Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

ELİF AKGÜN
1801042251

Course Assistant: Ayşe Şerbetçi Turan

1 INTRODUCTION

1.1 Problem Definition

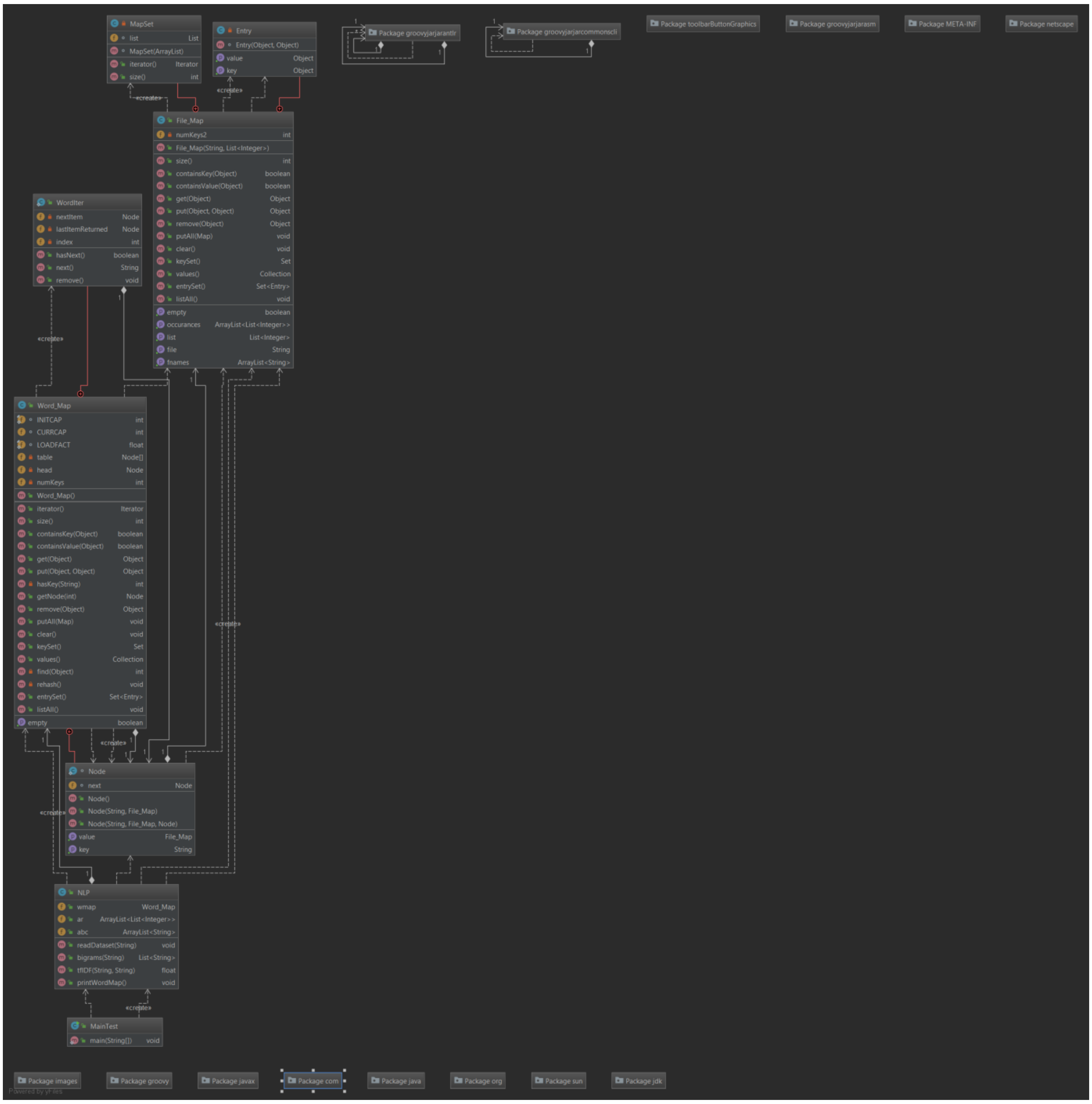
This task was asked to act as Natural Language Processing using the Map data structure. Two different HashMap classes have been implemented. The Word_Map class is a first-class thats key's type is a string, and the value is a File_Map. The second is the File_Map, thats key is the name of the file to be read, and the value of the ArrayList <List>, which specifies where the string of Word_Map is repeated in these files. Word_Map class is iterable to be more efficient program.

