

National College of Ireland

Higher Diploma in Science in Computing, Year 1 (HDSDEV_JAN24, HDCSDEV_INTJAN24,
HDSDEV_JANBL_YR1, HDWD_JAN24);
Certificate in Computing, Year 1, CIC_JAN24

Semester Two Terminal Assignment-based Assessment (TABA) – 2023/24

Release Date on Moodle: Friday 03rd May 2024 @09:00am
Online Moodle Submission Deadline: Sunday 05th May 2024 @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 carefully read 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 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 sub-questions 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 carefully read 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 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.

Note: Where user input is required, this should be provided via the keyboard. For reading user input from the keyboard, you can use either the **Scanner** class in conjunction with **System.in** or **JOptionPane**.

Question 1. A company has hired you to develop an application that enables a user to check if the provided piece of text conforms to the rules corresponding to the assigned item. The application prompts the user to provide one item (i.e. a piece of text). Next, the application uses the piece of text to determine if the provided text is valid (i.e. the text conforms to the rules assigned to you) or invalid (i.e. the text does NOT conform to those rules). The item assigned to you and the rules that you should implement to determine whether the item is valid or not have been assigned based on the penultimate digit of your student ID number. Please check [Table 1 Question 1. a. Item and Rules to Determine the Validity of an Item](#) for details about the functionality assigned to you.

Note that the application should work irrespective of how the user provides the piece of text, i.e. using upper case letters, lower case letters or a combination of both upper case and lower-case letters. Once the piece of text provided by the user is read, the input from the user should be converted to upper case letters, and the converted input will be further checked if it is valid or not. Question 1. a. and Question 1. b. are to be developed in conjunction, Question 1. a. is the instantiable class (ItemChecker), and Question 1. b. is the App class (ItemCheckerApp).

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 item.
- A compute method to determine if the provided item is valid according to item and the rules assigned to you based on *Table 1*, otherwise the item is deemed invalid. 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/classes which have not been addressed/covered in our module. **This is a submission requirement.**
- A getter method to return the validity of the item.
- Name the instantiable class **ItemChecker**.

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

(30 marks)

☐ **Assigned Item and Rules to Determine the Validity of an Item:** The item and the rules that you should implement to determine whether the item is valid or not are in assigned in *Table 1 Question*

1. a. *Item and Rules to Determine the Validity of an 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.

Penultimate (i.e. second to last) <u>digit</u> of your student ID	Assigned Item	Rules ¹ to Determine the Validity of the Item	Examples (given item and the validity that the compute method should determine)
0 OR 3 OR 6 OR 9	Irish vehicle registration plate	<p>Item format: YYYY-LL-SSSSSS</p> <p>A new Irish vehicle registration plate contains three groups of characters, each separated by a dash i.e. '-' as follows:</p> <ul style="list-style-type: none"> Starts with a three-digit year (i.e. YYY-LL-SSSSSS). <u>Note</u> for simplicity, the actual year does not have to be verified, namely any three-digit sequence will be considered valid. Continues with a dash i.e. '-' Followed by a two-letter county/city identifier (i.e. YYY-LL-SSSSSS) Continues with a dash i.e. '-' Ends with a one- to six-digit sequence number (i.e. YYY-LL-SSSSSS) <p>The possible county/city identifiers² (i.e. LL) are as follows: CK, CE, CN, CW, DN, DL, GY, KE, KK, KY, LK, LD, LH, LM, LS, MH, MN, MO, OY, RN, SO, TY, WD, WH, WX, WW</p> <p>A digit is any of the ten numbers from 0 to 9.</p>	<p>For example:</p> <ul style="list-style-type: none"> if the item is "222-DN-1055" then the compute method should determine that the item is valid, and store that information in the corresponding instance variable. if the item is "221-KE-1055" then the compute method should determine that the item is valid, and store that information in the corresponding instance variable. If the item is "2-DN-1055" then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable. if the item is "222-GY-1234567" then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable
1 OR 4 OR 7	Eircode	<p>Item format: LDD CCCC</p> <p>An Eircode³ contains two groups of characters separated by a space i.e. ' ' as follows:</p> <ul style="list-style-type: none"> Starts with a letter (i.e. LDD CCCC) Continues with two digits (i.e. LDD CCCC) Followed by a space i.e. ' ' Ends with a group of four-characters (i.e. LDD CCCC) that can be either letters or digits in any order. <p>A digit is any of the ten numbers from 0 to 9.</p>	<p>For example:</p> <ul style="list-style-type: none"> If the item is "D01 K6W2" then the compute method should determine that the item is valid, and store that information in the corresponding instance variable. if the item is "T12 DW6P" then the compute method should determine that the item is valid, and store that information in the corresponding instance variable. if the item is "123 DW6P" then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable. if the item is "T32 DW6P14" then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable

