Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 6 REPORT

EDA BAHRIOGLU
131044055

Course Assistant:

INTRODUCTION

1.1 Problem Definition

The goal of this assignment is to understand the hashmap structure. For this purpose, hashmap classes are implemented by using node structure and arraylist structures.In this project, the most appropriate structure for Hashmap structure is seen the bigram and tfidf .That's why we worked on these.

1.2 System Requirements

4 class is used for program design.These are Word_map, File_map, Test, and NPL.It contains all the functions required for hashmap using the Word_map node structure.
Word map has the following methods:
public Iterator iterator() : I used with iterator operation
public int size() : this method return size;
public boolean isEmpty() :this method make check
public boolean containsKey(Object key) :this method make check if it is validity it return true for key
public boolean containsValue(Object value) this method make check if it is validity it return true for value
public Object get(Object key) : It is get a key.
public Object put(Object key, Object value). It is add key and value.
public Object remove(Object key) .remove operation.
public void putAll(Map m) .All list add another list.
public void clear() :It is make clear list
public Set keySet() :It print key and return set
ublic Collection values(): print value and return collection.
public Set<Entry> entrySet() : This function empty.
Helper functions:
rehash and find : I used this methods from Koffman book.

In hashmap file_map is implemented with arraylist structure. File_map class methods such as Word_map .It different with only with arraylist .In the NLP class,ı used bigram, tfidf, read the data set and print the word map with iterator.

public void readDataset(String dir): Reads and parses the supplied dataset
public List<String> bigrams(String word): Bigram method finds is simply a piece of text consisting of two sequential words which occurs in a given text at least once.

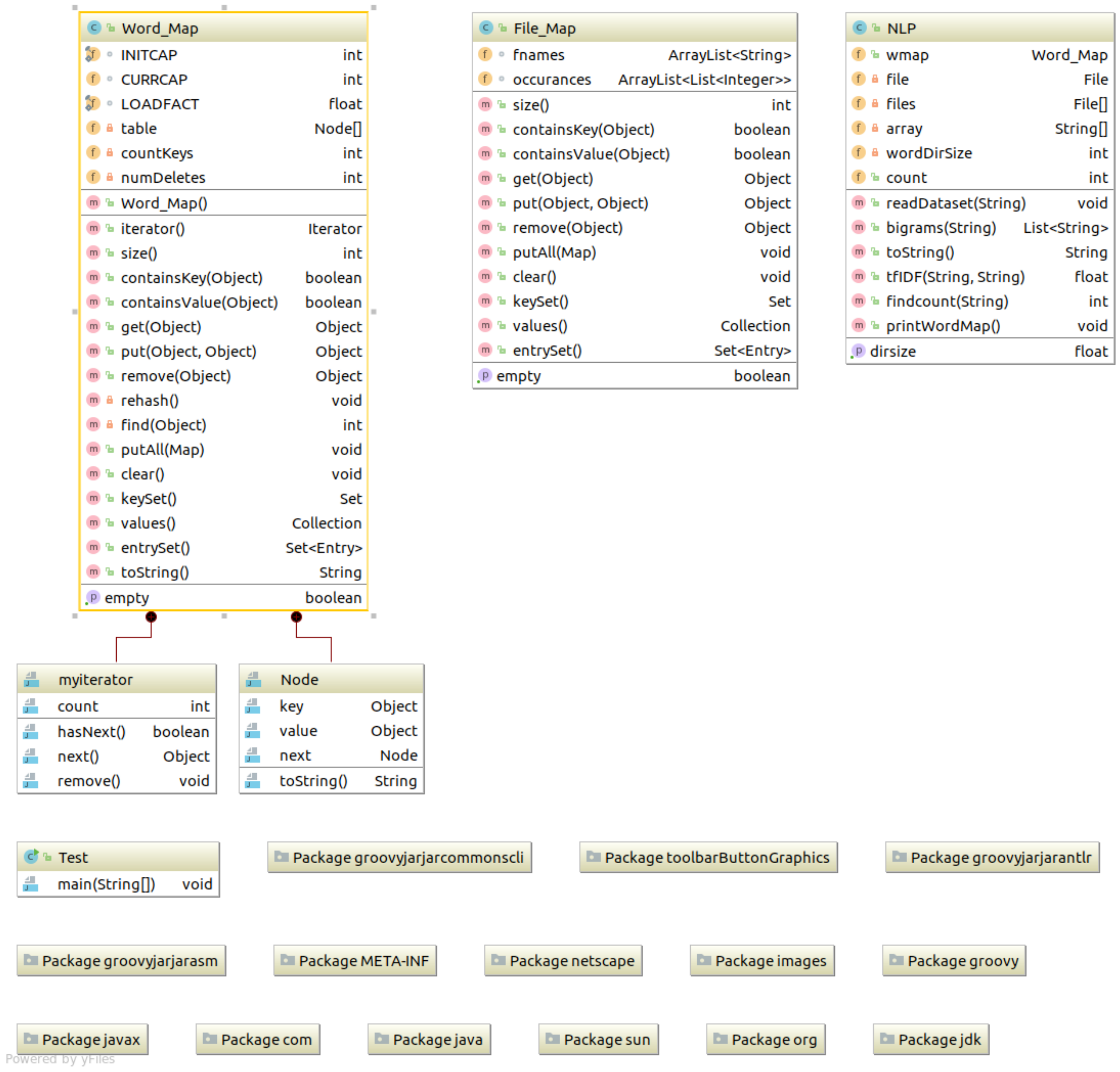
public float tfIDF(String word, String fileName): Aword is informative for a file to be categorized if it occurs frequently in that file in this method
public void printWordMap(): ı used with word mapi iterative.