1.2 System Requirements

My solution also does not require a specific piece of hardware. This program can work anywhere with JVM and can work with 128KB of memory, it does not keep memory so much and it needs the interface to work on the smart phone. It can work if it is provided

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

The only thing that a user needs to do is texts of dataset. User can run this program to click the run button.

2.3 Problem Solution Approach

During the project development phase, the objects that should be used first were considered and classes were created for those objects. Map data structure used. Also it was implemented. First of all Map methods were overrided. If needed, helper methods were written.

Methods and their complexties of Word_Map

Word_Map Class:

int size(): Returns the size of map

Complexiy: O(1), just returns size.

bolean isEmpty(): Check whether map is empty,

Complexiy: O(1), just returns true or false

boolean containsKey(Object key): Check whether this map contains a mapping for the specified key

Complexiy: O(1), just returns true or false

boolean containsValue(Object value): Check whether this map maps one or more keys to the specified value

Complexiy: O(1), just returns true or false

Object get(Object key): if this map contains a mapping from a key k to a value v, then this method returns v; otherwise it returns null.

Complexiy: It contains find methor and its compexty is O(n), so its O(n).

Object put(Object key, Object value): Associates the specified value with the specified key in this map

Complexiy: It has a for loop, so its complexty is O(n).

int hasKey(String key): Chech whether the specified key is exist

Complexiy: It has a for loop, so its complexty is O(n).

void putAll(Map m): Copies all of the mappings from the specified map to this map.

Complexiy: This methods complexty is O(1) because of the iterator.

void clear(): Removes all of the mappings from this map (optional operation). The map will be empty after this call returns.

Complexiy: It includes a for loop, so its complexty is O(n).

Set keySet(): Returns a Set view of the keys contained in this map.

Complexiy: It includes a for loop, so its complexty is O(n).

Collection values(): Returns a Collection view of the values contained in this map.,

Complexiy: It includes a for loop, so its complexty is O(n).

int find(Object key): Finds either the target key or the first empty slot in the search chain using linear probing.

Complexiy: It includes a loop, so its complexty is O(n).

void rehash(): Expands table size when loadFactor exceeds LOAD_THRESHOLD

Complexiy: It includes a loop, so its complexty is O(n).

void listAll(): Lists all fnames and occurances

Complexiy: It includes a loop, so its complexty is O(n).

Methods and their complexties of File_Map

All of the upper methods is valid to this class except the find and rehash method, Because two of them implements Map interface. File_Map does not need find and rehash becasu its structure is a list.

Additionally File_Map implements this methods:

Object remove(Object key): Removes the mapping for a key from this map if it is present
Complexiy: It includes a loop, so its complexty is O(n).

Set<Entry> entrySet(): Returns a Set view of the mappings contained in this map.

Complexiy: It includes a loop, so its complexty is O(n).

Methods and their complexties of NLP

void readDataset(String dir): Reads the dataset from the given dir and created a word map

Complexiy: It includes a loop, so its complexty is O(n).

List<String> bigrams(String word): Finds all the bigrams starting with the given Word

Complexiy: It includes a loop, so its complexty is O(n).

tfIDF(String word, String fileName): Calculates the tfIDF value of the given word for the given file

Complexiy: It does not include a loop or vs, so its complexty is O(1).

void printWordMap(): Print the WordMap

Complexiy: It includes a loop, so its complexty is O(n).

