CS 570: Homework Assignment 6

1   Assignment Policies

Under absolutely no circumstances any code can be exchanged among students. Excerpts of code presented in class can be used.

Assignments from previous offerings of the course **may** be re-used. Violations will be penalized appropriately.
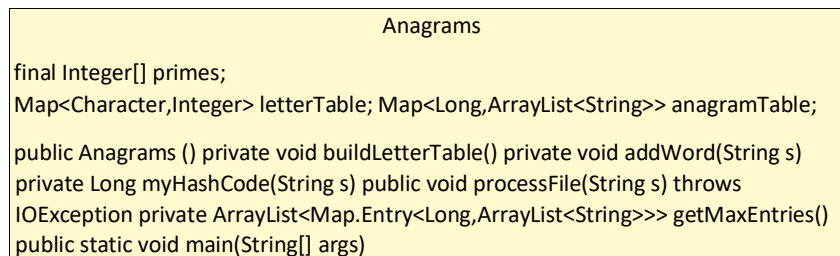
Check the course syllabus for the late submission policy.

2   Assignment

For this assignment, you will compute the list of words in a given dictionary that has the most number of anagrams. An anagram is word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. For example, "rail safety" and "fairy tales" are anagrams. In this assignment, to simplify matters, we will focus on anagrams of words formed from letters only. For example, here is a list of 15 anagrams of the word "alerts", including "alerts" itself, take from the dictionary:

<div align="center">

alerts, alters, artels, estral, laster, lastre,
rastle, ratels, relast, resalt,
salter, slater, staler, stelar, talers

</div>

This assignment requires the class Anagrams. We next describe the class through the following UML diagram:

Anagrams
final Integer[] primes; Map<Character,Integer> letterTable; Map<Long,ArrayList<String>> anagramTable; public Anagrams () private void buildLetterTable() private void addWord(String s) private Long myHashCode(String s) public void processFile(String s) throws IOException private ArrayList<Map.Entry<Long,ArrayList<String>>> getMaxEntries() public static void main(String[] args)

The constant primes should be initialized to an array consisting of the first 26 prime numbers:

```
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
    67, 71, 73, 79, 83, 89, 97, 101};
```

2

It will be used to set up the letterTable, a hash table that will associate each letter in the alphabet with a prime number. More precisely, it will associate "a" with 2, "b" with 3, "c" with 5, and so on.

The instance variable anagramTable will hold the main working hash table. Each entry in this hash table has the following format:

- Key (of type Long): the hash code of the word. It is important that this hash be the same for all anagrams of a word. The type Long is a 64-bit two's complement integer. More details on how to produce this key will be given below.

- Value (of type ArrayList<String>): a list of the words seen up until now that have Key as hash code. Note that all the strings in this list are anagrams.

We now describe each of the methods.

3   Method processFile

The main method is processFile which receives the name of a text file containing words, one per line, and builds the hash table anagramTable. For that it uses the addWord method. The implementation of

```
private void processFile(String s) throws IOException { FileInputStream fstream = new FileInputStream(s);
    BufferedReader br = new BufferedReader(new InputStreamReader(fstream)); String strLine;
    while ((strLine = br.readLine()) != null)     { this.addWord(strLine);
    } br.close();
}
```

this method is provided for you:

2

4

6

8

4   Method buildLetterTable()

This method should be invoked by the constructor for the class Anagrams and should build the hash table letterTable which consists of the following entries:

```
{a=2, b=3, c=5, d=7, e=11, f=13, g=17, h=19, i=23, j=29, k=31, l=37, m=41, n=43, o=47, p=53, q=59, r=61, s=67,
 t=71, u=73, v=79, w=83, x=89, y=97, z=101}
```

2

This table is to be used in myHashCode, described next, for constructing the hash code of strings.

5   Method myHashCode

This method, given a string s, should compute its hash code. A requirement for myHashCode is that all the anagrams of a word should receive the same hash code. With that aim, you should resort to the fundamental theorem of arithmetic. The fundamental theorem of arithmetic), also called the unique factorization theorem or the unique-prime-factorization theorem, states that every integer greater than 0 either is prime itself or is the product of a unique combination of prime numbers.

Every integer $n \geq 0$ can be expressed as a product of prime numbers: $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_n^{\alpha_n}$.

As an example, the words "alerts" and "alters" should both receive the key 2.36204078E8, if we follow the encoding of letters given above.

6   Method addWord

This method should compute the hash code of the string s passed as argument, and should add this word to the hash table anagramTable.

7   Method getMaxEntries

This method should return the entries in the anagramTable that have the largest number of anagrams. It returns a list of them since there ay be more than one list of anagrams with a maximal size. It will be called by the main method, whose code is described and supplied below.

8   Method main

The main method will read all the strings in a file, place them in the hash table of anagrams and then iterate over the hash table to report which words have the largest number of anagrams. Note that it refers to a file called words_alpha.txt. This file contains a dictionary of words; instructions on how to obtain it are given below. Here is the code for the main method:

```
public static void main(String[] args) { Anagrams a = new Anagrams();

    final long startTime = System.nanoTime(); try {
                a.processFile("words_alpha.txt");
    } catch (IOException e1) {
                e1.printStackTrace();
    }
    ArrayList<Map.Entry<Long,ArrayList<String>>> maxEntries = a.getMaxEntries(); final long estimatedTime =
    System.nanoTime() - startTime; final double seconds = ((double) estimatedTime/1000000000);
    System.out.println("Time: "+ seconds);
    System.out.println("List of max anagrams: "+ maxEntries);
}
```

2

4

6

8

10

12

14

Here is the expected output of your solution (the elapsed time may vary):

```
Elapsed Time: 0.689135767
Key of max anagrams: 236204078
List of max anagrams: [alerts, alters, artels, estral, laster, lastre, rastle, ratels, relast, resalt, salter, slater, staler, stelar,
    talers]
Length of list of max anagrams: 15
```

1

3

5

9   Testing Your Solution

In order to test your solution you are provided with a dictionary (words_alpha.txt) that has 370099 words. You may download this file from this link:

   https://github.com/dwyl/english-words/blob/master/words_alpha.txt

   Place it in the folder where your project is located (the same folder where the bin and src folders are).

10   Submission instructions

Please submit a zip file containing the file Anagrams.java on Canvas. The main method of the class Anagrams.java should include your tests. No report is required. Some further guidelines:

- Use JavaDoc to comment your code.

- Check the arguments of methods.