**package** com.google.engedu.anagrams;  
  
**import** android.util.Log;  
  
**import** java.io.BufferedReader;  
**import** java.io.IOException;  
**import** java.io.InputStream;  
**import** java.io.InputStreamReader;  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.HashMap;  
**import** java.util.HashSet;  
**import** java.util.Random;  
  
**public class** AnagramDictionary {  
  
 **private static final int *MIN\_NUM\_ANAGRAMS*** = 5;  
 **private static final int *DEFAULT\_WORD\_LENGTH*** = 3;  
 **private static final int *MAX\_WORD\_LENGTH*** = 7;  
 **private static int** *wordLength* = ***DEFAULT\_WORD\_LENGTH***;  
 **private** Random **random** = **new** Random();  
  
 HashSet<String> **wordSet** = **new** HashSet<String>();  
 ArrayList<String> **wordList** = **new** ArrayList<String>();  
 HashMap<String,ArrayList<String>> **lettersToWord** = **new** HashMap<String,ArrayList<String>>();  
 HashMap<Integer,ArrayList<String>> **sizeToWord** = **new** HashMap<Integer,ArrayList<String>>();  
  
 **public** AnagramDictionary(InputStream wordListStream) **throws** IOException {  
 BufferedReader in = **new** BufferedReader(**new** InputStreamReader(wordListStream));  
 String line;  
 **while**((line = in.readLine()) != **null**) {  
 String word = line.trim();  
  
 **wordSet**.add(word);  
 **wordList**.add(word);  
  
 ArrayList<String> temp\_words1 = **new** ArrayList<String>();  
 ArrayList<String> temp\_words2 = **new** ArrayList<String>();  
  
 **int** l = word.length();  
 **if**(**sizeToWord**.containsKey(l))  
 {  
 temp\_words2 = **sizeToWord**.get(l);  
 temp\_words2.add(word);  
 **sizeToWord**.put(l,temp\_words2);  
 }  
 **else** {  
 temp\_words2.add(word);  
 **sizeToWord**.put(l,temp\_words2);  
 }  
  
 String sortWord;  
 sortWord = alphabeticalOrder(word);  
  
 **if**(**lettersToWord**.containsKey(sortWord)){  
  
 temp\_words1 = **lettersToWord**.get(sortWord);  
 temp\_words1.add(word);  
 }  
 **else**{  
 temp\_words1.add(word);  
 **lettersToWord**.put(sortWord,temp\_words1);  
 }  
 }  
 }  
  
 **public** String alphabeticalOrder(String word) {  
 **char**[] charArray = word.toCharArray();  
 Arrays.*sort*(charArray);  
 String newWord = **new** String(charArray);  
  
 **return** newWord;  
 }  
  
 **public boolean** isGoodWord(String word, String base) {  
  
 **if** (**wordSet**.contains(word) && !word.contains(base))  
 **return true**;  
 **else  
 return false**;  
  
 }  
  
 **public** ArrayList<String> getAnagramsWithOneMoreLetter(String word) {  
 ArrayList<String> result = **new** ArrayList<String>();  
  
 String newWord;  
  
 **for**(**char** c = **'a'**; c<= **'z'**; c++){  
 newWord = c + word;  
 newWord = alphabeticalOrder(newWord);  
  
 **if**(**lettersToWord**.containsKey(newWord)){  
 result.addAll(**lettersToWord**.get(newWord));  
 }  
 }  
  
 **for**(**int** i = result.size() -1; i>=0;i--){  
 Log.*d*(**"AD list "**,result.get(i));  
 **if**(!isGoodWord(result.get(i),word)) {  
  
 Log.*d*(**"AD removed "**,result.remove(i));  
 }  
 }  
  
 **return** result;  
 }  
  
 **public** String pickGoodStarterWord() {  
  
 String word = **new** String();  
  
 **int** j;  
  
 ArrayList<String> lengthWords = **new** ArrayList<>();  
  
 **if**(*wordLength* <= ***MAX\_WORD\_LENGTH***){  
 lengthWords = **sizeToWord**.get(*wordLength*);  
 }  
  
 **int** i = **random**.nextInt(lengthWords.size());  
  
 **for**(j = i; j < lengthWords.size(); j++) {  
  
 **if**(getAnagramsWithOneMoreLetter(lengthWords.get(j)).size() >= ***MIN\_NUM\_ANAGRAMS***)  
 {  
 Log.*d*(**"word "**,lengthWords.get(j));  
 word = lengthWords.get(j);  
 **break**;  
 }  
 }  
  
 **if**(j == lengthWords.size() - 1 && word == **null**) {  
  
 **for** (j = 0; j < i; j++) {  
 **if** (getAnagramsWithOneMoreLetter(lengthWords.get(j)).size() >= ***MIN\_NUM\_ANAGRAMS***) {  
 word = lengthWords.get(j);  
 **break**;  
 }  
 }  
  
 }  
  
 **if**(*wordLength* < ***MAX\_WORD\_LENGTH***)  
 *wordLength*++;  
  
 **return** word;  
  
 }  
}