

**Gebze Technical University
Computer Engineering**

CSE 222 - 2019 Spring

HOMEWORK 6 REPORT

**ÖMER ÇEVİK
161044004**

Course Assistant: Ayşe Şerbetçi TURAN

1 INTRODUCTION

1.1 Problem Definition

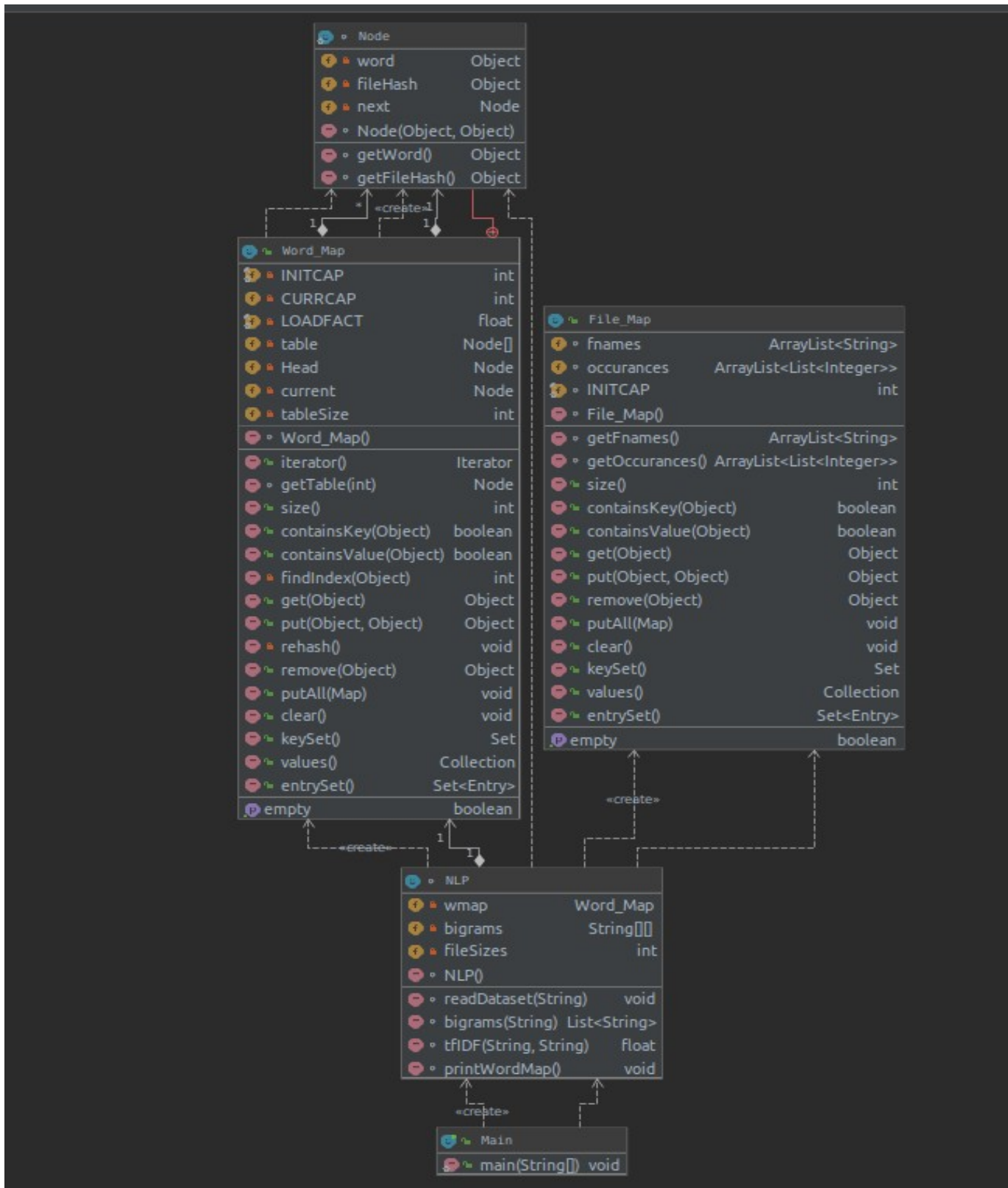
In that programme, there is a dataset which has text datas in that dataset. User wants to find bigrams of words in that files. The needed structures are linkedlist and hashmap. Word_Map class and File_Map class needed to implement to solve that problem. Also the wanted is TFIDF evaluation for all word in file.

