

National College of Ireland

**Certificate in Computing, Year 1, CIC\_MARBL23**

**Terminal Assignment-based Assessment (TABA) – 2022/23**

**Release Date on Moodle: 28th of June 2023 @09:00am**

**Online Moodle Submission Deadline: 1st of July 2023 @23:55pm**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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**](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.

1. A company has hired you to develop an application that enables a user to check if the user name of your WiFi network, which is identified as SSID (Service Set Identifier) hereafter is of a standard format that 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 (SsidChecker), and Question 1. b. is the App class (SsidCheckerApp).

1. 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 (i.e. piece of text)
   * A compute method to determine if the provided item is valid according to the item and 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/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 **SsidChecker**.

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.

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

| **Penultimate (i.e. second to last) digit of your student ID** | **Assigned Item** | **Rules**[[1]](#footnote-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** | Virgin Media network | Item format: PP-SSSSS-LLL  The new format for Virgin Media WiFi SSID contains three groups of characters, each separated by a dash i.e. ‘-‘ as follows:   * Starts with two letters   (i.e. **PP**-SSSSS-LLL) unique to the service provider and is VM   * Continues with a dash i.e. ‘**-**‘ * Followed by a five-digit sequence number (i.e. PP-**SSSSS**-LLL)   Note The five-digit sequence should be in ascending order be considered valid.   * Continues with a dash i.e. ‘**-**‘ * Ends with a three-letter sequence   (i.e. PP-SSSSS-**LLL**)  Note The three-letter sequence should be vowels to be considered valid.  A digit is any of the ten numbers from 0 to 9. Further, the same digit and letter can repeat in five-digit and three-letter sequences. | For example:   * if the item is “VM-13599-AOU” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable * if the item is “VM-12578-EII” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable * if the item is “DM-12358-OUI” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable * if the item is “VM-12328-OAI” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable * if the item is “VM-15789-BAI” 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** | Three Mobile | Item format: PP SSSSS CCCC  The new format for Three WiFi SSID contains three groups of characters, each separated by a space i.e. ‘ ‘ as follows:   * Starts with two characters   (i.e. **PP** SSSSS CCCC) unique to the service provider and is 3M   * Continues with a space i.e. ‘‘ * Followed by a five-digit sequence number (i.e. PP **SSSSS** CCCC)   Note Every second digit of the five-digit sequence should be even numbers and to be considered valid. The other digits can be of any type, (odd or even).   * Continues with a space i.e. ‘‘ * Ends with a four-character sequence   (i.e. PP SSSSS **CCCC**)  Note The four-character sequence should be either 2GHz or 5GHz to be valid  A digit is any of the ten numbers from 0 to 9. Further, the same digit can repeat in the five-digit sequence | For example:   * if the item is “3M 12543 2GHz” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable * if the item is “3M 26383 5GHz” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable * if the item is “2M 12581 5GHz” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable * if the item is “3M 11781 2GHz” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable * if the item is “3M 98142 5G” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable |
| **2**  OR  **5**  OR  **8** | Eir Network | Item format: SSSSS-LLL CC  The new format for Three WiFi SSID contains three groups of characters, each separated by a dash i.e. ‘-‘ and then a space i.e. ‘ ‘ as follows   * Starts with five-digit sequence number (i.e. **SSSSS**-LLL CC)   Note The five-digit sequence should be in descending order be considered valid   * Continues with a dash i.e. ‘**-** ‘ * Followed by three letters   (i.e. SSSSS-**LLL** CC) unique to the service provider and is Eir   * Continues with a space i.e. ‘ ‘ * Ends with a two-character sequence   (i.e. SSSSS-LLL **CC**)  Note The first character of the two-character sequence should be a digit followed by a letter randomly selected from English alphabet  A digit is any of the ten numbers from 0 to 9. Further, the digits should be unique in the five-digit sequence | For example:   * if the item is “95432-Eir 1A” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable * if the item is “76321-Eir 5F” then the compute method should determine that the item is valid, and store that information in the corresponding instance variable * if the item is “89632-Eir 8Q” then the compute method should determine that the item is invalid, and store that information in the corresponding instance variable * if the item is “54321 Eir 7S” 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 = 219876**5**4 is assigned the item *Eir Network*, and therefore would implement the assigned rules to determine the validity of a given piece of text according to the *Eir Network rules* (because the penultimate digit of that student ID is **5**).

