National College of Ireland

Higher Diploma in Science in Computing, Year 1, HDSDEV_SEP
Higher Diploma in Science in Computing, Year 1, HDCSDEV_INT
Higher Diploma in Science in Computing, Year 1, HDAIML_SEPOL
Higher Diploma in Science in Computing, Year 1, HDBC_SEPOL
Higher Diploma in Science in Computing, Year 1, HDSDEV_SEPOL_YR1
Higher Diploma in Science in Computing, Year 1, HDWD_SEPOL
Higher Diploma in Science in Computing, Year 1, HDCYB_SEPOL
Higher Diploma in Science in Computing, Year 1, HDSDEV_JAN21_O
Higher Diploma in Science in Computing, Year 1, HDWD_JAN21OL_O
Higher Diploma in Science in Computing, Year 1, HDCYB_JAN21OL_O
Higher Diploma in Science in Computing, Year 1, HDAIML_JAN21OL_O
Certificate in Computing, Year 1, CIC_OCTOL
Fundamentals of Software Development, Year 1, MCFSD_OCT

Semester One Terminal Assignment-based Assessment (TABA) – 2021/22

Release Date on Moodle: 7th of January 2022 @09:00am
Online Moodle Submission Deadline: 9th of January 2022 @18:00

Software Development

IMPORTANT: It is your responsibility to avoid plagiarism. Please read the comprehensive guidelines on academic honesty and academic integrity, and how to avoid plagiarism made available by the NCI Library (https://libguides.ncirl.ie/referencingandavoidingplagiarism).

NOTE: YOU ARE NOT ALLOWED TO PUBLISH THIS ASSIGNMENT BRIEF OR A PART THEREOF ON ANY WEBSITES. YOU ARE NOT ALLOWED TO PUBLISH/SHARE YOUR SOLUTION WITH OTHERS. All work submitted **should be your own** and should be carried out **using only the programming concepts and data types covered in this module.** Conferring with others **is not permitted.**

Note that all submissions will be electronically screened (via Turnitin) for evidence of academic misconduct (i.e. plagiarism and collusion).

Note:

- This is an open book assessment.
- The requirements that you have to implement are assigned based on **different digits** of **your student ID number**. Please read carefully the ***Assignment Description*** section for guidelines about the functionalities you have to implement. **This is a submission requirement**. If the incorrect functionalities are implemented, no marks will be provided for those particular functionalities.

Assignment Description

For this assessment you are required to implement an application that should work according to the two main questions and all their subquestions described in this section. The requirements that you have to implement in your application are assigned based on different digits of your student ID number. Please read carefully the instructions and details about the requirements you have to implement. Please note that if the incorrect requirements are implemented, no marks will be provided for those particular functionalities.

For each of the two main questions, you have to plan, develop, and manually test that the application provides a solution according to the question specification in this section and the requirements assigned to you.

Question 1. A company has hired you to develop an application to generate certain items for their employees' accounts. The application prompts the user to provide one single line of text with an employee's full name in the following format "forename surname" at a time. (**Please note that you are not required to validate the input, we assume that the input is well formed.**) Next, the application uses the given full name to create the corresponding item. The item is created using several rules. The item assigned to you and the rules that you should implement to create the item have been assigned based on the **penultimate digit of your student ID number**. Please check *Table 1 Question 1. a. Item and Rules to Create the Item* for details about the functionality assigned to you.

Note that the application should work irrespective of how the user provides the full name i.e. using upper case letters, lower case letters or a combination of both upper case and lower case letters. You should use the full name as provided i.e. you should not modify the given full name to a different letter case.

a. Develop an **instantiable class** for this application which contains:

- A class definition
- Suitable data members (instance variables)
- A constructor
- A setter method to set the given full name
- A compute method to generate/create the item assigned to you according to the rules assigned to you based on *Table 1*. Note: This method should demonstrate the use of programming concepts covered in our module. Marks will not be awarded for solutions that use concepts and data types which have not been addressed/covered in our module.
- A getter method to return the item

Name the instantiable class **ItemGenerator**.