Space Complexity

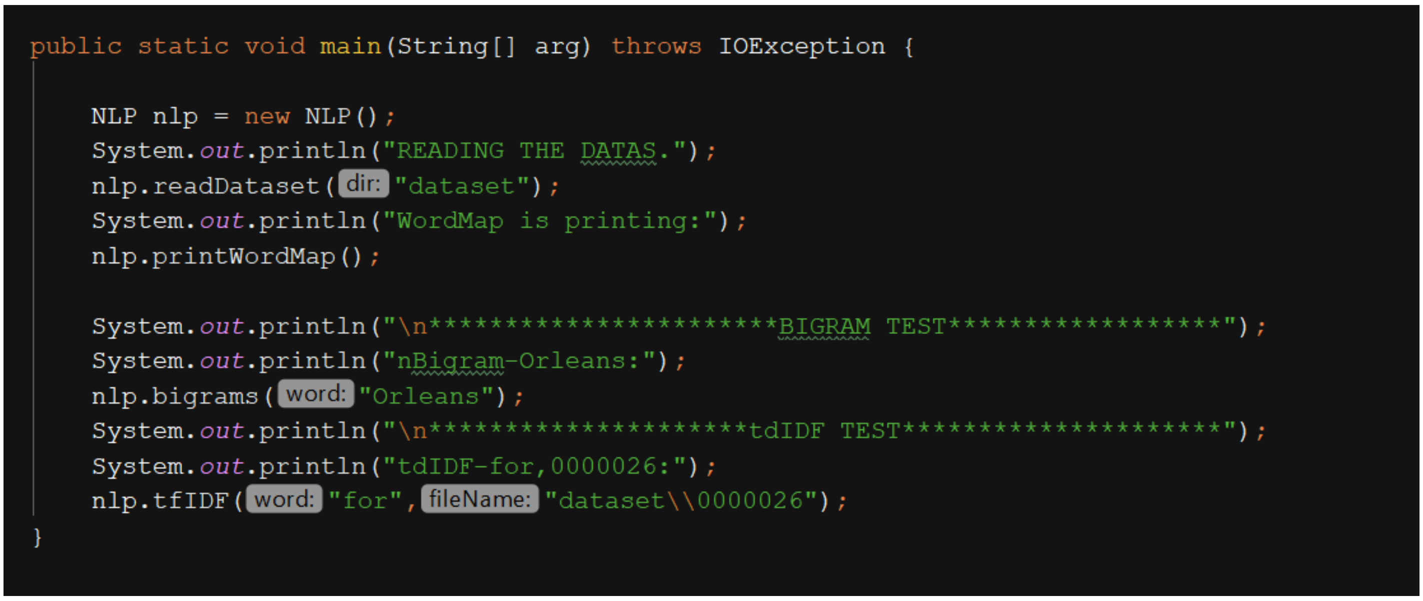
File_Map: Its space complexity is O(n^2) because of ArrayList<List<Integer>> occurances.

Word_Map: Its space complexity is O(n) because of Node table[]

NLP: : Its space complexity is O(n^2) because of ArrayList<List<Integer>> occurances

