import java.io.\*;

import java.util.\*;

public class autocomplete {

class Trie {

public boolean isEnd;

public Map<String, Integer> frequency;

public Map<Character, Trie> children;

public Trie() {

this.frequency = new HashMap<>();

this.children = new HashMap<>();

}

}

class Pair {

String s;

int count;

public Pair(String s, int count) {

this.s = s;

this.count = count;

}

}

Trie root, curr;

StringBuffer sb;

public autocomplete(String[] sentences, int[] times) {

if (sentences == null || times == null || sentences.length != times.length) return;

reset();

root = new Trie();

for (int i = 0; i < times.length; i++) {

insert(sentences[i], times[i]);

}

}

public List<String> input(char c) {

List<String> new1 = new ArrayList<>();

if (curr == null) curr = root;

sb.append(c);

curr.children.putIfAbsent(c, new Trie());

curr = curr.children.get(c);

new1.addAll(findTopK(curr, 10));//search is limitted to top 10

return new1;

}

public List<String> string\_search(String s2)

{

char[] s= s2.toCharArray();

List<String> ans= new ArrayList<String>();

for(char s1 : s)

{

ans=input(s1);

}

return ans;

}

private List<String> findTopK(Trie node, int k) {

List<String> rst = new ArrayList<>();

if (node.frequency.isEmpty()) return rst;

PriorityQueue<Pair> queue = new PriorityQueue<>(

(a, b) -> a.count == b.count ? b.s.compareTo(a.s) : a.count - b.count);

for (Map.Entry<String, Integer> entry : node.frequency.entrySet()) {

if (queue.size() < 10 || entry.getValue() >= queue.peek().count) {

queue.offer(new Pair(entry.getKey(), entry.getValue()));

}

if (queue.size() > 10) queue.poll();

}

while (!queue.isEmpty()) {

rst.add(0, queue.poll().s);

}

return rst;

}

private void reset() {

curr = null;

sb = new StringBuffer();

}

public void insert(String sentence, int count) {

if (sentence == null || sentence.length() == 0) return;

Trie node = root;

for (char c : sentence.toCharArray()) {

node.children.putIfAbsent(c, new Trie());

node = node.children.get(c);

node.frequency.put(sentence, node.frequency.getOrDefault(sentence, 0) + count);

}

node.isEnd = true; //can set word to node as well, if needed

}

public static void main(String[] args)

{

String[] temp1= {"blue jeans","blue shirt","blue pyjama","red jeans","red jeans","denim jeans","denim shirt", "henley jeans","henley Tshirt","Mens Polo","Melange Tshirt","Track Pants","grey shirt","grey jeans","slim fit jeans","slim fit Tshirt","slim fit denim","slim fit solid jeans","black women leggings","blue leggings","black leggings","white Tshirt","navy blue jeans","white shirt","black denim jeans","grey solid tshirt","blue solid jeans"};

int[] temp2={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1};

autocomplete obj = new autocomplete(temp1,temp2);

List<String> lists= obj.string\_search("bl");

for( String it : lists)

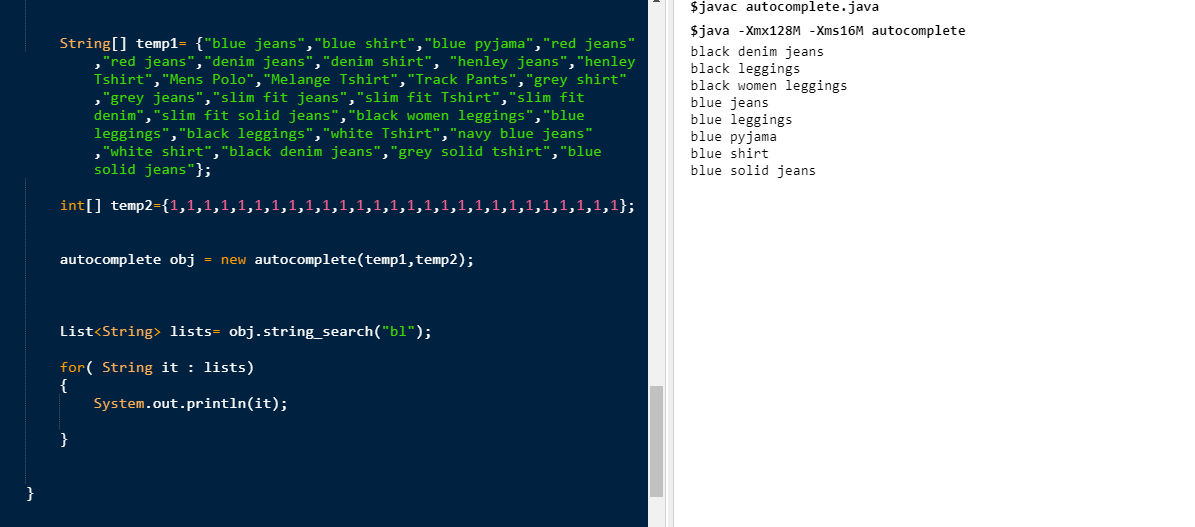
{

System.out.println(it);

}

}

}



import com.google.gson.Gson;  
import java.io.File;  
import java.io.IOException;  
import java.util.Scanner;  
  
public class JsonParser {  
 private static String[] tokenizeString(String commaSeparatedString) {  
 String [] tokens = commaSeparatedString.split(",");  
 for (int idx = 0; idx < tokens.length; idx++) {  
 tokens[idx] = tokens[idx].trim();  
 }  
 return tokens;  
 }  
 public static void main(String args[])throws IOException {  
 String filename = "M:\\FILES\\work\\Important Docs\\INTERN 2020\\Bigphi\\product.json";  
 String content = new Scanner(new File(filename)).useDelimiter("\\Z").next();  
 Gson gson = new Gson();  
 ProductDatatype[] target = gson.fromJson(content, ProductDatatype[].class);  
 for (ProductDatatype entry: target) {  
 String title = entry.title;  
 String product\_type = entry.product\_type;  
 String[] tags = *tokenizeString*(entry.tags);  
 String[] searchTerms = *tokenizeString*(entry.searchTerms);  
 }  
 }  
}