The source code should be commented throughout highlighting and explaining where the key functionality is being addressed.

(30 marks)

☐ **Assigned Item and Rules to Create the Item**: The item and the rules that you should implement to create the item are in assigned in *Table 1 Question 1. a. Item and Rules to Create the Item* based on the **penultimate digit of your student ID**. **IMPORTANT: This is a submission requirement.** If the incorrect rules are implemented, no marks will be provided for that functionality.

Table 1 Question 1. a. Item and Rules to Create the Item

Penultimate (i.e. second to last) digit of your student ID	**Assigned Item**	**Rules to Create the Item**	**Examples (given full name and corresponding item that should be created)**
0 OR **1** OR **2** OR **3** OR **4**	password	The password is created using the following rules: • The letters 'a', 'e', and 't' from the given full name will not be used in the password • Each vowel (except 'a' and 'e' which are eliminated) is going to be added twice • Each space is replaced by the letter 'S' followed by a '&' and a '?' • All the other characters will remain the same as in the given full name • The password ends with the total number of letters eliminated (i.e. the total number of letters 'a', 'e', and 't' from the given full name that were not used in the password)	For example, if the instantiable class receives: • The full name "JANE dOe" then the compute method should create the password "JNS&?dOO3" (note that 3 is the total number of letters eliminated i.e. as the letters 'a' and 'e' from the given full name were not used in the password) • The full name "conOr MUrphy" then the compute method should create the password "coonOOrS&?MUUrphy0" (note that 0 is the total number of letters eliminated i.e. as the given full name did not contain any of the letters 'a', 'e', or 't')
5 OR **6** OR **7** OR **8** OR **9**	username	The username is created using the following rules: • The username starts with the second to last character of the given full name, written using the same letter case as in the given full name • Each upper case vowel is replaced by its lower case version. Vowels entered in lower case, remain lower case in the username. • Each space is replaced by the total number of characters in the given full name (note that the total number of characters includes the space), followed by a hyphen character '-' • All the other characters will remain the same as in the given full name • The username ends with the total number of upper case vowels that had been replaced by their lower case versions	For example, if the instantiable class receives: • The full name "JANE dOe" then the compute method should create the username "OJaNe8-doe3" (note that 8 is the total number of characters in the given full name which includes the spaces and 3 is the total number of upper case vowels that had been replaced by their lower case versions) • The full name "conOr MUrphy" then the compute method should create the username "hconor12-Murphy2" (note that 12 is the total number of characters in the given full name which includes the spaces and 2 is the total number of upper case vowels that had been replaced by their lower case versions)

Example: A student with the student ID = 219876**5**4 is assigned the item *username*, and therefore would implement the assigned rules to create *usernames* (because the penultimate digit of that student ID is **5**).

b. Develop an application that uses the instantiable class *ItemGenerator* (the instantiable class previously developed) to create items. The application should allow a user to enter multiple employees' full names. Please check *Table 2 Approaches to entering multiple full names to create corresponding items* for details about the approach you have to implement in order for the application to create multiple items. The approach you have to implement has been assigned to you based on the **last digit of your student ID number**. The application will display the created items on the screen. In the application class, please add a short comment for each method of the *ItemGenerator* class that you use/call in your application to explain why that method is needed.
Name the application class **ItemGeneratorApp.**

(20 marks)

☐ **Approach for entering multiple full names to create corresponding assigned item**: Use *Table 2 Approaches to entering multiple full names to create corresponding items* and based on the **last digit of your student ID** find the approach you have to implement in your application.
IMPORTANT: This is a submission requirement. If the incorrect approach is implemented, no marks will be provided for that functionality.

