

Java OOP4 CA Report

Aleksandar Zoric



November 16, 2015

**CA Report – Due 11am Monday, 16th November**

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

1. (Aleksandar Zoric, 16 Nov 2015)
2. (a) *Code for height method of BinarySearchTreeMap:*

|  |
| --- |
| **public int height()**  **{**  Node node = root;  return heightSub(node);  **}**    **private int heightSub(Node currentNode)**  **{**  int height = 0;  if (currentNode == null) {  return 0;  } else {  height = Math.max(heightSub(currentNode.left), heightSub(currentNode.right)) + 1;  return height;  }    **}** |

(b) *Can this method be implemented easily without using recursion? Please explain your answer.*

***Answer:***

In my opinion, no. I imagine this because if we were going to implement this method without using recursion we would need two stacks, one for additional nodes and one always containing the current path from the root and when we see the same node on top of both stacks it means we have explored everything below it and can pop it.

(c) Test code for height method of BinarySearchTreeMap – just give the JUnit test method for this. Specify if it passed or failed. The method should contain all the code to build the BinarySearchTreeMap as well as call the size method. (E.g. add the following to an instance of BinarySearchTreeMap: “Meg”, “Rob”, “Bob”. The associated values can be anything. What height will this tree be?)

|  |
| --- |
| /\*\*  \* Test of height method, of class BinarySearchTreeMap.  \*/  @Test  **public void testHeight()**  **{**  System.out.println("height");  BinarySearchTreeMap instance = new BinarySearchTreeMap();  instance.put("Meg","0874453453");  instance.put("Rob","0854652781");  instance.put("Bob","0864325673");  int expResult = 2;  int result = instance.height();  assertEquals(expResult, result);  *// TODO review the generated test code and remove the default call to fail.*  *// fail ("The test case is a prototype.");*  **}** |

1. *Specify the name and source of the .txt file you used to create the index in IndexMain.*

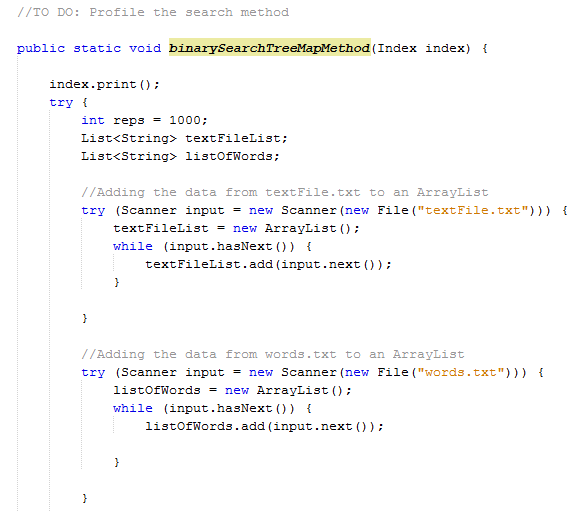
**Name:** RandomTextGenerator.com

**URL:** http://randomtextgenerator.com/

1. *Describe how you profiled the search method of Index class. Give code segments to illustrate your answer.*

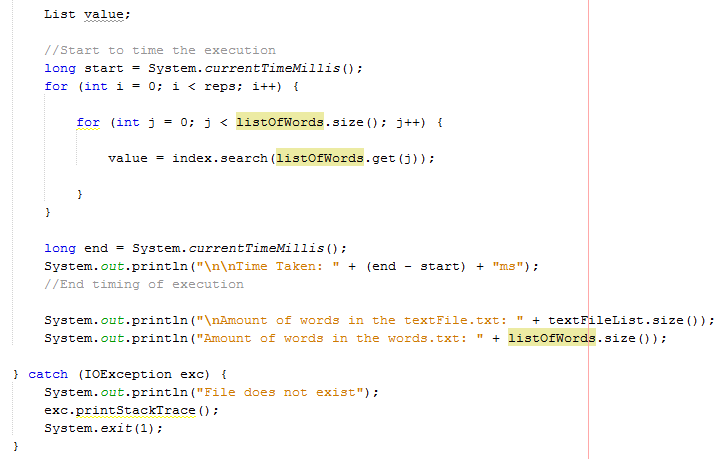
I first created a method ( binarySearchTreeMapMethod() ) in the IndexMain class which has a parameter of type ‘Index’. Inside the binarySearchTreeMapMethod() I first print my index, which prints all the words from the textFile.txt and declare a try-catch block. I then proceeded and declared my variables, I have ‘reps’ as in int, which holds a value of how many repetitions will occur, then I have two arraylists, one that will hold my large index file and one that will hold the words which I use later in the code. Once I have declared the variables, I then went and added the data from the two text files into the arraylist using a Scanner object as such:

