Output:

Processing will take few seconds...

Longest word: electroencephalographically

Second Longest word: immunoelectrophoretically

Number of words that can be formed: 50603

Code:

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

/\*\*

\*

\* @author Sachin Shinde

\* 9th April 2015

\*/

/\*

\* This Program scans a text file with large number of words and finds the longest and second longest words that can be formed by other words in the file.

\* It also finds the count of all those words which can be formed by other words in the file.

\* The program assumes that least word length is 2

\*/

public class WordProblem {

public static void main(String[] args) {

String wordFile = "wordsforproblem.txt"; // File Name

String line = null; // Holds the line from the file

String tempWordHolder=null; // Holds the word. As a temporary location

String reference = "abcdefghijklmnopqrstuvwxyz"; // Reference String to used to store the starting addresses of words

StringBuffer st = new StringBuffer();

FileInputStream fis = null;

BufferedReader reader = null;

ArrayList wordsList = new ArrayList(); // Contains the words list from the file

ArrayList hashingList = new ArrayList(); // Holds 2 charachters for reference eg: aa, ab, ac..

int[] wordIndex = new int[676]; // stores the starting address of the words

int longestWordIndex = 0; // Index of longest word in the Arraylist

int secondLongestWordIndex = 0; // Index of second longest word in the Arraylist

int validWordsCount = 0; // Count of words which can be formed by other words in the file

int wordListSize = 0; // Size of the wordslist

int i=1;

int j=0;

List longestWords = new LinkedList(); // It will hold the longest words. first 2 for this program

longestWords.add("xx"); // garbage value

longestWords.add("y"); // garbage value to avoid nullpointer exception

System.out.println("Processing will take few seconds...");

try {

fis = new FileInputStream(wordFile);

reader = new BufferedReader(new InputStreamReader(fis));

line = reader.readLine(); // reading the first line

// This loop prepares the list of starting 2 letters of the words eg: aa,ab,ac. To use it for storing the addresses of the words in the next loop

for(int k=0;k<reference.length();k++){

for(int l=0;l<reference.length();l++){

st.append(reference.charAt(k));

st.append(reference.charAt(l));

hashingList.add(st.substring(0, 2));

st.replace(0, st.length(), "");

}

}

// loop to process the file

while(line != null){

wordsList.add(line.trim());

wordIndex[hashingList.indexOf(line.toString().substring(0,2))]=wordsList.size()-1;

line = reader.readLine();

}

// Loop to search for the words

while(i<wordsList.size()){

j=i-1;

tempWordHolder=wordsList.get(i).toString();

if(wordsList.get(i).toString().length()<=2){

i++;

continue;

}

while(j>=0){

if(tempWordHolder.length() < wordsList.get(j).toString().length()){

j--;

continue;

}

if(tempWordHolder.contains(wordsList.get(j).toString())){

// If found then removes the word from the word under consideration

tempWordHolder=tempWordHolder.replaceAll(wordsList.get(j).toString(),"");

if(tempWordHolder.length() > 1){

// getting the index of the last word which starts from the first letter of the word under consideration

j=wordIndex[hashingList.indexOf(tempWordHolder.toString().substring(0,2))];

}

}

if((tempWordHolder.length() <= 1) || !(tempWordHolder.toString().substring(0,1).equals(wordsList.get(j).toString().substring(0,1)))) {

break;

}

j--;

}

// If the final length of the string is 0 then it is a valid word and can be counted in validWordsCount

if(tempWordHolder.length()==0){

validWordsCount++;

if(longestWords.get(0).toString().length() < wordsList.get(i).toString().length()){

longestWords.add(0, wordsList.get(i).toString());

}

else if(longestWords.get(0).toString().length() != wordsList.get(i).toString().length()){

if(longestWords.get(1).toString().length() < wordsList.get(i).toString().length()){

longestWords.add(1, wordsList.get(i).toString());

}

}

}

i++;

}

// Printing the required Data

System.out.println("Longest word: "+longestWords.get(0));

System.out.println("Second Longest word: "+longestWords.get(1));

System.out.println("Number of words that can be formed: "+validWordsCount);

}catch (FileNotFoundException e) {

System.out.println("File not found");

}

catch (IOException e) {

e.printStackTrace();

}

}

}