3 RESULT

3.1 Test Cases

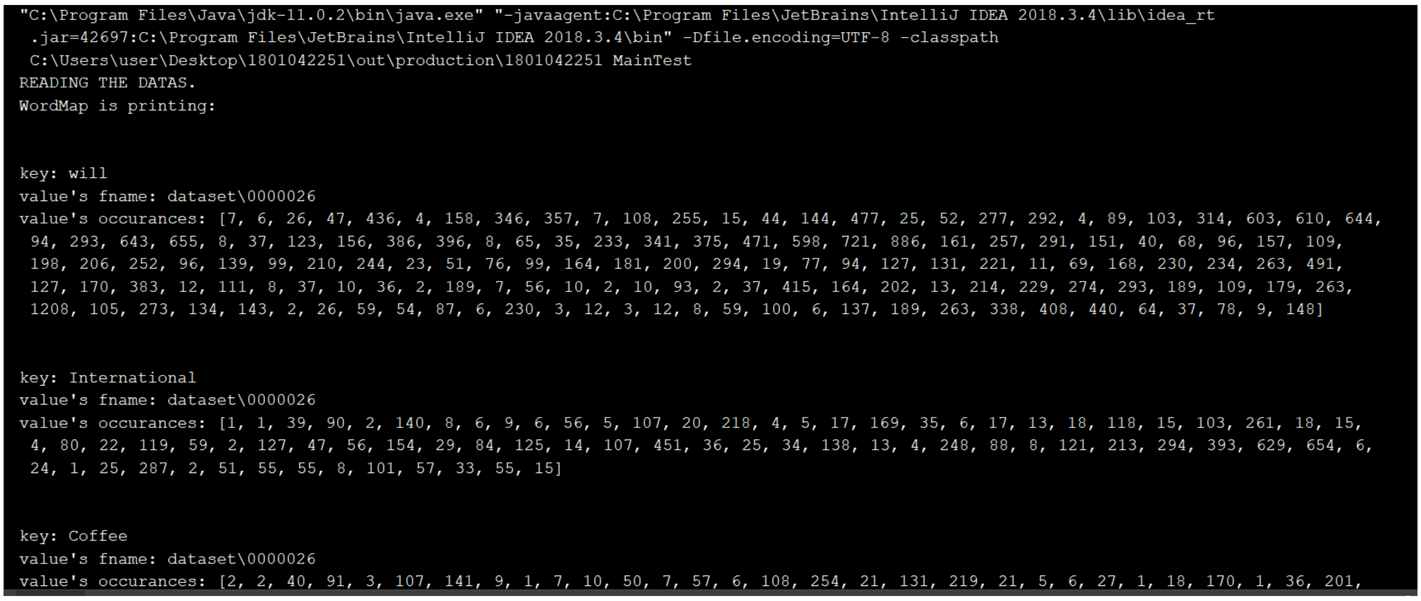


```
public static void main(String[] arg) throws IOException {

    NLP nlp = new NLP();
    System.out.println("READING THE DATAS.");
    nlp.readDataset( dir: "dataset");
    System.out.println("WordMap is printing:");
    nlp.printWordMap();

    System.out.println("\n***********************BIGRAM TEST******************");
    System.out.println("nBigram-Orleans:");
    nlp.bigrams( word: "Orleans");
    System.out.println("\n*********************tdIDF TEST*********************");
    System.out.println("tdIDF-for,0000026:");
    nlp.tfIDF( word: "for", fileName: "dataset\\0000026");
}
```

