// ------------------------------------------

// Author: Lauren Escobedo

// Assignment: Chapter 14 Problem 14.17

// Date: 03/22/2023

// Language: Java

// File Name: Exercise\_14\_17.java

// Description: Exercise 14.17

// - Three letter string from five letter word

// ------------------------------------------

import java.util.ArrayList;

import java.util.Scanner;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.URL;

public class Exercise\_14\_17 {

// Instance variables

private ArrayList<Character> letters = new ArrayList<>();

private ArrayList<String> combinations = new ArrayList<>();

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

// Variables

Scanner input = new Scanner(System.in);

Exercise\_14\_17 driver = new Exercise\_14\_17();

// Get user input, truncate to 5 letters if longer

System.out.println("Please enter a five letter word: ");

String word = input.nextLine();

word = word.substring(0, 5);

// Get the input in the char list

for (char letter : word.toCharArray()) {

driver.letters.add(letter);

}

// Get the combinations in the combo list

driver.fillCombinations(driver.letters, driver.combinations);

// Check if real words

driver.checkCombinations(driver.combinations);

// Close resources

input.close();

}

private void checkCombinations(ArrayList<String> combinations) {

// Set variables

String url = "https://www.merriam-webster.com/dictionary/";

int flag = 0;

System.out.println("\n");

// Iterate through array list to check all combinations

for (String combo : combinations) {

// Set full url based on which combo we're on

url = url + combo;

// Go online and see if word is real

try (BufferedReader reader = new BufferedReader(new InputStreamReader(new URL(url).openStream()))) {

System.out.printf("This combo is a real word (or abbreviation): %s\n", combo);

flag = 1;

} catch (IOException ex) {

System.out.print("");

}

}

// Defualt message

if (flag == 0) {

System.out.println("This word does not have any combinations that are real words. :(");

}

System.out.println("\n\n");

}

private void fillCombinations(ArrayList<Character> letters, ArrayList<String> combinations) {

// Brute force combinations for the word and add to combinations list

for (int i = 0; i < 5; i++) {

for (int j = i + 1; j < 5; j++) {

for (int k = j + 1; k < 5; k++) {

int flag = 0;

// Compose the combination

String combo = letters.get(i).toString() + letters.get(j).toString() + letters.get(k).toString();

// Check if it's already accounted for

for (String word : combinations) {

if (combo == word) {

flag = 1;

break;

}

}

// If not accounted for, add

if (flag == 0) {

combinations.add(combo);

}

}

}

}

// Output all combinations

System.out.print("\n\nAll combinations for the word are...\n");

for (String combo : combinations) {

System.out.println(combo);

}

}

}

// ------------------------------------------

// Author: Lauren Escobedo

// Assignment: Chapter 14 Problem 14.18

// Date: 03/22/2023

// Language: Java

// File Name: Exercise\_14\_18.java

// Description: Exercise 14.18

// - Text Analysis

// ------------------------------------------

import java.util.Scanner;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class Exercise\_14\_18 {

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

Exercise\_14\_18 driver = new Exercise\_14\_18();

Scanner input = new Scanner(System.in);

System.out.println("\n\nPlease enter a sentence for analysis:");

if (input.hasNextLine()) {

String sentence = input.nextLine();

driver.letterCount(sentence);

driver.wordLengthCount(sentence);

driver.wordCount(sentence);

}

input.close();

System.exit(0);

}

private void letterCount(String sentence) {

// Variables

ArrayList<Character> phrase = new ArrayList<>();

Map<Character,Integer> alphabet = new HashMap<Character,Integer>();

// Format sentence

sentence = sentence.trim().replaceAll("\\s", "");

// Get sentence down to it's characters

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

phrase.add(letter);

}

// Count characters

for (char letter : phrase) {

Integer count = alphabet.get(letter);

int newCount = (count==null ? 1 : count+1);

alphabet.put(letter, newCount);

}

// Print the map

System.out.println("\nEach appearing letter and count of appearances: ");

for (Map.Entry<Character, Integer> letter : alphabet.entrySet()) {

System.out.println(letter.getKey() + ":" + letter.getValue().toString());

}

}

private void wordLengthCount(String sentence) {

// Variables

int[] count = new int[10];

// Split on space

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

// Count them up

for (String word : words) {

switch(word.length()) {

case 1:

count[0] ++;

break;

case 2:

count[1] ++;

break;

case 3:

count[2] ++;

break;

case 4:

count[3] ++;

break;

case 5:

count[4] ++;

break;

case 6:

count[5] ++;

break;

case 7:

count[6] ++;

break;

case 8:

count[7] ++;

break;

case 9:

count[8] ++;

break;

case 10:

count[9] ++;

break;

default:

break;

}

}

// Start table

System.out.printf("\n%13s%2s|%11s%4s\n", "Word Length", "", "Letters", "");

System.out.println("------------------------------");

// Print data

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

if (count[i] > 0) {

System.out.printf("%8d%7s|%8d\n", i+1, "", count[i]);

System.out.println("------------------------------");

}

}

}

private void wordCount(String sentence) {

// Variables

Map<String,Integer> wordCount = new HashMap<String,Integer>();

// Split on space

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

// Count the words

for (String w : words) {

Integer count = wordCount.get(w);

int newCount = (count==null ? 1 : count+1);

wordCount.put(w, newCount);

}

// Print the map

System.out.println("\nEach appearing word and count of appearances: ");

for (Map.Entry<String, Integer> word : wordCount.entrySet()) {

System.out.println(word.getKey() + ":" + word.getValue().toString());

}

System.out.println();

}

}

Text

Description automatically generated

Text

Description automatically generated