¹ **Disclaimer:** note the rules presented in this table may be simplified and/or modified from their real-world counterparts. You are required to implement the rules as described in this assessment.

² https://en.wikipedia.org/wiki/Vehicle_registration_plates_of_the_Republic_of_Ireland#Current_index_mark_codes

³ <https://www.gov.ie/en/policy-information/01f07-eircode/>

Penultimate (i.e. second to last) <u>digit</u> of your student ID	Assigned Item	Rules ¹ to Determine the Validity of the Item	Examples (given item and the validity that the compute method should determine)
2 OR 5 OR 8	Passport number	<p>Item format: C X- DDDDDDD A passport number:</p> <ul style="list-style-type: none"> Starts with <u>EITHER</u> the letter P <u>OR</u> letter L (i.e. C X- DDDDDDD) Continues with a space i.e. ‘ ‘ Followed by a letter (i.e. C X- DDDDDDD) Continues with a dash i.e. ‘-‘ Ends with a seven-digit number (i.e. C X-DDDDDDD) <p>A digit is any of the ten numbers from 0 to 9.</p>	<p>For example:</p> <ul style="list-style-type: none"> if the item is “P A-1234123” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable. If the item is “L Q-7612543” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable. if the item is “PA-98765” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable. if the item is “P X-12345678” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable

Example: A student with the student ID = 21987654 is assigned the item *Passport number*, and therefore would implement the assigned rules to determine the validity of a given piece of text according to the *Passport number rules* (because the penultimate digit of that student ID is 5).

- b. **Develop an application** that uses the instantiable class *ItemChecker* (the instantiable class previously developed in Question 1. a.) to check the validity of the given items. The application should allow a user to enter multiple items. Please *check Table 2 Approaches to entering multiple items to check their validity* for details about the approach you have to implement in order for the application to check 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 on the screen if the given items are valid or not. In the application class, please add a short comment for each method of the *ItemChecker* class that you use/call in your application to explain why that method is needed. Name the application class **ItemCheckerApp**.
(20 marks)

- ☐ **Approach to entering multiple items to check their validity:** Use *Table 2 Approaches to entering multiple items to check their validity*, 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 items to check their validity.

Last digit of your student ID	Approach ID	Approaches to entering multiple items to check their validity ⁴ (MIA)
0 OR 1 OR 2 OR 3 OR 4	MIA1	Ask the user to provide an item, and after the validity of the item is checked and the result is displayed on the screen, ask the user if they would like to check another item. As long as the user enters "yes" the application should work as described in the previous sentence. When the user enters anything other than "yes", no other items are checked.
5 OR 6 OR 7 OR 8 OR 9	MIA2	Ask the user at the beginning of the application how many items they would like to check and ensure that the application enables the user to provide that number of items and for each item check its validity.

Example: A student with the student ID = 21987654 would implement the approach corresponding to MIA1 (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 *ItemChecker* (the instantiable class previously developed at Question 1. a.) **another method** which takes in as a parameter a one-dimensional array of String handles/usernames. The method should compute whether the given handles/usernames are valid (i.e. a given handle/username conforms to the rules assigned to you) or invalid (i.e. a given handles/usernames does NOT conform to those rules). The method should return an array of boolean values, where a true value means that a particular handle/username is valid, and a false value means that a particular handle/username is invalid. The handle/username assigned to you and the rules that you should implement to determine if the handles/usernames are valid or not have been assigned based on the **penultimate digit of your student ID number**. Please check *Table 1 Question 2. a. Handle/Username and Rules to Determine the Validity of a Handle/Username* to identify the functionality assigned to you.

