**TITLE PAGE**

**Course:** CS1073

**Section:** FR03B

**Assignment number:** 6

**Name:** Zohaib Hassan Khan

**UNB student number:** 3740572

Anagram.java:

/\*\*

This is a GUI application program for an anagram tester.

@author Zohaib Khan 3740572

\*/

import javafx.application.Application;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.control.Button;

import javafx.scene.text.Text;

import javafx.scene.layout.FlowPane;

import javafx.geometry.Pos;

import javafx.event.ActionEvent;

import java.util.Scanner;

public class Anagram extends Application {

private TextField wordTextField1, wordTextField2;

private Text result;

public void start (Stage primaryStage) {

primaryStage.setTitle ("Anagram Tester");

Label firstW = new Label ("1st word:");

wordTextField1 = new TextField();

wordTextField1.setPrefWidth(145);

Label secondW = new Label ("2nd word:");

wordTextField2 = new TextField();

wordTextField2.setPrefWidth(145);

Button anagram = new Button ("Are these anagrams?");

anagram.setOnAction(this::anagramProcess);

result = new Text ("Let's test some possible anagrams!");

FlowPane pane = new FlowPane (firstW, wordTextField1,

secondW, wordTextField2,

anagram, result);

pane.setAlignment(Pos.CENTER);

pane.setHgap(40);

pane.setVgap(40);

Scene scene = new Scene (pane, 320, 300);

primaryStage.setScene(scene);

primaryStage.show();

}

public void anagramProcess (ActionEvent event){

String word1 = wordTextField1.getText();

String word2 = wordTextField2.getText();

int counter = 0;

char[] array1 = word1.toLowerCase().toCharArray();

char[] array2 = word2.toLowerCase().toCharArray();

if (word1.length() == word2.length()){

for(int i=0; i<word1.length(); i++){

boolean isAnagram = false;

for(int j=0; j<word2.length() &&

!(isAnagram); j++){

if (array1[i] == array2[j]){

counter++;

isAnagram = true;

array2[j] = (char) 0;

}

}

}

}

if (counter == word1.length()){

result.setText(word2 + " is an anagram of " +

word1);

}

else {

result.setText(word2 + " is not an anagram of " +

word1);

}

// ANOTHER WAY WITHOUT USING ARRAYS:

// word1 = word1.toLowerCase();

// word2 = word2.toLowerCase();

// int counter = 0;

// if (word1.length() == word2.length()){

//

// for (int i=0; i<word1.length(); i++){

//

// boolean isAnagram = false;

//

// for (int j=0; j<word2.length() && !(isAnagram);

// j++){

//

// if(word1.charAt(i) == word2.charAt(j)){

// counter++;

// isAnagram = true;

// word2 = word2.substring(0, j) +

// word2.substring(j+1);

// }

// }

// }

// }

//

// if(counter == word1.length()){

// result.setText(word2 + "is an anagram of " + word1);

// }

// else{

// result.setText(word2 + " is not an anagram of " +

// word1);

// }

}

}

OUTPUT:

Graphical user interface, application

Description automatically generated

Here is a screenshot showing the application when it is first launched:

Graphical user interface, text, application

Description automatically generated

Here is a view after the user has entered two words (that are anagrams) and pressed the "Are these anagrams?" button:

Graphical user interface, application

Description automatically generated

Here is a view after the user has entered two words (that are NOT anagrams) and pressed the "Are these anagrams?" button:

Graphical user interface, application

Description automatically generated

Here is a view after the user has entered two words (that are anagrams) and pressed the "Are these anagrams?" button:

Graphical user interface, application

Description automatically generated

Here is a view after the user has entered two words (that are NOT anagrams) and pressed the "Are these anagrams?" button:

DiceAnalyzer.java:

public abstract class DiceAnalyzer {

public static int getLongestIncreasingSequence (int[] diceRolls) {

int maxSequence = 0;

int currentSequence = 0;

for (int i = 1; i < diceRolls.length; i++) {

if (diceRolls[i] > diceRolls[i-1]) {

currentSequence++;

}

else {

if (currentSequence > maxSequence) {

maxSequence = currentSequence;

}

currentSequence = 1;

}

}

if (currentSequence > maxSequence) {

maxSequence = currentSequence;

}

return maxSequence;

}

public static boolean isTargetSumPossible (int[] diceRolls, int

target) {

boolean isPossible = false;

for (int i = 0; i < diceRolls.length && !(isPossible); i++) {

for (int j = i + 1; j < diceRolls.length && !(isPossible);

j++) {

if(diceRolls[i] + diceRolls[j] == target) {

isPossible = true;

}

}

}

return isPossible;

}

}

Q2Output.txt:

rollsArrayA: 2, 5, 2, 3, 5, 6

longest increasing sequence (should be 4): 4

rollsArrayB: 2, 3, 1, 6, 5, 1, 2, 4, 5, 6, 2, 3, 6

longest increasing sequence (should be 5): 5

rollsArrayC: 6, 2, 4, 3, 1, 2, 3, 3, 5, 2, 4, 4, 5, 5

longest increasing sequence (should be 3): 3

rollsArrayD: 6, 5, 5, 4, 3, 3, 3, 2, 1

longest increasing sequence (should be 1): 1

rollsArrayE: array with length of 0 (no elements)

longest increasing sequence (should be 0): 0

Searching for target total in dice roll pairs:

3 from rollsArrayA (should be false): false

4 from rollsArrayA (should be true): true

5 from rollsArrayA (should be true): true

6 from rollsArrayA (should be false): false

2 from rollsArrayC (should be false): false

7 from rollsArrayC (should be true): true

12 from rollsArrayC (should be false): false

7 from rollsArrayE (should be false): false