Table 2 Approaches to entering multiple full names to create corresponding items

Last digit of your student ID	Approach ID	Approaches to entering multiple full names to create the corresponding items[1] (MFNA)
0 OR **2** OR **4** OR **6** OR **8**	MFNA1	Ask the user to provide a full name and after the corresponding item is created and displayed on the screen, ask the user if they would like to create another item. As long as the user enters "yes" the application should work as described in the previous sentence. When the user enters anything else than "yes", no other items are created.
1 OR **3** OR **5** OR **7** OR **9**	MFNA2	Ask the user at the beginning of the application how many items they would like to create, and ensure that the application enables the user to provide that amount of full names and for each full name creates the corresponding item.

Example: A student with the student ID = 2198765**4** would implement the approach corresponding to *MFNA1* (because the last digit of that student ID is **4**).

Question 2. Develop further the application as follows:

a. First, implement in the instantiable class *ItemGenerator* (the instantiable class previously developed at Question 1. a.) **another method** which takes in as a parameter a one-dimensional array of String paragraphs. A paragraph is a short piece of text, consisting of at least one sentence. The text can contain only letters, spaces (i.e. ' '), commas (i.e. ','), dots (i.e. '.'), exclamation marks (i.e. '!') and question marks (i.e. '?'). Each sentence ends with

[1] Note that the item that has to be created is the one assigned to you in *Table 1 Question 1. a. Item and Rules to Create the Item*

either a dot, an exclamation mark or a question mark. (**Please note that you are not required to validate the paragraphs, we assume that each paragraph is well formed.**)
The method should compute and return an array of numbers. The functionality that you should implement for this method has been assigned based on the **penultimate digit of your student ID number**. Please check *Table 3 Question 2. a. Functionality* to find the functionality assigned to you.

Note 1: the method should work irrespective of the letter case used in the paragraphs i.e. upper case letters, lower case letters or a combination of both upper case and lower case letters. The method should not modify the letter case of the given paragraphs.

Note 2: this method should demonstrate the use of programming concepts covered in our module. Marks will not be awarded for solutions that use concepts and data types which have not been addressed/covered in our module.

The source code should be commented throughout highlighting and explaining where the key functionality is being addressed.

(20 marks)

☐ **Method**: You are required to implement the functionality assigned to you in *Table 3 Question 2. a. Functionality* based on the **penultimate digit of your student ID**. **IMPORTANT: This is a submission requirement.** If the incorrect functionality is implemented, no marks will be provided for that functionality.

Table 3 Question 2. a. Functionality

Penultimate (i.e. second to last) digit of your student ID	**Functionality ID**	**Functionality**	**Example**
0 OR **5** OR **6**	F1	The method should traverse the array of paragraphs and calculate the total number of declarative sentences, interrogative sentences, and exclamatory sentences per paragraph. The total number of sentences computed per paragraph should be stored in an array of numbers. The method should return the computed array of numbers. Note: A declarative sentence ends with a dot (i.e. .). An exclamatory sentence ends with an exclamation mark (i.e. !). An interrogative sentence ends with a question mark (i.e. ?).	For example, if the method is passed in an array of paragraphs containing the following values/elements: • "YOU are your BEST thing!" • "Omar learned Java. Did John learn C? Emma programs in Java and Scala." • "Who in the world am I? Ah, that is the GreAT puzzle!" Then the method should calculate and return the array of numbers that contains the total number of declarative sentences, interrogative sentences, and exclamatory sentences per paragraph, which in this example is the array with the following values/elements: 1, 3, 2