All methods in the test class were checked for correct operation.Also ı read the commands in the input file.

HardWare requirements:
System requirements for Intelig Ide:
GNOME or KDE desktop
2 GB RAM minimum, 4 GB RAM recommended
1.5 GB hard disk space + at least 1 GB for caches
1024x768 minimum screen resolution
The minimum system requirements for Java Virtual Machine are as follows:
Windows 8/7/Vista/XP/2000. ...
Windows Server 2008/2003.
Intel and 100% compatible processors are supported.
Pentium 166 MHz or faster processor with at least 64 MB of physical RAM.
98 MB of free disk space.

METHOD

2.1 Class Diagrams


Word_Map
INITCAP int
CURRCAP int
LOADFACT float
table Node[]
countKeys int
numDeletes int
Word_Map()
iterator() Iterator
size() int
containsKey(Object) boolean
containsValue(Object) boolean
get(Object) Object
put(Object, Object) Object
remove(Object) Object
rehash() void
find(Object) int
putAll(Map) void
clear() void
keySet() Set
values() Collection
entrySet() Set<Entry>
toString() String
empty boolean
File_Map
fnames ArrayList<String>
occurances ArrayList<List<Integer>>
size() int
containsKey(Object) boolean
containsValue(Object) boolean
get(Object) Object
put(Object, Object) Object
remove(Object) Object
putAll(Map) void
clear() void
keySet() Set
values() Collection
entrySet() Set<Entry>
empty boolean
NLP
wmap Word_Map
file File
files File[]
array String[]
wordDirSize int
count int
readDataset(String) void
bigrams(String) List<String>
toString() String
tfIDF(String, String) float
findcount(String) int
printWordMap() void
dirsize float
myiterator
count int
hasNext() boolean
next() Object
remove() void
Node
key Object
value Object
next Node
toString() String
Test
main(String[]) void
Package groovyjarjarcommonscli
Package toolbarButtonGraphics
Package groovyjarjarantlr
Package groovyjarjarasm
Package META-INF
Package netscape
Package images
Package groovy
Package javax
Package com
Package java
Package sun
Package org
Package jdk
Powered by yFiles

2.2 Use Case Diagrams

This program will get an input file. It will print the screen by reading all such commands.A sample driver class is also written.

