**Data Structures and Concurrency**

**Continuous Assessment 1**

**Due Monday 11th 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 Anna Kovalenko

Submission Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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 | 18 | 34 | O(n) |
| ArrayList | 7 | 11 | O(n) |
| HashSet | 0 | 0 | O(1) |
| TreeSet | 7 | 11 | O(log n) |
| Queue | 7 | 11 | O(n) |

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

1. Obtained with Intel Core i7 processor, 2.8\_ GHz, Java Version \_\_22.0.2\_, Windows 11 (or specify if not…)

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.

A computer screen shot of a program

Description automatically generated

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

Using the `binarySearch()` method rather than the `contains()` method on an `ArrayList` gives some performance improvements for large, sorted lists. The `binarySearch()` is optimized for sorted collections and execute a divide-and-conquer algorithm that can detect absent elements with a time complexity of O(log n). That makes it much quicker than the `contains()` method, which is doing a linear search with a time complexity of O(n) since it goes through each element one by one. For large data, this can get quite slower and more inefficient since it will take more time.

1. Specify the changes that you made to SpellCheck.java for ArrayList version
2. I changed Set<String> dictionaryWords to ArrayList<String> dictionaryWords
3. I added Collections.*sort*(dictionaryWords) as binary search requires a sorted list.
4. I changed the readDictionary method to return an ArrayList instead of a Set.
5. I updated the search method to use binarySearch.

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.

LinkedList

A computer screen shot of a program

Description automatically generated

ArrayList

A computer screen shot of a program

Description automatically generated

HashSet

A computer screen shot of a program

Description automatically generated

TreeSet

A computer screen shot of a program

Description automatically generated

Queue

A screenshot of a computer

Description automatically generated

1. File used for dictionary (if you used a different one):

(double click to open the file)



(b) Source of this file (give URL): <https://www.randomlists.com/random-words>

(c) Value of n (size of the dictionary) 1000

(d) File on which spell checking is done:



(e) Source of this file (give URL):­­­­­­­­­­ <https://archive.org/details/cu31924011498676>

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

For the SpellCheck application, I would recommend using a HashSet to store the dictionary.

1. Explain your answer

The HashSet provides the best performance for checking membership with its contains() method, gives an average time complexity of O(1), meaning it can check if a word is in the dictionary in constant time. This is significantly faster than other collection types, such as ArrayList, LinkedList, TreeSet, and Queue, which have time complexities of O(n) or O(log n) for the contains() method. As the dictionary grows larger, the performance of ArrayList and LinkedList decreases because their contains() method performs a linear search. TreeSet has a O(log n) time complexity, which is slower than O(1). Even though ArrayList and TreeSet take similar amounts of time (11 ms), HashSet performs much better, taking 0 ms for the contains() method. When handling large datasets, the O(1) time complexity of HashSet ensures that the dictionary search remains fast and scalable.

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

1. Use a TreeSet instead of ArrayList for automatic sorting to avoid sorting the dictionary each time.

2. Add error handling for missing or invalid files.

(10) References/Sources of information. Specify any sources you used.

Canvas: [Data Structures & Concurrency-43862 (PROG71000)](https://mtukerry.instructure.com/courses/4769) materials

Book file: <https://archive.org/details/cu31924011498676>

RandomWords file: <https://www.randomlists.com/random-words>

Time Complexity: <https://www.geeksforgeeks.org/understanding-time-complexity-simple-examples/>

***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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  T Number \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Class Group\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Assignment Title\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |
| **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 |  |
| Used the University’s approved referencing style throughout |  |
| 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 |    |
| Not sought or used the services of any professional agencies to produce this work |  |
| In addition, I understand that any false claim in respect of this work will result in disciplinary action in accordance with University regulations |  |
|  |  |
| 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 \_\_\_\_\_\_\_\_\_­­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |