e) Source of this file (give URL):­­­­­­­­­­https://www.gutenberg.org/ebooks/996.txt.utf-8

1. What Collection class would you recommend for the SpellCheck application?

\_\_ My recommendation for the SpellCheck application(CA1.java) in this project is to use the HashSet class. \_\_

1. Explain your answer

\_\_ The HashSet class is ideal for this application since it is time-efficient for big data. In this project, large files are dealt with, and hence, ArrayList and LinkedList classes are not ideal for such operations. Searches in both ArrayList and LinkedList run at a time complexity of O(n); this is really inefficient for an application which often has to look up some piece of data from some big data. These structures might result in very high consumption of memory and huge processing time, affecting performance much.  
  
On the other hand, the class TreeSet has an improved time complexity of O(logn) in its search operations but still performs worse in this application compared with HashSet. However, the average time complexity of search operations in HashSet is O(1); it is thus faster and more memory-efficient. Such nature makes HashSet particularly apt for spelling applications where retrieving a word as fast as possible can lead to great discrepancies in performance when dealing with huge data files.\_\_

1. Any other suggestions you have for improving or extra ideas for this exercise

It can be extended further to use a caching mechanism in storing frequently checked words so as to reduce repeated lookups, hence fastening the lookups. In this case, the program can quickly access words that occur more often with increased efficiency through the saving of time and memory.  
  
Concurrency through multi-threading will also ensure that the program processes really large-sized files with increased speed. This will give the application the capability of processing different portions of the text simultaneously and, hence, reduce the overall runtime while enhancing scalability towards larger datasets.

1. References/Sources of information. Specify any sources you used.

https://www.gutenberg.org/ebooks/996\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

https://mtukerry.instructure.com/courses/4769/files/683620?module\_item\_id=223313 \_\_\_

***Important Note: Plagiarism and Academic Dishonesty***

Please familiarize yourself with MTU-Kerry Campus Anti-plagiarism Policy and Procedures document – available at <http://www.ittralee.ie/en/InformationAbout/QualityAssurance/>

At this link you will see: A5 Assessment of Learners -> A5.2 Anti-Plagiarism Policy and Procedures

Please note that if the work you submit is not your own, a mark of 0 will be awarded.

**Appendix A**

**‘Declaration of Originality Form’**- MTU-Kerry.

|  |  |
| --- | --- |
| This form **must** be completed and signed and submitted with all assignments. | |
| Please complete the information below (using BLOCK CAPITALS). | |
| Name \_\_Nazim Alperen Akcakaya\_\_\_  T Number \_\_T00260896\_\_\_  Class Group\_\_\_\_KCOMP\_D\_3S\_\_\_\_\_  Assignment Title \_\_Data Structures and Concurrency CA1\_\_\_ | |
| **Students are advised to inform themselves of the University Anti-Plagiarism Policy.** | |
| **I confirm that this assignment is my own work and that I have:** | |
| Familiarised myself with the University Anti-Plagiarism Policy | a |
| Used the University’s approved referencing style throughout | a |
| Clearly referenced, in both the text and the bibliography or references, all sources used in the work  Not made use of the work of any other student(s) past or present without acknowledgement. This includes any of my own work, that has been previously, or concurrently, submitted for assessment, either at this or any other educational institution | a  a |
| Not sought or used the services of any professional agencies to produce this work | a |
| In addition, I understand that any false claim in respect of this work will result in disciplinary action in accordance with University regulations | a |
|  |  |
| DECLARATION:I am aware of and understand the University’s policy on plagiarism and I certify that this assignment is my own work, except where indicated by referencing, and that I have followed the good academic practices noted aboveSigned Nazim Alperen Akcakaya | |