1.2 System Requirements

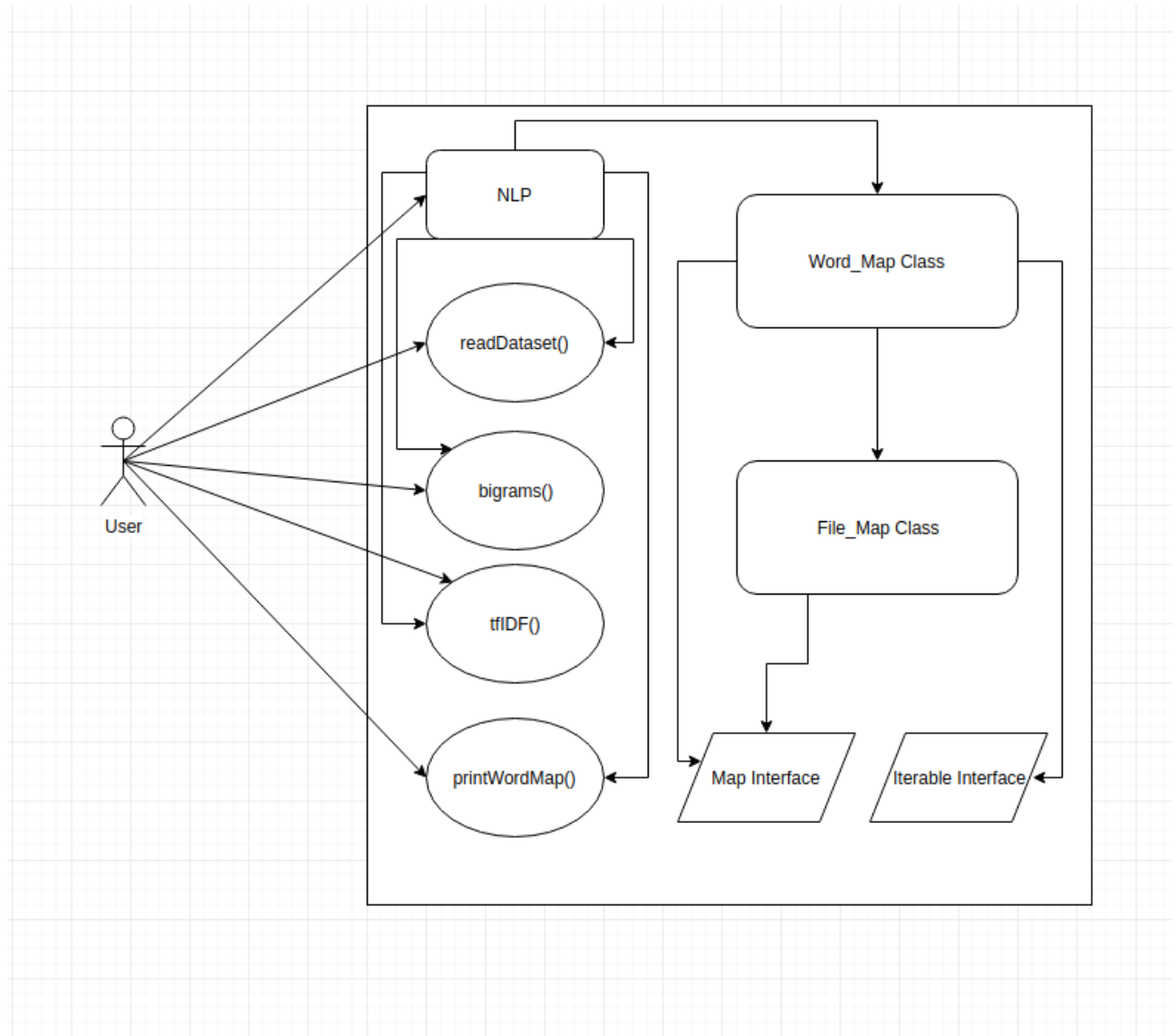
My solution doesn't require a specific piece of hardware, or maybe a certain minimal amount of memory, or a certain operating system, or a software library to be installed in order to run properly. It built by IntelliJ IDEA and used java programming language. It might run correctly if you use Java Virtual Machine and JDK is at least 8. Will my program work with 128KB of memory? I don't know, I didn't work with 128KB of memory but maybe it can work. If your nephew's smartphone's operating system is Android then it must run on it.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams



