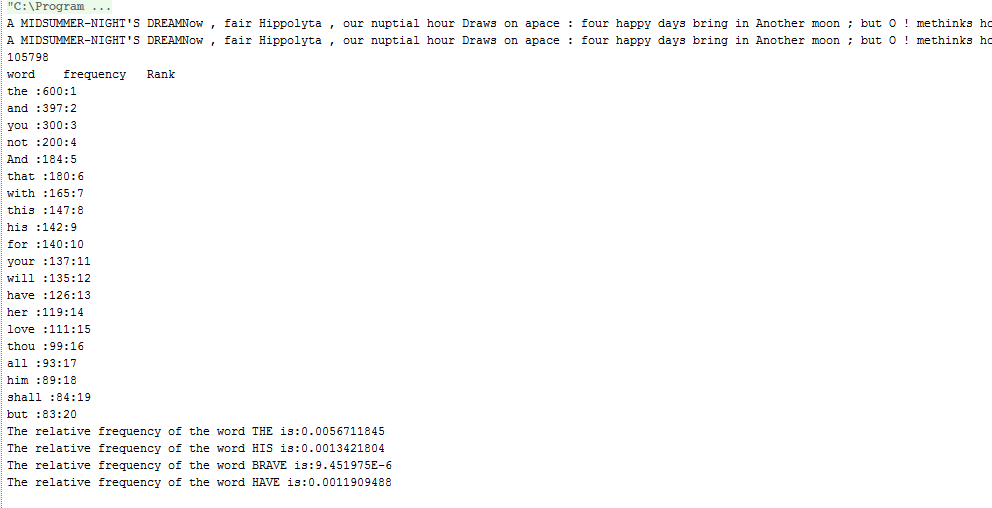
*/\*\*  
 \* Created by asmita on 6/3/2016.  
 \*/***import** java.io.BufferedReader;  
**import** java.io.File;  
**import** java.io.FileReader;  
**import** java.util.\*;  
**import** java.util.Set;  
**import** java.util.List;  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
**public class** Probability {  
 **public static void** main(String[] args) **throws** Exception  
 {  
 *// System.out.println("aSHMITA");* FileReader file = **new** FileReader(**"C:/Users/asmita/Desktop/Text1.txt"**);  
  
 BufferedReader reader= **new** BufferedReader(file);  
 String text=**""**;  
 String line=reader.readLine();  
 **while**(line!=**null**){  
 text+=line;  
 line=reader.readLine();  
 }  
 System.***out***.println(text);  
 String text1=text.replace(**"."**, **" "**);  
 System.***out***.println(text1);  
 String text2[]=text1.split(**"[(' ),.!;-]"**);  
 **int** totalWords=text.length();  
 System.***out***.println(totalWords);  
 HashMap<String, Integer> hasmap= **new** HashMap<String, Integer>();  
 *//System.out.println("hghg");* **for** (String word:text2)  
 {  
 **if**(word.length()<=2){  
 **continue**;  
 }  
 Integer existingCount=hasmap.get(word);  
 *//System.out.println(existingCount);* hasmap.put(word,existingCount==**null**?1:(existingCount+1));  
 *//Collections.sort(existingCount);* }  
  
  
 Map<Integer, String> map = *sortByValues*(hasmap);  
 *//System.out.println("After Sorting:");  
 //int i=0;* **int** m=0;  
 Set set2 = map.entrySet();  
 Iterator iterator2 = set2.iterator();  
 *// while(iterator2.hasNext())* System.***out***.println(**"word\t"**+ **"frequency\t"**+**"Rank"**);  
 **while**(m<20){  
 Map.Entry me2 = (Map.Entry)iterator2.next();  
  
 System.***out***.print(me2.getKey() + **" :"**);  
 System.***out***.print(me2.getValue()+**":"**);  
 System.***out***.println(++m);  
  
 }  
  
 **float** countThe=(hasmap.get(**"the"**));  
 *//float countBecome=(hasmap.get("become"));* **float** countBrave=(**float**)(hasmap.get(**"brave"**));  
 *//float countTreason=hasmap.get("treason");  
 // Calculating the relative frequency (probability estimate) of the words:* System.***out***.println(**"The relative frequency of the word THE is:"** + countThe/totalWords);  
 *//System.out.println("The relative frequency of the word THE is:" + countBecome/totalWords);* System.***out***.println(**"The relative frequency of the word BRAVE is:"** + countBrave/totalWords);  
 *//System.out.println("The relative frequency of the word TREASON is:" + countTreason/totalWords);  
  
  
  
  
/\*  
  
 Set<Map.Entry<String, Integer>> uniqueSet=hasmap.entrySet();  
 //List<Object> list = new ArrayList<Object>(uniqueSet);  
  
 for (Map.Entry<String, Integer> entry:uniqueSet)  
 {  
 if (entry.getValue()>1)  
 {  
 System.out.println(entry.getKey()+" "+entry.getValue());  
 }  
  
 }  
\*/* }  
 **private static** HashMap sortByValues(HashMap map) {  
 List list = **new** LinkedList(map.entrySet());  
 *// Defined Custom Comparator here* Collections.*sort*(list, **new** Comparator() {  
 **public int** compare(Object o1, Object o2) {  
 **return** ((Comparable) ((Map.Entry) (o2)).getValue())  
 .compareTo(((Map.Entry) (o1)).getValue());  
 }  
 });  
  
 *// Here I am copying the sorted list in HashMap  
 // using LinkedHashMap to preserve the insertion order* HashMap sortedHashMap = **new** LinkedHashMap();  
 **for** (Iterator it = list.iterator(); it.hasNext();) {  
 Map.Entry entry = (Map.Entry) it.next();  
 sortedHashMap.put(entry.getKey(), entry.getValue());  
 }  
 **return** sortedHashMap;  
 }  
  
  
}

OUTPUT



Note: Since the text file you provided us was very large which hanged by computer every time, therefore I used certain portion of it.