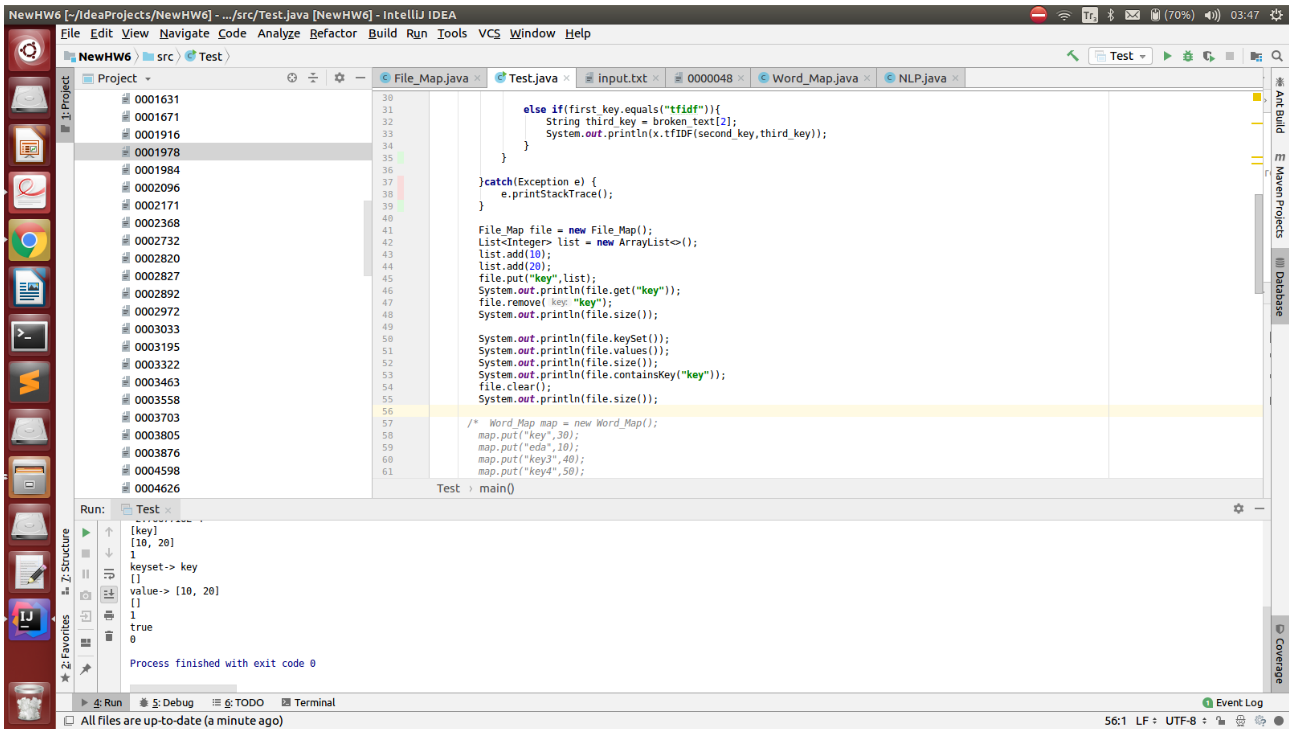
2.3 Problem Solution Approach

1.1 In this assignment, ı apply the world_map class completely.But putAll method a bit missing.I've reached my goal for this class, and I've used my bigram.There are some deficiencies in the File Map class.I was able to test the methods of this class more in the driver class.I was able to run bigram method completely correctly.But TFIDF in some cases is not working properly because of the IDF is working incorrectly. I tried so hard for this part.I used the hashmap data structure for the program .I wrote my code with the Java programming language, paying attention to the object oriented structure.