2.3 Problem Solution Approach

In my programme, I used HashMap data structure in Word_Map class. It uses linear probing/chaining index while putting datas in map truly. File_Map class have two main fields that occurances and filenames. File_Map class is not actually acting like HashMap data structure. Firstly I read dataset and saved datas into maps. While putting the datas in Word_Map class, I create linkedlist on words to use iterator. I couldn't find the real TFIDF calculation. I didn't have time for it but my maps works correctly.

Time Complexity:

Word_Map class:

- Word_Map() : Initialization of table, $O(1)$.
- iterator() : hasNext() and next() have constant time complexity, $O(1)$.
- Node class : Node() constructor, getWord(), getFileHash() have constant, $O(1)$.
- getTable() : Just returns table, $O(1)$.
- size() : Just returns size of table, $O(1)$.
- isEmpty() : Just returns boolean expression, $O(1)$.
- containsKey() : Loop in table to find key, $O(n)$.
- containsValue() : Loop in table to find values, $O(n)$.
- findIndex() : For worst case scenerio $O(\log(n))$, otherwise $O(1)$.
- get() : Uses findIndex(), $O(1)$.
- put() : Uses findIndex(), $O(1)$.
- rehash() : Reallocates table, $O(n)$.
- remove() : Not implemented, $O(1)$.
- putAll() : Puts all items in map, $O(n)$.
- clear() : Constant clear instructions, $O(1)$.
- keySet() : Adds into set all table keys, $O(n)$.
- values() : Adds into collection all table values, $O(n)$.
- entrySet() : Not implemented, $O(1)$.

File_Map class:

- File_Map() : Initialization of fields, $O(1)$.
- getFnames() : Returns fnames, $O(1)$.
- getOccurances() : Returns occurances, $O(1)$.
- size() : Returns map size, $O(1)$.
- isEmpty() : Returns boolean expression, $O(1)$.
- containsKey() : Searches key in fnames, $O(n)$.
- containsValue() : Searches values in occurances, $O(n)$.
- get() : Returns occurances given key, $O(1)$.
- put() : Puts given key and value into map, $O(1)$.
- remove() : Uses containsKey() and removes, $O(n)$.
- putAll() : Puts all datas into map, $O(n)$.
- clear() : Clears fields constant times, $O(1)$.

- `keySet()` : Adds into set all frames keys, $O(n)$.
- `values()` : Adds into collection all occurrences values, $O(n)$.
- `entrySet()` : Adds into HashSet all keys and values, $O(n)$.

NLP class:

- `readDataset()` : Reads dataset and checks bigrams, $O(n^2)$.
- `bigrams()` : Finds bigrams in map, $O(n)$.
- `tfidf()` : Evaluate TFIDF in map, $O(\log(n))$.
- `printWordMap()` : Using iterator and print Word_Map, $O(n)$.

