

bible-as.txt:

The total number of paragraphs processed: 31103
The total number of unique words observed: 12780
The total number of words is: 784893

1 the- cf: 59143 and numOccurs: 23168
2 and- cf: 52095 and numOccurs: 23855
3 of- cf: 34517 and numOccurs: 18172
4 to- cf: 13771 and numOccurs: 9794
5 that- cf: 13575 and numOccurs: 10283
6 in- cf: 12770 and numOccurs: 9624
7 he- cf: 10634 and numOccurs: 7671
8 unto- cf: 9257 and numOccurs: 7572
9 for- cf: 9170 and numOccurs: 7212
10 shall- cf: 9105 and numOccurs: 5669
11 i- cf: 8879 and numOccurs: 6006
12 a- cf: 8816 and numOccurs: 6594
13 his- cf: 8151 and numOccurs: 5775
14 they- cf: 7591 and numOccurs: 5638
15 is- cf: 7181 and numOccurs: 5628
16 jehovah- cf: 6775 and numOccurs: 5756
17 him- cf: 6609 and numOccurs: 4974
18 not- cf: 6573 and numOccurs: 5571
19 be- cf: 6470 and numOccurs: 5104
20 them- cf: 6407 and numOccurs: 4951
21 it- cf: 6163 and numOccurs: 4759
22 with- cf: 5990 and numOccurs: 4837
23 all- cf: 5596 and numOccurs: 4651
24 thou- cf: 5513 and numOccurs: 3889
25 thy- cf: 4956 and numOccurs: 3241
26 my- cf: 4497 and numOccurs: 3164
27 was- cf: 4467 and numOccurs: 3618
28 will- cf: 4115 and numOccurs: 3021
29 me- cf: 4086 and numOccurs: 3069
30 god- cf: 4077 and numOccurs: 3553
100 therefore- cf: 1149 and numOccurs: 1131
500 zion- cf: 161 and numOccurs: 161
1000 syrians- cf: 64 and numOccurs: 57

Number of words that occur in exactly one document: 4042

lesmis.txt:

The total number of paragraphs processed: 13930

The total number of unique words observed: 9903

The total number of words is: 131775

1 the- cf: 8944 and numOccurs: 6449
2 - cf: 4656 and numOccurs: 2587
3 of- cf: 3085 and numOccurs: 2741
4 a- cf: 3084 and numOccurs: 2746
5 he- cf: 2671 and numOccurs: 2324
6 to- cf: 2628 and numOccurs: 2339
7 and- cf: 2360 and numOccurs: 2279
8 in- cf: 2256 and numOccurs: 2116
9 was- cf: 2105 and numOccurs: 1897
10 that- cf: 1771 and numOccurs: 1646
11 is- cf: 1584 and numOccurs: 1433
12 had- cf: 1429 and numOccurs: 1302
13 his- cf: 1402 and numOccurs: 1252
14 it- cf: 1386 and numOccurs: 1288
15 you- cf: 1373 and numOccurs: 1161
16 this- cf: 1248 and numOccurs: 1206
17 i- cf: 1212 and numOccurs: 1067
18 said- cf: 1022 and numOccurs: 1021
19 at- cf: 1004 and numOccurs: 963
20 on- cf: 976 and numOccurs: 940
21 not- cf: 907 and numOccurs: 879
22 with- cf: 812 and numOccurs: 788
23 she- cf: 784 and numOccurs: 697
24 which- cf: 761 and numOccurs: 742
25 one- cf: 716 and numOccurs: 684
26 marius- cf: 710 and numOccurs: 698
27 as- cf: 675 and numOccurs: 603
28 have- cf: 666 and numOccurs: 640
29 what- cf: 662 and numOccurs: 642
30 for- cf: 601 and numOccurs: 582
100 like- cf: 160 and numOccurs: 154
500 dropped- cf: 28 and numOccurs: 28
1000 prevent- cf: 13 and numOccurs: 13