1. **Develop an application** that uses the instantiable class *SsidChecker* (the instantiable class previously developed in Question 1. a.) to check the validity of the given items (i.e. pieces of text). 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 *SsidChecker* class that you use/call in your application to explain why that method is needed.

Name the application class **SsidCheckerApp.**

**(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[[2]](#footnote-2) (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 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 amount of items and for each item check its validity. |

Example: A student with the student ID = 2198765**4** would implement the approach corresponding to *MIA1* (because the last digit of that student ID is **4**).

1. Develop further the application as follows:
2. First, implement in the instantiable class *SsidChecker* (the instantiable class previously developed at Question 1. a.) **another compute method** which takes parameters (i) the number of passwords to be generated, and (ii) The Personal Public Services Number (PPSN) from the user which is expected to be entered in format of seven digits followed by 2 letters (i.e. DDDDDDDPP), it’s not required to check if PPSN is entered in the valid format. And (iii) an array of numbers between 21 and 29 to be used in a modulo/remainder calculation as part of the password generation for each of the passwords e.g., if 3 passwords are to be generated, then the array should contain 3 different numbers between 21 and 29 inclusive. The method should generate/create as many passwords as the given number and should store those passwords in an array of String items. The method should return the array of computed passwords. The items that you should generate and the rules to implement to generate the passwords in this method have been assigned based on the **penultimate digit of your student ID number**. Please check Table 3 Question 2. a. Rules to generate passwords (PW1 or PW2) assigned to you find the functionality assigned to you.

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.**

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. Item and Rules to generate the Item* 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. 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** | **Example** |
| **1**  OR  **3**  OR  **5**  OR  **7**  OR  **9** | PW1 | The method should create a password as follows. The password is a variable length code (i.e., a piece of text/String) formed from digits, special characters and upper-case letters.   * Using the seven sequence of digits   (i.e. **DDDDDDD**PP) of the PPSN number entered by the user, calculate the remainder based on the random numbers generated in the method.   * Followed by an ‘$’ symbol * Followed by 2 letters of the PPSN number   (i.e. DDDDDDD**PP**) entered by the user   * Followed by 4 upper case letters (i.e. **LLLL**), selected at random from the English alphabet.   Note that a digit or a letter can appear more than once in a password. Further all the letters in the created passwords should be in Uppercase.  The password should follow the following format:  Moduloremainder$PPLLLL | For example:   * if the number 4 is passed in as an argument in the compute method * then the method should create and return an array with 4   passwords, such as   * + “15$QABAGH”   + “28$AEJKRT”   + “7$HGTRWS” |
| **0**  OR  **2**  OR  **4**  OR  **6**  OR  **8** | PW2 | The method should create a password as follows. The password is a variable length code (i.e., a piece of text/String) formed from digits, special characters and lower-case letters.   * Starts with 4 lower case letters, selected at random from the English alphabet. * Followed by an ‘@’ symbol * Using the seven sequence of digits   (i.e. **DDDDDDD**PP)  of the PPSN number entered by the user, calculate the remainder based on the random numbers generated in the method.   * Followed by 2 letters of the PPSN number   (i.e. DDDDDDD**PP**) entered by the user  Note that a digit or a letter can appear more than once in a password. Further all the letters in the created passwords should be in Lowercase.  The password should follow the following format:  LLLL@ModuloremainderPP | For example:   * if the number 4 is passed in as an argument in the compute method * then the method should create and return an array with 3 user IDs, such as   + “artw@5tr”   + “bhrt@11yu”   + “ufer@21uk”   + “gpos@17pw” |

Example: A student with the student ID = 219876**5**4 would generate passwords as per PW1 (because the penultimate digit of that student ID is **5**).

1. Next, develop further the application class *SsidCheckerApp* (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 passwords they would like to be created. Next, call/invoke/use the method defined at *Question 2. a*. to generate the passwords according to the functionality assigned to you. Finally, the application should display on the screen the passwords created 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](https://ncisupporthub.ncirl.ie/hc/en-ie/articles/4407343669394-How-to-I-access-the-student-virtual-desktop-)).

# 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. **This is a submission requirement.**

1. 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. **This is a submission requirement.**

# 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.

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. [↑](#footnote-ref-1)
2. 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* [↑](#footnote-ref-2)