Space Complexity:

Word_Map class:

- $4 + 4 + 8 + 4 + \text{Node} \times 100 + \text{Node} + \text{Node} = 102 \times \text{Node} + 20 \text{ bytes.}$

Node class:

- $\text{String} + \text{File_Map} + \text{Node}$

File_Map class:

- $\text{ArrayList<String>} + \text{ArrayList<List<Integer>>} + 4$

NLP class:

- $\text{String}[][] + 4 + \text{Word_Map}$

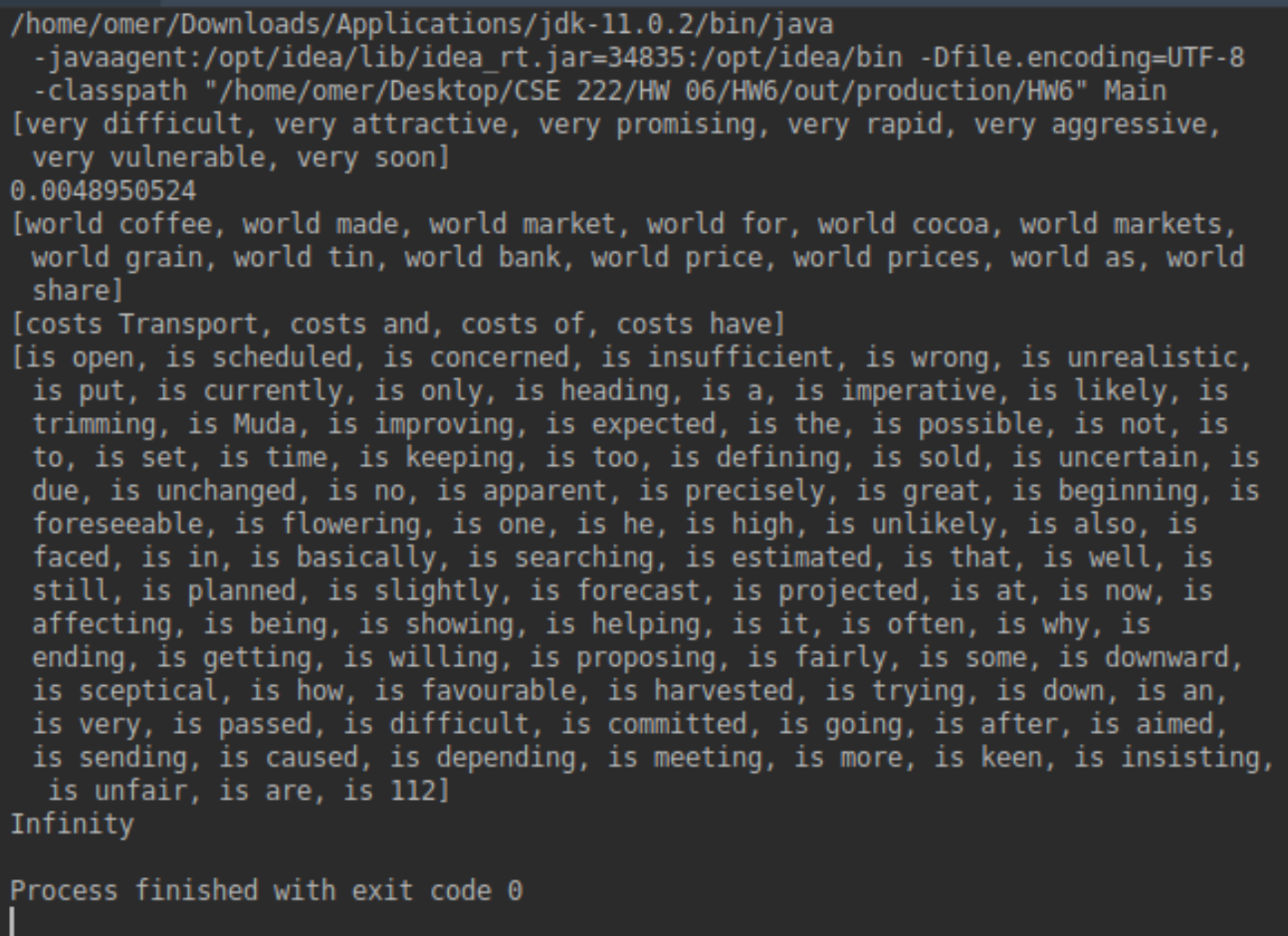
3 RESULT

3.1 Test Cases

Main Test

I used main method to test my programme. The results are in **3.2 Running Results**.