RESULT

3.1 Test Cases

Test Cases For Filemap

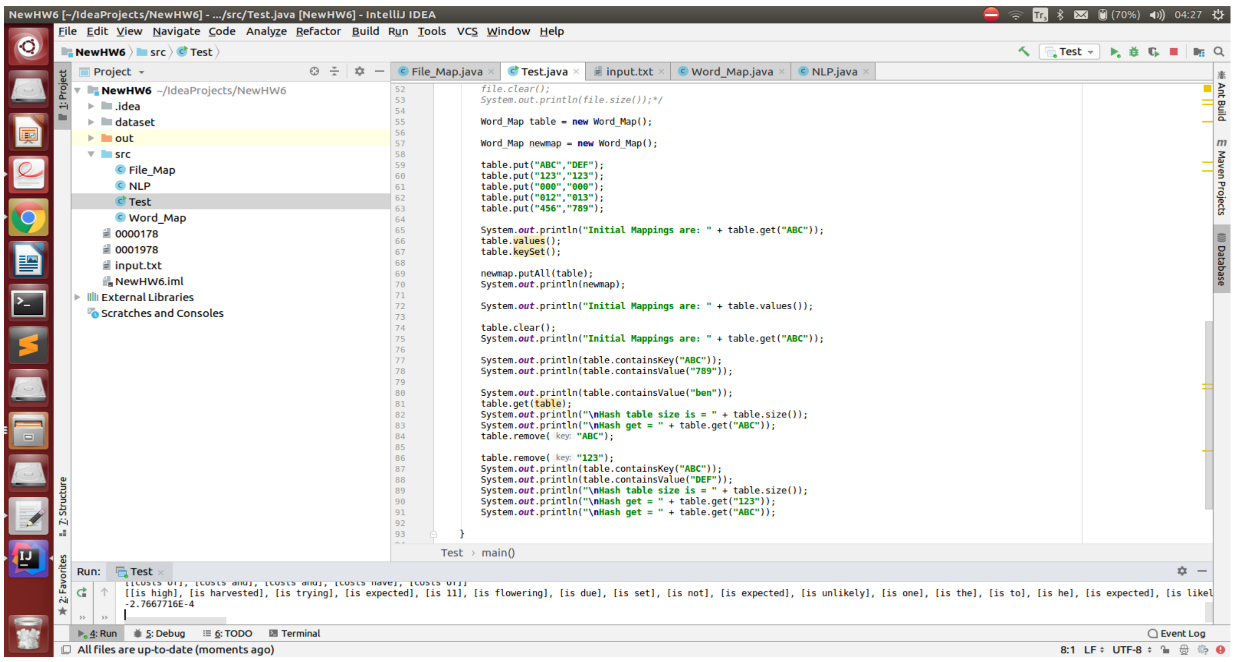


```java
            else if(first_key.equals("tfidf")){
                String third_key = broken_text[2];
                System.out.println(x.tfIDF(second_key,third_key));
            }
        }

    }catch(Exception e) {
        e.printStackTrace();
    }

    File_Map file = new File_Map();
    List<Integer> list = new ArrayList<>();
    list.add(10);
    list.add(20);
    file.put("key",list);
    System.out.println(file.get("key"));
    file.remove( key: "key");
    System.out.println(file.size());

    System.out.println(file.keySet());
    System.out.println(file.values());
    System.out.println(file.size());
    System.out.println(file.containsKey("key"));
    file.clear();
    System.out.println(file.size());

 /*  Word_Map map = new Word_Map();
    map.put("key",30);
    map.put("eda",10);
    map.put("key3",40);
    map.put("key4",50);
```

```
[key]
[10, 20]
1
keyset-> key
[]
value-> [10, 20]
[]
1
true
0

Process finished with exit code 0
```

All files are up-to-date (a minute ago)

