WordCount.java

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
public class WordCount {
    //Data structure to map a word to its frequency i.e. a Map
    HashMap<String,Integer> wordMap = new HashMap<String,Integer>();//    KEY VALUE ie word & its count
    HashMap<String,Integer> pairMap = new HashMap<String,Integer>();//KEY VALUE ie word & its count

    public void readFile(String file) {
        String word = "";
        //var for pairMap
        String prevWord = "";
        String nextWord = "";
        String tempKey ="";
        Scanner sc;
        try {
            sc = new Scanner(new File("C:\\Users\\bhupe\\Downloads\\"+file));
            //Split the text into words using whitespace character
            sc.useDelimiter(" ");//("\\s +") for multiple spaces ie one of more occurance of space
            //Get a word at a time
            while(sc.hasNext()) {
                word = sc.next(); //Get a word
                word = word.toLowerCase(); //Convert words to lowercase
                word = word.replaceAll("[,;.]", ""); //Delete punctuation
                //Store words as keys and frequencies as values
                if(!word.equals("")) {
                    if(wordMap.containsKey(word))
                        wordMap.put(word, wordMap.get(word) + 1);
```

```
                    else
                        wordMap.put(word, 1);
                }
            }

            //for keyMap
            sc = new Scanner(new
File("C:\\Users\\bhupe\\Downloads\\"+file));
            //Split the text into words using whitespace
character
            sc.useDelimiter(" ");//("\\s +") for multiple spaces
ie one of more occurance of space
            //Get a word at a time
            while(sc.hasNext()) {
                nextWord = sc.next(); //Get a word
                nextWord = nextWord.toLowerCase(); //Convert
words to lowercase
                nextWord = nextWord.replaceAll("[,;]", "");
//Delete punctuation
                //Store words as keys and frequencies as values
                if(!nextWord.equals("")&& !prevWord.equals(""))
{
                    tempKey = prevWord + ":" +nextWord;
                    if(pairMap.containsKey(tempKey))
                        pairMap.put(tempKey,
pairMap.get(tempKey) + 1);
                    else
                        pairMap.put(tempKey, 1);
                }
                prevWord = nextWord;
            }


            sc.close();
        } catch (FileNotFoundException e) {
            System.out.println("File does not exist!" + e);
        }
    }
    public void printPairFreq() {
        System.out.print("check");
        for(String pairWord: pairMap.keySet()) {
                    System.out.print("check");
```

```
66
67              int count = pairMap.get(pairWord);
68              System.out.print("{"+pairWord+", "+count+"}"+"\n");
69          }
70      }
71
72      public void printWordFreq() {
73
74          for(String word: wordMap.keySet()) {
75              int count = wordMap.get(word);
76              System.out.print("{"+word+", "+count+"}"+"\n");
77          }
78      }
79
80      double probWord(String word){//tf
81          double p;//probality
82          int total_count=0;
83          int word_count=0;
84          for(String wordInMap: wordMap.keySet()) {
85              total_count=total_count+1;
86           }
87          word_count = wordMap.get(word);
88          p = word_count*1.0/total_count*1.0; // type casting
89          return p;
90      }
91
92
93      double probOfWordPair(String first,String second){
94          String word = first + ":" +second;
95          double p;//probality
96          int total_count=0;
97          int wordPair_count=0;
98          for(String wordInMap: pairMap.keySet()) {
99              total_count=total_count+1;
100          }
101         wordPair_count = pairMap.get(word);
102         p = wordPair_count*1.0/total_count*1.0; // type casting
103         return p;
104     }
105
```

```
106
107     double idf(String[] arr,String term){
108         int length = arr.length;
109         double idft=0;
110         double p;
111         int dft=0;
112         for (int i = 0; i < length; i++) {
113              readFile(arr[i]+".txt");//file name
114              p=probWord(term);
115              if(p>0){
116                  dft++;
117              }
118          }
119
120         idft=Math.log(length/dft); //length is total number of
    files & dft is the total number of files that have the
    particular tern that is passed in the function
121
122         return idft;
123     }
124
125
126     double tf_idf(String[] arr,String term){
127         double overAllProb=0;
128         double p;
129         for(int i = 0; i < arr.length; i++) {
130              readFile(arr[i]+".txt");//file name
131              p=probWord(term);
132              if(p>0){
133                  overAllProb+=p;
134              }
135          }
136
137         double tf = overAllProb/arr.length;
138         double idf = idf(arr, term);//this logic is worng
139
140         return tf*idf;
141     }
142
143
```

```
144     public void main(String[] args) {
145         WordCount unigram = new WordCount();
146         unigram.readFile("file.txt");
147         //unigram.printWordFreq();
148         //unigram.printPairFreq();
149         double pWord = unigram.probWord("that");
150         System.out.print(pWord+"\n");
151         double pWordPair = unigram.probOfWordPair("is","the");
152         System.out.print(pWordPair+"\n");
153 
154         String[] numbers = {"file1","file2","file3"};
155         double idf = idf(numbers,"the");
156         System.out.println(idf);
157 
158         double id_idf = tf_idf(numbers,"the");
159         System.out.println(id_idf);
160 
161     }
162 }
163 
```