3.2 Running Results

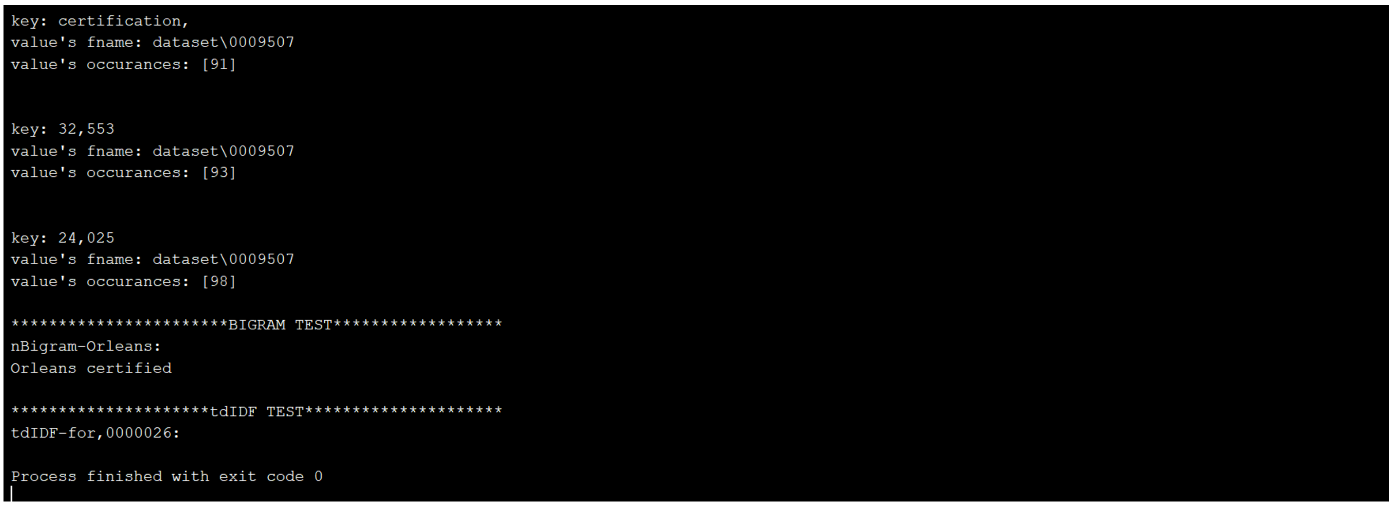


```
"C:\Program Files\Java\jdk-11.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2018.3.4\lib\idea_rt
 .jar=42697:C:\Program Files\JetBrains\IntelliJ IDEA 2018.3.4\bin" -Dfile.encoding=UTF-8 -classpath
 C:\Users\user\Desktop\1801042251\out\production\1801042251 MainTest
READING THE DATAS.
WordMap is printing:


key: will
value's fname: dataset\0000026
value's occurances: [7, 6, 26, 47, 436, 4, 158, 346, 357, 7, 108, 255, 15, 44, 144, 477, 25, 52, 277, 292, 4, 89, 103, 314, 603, 610, 644,
 94, 293, 643, 655, 8, 37, 123, 156, 386, 396, 8, 65, 35, 233, 341, 375, 471, 598, 721, 886, 161, 257, 291, 151, 40, 68, 96, 157, 109,
 198, 206, 252, 96, 139, 99, 210, 244, 23, 51, 76, 99, 164, 181, 200, 294, 19, 77, 94, 127, 131, 221, 11, 69, 168, 230, 234, 263, 491,
 127, 170, 383, 12, 111, 8, 37, 10, 36, 2, 189, 7, 56, 10, 2, 10, 93, 2, 37, 415, 164, 202, 13, 214, 229, 274, 293, 189, 109, 179, 263,
 1208, 105, 273, 134, 143, 2, 26, 59, 54, 87, 6, 230, 3, 12, 3, 12, 8, 59, 100, 6, 137, 189, 263, 338, 408, 440, 64, 37, 78, 9, 148]


key: International
value's fname: dataset\0000026
value's occurances: [1, 1, 39, 90, 2, 140, 8, 6, 9, 6, 56, 5, 107, 20, 218, 4, 5, 17, 169, 35, 6, 17, 13, 18, 118, 15, 103, 261, 18, 15,
 4, 80, 22, 119, 59, 2, 127, 47, 56, 154, 29, 84, 125, 14, 107, 451, 36, 25, 34, 138, 13, 4, 248, 88, 8, 121, 213, 294, 393, 629, 654, 6,
 24, 1, 25, 287, 2, 51, 55, 55, 8, 101, 57, 33, 55, 15]


key: Coffee
value's fname: dataset\0000026
value's occurances: [2, 2, 40, 91, 3, 107, 141, 9, 1, 7, 10, 50, 7, 57, 6, 108, 254, 21, 131, 219, 21, 5, 6, 27, 1, 18, 170, 1, 36, 201,
```



```
key: certification,
value's fname: dataset\0009507
value's occurances: [91]


key: 32,553
value's fname: dataset\0009507
value's occurances: [93]


key: 24,025
value's fname: dataset\0009507
value's occurances: [98]

***********************BIGRAM TEST******************
nBigram-Orleans:
Orleans certified

*********************tdIDF TEST*********************
tdIDF-for,0000026:

Process finished with exit code 0
```