package binaryproject;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

/\*\*

\*

\* @author lucasmarques

\*

\*/

public class BinaryProject {

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic

morseCode mc = new morseCode();

TreeNode tree = new TreeNode(); //create the first and empty node of the tree

mc.create("•-", 'A', tree); // the next lines will add the alphabet into the tree

mc.create("-•••", 'B', tree);

mc.create("-•-•", 'C', tree);

mc.create("-••", 'D', tree);

mc.create("•", 'E', tree);

mc.create("••-•", 'F', tree);

mc.create("--•", 'G', tree);

mc.create("••••", 'H', tree);

mc.create("••", 'I', tree);

mc.create("•---", 'J', tree);

mc.create("-•-", 'K', tree);

mc.create("•-••", 'L', tree);

mc.create("--", 'M', tree);

mc.create("-•", 'N', tree);

mc.create("---", 'O', tree);

mc.create("•--•", 'P', tree);

mc.create("--•-", 'O', tree);

mc.create("•-•", 'R', tree);

mc.create("•••", 'S', tree);

mc.create("–", 'T', tree);

mc.create("••-", 'U', tree);

mc.create("•••-", 'V', tree);

mc.create("•--",'W',tree);

mc.create("-••-", 'X',tree);

mc.create("-•--", 'Y',tree);

mc.create("--••", 'Z',tree);

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String code = br.readLine(); //read the morse code

String result = mc.morseTranslate(code, tree); //translate the morse code

String fresult = new String();

String[] words = result.split(" ");

fresult = words[0].charAt(0) + words[0].substring(1).toLowerCase();

for(int i = 1; i < words.length ; i++){ //capitalize first letter of each word

fresult = fresult + " " + words[i].charAt(0) + words[i].substring(1).toLowerCase();

}

System.out.println(fresult);

}

}

package binaryproject;

import java.util.stream.IntStream;

/\*\*

\*

\* @author lucasmarques

\*/

public class morseCode {

public String morseTranslate(String code, TreeNode tree){

int pos = 0;

String uCode = code.toUpperCase();

String result = new String();

char[] morsecode = uCode.toCharArray();

TreeNode aux = tree; //tree always will be the beggining of the tree, value never changed

while(morsecode[pos] != '#'){

if(morsecode[pos] == '•'){ //if left

aux = aux.l; //walk to the tree through aux

}else if(morsecode[pos] == '-'){ //if right

aux = aux.r;

}else if(morsecode[pos] == ' '){//end of a letter

if(morsecode[pos+1] == ' '){//space in the code

result = result + aux.letter + " "; //add letter + space

pos++;

}else{ //case isn't a space

result = result + aux.letter; //add letter

}

aux = tree; //back to begging of the tree

}

pos++;

}

result = result + aux.letter;// add the last letter, because the while just add the character after the end of the letter

return result;

}

public void create(String code, char letter, TreeNode t){

char[] a = code.toCharArray();

if(code.length() > 1){ //we haven't find where to place the letter

if(a[0] == '•'){

if(t.l == null){ //if the node doesn't exist then we create a node in the left.

TreeNode left = new TreeNode();

t.l = left;

}

this.create(code.substring(1), letter, t.l); //call the function again with 1 less character and in the next node

}else{

if(t.r == null){//if the node doesn't exist then we create a node in the right.

TreeNode right = new TreeNode();

t.r = right;

}

this.create(code.substring(1), letter, t.r); //same of call function above but now to the right

}

}else{ //else it's the last character in the string and we now where to place the letter

if(a[0] == '•'){ //place in the left

if(t.l == null){

TreeNode left = new TreeNode();

t.l = left;

}

t.l.letter = letter;

}else{ //place in the right

if(t.r == null){

TreeNode right = new TreeNode();

t.r = right;

}

t.r.letter = letter;

}

}

}

}

package binaryproject;

/\*\*

\*

\* @author lucasmarques

\*/

public class TreeNode {

char letter;

TreeNode l = null, r = null;

}

Screenshots - Letters separated by one space, words separated by two spaces