			as the first paragraph has 1 sentence; the second paragraph has 3 sentences; and the third paragraph has 2 sentences.
1 OR **4** OR **7**	F2	The method should traverse the array of paragraphs and calculate the total number of vowels per paragraph. The total number of vowels per paragraph should be stored in an array of numbers. The method should return the computed array of numbers.	For example, if the method is passed in an array of paragraphs containing the following values/elements: • "YOU are your BEST thing!" • "Omar learned Java. Did John learn C? Emma programs in Java and Scala." • "Who in the world am I? Ah, that is the GreAT puzzle!" Then the method should calculate and return the array of numbers that contains the total number of vowels per paragraph, which in this example is the array with the following values/elements: 8, 21, 14 as the first paragraph has 8 vowels; the second paragraph has 21 vowels; and the third paragraph has 14 vowels.
2 OR **3** OR **8** OR **9**	F3	The method should traverse the array of paragraphs and calculate the total number of consonants per paragraph. The total number of consonants per paragraph should be stored in an array of numbers. The method should return the computed array of numbers.	For example, if the method is passed in an array of paragraphs containing the following values/elements: • "YOU are your BEST thing!" • "Omar learned Java. Did John learn C? Emma programs in Java and Scala." • "Who in the world am I? Ah, that is the GreAT puzzle!" Then the method should calculate and return the array of numbers that contains the total number of consonants per paragraph, which in this example is the array with the following values/elements: 11, 33, 24 as the first paragraph has 11 consonants; the second paragraph has 33 consonants; and the third paragraph has 24 consonants.

Example: A student with the student ID = 219876**5**4 would implement the functionality identified by *F1* (because the penultimate digit of that student ID is **5**).

b. Next, develop further the application class *ItemGeneratorApp* (the class previously developed at Question 1. b) to use the method defined at *Question 2. a*. First, prompt the user to provide the number of paragraphs they would like to enter, and then prompt the user to provide those paragraphs by reading one paragraph at a time from the keyboard, and store those paragraphs in an array of paragraphs. Next, call/invoke/use the method defined at *Question 2. a*. to compute the numbers according to the functionality assigned to you. Finally, the application should display on the screen the numbers computed by the method implemented at *Question 2. a*.

(10 marks)

Application Development

The applications should be developed, compiled, and run using a text editor such as TextPad (or similar Text Editor alternative for macOS or Linux users) which enables you to compile and run Java applications. Note that Java Development Kit (JDK) has to be installed on your machine in order to compile and run your application.
Alternatively, you should be able to access these tools through the Student Virtual Desktop (details available here).

Deliverables

The Terminal-Assignment Bases Assessment deliverables **must be submitted via Moodle**, they cannot be submitted by email. You are required to submit the following deliverables:

1) Complete Java source code (.java files) for the questions assigned to you
   Submit the complete Java source code as a .zip file via the submission page ***TABA Source-Code*** available on the Software Development Moodle page.

2) A report (in .pdf or .doc format) which includes:
   - A description of the input, main processing, and output (IPO) for each of the two main questions
   - The class diagram for your application
     - For creating the class diagram, you can either use an online tool, such as http://app.diagrams.net, or draw it by hand and take a photo
   - Any decisions you take in designing and implementing your application should be specified in the report
   - ***Note that the entire java source code of your application must also be included as an appendix in your report! Note that the java source code must be included as text (i.e. copy and paste your code in the report, do not take a screenshot of your code!).*** This is a submission requirement.

   Submit the report document via the ***TABA Report*** Turnitin submission page available on the Software Development Moodle page.

Marking Scheme

The marks for this assignment will be allocated as follows:

- **Application Implementation (80 marks)**
  - Question 1. a. (30 marks)
  - Question 1. b. (20 marks)
  - Question 2. a. (20 marks)
  - Question 2. b. (10 marks)
  - Note that the implementation evaluation includes the following
    - Fully compiling and executing application with no syntax or logical errors which addresses the full requirements of the application
    - All requirements have been implemented, and the application works according to the specification assigned to you
    - The application produces accurate output
- **Good coding practices and code understanding (13 marks)**
  - The application should be fully commented throughout highlighting and explaining where the key functionalities of the application are being addressed (10 marks)
  - Well formatted and properly indented code. Appropriately named variables, methods, classes using the Java naming conventions (3 marks)
- **Report (7 marks)**
  - Evidence of designing and planning of the application prior to coding (for example, the report should include the IPO and the class diagram) (7 marks)

NOTE: the examiners reserve the right to conduct mini presentations with a sample of the students, where students will provide answers to questions related to their assignment.