3.2 Running Results



```
/home/omer/Downloads/Applications/jdk-11.0.2/bin/java
-javaagent:/opt/idea/lib/idea_rt.jar=34835:/opt/idea/bin -Dfile.encoding=UTF-8
-classpath "/home/omer/Desktop/CSE 222/HW 06/HW6/out/production/HW6" Main
[very difficult, very attractive, very promising, very rapid, very aggressive,
very vulnerable, very soon]
0.0048950524
[world coffee, world made, world market, world for, world cocoa, world markets,
world grain, world tin, world bank, world price, world prices, world as, world
share]
[costs Transport, costs and, costs of, costs have]
[is open, is scheduled, is concerned, is insufficient, is wrong, is unrealistic,
is put, is currently, is only, is heading, is a, is imperative, is likely, is
trimming, is Muda, is improving, is expected, is the, is possible, is not, is
to, is set, is time, is keeping, is too, is defining, is sold, is uncertain, is
due, is unchanged, is no, is apparent, is precisely, is great, is beginning, is
foreseeable, is flowering, is one, is he, is high, is unlikely, is also, is
faced, is in, is basically, is searching, is estimated, is that, is well, is
still, is planned, is slightly, is forecast, is projected, is at, is now, is
affecting, is being, is showing, is helping, is it, is often, is why, is
ending, is getting, is willing, is proposing, is fairly, is some, is downward,
is sceptical, is how, is favourable, is harvested, is trying, is down, is an,
is very, is passed, is difficult, is committed, is going, is after, is aimed,
is sending, is caused, is depending, is meeting, is more, is keen, is insisting,
is unfair, is are, is 112]
Infinity

Process finished with exit code 0
```

```
Run: Main x
[7], [120, 293], [56], [5], [7], [52], [108], [3], [16], [18], [60], [85, 126],
[6], [7]]

Word : Coffee

File_HashMap_FileNames :

[0008350, 0007678, 0005750, 0004983, 0005522, 0001978, 0008911, 0000048, 0002827,
0000165, 0000402, 0003033, 0004598, 0008477, 0009253, 0005146, 0008606,
0007110, 0000828, 0006240, 0000202, 0000527, 0007915, 0007985, 0000458,
0001671, 0007660, 0003195, 0008857, 0000026, 0003805, 0003558, 0009303,
0001916, 0008891, 0001603, 0009321, 0000605, 0006812, 0006796, 0000194,
0003322, 0007964, 0008777, 0003463, 0001628, 0003876, 0008859, 0009280,
0009507, 0001004, 0007882, 0002820, 0008051, 0007052, 0007485, 0008792,
0001085, 0000677, 0005707, 0000283, 0008656, 0005674, 0001184, 0008425,
0008834, 0000178, 0002096, 0001327, 0007851, 0000764, 0002972, 0004842,
0004997, 0005172, 0000783, 0000641, 0008364, 0000759, 0007709, 0009271,
0000503, 0001631, 0001155]

File_HashMap_Occurances :

[[1, 130], [4, 183], [2, 240], [33], [6], [22, 80], [0, 33], [2], [34, 119], [39,
89], [9], [2], [29], [25, 235, 287], [2], [27], [2], [25, 120], [0, 35, 200],
[67, 86], [8], [5, 107, 253], [6, 221], [24, 62], [50], [4, 73], [13], [127],
[101], [2], [57, 355], [47], [3], [26], [57], [261, 420], [15, 103], [21, 131,
219], [12, 36], [1], [140], [143, 163], [1, 14], [0, 19, 32, 55], [12, 47],
[18], [154, 181], [34], [2], [20, 46], [6], [292, 626], [26], [1], [3], [34,
127, 138], [0, 18, 33, 55], [17], [4], [23], [0, 6], [51], [107, 152], [18,
195], [196, 201], [8, 34], [2, 105], [12], [15, 25], [20, 88], [0, 17], [2, 59,
87], [1], [22, 84, 125], [14], [169], [20], [220, 225], [5, 26], [2, 248], [2,
55], [6, 55], [15, 33], [13, 38, 221]]

Word : International

File_HashMap_FileNames :

[0008350, 0007678, 0005750, 0001978, 0008911, 0000048, 0002827, 0001234, 0000165,
0000402, 0003033, 0004598, 0008477, 0008606, 0007110, 0000828, 0000202,
0000527, 0007915, 0007985, 0001671, 0007660, 0003195, 0008857, 0000026,
0003805, 0008891, 0001603, 0009321, 0000605, 0006812, 0002171, 0000194,
0008777, 0003463, 0001628, 0003876, 0001004, 0007882, 0007485, 0008792,
0001085, 0000677, 0000283, 0008656, 0005674, 0001184, 0008834, 0000178,
0001327, 0007851, 0000764, 0002972, 0004997, 0005172, 0000783, 0000759,
0007709, 0009271, 0000503, 0001631, 0001155]

File_HashMap_Occurances :

[[0], [3], [450], [79], [32], [1], [118], [117], [38, 88], [8], [1], [28], [24,
286], [1], [24], [34], [7], [4, 106], [5], [23], [3], [12], [126], [100], [1],
[56], [56], [102, 260], [14], [20, 218], [35], [21], [139], [54], [46], [17],
[153], [5], [7, 118, 210, 291, 390, 625, 649], [33, 137], [54], [16], [3], [5],
[50], [106], [17], [7], [1], [14], [87], [16], [58], [83, 124], [13], [168],
[4], [247], [54], [5, 54], [14], [12]]

Process finished with exit code 0
```


