**University of the People**

**Programming-2**

**Solutions for Assignment Unit-5**

import java.io.\*;  
import java.util.\*;

import javax.swing.JFileChooser;

public class Dictionary {

 public static void main(String args[]) throws FileNotFoundException{

                try {  
                    // Reading the words.txt file.  
                    Scanner filein = new Scanner(new File ("words.txt"));

                    // Creatin the new dictionary data structure  
                    HashSet<String> hash = new HashSet();

                    while (filein.hasNext()) {  
                     String tk = filein.next();

                        //Adding words into dictionary from "words.txt"  
   hash.add(tk.toLowerCase());  
                    }

                    Scanner userFile = new Scanner(getInputFileNameFromUser());  
                    userFile.useDelimiter("[^a-zA-Z]+");

                    while (userFile.hasNext()){  
                         String two = userFile.next();  
                                String two1 = two.toLowerCase();  
                                if(!hash.contains(two1)){  
            System.out.println(two1 + ":" + corrections(two1, hash));  
                     }  
                    }  
                }  
                 catch (IOException e) {  
                    System.out.println("File not found - words.txt");  
  }  
 }

 /\*\*  
  \* Stores the variations in a TreeSet, then  
  \* Returns the possible variations on the misspelled word.  
  \* Since the corrections are stored in a tree set, they are automatically  
  \* printed out in alphabetical order with no repeats.  
         \*  
         \* The possible corrections that the program considers are as follows:  
         \*  
         \* Delete any one of the letters from the misspelled word.  
      \* Change any letter in the misspelled word to any other letter.  
      \* Insert any letter at any point in the misspelled word.  
      \* Swap any two neighboring characters in the misspelled word.  
      \* Insert a space at any point in the misspelled word (and check that both of the words that are produced are in the dictionary)  
  \*/

        static TreeSet corrections(String badWord, HashSet dictionary){

                TreeSet<String> tree = new TreeSet<String>();

  //Delete any one of the letters from the misspelled word, then check if exist in the dictionary.

                for (int i=0; i<badWord.length(); i++){  
   String s = badWord.substring(0,i) + badWord.substring(i+1);  
   if(dictionary.contains(s)){  
    tree.add(s);  
   }  
  }

  //Change any letter in the misspelled word to any other letter , then check if exist in the dictionary.

                for (int i=0; i<badWord.length(); i++){  
   for (char ch = 'a'; ch <= 'z'; ch++) {  
    String s = badWord.substring(0,i) + ch + badWord.substring(i+1);  
    if(dictionary.contains(s)){  
     tree.add(s);  
    }  
   }  
  }  
  //Insert any letter at any point in the misspelled word ,then check if exist in the dictionary.

                for (int i=0; i<=badWord.length(); i++){  
   for (char ch = 'a'; ch <= 'z'; ch++) {  
    String s = badWord.substring(0,i) + ch + badWord.substring(i);  
    if(dictionary.contains(s)){  
     tree.add(s);  
    }  
   }  
  }

  //Swap any two neighboring characters in the misspelled word, then check if exist in the dictionary.

                for(int i=0; i< badWord.length()-1; i++){  
   String s = badWord.substring(0,i)+ badWord.substring(i+1, i+2) + badWord.substring(i,i+1)+ badWord.substring(i+2);  
   if(dictionary.contains(s)){  
    tree.add(s);  
   }  
  }

  //Insert a space at any point in the misspelled word (and check that  
  //both of the words that are produced are in the dictionary)

  for (int i=1; i<badWord.length(); i++){  
   String stringInput = badWord.substring(0,i) + " " + badWord.substring(i);  
   String tempString = "";

                        //break a string into tokens and add it to tempWords  
                        StringTokenizer tempWords = new StringTokenizer(stringInput);

   //Loop over all words in tempWords.  
                        while(tempWords.hasMoreTokens()){  
                                //Store each word in a temp string.  
    String stringWord1 = tempWords.nextToken();  
    String stringWord2 = tempWords.nextToken();  
    //Check if temp words exist in dictionary otherwise break  
                                if(dictionary.contains(stringWord1) && dictionary.contains(stringWord2)){  
     tempString = stringWord1 + " " + stringWord2;  
    }  
    else  
     break;  
    tree.add(tempString);  
   }  
  }

  if(tree.isEmpty()){  
   tree.add("no suggestions");  
  }  
  return tree;  
 }

 /\*\*  
  \* Lets the user select an input file using a standard file  
  \* selection dialog box.  If the user cancels the dialog  
  \* without selecting a file, the return value is null.  
  \*/  
 static File getInputFileNameFromUser() {  
  JFileChooser fileDialog = new JFileChooser();  
  fileDialog.setDialogTitle("Select File for Input");  
  int option = fileDialog.showOpenDialog(null);  
  if (option != JFileChooser.APPROVE\_OPTION)  
   return null;  
  else  
   return fileDialog.getSelectedFile();  
 }  
}