Note 1: the method should work irrespective of the letter case used in the handles/usernames i.e. upper-case letters, lower case letters or a combination of both upper case and lower-case letters. The method **can modify** the letter case of the given handles/usernames. Each handle/username should be converted to lower case letters, and the converted handle/username will be further checked, to see if it is valid or not.

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/classes which have not been addressed/covered in our module. **This is a submission requirement.**

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

(22 marks)

- ☐ **Method:** You are required to implement the functionality assigned to you in *Table 1 Question 2. a. Handle/Username and Rules to Determine the Validity of a Handle/Username*. The functionality is assigned 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.

⁴ Note that the item that has to be checked is the one assigned to you in *Table 1 Question 1. a. Item and Rules to Determine the Validity of an Item*

Table 1 Question 2. a. Handle/Username and Rules to Determine the Validity of a Handle/Username

Penultimate (i.e. second to last) digit of your student ID	Assigned handle/username	Rules ⁵ to determine the validity of a handle/username	Example
0 OR 1 OR 2 OR 3 OR 4	Instagram handles/ usernames	<p>Functionality: Check the validity of an Instagram handle/username.</p> <p>A valid Instagram handle/username satisfies all the following rules:</p> <ul style="list-style-type: none"> It cannot start with a period '.' It cannot exceed 30 characters. It cannot be shorter than 6 characters. It can only contain letters (i.e. a – z inclusive, A – Z inclusive), digits (i.e. 0 – 9 inclusive), and periods (i.e. '.') 	<p>For example,</p> <ul style="list-style-type: none"> if the method is passed in an array with the following handles/usernames: <ul style="list-style-type: none"> theSW.Developer .theSWDeveloper @theRealSWDeveloper the2Hikers l.mD1 then the method should determine for each username/handle if it's valid or not, and return an array of boolean values, which in this example is the array with the following values/elements: true, false, false, true, false.
5 OR 6 OR 7 OR 8 OR 9	Twitter handles/ usernames	<p>Functionality: Check the validity of a Twitter handle/username</p> <p>A valid Twitter handle/username satisfies all the following rules:</p> <ul style="list-style-type: none"> It must start with the symbol '@' It cannot be shorter than 5 characters. It cannot be longer than 16 characters. It can only contain letters (i.e. a – z inclusive, A – Z inclusive), digits (i.e. 0 – 9 inclusive), and underscores (i.e. '_') 	<p>For example,</p> <ul style="list-style-type: none"> if the method is passed in an array with the following handles/usernames: <ul style="list-style-type: none"> @The1SWDeveloper theSW_Developer @the.RealDev @lAm @theHiker Then the method should determine for each username/handle if it's valid, and return an array of boolean values, which in this example is the array with the following values/elements: true, false, false, false, true.

Example: A student with the student ID = 22987654 would validate Twitter handles/usernames (because the penultimate digit of that student ID is 5).

- a. Next, develop further the application class *ItemCheckerApp* (the class previously developed at Question 1. b) to use the method defined in *Question 2. a.* to demonstrate its functionality. First, prompt the user to provide

⁵ **DISCLAIMER:** note that some of the rules specified for this functionality are provided for this assessment purposes only and may not represent the actual rules used in their real-world counterparts. You are required to implement the rules as described in this assessment.

the number of handles/usernames they would like to validate, and then prompt the user to provide those handles/usernames by reading one handle/username at a time from the keyboard, and store those handle/usernames in an array of handles/usernames. Next, call/invoke/use the method defined at *Question 2. a.* to validate the usernames according to the functionality assigned to you. Finally, the application should display on the screen the values computed by the method implemented at *Question 2.a.*

(8 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](#)).

Note: Where user input is required, this should be provided via the keyboard. For reading user input from the keyboard, you can use either the **Scanner** class in conjunction with **System.in** or **JOptionPane**.

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:
 - The specific digits, mapped to the options/requirements that you have used to develop your application. E.g., 229876**54**, 5 = Passport Number; 4= MIA1 for Q1 & Twitter handles/usernames for Q2.
 - 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.
 - Be careful to adhere to appropriate syntax and structure and identify suitable data types for each data member/ instance variable.
 - **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. (22 marks)
 - Question 2. b. (8 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 (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.