Run: Main x

▶

⬆

⬇

📷

🔍

📄

🗑

🚀

Word : GMT

File_HashMap_FileNames :
[0008350, 0008606]

File_HashMap_Occurances :
[[9, 216], [88]]

Word : 1500

File_HashMap_FileNames :
[0008350, 0001085]

File_HashMap_Occurances :
[[8], [182]]

Word : at

File_HashMap_FileNames :
[0008350, 0007678, 0005750, 0000639, 0002827, 0000165, 0000402, 0004598, 0008477, 0009253, 0008606, 0007110, 0000828, 0005743, 0000202, 0000527, 0007915, 0007985, 0000458, 0003195, 0008857, 0003805, 0002892, 0009303, 0001603, 0007320, 0000605, 0006812, 0006272, 0006796, 0000194, 0003322, 0002732, 0007218, 0008777, 0001628, 0008859, 0001004, 0007882, 0002820, 0007485, 0008792, 0001085, 0000283, 0008656, 0005674, 0001184, 0005012, 0001327, 0007851, 0000764, 0000167, 0002972, 0005172, 0000783, 0007709, 0001631, 0001155]

File_HashMap_Occurances :
[[7, 189, 214], [94, 457, 467], [190, 373, 454], [39], [114, 399, 487], [370], [1], [92], [87, 96, 130, 591], [22], [86], [113], [66, 78, 96, 147, 224, 227, 280, 503, 507, 569], [102], [5, 16, 45, 58], [25], [17], [42, 86], [91], [351], [116], [296, 424], [38, 51], [34], [104, 370, 401], [241], [86, 144, 178], [120], [9, 27], [50, 60], [197, 230, 263], [2, 71, 83, 141], [34, 78], [242, 451], [25, 128, 150], [27], [3, 62], [436, 575], [557, 745], [86, 127], [78], [24, 128, 150], [225], [3, 88], [66, 205, 233, 251, 361, 495, 503, 551], [16, 18, 82], [85, 108, 120, 121, 208, 294], [6, 195], [347], [70], [158], [77], [19, 213, 311, 365, 456, 563], [21], [196], [145], [63, 83], [62, 192]]

Word : meet

File_HashMap_FileNames :
[0008350, 0001978, 0008911, 0003033, 0004598, 0008477, 0008606, 0007110, 0000828, 0000202, 0000527, 0000458, 0007218, 0000283, 0004626, 0004997, 0000759]

File_HashMap_Occurances :
[[6], [183], [9], [51], [12], [304], [100], [35], [651], [44], [477, 488], [43], [636], [62], [9], [131], [89, 104]]

Maven
Ant Build
Database
4: Run

Event Log