NATIONAL COLLEGE OF IRELAND

HIGHER DIPLOMA IN SCIENCE IN COMPUTING

MODULE : SOFTWARE DEVELOPMENT

CA2 REPORT

STUDENT: DANIEL GONZALEZ

STUDENT NUMBER: 21111383

Table of Contents

[QUESTION A 3](#_Toc92619522)

[INPUT PROCESSING AND OUTPUT: 3](#_Toc92619523)

[DECISIONS: 3](#_Toc92619524)

[QUESTION B 4](#_Toc92619525)

[INPUT PROCESSING AND OUTPUT: 4](#_Toc92619526)

[DECISIONS: 4](#_Toc92619527)

[CLASS DIAGRAM 5](#_Toc92619528)

[ANNEX 1. SCREENSHOTS OF APPLICATION. 6](#_Toc92619529)

[CHOICE 1: GENERATE USERNAME 6](#_Toc92619530)

[CHOICE 2: COUNT CONSONANTS IN PARAGRAPH 7](#_Toc92619531)

[ANNEX 2. SOURCE CODE: 8](#_Toc92619532)

[ItemGenerator: 8](#_Toc92619533)

[ItemGeneratorMenu 13](#_Toc92619534)

[ItemGeneratorApp 14](#_Toc92619535)

# QUESTION A

## INPUT PROCESSING AND OUTPUT:

1. INPUT:
   1. String name;
2. PROCESSING:
   1. firstCharacter(): Calculates the first character of the username.
   2. calculateItem(): calculates the body of the username.
   3. substituteSpace(): returns the length of the name to substitute space characters.
   4. upperToLower(): calculates the last character of the username.
   5. calculateUsername(): invokes all previous methods to generate the full username.
3. OUTPUT:
   1. The provided name || username.

## DECISIONS:

At first I had fewer methods with the same result. But I decided to break down the processes according to the specifications of the username so that the application could be easy to modify in the future.

# QUESTION B

## INPUT PROCESSING AND OUTPUT:

1. INPUT:
   1. String paragraph.
2. PROCESSING:
   1. consonantsCount():
      1. iterates through the provided paragraph and an array of consonant letters regardless of the case (upper or lower).
      2. When there’s a match the count increases by 1.
      3. Returns the final count.
3. OUTPUT:
   1. The index of the given paragraph and the total number of consonants present.
   2. An integer array with the consonants present for each paragraph.

## DECISIONS:

For this questions I hesitated between two options: reuse a previous method or create a new one. I could have reused the method upperToLower(). This method was first created for question A to count the number of upper case vowels in the name. The process is the same as the one required for question b: iterate through the given string and an array of characters, when there´s a match the count increases. In order to do that, what I would have to do is to modify the upperToLower() to take in two parameters (String, char[]). For question A it would take in the string name and compare it to the array upper; for question b, the string paragraph and the array consonants.

Even though the previous process could have been more efficient, I decided to create a specific method for this questions as that was the requirement.

I also decided to reuse the private String name instead of creating another data member for the paragraph.

# CLASS DIAGRAM

Diagram

Description automatically generated

# ANNEX 1. SCREENSHOTS OF APPLICATION.

## CHOICE 1: GENERATE USERNAME

Text

Description automatically generated

## CHOICE 2: COUNT CONSONANTS IN PARAGRAPH

Table

Description automatically generated

# ANNEX 2. SOURCE CODE:

## ItemGenerator:

/\*\*

\* NCIRL.

\* Module = Software development

\* CA2.

\* Class for creating username from employee's name,

\* and counting consonants in a paragraph.

\* author: Daniel Gonzalez.

\* student number: 21111383.

\*/

// This file contains the following:

// Instanciable class ItenGenerator:

// QUESTION A: Generates a username from a given name

// QUESTION B: Counts the number of consonants in a given paragraph.

// Constuctor

// setter for name

// getter for name and username

// QUESTION A:

// The processing has been divided in four methods:

// firstCharacter(): calculates the first character of username

// calculateItem(): calculates the body of the username

// substituteSpace(): returns the length to substitute space characters.

// upperToLower(): calculates the last character of the username

// calculateUsername(): invokes all previous methods to generate the full

// username

// QUESTION B:

// Processing:

// consonantsCount() returns int number of consonants in the given string.

public class ItemGenerator {

// declare instanciable classes / data members:

private String name;

private String username;

private char[] upper = { 'A', 'E', 'I', 'O', 'U' };

private char[] lower = { 'a', 'e', 'i', 'o', 'u' };

private char[] consonants = { 'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v',

'w', 'x', 'y', 'z' };

private int numberOfConsonants;

// Constructor.

// Initialised with empty string to prevent runtime error

public ItemGenerator() {

name = "";

}

// Setter:

public void setName(String aName) {

name = aName;

}

// getter:

public String getUsername() {

return username;

}

public int getNumberOfConsonants() {

return numberOfConsonants;

}

public String getName() {

return name;

}

// PART 1:

// Processing:

// Method to calculate the first character of

// the username (second to last character of the name).

public char firstCharacter() {

int length = name.length();

char firstChar = (char) name.charAt(length - 2);

return firstChar;

}

// method to calculate the body of the username (except the last character)

// If the character is upper will return the lower case version,

// else will return the same character

private char calculateItem(char letter) {

// Indexing array upper:

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

char currentLetter = upper[i];

// Substitute the position in upper

// for the character in the same position in string lower

if (currentLetter == letter) {

letter = lower[i];

}

}

return letter;

}

// Method to calculate the length of the given name to

// substitute a space character.

public int substituteSpace() {

int length = name.length();

return length;

}

// Method to calculate the last character of the username

public int upperToLower() {

int upperToLower = 0;

// Indexing and traversing the given name:

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

char c = name.charAt(i);

// Indexing and traversing the array upper:

for (int y = 0; y < upper.length; y++) {

char upperTo = upper[y];

// substitution of upper to lower:

if (c == upperTo) {

upperToLower++;

}

}

}

return upperToLower;

}

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

// Method to calculate the full username:

public String calculateUsername() {

// declare buffer to traverse the given name:

StringBuffer buffer = new StringBuffer();

// Local variables:

char usernameChar; // to append to buffer.

char afterSpace = '-'; // to add after length when char is space.

