Name Abinet Kenore

CS 2050

HW #3

Chapter #3.1

3.1.8

Max)word)is)Max)Count)7515

distinct)=)19695

words)=)142707

3.1.6

Calculation)for)Putters)is)#)Words)+)one)for)MAX)Ð)Words)<)minlen)

Calculation)for)Getters)is)#)Words)+)1)+)#Distinct)Words)+)1)Ð)minlen

Putters)139044)Getters)158740

Number)of)Words)<)minlen)3664

3.1.8

Length)of)1)Cutoff)Max)Word)the)7515)

Length)of)8)Cutoff)Max)Monsieur)93

Length)of)10)Cutoff)Max)GutenbergUtm 53

3.1.9

Length 1 prior)Word Count 139042 Last Word***

Length 8 prior Word Count 18225 Last Word newsletter***

Length 10 prior Word Count 6694 Last Word newsletter***

3.1.19 lists of top ten words

He = 7515

and = 4751

of = 4071

to = 3458

a = 2830

in = 2447

his = 1911

was = 1675

that = 1663

I = 1451

Source Codes for Selected problems.

```
1 import java.io.File;
2 import java.io.FileNotFoundException;
3 import java.io.*;
4 import java.util.*;
5
6
7 public class CS2Algs316FC {
8
9   // Do not instantiate.
10   private CS2Algs316FC() { }
11
12       String[] maxWord = new String[10];
13      int[] maxCntr = new int[10];
14      int maxWordLen = 0;
15      int cntM = 0;
16      int diffOfLen = 0;
17      int loopCtr = 0;
18      int wordCnt = 0;
19
20      String key;
21      int keyleng;
22
23      String fileName="src/tale.txt";
24      int wordLen = 0;
25      String wordTxt = "";
26      /*String filename = "Ak3.dat"; // a path name may be necessary
27    StringTokenizer st;
28    PrintWriter outputFile = new PrintWriter ("ak3.out");
29    outputFile.println("output File");*/
30      // String fileName="src/tinyTale.txt";
31
32      //Instantiate the BufferedReader Class
33      try (Scanner inFile = new Scanner(new FileReader (filename));
34      //BufferedReader in = new BufferedReader(new FileReader(fileName)))
35      {
36            String line;
37            while ((line = in.readLine()) != null)
38            {
39               String[] pieces = line.split("\\s+");
40
41               for (String pieceKey : pieces) {
42                  words++;
43
44               if (pieceKey.length() < minlen) {
45                  wordCnt ++;
46                  continue;
47               }
48               if (st.contains(pieceKey)) {
49                  getters ++;
50                  putters ++;
51                  lastWord = pieceKey;
52                  priorCnt ++;
53
54                  st.put(pieceKey, st.get(pieceKey) + 1);
55               }
56               else {
```

```
                    lastWord = pieceKey;
                    priorCnt ++;
                    st.put(pieceKey, 1);
                    putters ++;
                    distinct++;
                 }
              }
           }   // find a key with the highest frequency count
           String max;
           putters ++;
           for (cntM = 0; cntM < 10; cntM++) {
              max = "";
              st.put(max, 0);
              for (String word : st.keys()) {

                 if (st.get(word) > st.get(max))
                    max = word;
              }   // System.out.println(max + " " + st.get(max));
              maxWord[cntM] = max;
              if (maxWordLen < max.length())
                 maxWordLen = max.length();
              maxCntr[cntM] = st.get(max);
              st.delete(max);
           }
           maxWordLen += 4;
           System.out.println("Word Count for words < minlen  " + wordCnt + "\n");
           priorCnt --;
           // System.out.println("Last Word " + lastWord + " Prior Cnt " + priorCnt + "\n");
           System.out.println("List Top 10 words and their counts:");
           for (cntM = 0; cntM < 10; cntM++) {
                wordTxt = maxWord[cntM];
                wordLen = wordTxt.length();
                diffOfLen = maxWordLen - wordLen;
                System.out.print(maxWord[cntM]);
                for (loopCtr = 1; loopCtr <= diffOfLen; loopCtr++)
                   System.out.print(" ");
                System.out.println(maxCntr[cntM]);
           }

           in.close();
        }
        catch (Exception e)
        {
        System.err.format("Exception occurred trying to read '%s': " + e.getMessage(),        fileName);
           e.printStackTrace();
           // return null;
         }
     }
 }


          1 import java.util.*;
 2
 3 public class CS2Algs316ST<Key extends Comparable<Key>, Value> implements Iterable<Key> {
 4
 5    private TreeMap<Key, Value> st;
 6
 7    public CS2Algs316ST() {
 8       st = new TreeMap<Key, Value>();
 9    }
10
11
12
```

```
13  public Value get(Key key) {
14     if (key == null) throw new NullPointerException("called get() with null key");
15     return st.get(key);
16  }
17
18
19  public void put(Key key, Value val) {
20     if (key == null) throw new NullPointerException("called put() with null key");
21     if (val == null) st.remove(key);
22     else             st.put(key, val);
23  }
24
25
26  public void delete(Key key) {
27     if (key == null) throw new NullPointerException("called delete() with null key");
28     st.remove(key);
29  }
30
31
32  public boolean contains(Key key) {
33     if (key == null) throw new NullPointerException("called contains() with null key");
34     return st.containsKey(key);
35  }
36
37
38  public int size() {
39     return st.size();
40  }
41
42     public boolean isEmpty() {
43     return size() == 0;
44  }
45
46
47  public Iterable<Key> keys() {
48     return st.keySet();
49  }
50
51  @Deprecated
52  public Iterator<Key> iterator() {
53     return st.keySet().iterator();
54  }
55  public Key max() {
56     if (isEmpty()) throw new NoSuchElementException("called max() with empty symbol table");
57     return st.lastKey();
58  }
59
60
61  public Key ceiling(Key key) {
62     if (key == null) throw new NullPointerException("called ceiling() with null key");
63     Key k = st.ceilingKey(key);
64     if (k == null) throw new NoSuchElementException("all keys are less than " + key);
65     return k;
66  }
67
68
69  public Key floor(Key key) {
70     if (key == null) throw new NullPointerException("called floor() with null key");
71     Key k = st.floorKey(key);
72     if (k == null) throw new NoSuchElementException("all keys are greater than " + key);
73     return k;
74  }
75
```

```
76
77    public static void main(String[] args) {
78       Scanner in = new Scanner(System.in);
79       ST<String, Integer> st = new ST<String, Integer>();
80       String key;
81       // key = in.nextLine( )).length( ) > 0
82       for (int i = 0; (key = in.next( )).length( ) > 0; i++) {
83          // key = StdIn.readString();
84          String[] parts = key.split("\\s+");
85          for (String partKey : parts)
86              st.put(partKey, i);
87       }
88       // for (String s : st.keys())
89          // System.out.println(s + " " + st.get(s));
90    }
91 }
```