Figure 1.1:



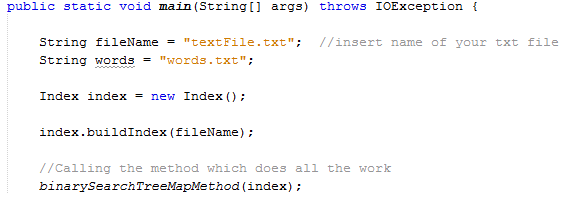
Once this section of the method was completed, I then created a nested for loop as shown in the question sheet. The outer loop loops as long as ‘I’ is less than reps and the inner loop loops as long as ‘j’ is less than the arraylist which holds the words. Inside the inner for loop I then declared a List named ‘value’ since our search() return a List. I then called the search method from the index class (which I instantiated in my main method) and whatever is returned is stored in the List I just created called ‘value’. I then declared two variables of type long named ‘start’ and ‘end’, the value that these two variables have is ‘System.currentTimeMillis()’ which I use to time the execution and print it to the user. Finally, we have a ‘catch’ section which handles our IOException in case the file is not found and then I called the size() on the arraylist which holds my large text file and the arraylist that holds the words.txt and print the amount of words that are in both text files.

Figure 1.2



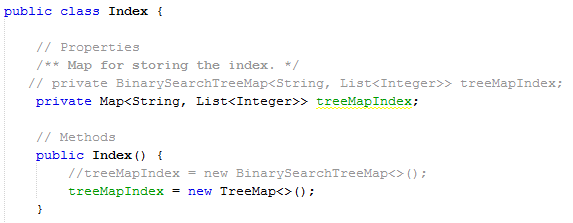
Now, once the binarySearchTreeMapMethod() was complete, in my main method I had the following: I declared two string which held the path of my two text files, I then instantiated the Index class and named it ‘index’, then we called the buildIndex() from the Index class and passed the large text file as a parameter.

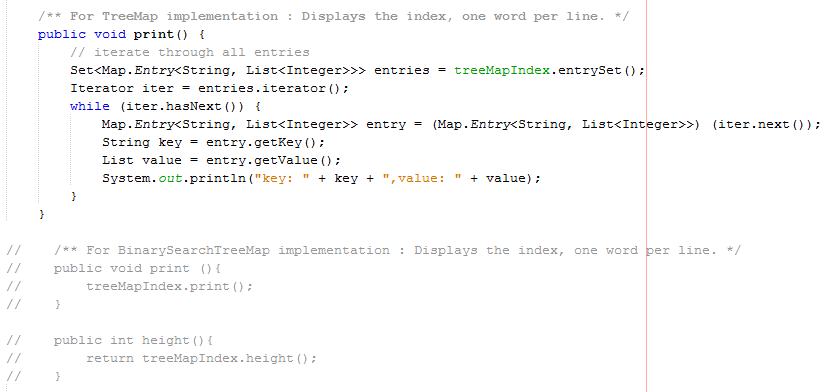
Figure 1.3



Once I ran the test using the BinarySearchTreeMap, I then went back to the Index class and commented out certain lines of code and uncommented others in order to test it using the java treeMap. As all the code was already written, it was just a matter of commenting and uncommenting to implement this.

Figure 1.4





1. *Complete the following table, giving details of profiling search method for index based on (i) BinarySearchTreeMap and (ii) TreeMap*

**Time taken for multiple searches of index**

|  |  |
| --- | --- |
| **Index Based on:** | **Time taken in milliseconds** |
| BinarySearchTreeMap | 6566 |
| TreeMap | 5744 |

Obtained with: Intel Q6600 quad core processor, 2.8GHz (OC), Java 8, Windows 10

1. *What conclusions can you draw from table:*

From the results I gathered from the table, it is obvious that the Java treepMap is faster than our BinarySearchTreeMap. In the code, I declared my ‘reps’ to a 1000, which was enough for the task given but even anything above that would produce the same result, not the same difference in time of course but the same result being that the Java treeMap is faster than our BinarySearchTreeMap.

7. *References/Sources of information*

***Words.txt file:***

Name: Street-Smart Language Learning

URL: <http://wortschatz.uni-leipzig.de/Papers/top10000en.txt>

***textFile.txt file:***

Name:RandomTextGenerator.com

URL: <http://randomtextgenerator.com/>

***Size():***

Name: Java API

URL: http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html#size%28%29

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

Please familiarize yourself with IT Tralee Anti-plagiarism Policy and Procedures document. There is a copy of the document available in blackboard under Admin- Academic Affairs and Student Supports>Policies and Protocols>Anti-Plagiarism Policy.

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