Test Cases for WordMap



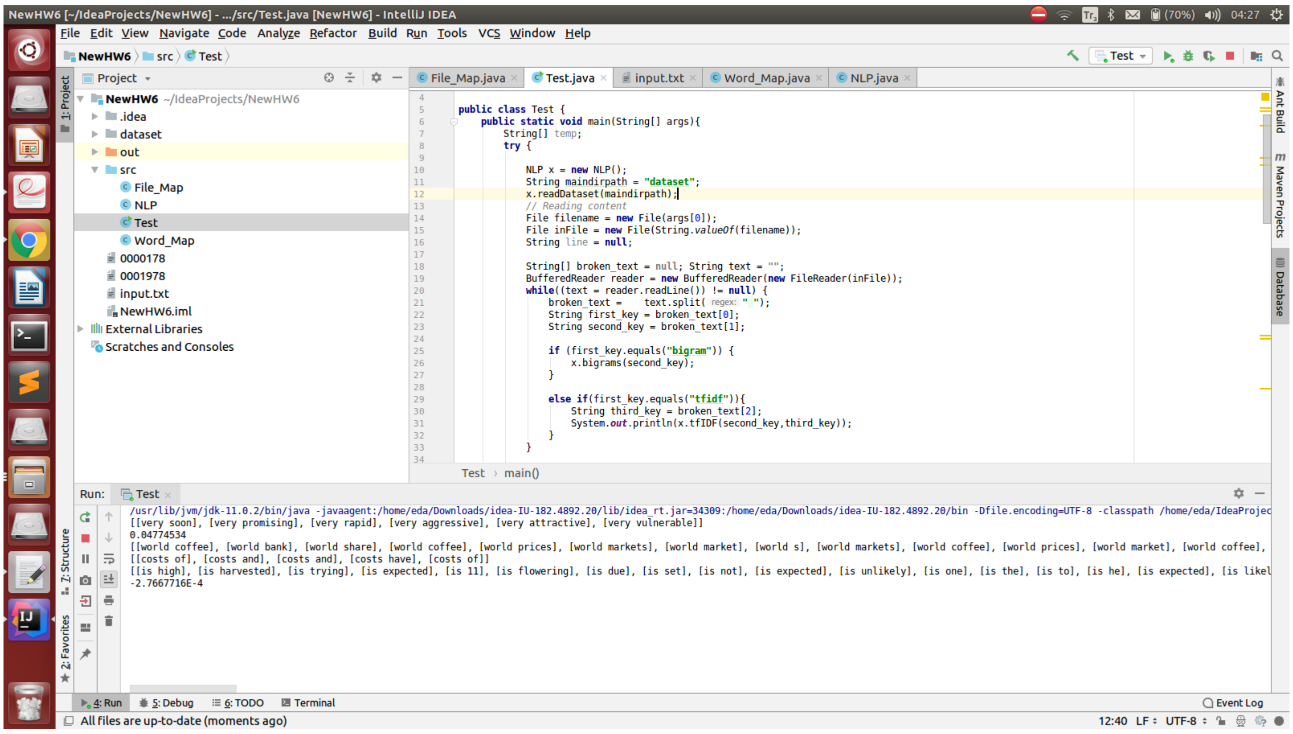
```java
    file.clear();
    System.out.println(file.size());*/

    Word_Map table = new Word_Map();

    Word_Map newmap = new Word_Map();

    table.put("ABC","DEF");
    table.put("123","123");
    table.put("000","000");
    table.put("012","013");
    table.put("456","789");

    System.out.println("Initial Mappings are: " + table.get("ABC"));
    table.values();
    table.keySet();

    newmap.putAll(table);
    System.out.println(newmap);

    System.out.println("Initial Mappings are: " + table.values());

    table.clear();
    System.out.println("Initial Mappings are: " + table.get("ABC"));

    System.out.println(table.containsKey("ABC"));
    System.out.println(table.containsValue("789"));

    System.out.println(table.containsValue("ben"));
    table.get(table);
    System.out.println("\nHash table size is = " + table.size());
    System.out.println("\nHash get = " + table.get("ABC"));
    table.remove( key: "ABC");

    table.remove( key: "123");
    System.out.println(table.containsKey("ABC"));
    System.out.println(table.containsValue("DEF"));
    System.out.println("\nHash table size is = " + table.size());
    System.out.println("\nHash get = " + table.get("123"));
    System.out.println("\nHash get = " + table.get("ABC"));

}
```

```
[[is high], [is harvested], [is trying], [is expected], [is 11], [is flowering], [is due], [is set], [is not], [is expected], [is unlikely], [is one], [is the], [is to], [is he], [is expected], [is likel
-2.7667716E-4
```

All files are up-to-date (moments ago)

3.2 Running Results

Results is Expected from me .



```java
public class Test {
    public static void main(String[] args){
        String[] temp;
        try {

            NLP x = new NLP();
            String maindirpath = "dataset";
            x.readDataset(maindirpath);
            // Reading content
            File filename = new File(args[0]);
            File inFile = new File(String.valueOf(filename));
            String line = null;

            String[] broken_text = null; String text = "";
            BufferedReader reader = new BufferedReader(new FileReader(inFile));
            while((text = reader.readLine()) != null) {
                broken_text =    text.split( regex: " ");
                String first_key = broken_text[0];
                String second_key = broken_text[1];

                if (first_key.equals("bigram")) {
                    x.bigrams(second_key);
                }

                else if(first_key.equals("tfidf")){
                    String third_key = broken_text[2];
                    System.out.println(x.tfIDF(second_key,third_key));
                }
            }
```

```
/usr/lib/jvm/jdk-11.0.2/bin/java -javaagent:/home/eda/Downloads/idea-IU-182.4892.20/lib/idea_rt.jar=34309:/home/eda/Downloads/idea-IU-182.4892.20/bin -Dfile.encoding=UTF-8 -classpath /home/eda/IdeaProjec
[[very soon], [very promising], [very rapid], [very aggressive], [very attractive], [very vulnerable]]
0.04774534
[[world coffee], [world bank], [world share], [world coffee], [world prices], [world markets], [world market], [world s], [world markets], [world coffee], [world prices], [world market], [world coffee],
[[costs of], [costs and], [costs and], [costs have], [costs of]]
[[is high], [is harvested], [is trying], [is expected], [is 11], [is flowering], [is due], [is set], [is not], [is expected], [is unlikely], [is one], [is the], [is to], [is he], [is expected], [is likel
-2.7667716E-4
```

- Main titles -> 16pt , 2 line break
- Subtitles -> 14pt, 1.5 line break
- Paragraph -> 12pt, 1.5 line break