// First character of username::

buffer.append(firstCharacter());

// Body of the username:

// traverse the given name:

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

char c = name.charAt(i);

// if char is not empty space:

if (c != ' ') {

// invoke calculateItem:

usernameChar = calculateItem(c);

buffer.append(usernameChar);

// If char is empty space:

} else {

buffer.append(substituteSpace());

buffer.append(afterSpace);

}

}

// final character of username:

buffer.append(upperToLower());

// Converting back to string:

username = buffer.toString();

return username;

}

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

// PART 2

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

public int consonantsCount() {

int numberOfConsonants = 0;

// Indexing and traversing the given paragraph:

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

char c = Character.toLowerCase(name.charAt(i));

// Indexing and traversing the array consonants:

for (int y = 0; y < consonants.length; y++) {

char currentConsonant = consonants[y];

// updating count

if (c == currentConsonant) {

numberOfConsonants++;

}

}

}

return numberOfConsonants;

}

}

## ItemGeneratorMenu

/\*\*

\* NCIRL.

\* Module = Software development

\* CA2.

\* Menu for displaying in the ItemGeneratorApp

\* author: Daniel Gonzalez.

\* student number: 21111383.

\*/

public class ItemGeneratorMenu {

public void displayMenu() {

System.out.println("");

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

System.out.println("ItemGeneratorApp Menu");

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

System.out.println("1 - Generate username");

System.out.println("2 - Count consonants in paragraph");

System.out.println("Any other - exit");

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

}

}

## ItemGeneratorApp

/\*\*

\* NCIRL.

\* Module = Software development

\* CA2.

\* Application to use the instanciable class ItemGEnerator.

\* author: Daniel Gonzalez.

\* student number: 21111383.

\*/

/\*\*

This file contains the following:

Main method to generate an array of employees and their

corresponding username by instantiating the class ItemGenerator.

Local variables for String arrays populating

output:

Each name and the corresponding username.

\*/

import java.util.Arrays;

import java.util.Scanner;

public class ItemGeneratorApp {

public static void main(String[] args) {

// Declare local variables:

Scanner input = new Scanner(System.in);

String[] employees;

String[] usernames;

int numberOfEmployees;

String[] paragraphs;

int[] consonantsPresent;

int numberOfParagraphs;

ItemGeneratorMenu m = new ItemGeneratorMenu();

ItemGenerator generator = new ItemGenerator();

// Processing:

m.displayMenu();

while (true) {

int choice = 0;

System.out.println("Choice: ");

try {

choice = input.nextInt();

} catch (Exception e) {

System.out.println("Choice must be an integer between 1 & 2");

System.out.println("Good bye.");

break;

}

// Valid choice input:

if (choice >= 1 && choice <= 2) {

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

// PART 1

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

// Choice 1 = generate username:

if (choice == 1) {

// Prompt user to give a numbre of employeees to generate usernames for:

System.out.println("For how many employees would you like to generate usernames: ");

numberOfEmployees = input.nextInt();

input.nextLine();

// Arrays initialized;

employees = new String[numberOfEmployees];

usernames = new String[numberOfEmployees];

// populate employee array with user input:

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

int currentEmployee = i + 1;

// prompt user for a name:

System.out.println("Enter the name for employee number: " + currentEmployee);

String name = input.nextLine();

employees[i] = name;

}

// populate the usernames array with values in employees array:

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

generator.setName(employees[i]);

usernames[i] = generator.calculateUsername();

}

System.out.println("");

// Output:

// Prints each employee name and the corresponding username:

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

System.out.println("Name: " + employees[i] + " || " + "Username: " + usernames[i]);

}

m.displayMenu();

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

// PART 2

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

} else if (choice == 2) {

// Prompt user for a number of paragraphs:

System.out.println("How many paragraphs would you like to enter: ");

numberOfParagraphs = input.nextInt();

input.nextLine();

// Array initialized;

paragraphs = new String[numberOfParagraphs];

consonantsPresent = new int[numberOfParagraphs];

// populate array array with user input:

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

int currentParagraph = i + 1;

// prompt user for a paragraph:

System.out.println("Enter the paragraph " + currentParagraph);

String parrag = input.nextLine();

paragraphs[i] = parrag;

}

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

generator.setName(paragraphs[i]);

consonantsPresent[i] = generator.consonantsCount();

}

System.out.println("");

// Output:

// Prints the index of each paragraph and the corresponding number of

// consonants:

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

int currentParagraph = i + 1;

System.out.println("Paragraph " + currentParagraph + ", contains " + consonantsPresent[i]

+ " consonants.");

}

// Prints the array of consonants:

System.out.print(Arrays.toString(consonantsPresent));

m.displayMenu();

}

}

}

}

}