Number of words that occur in exactly one document: 4599

source code:

```

import java.io.*;
import java.util.*;
import java.lang.String;

/**
 * PA1. java
 * @version 2.0, 2016-02-12
 * @author Chin-Ting Ko
 *
 */

public class PA1 {

    public static void main(String[] args) throws IOException {
        // TODO Auto-generated method stub

        /* load input file "input.txt"*/
        Scanner input= new Scanner (new FileReader("bible-asv.txt"));
        /* generate output file "output.txt"*/
        PrintWriter output = new PrintWriter (new FileWriter("output.txt"));

        int number_paragraphs = 0;
        int total_words = 0;
        int single_occurs=0;
        String input_string;
        HashMap<String, Integer> collectionFrequency= new HashMap<String, Integer>() ;
        HashMap<String, Integer> documentFrequency= new HashMap<String, Integer>() ;
        HashMap<String, Integer> paragraphFrequency= new HashMap<String, Integer>() ;

        while (input.hasNextLine()) {
            input_string = input.nextLine();

            if (input_string.startsWith("<P ID=")){
                number_paragraphs++;
                String paragraph = input.nextLine();
                String words[] = paragraph.replaceAll("[^a-zA-Z ]",
"".toLowerCase().split(" ");

                for(int i=0; i<words.length; i++){
                    total_words++;

                    if(collectionFrequency.containsKey(words[i])==true){
                        int cf= collectionFrequency.get(words[i]);
                        collectionFrequency.put(words[i],++cf);
                    }
                    else if (collectionFrequency.containsKey(words[i])==false){
                        collectionFrequency.put(words[i],1);
                    }

                    if (documentFrequency.containsKey(words[i])==false){
                        documentFrequency.put(words[i],1);
                    }
                    else if (documentFrequency.containsKey(words[i])==true &&
paragraphFrequency.containsKey(words[i])==false){
                        int df= documentFrequency.get(words[i]);
                        documentFrequency.put(words[i],++df);

```

```

        }

        if (paragraphFrequency.containsKey(words[i])==false){
            paragraphFrequency.put(words[i],1);
        }
        else if(paragraphFrequency.containsKey(words[i])==true){
            int pf= paragraphFrequency.get(words[i]);
            paragraphFrequency.put(words[i],++pf);
        }

        if (i==words.length-1){
            paragraphFrequency.clear();
        }
    }
}

Set<Map.Entry<String, Integer>> entriesCf = collectionFrequency.entrySet();
Comparator<Map.Entry<String, Integer>> valueComparatorCf = new
Comparator<Map.Entry<String,Integer>>() {
    public int compare(Map.Entry<String, Integer> e1, Map.Entry<String,
Integer> e2) {
        Integer v1 = e1.getValue();
        Integer v2 = e2.getValue();
        return v2.compareTo(v1);
    }
};
//convert Set to List in Java
List<Map.Entry<String, Integer>> listOfEntriesCf = new
ArrayList<Map.Entry<String, Integer>>(entriesCf);
// sorting HashMap by values using comparator
Collections.sort(listOfEntriesCf, valueComparatorCf);
LinkedHashMap<String, Integer> sortedCfByValue = new LinkedHashMap<String,
Integer>(listOfEntriesCf.size());
// copying entries from List to Map
for(Map.Entry<String, Integer> entryCf : listOfEntriesCf){
    sortedCfByValue.put(entryCf.getKey(), entryCf.getValue());
}

Set<Map.Entry<String, Integer>> entriesDf = documentFrequency.entrySet();
Comparator<Map.Entry<String, Integer>> valueComparatorDf = new
Comparator<Map.Entry<String,Integer>>() {
    public int compare(Map.Entry<String, Integer> e1, Map.Entry<String,
Integer> e2) {
        Integer v1 = e1.getValue();
        Integer v2 = e2.getValue();
        return v2.compareTo(v1);
    }
};
// convert Set to List in Java
List<Map.Entry<String, Integer>> listOfEntriesDf = new
ArrayList<Map.Entry<String, Integer>>(entriesDf);
// sorting HashMap by values using comparator
Collections.sort(listOfEntriesDf, valueComparatorDf);
LinkedHashMap<String, Integer> sortedDfByValue = new LinkedHashMap<String,
Integer>(listOfEntriesDf.size());
// copying entries from List to Map
for(Map.Entry<String, Integer> entryDf : listOfEntriesDf){
    sortedDfByValue.put(entryDf.getKey(), entryDf.getValue());
}

```

```
        if (entryDf.getValue()==1){
            single_occurs++;
        }
    }

    output.println("The total number of paragraphs processed: "+number_paragraphs);
    output.println("The total number of unique words observed:
"+collectionFrequency.size());
    output.println("The total number of words is: "+total_words);
    output.println();
    Iterator<String> itCf= sortedCfByValue.keySet().iterator();
    Iterator<String> itDf= sortedDfByValue.keySet().iterator();
    for (int i=1; i<1001; i++){
        String keyCf=(String)itCf.next();
        if (i<31 || i==100 || i==500 || i==1000){
            output.println(i+" "+keyCf+" - cf: " + sortedCfByValue.get(keyCf)+ " and
numOccurs: " + sortedDfByValue.get(keyCf));
        }
    }
    output.println();
    output.println("Number of words that occur in exactly one document:
"+single_occurs);
    output.println();

    //below is for tag cloud
    Iterator<String> itCf_TagCloud= sortedCfByValue.keySet().iterator();
    for (int i=1; i<51; i++){
        String keyCf=(String)itCf_TagCloud.next();
        output.println(keyCf+": "+ sortedCfByValue.get(keyCf));
    }
    output.close();
}
}
```