National College of Ireland

National College of Ireland

Higher Diploma in Science in Computing, Year 1, HDSDEV_SEP
Higher Diploma in Science in Computing, Year 1, HDSDEV_SEPOL_YR1
Higher Diploma in Science in Computing, Year 1, HDCSDEV_INT
Higher Diploma in Science in Computing, Year 1, HDAIML_SEP
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_JANOL

Autumn/Repeat Terminal Assignment-based Assessment (TABA) – 2020/21
Deferred Terminal Assignment-based Assessment (TABA) – 2020/21

Release Date on Moodle: 16th of August 2021 @09:00am
Online Moodle Submission Deadline: 18th of August 2021 @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 instructions presented in the ***Assignment Description*** section for details about the functionalities you have to implement. **This is a submission requirement**. Note that if the incorrect functionalities are implemented, no marks will be provided for those particular functionalities.

Assignment Description

For this assessment you are required to develop an application for Strings analysis and processing. The application you have to implement 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 commissioned the development of an application that enables a user to check the strength of their password for a particular account. The application prompts the user to enter a password. Next, the application uses the given password to determine its strength. The functionality that you should implement to compute the password strength has been assigned based on the penultimate digit of your student ID number. Please check the *Table 1 Question 1. a. Compute Method Functionality* for details about the functionality assigned to you.

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 password
- A compute method to determine the password strength 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 message about the password strength

Name the instantiable class **PasswordAnalyser**.

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

(30 marks)

☐ **Compute Method Functionality**: You are required to implement the functionality assigned to you in *Table 1 Question 1. a. Compute Method 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 1 Question 1. a. Compute Method Functionality

Penultimate (i.e. second to last) digit of your student ID	**Functionality ID**	**Functionality**[1]	**Examples (password and corresponding strength message)**
1 or 3 or 5 or 7 or 9	F1	The method should determine whether the given password is a weak password or a good password, and set a corresponding message. A good password is a password which • has at least 10 characters • and contains one of the following characters '&', '+', '@' or '?' • and contains at least two digits (i.e. any of the numbers from 0 to 9). The method should store in the message one of the following texts: • "password is good" – when all the above properties/characteristics are met • "password is weak", otherwise Note that for the rule regarding the existence of at least two digits, the very same digit may be repeated to count towards the two required digits (please see the examples in the next column).	For example • if the password is "apple" then the method should determine that the password is weak, and store in the message the text *"password is weak"* (as the given password has less than 10 characters) • if the password is "qwerty12345" then the method should determine that the password is weak, and store in the message the text *"password is weak"* (as the given password does not contain any of the characters '&', '+', '@' or '?') • if the password is "apple?fruits1" then the method should determine that the password is weak, and store in the message the text *"password is weak"* (as the given password does not contain at least two digits) • if the password is "1apple?fruits3" then the method should determine that the password is good, and store in the message the text *"password is good" (as all the characteristics/rules are met)* • if the password is "1app1e?@ruits" then the method should determine that the password is good, and store in the message the text *"password is good" (as all the characteristics/rules are met)*

[1] **Disclaimer**: note that the rules specified for this functionality are provided for this assessment purposes only and are not intended to determine the strength/quality of a password in a real world setting.

Penultimate (i.e. second to last) digit of your student ID	**Functionality ID**	**Functionality**[1]	**Examples (password and corresponding strength message)**
0 or 2 or 4 or 6 or 8	F2	The method should determine whether the given password is a strong password or not, and set a corresponding message. A strong password is a password which • has a minimum of 14 characters • and contains at least three of any of the following special characters '!', '$', '*' or '%' • and contains at least a digit (i.e. any of the numbers from 0 to 9). The method should store in the message one of the following texts: • "password is strong" – when all the above properties/characteristics are met • "password is NOT strong", otherwise Note that for the rule regarding the existence of the three special characters, the very same special character may be repeated multiple times to count towards the three required special characters (please see the examples in the next column).	For example • if the password is "tree" then the method should determine that the password is not strong, and store in the message the text *"password is NOT strong"* (as the given password does not have a minimum of 14 characters) • if the password is "fruit$tree1%zy" then the method should determine that the password is not strong, and store in the message the text *"password is NOT strong"* (as the given password does not have at least three of any of the special characters '!', '$', '*' or '%') • if the password is "fruit$$tree%zy" then the method should determine that the password is not strong, and store in the message the text *"password is NOT strong"* (as the given password does not have at least a digit) • if the password is "fru1t$tree%z$yx" then the method should determine that the password is strong, and store in the message the text *"password is strong"* (*as all the characteristics/rules are met*) • if the password is "fru1t$tree$z$yx" then the method should determine that the password is strong, and store in the message the text *"password is strong"* (*as all the characteristics/rules are met*)

Example: A student with the student ID = 203456**7**8 would implement the functionality identified by *F1* (because the penultimate digit of that student ID is **7**).

b. Develop an application that uses the instantiable class *PasswordAnalyser* (the instantiable class previously developed) to compute the strength of a password. The application will display the computed message about the strength of the given password on the screen. In the application class, please add a short comment for each method of the *PasswordAnalyser* class that you use/call in your application to explain why that method is needed.
Name the application class **PasswordAnalyserApp.**

(15 marks)

Question 2. Develop further the application as follows:

a. First, implement in the instantiable class *PasswordAnalyser* (the instantiable class previously developed at Question 1. a.) **another method** which takes in as a parameter a sentence, i.e. a short piece of text that can contain only letters, spaces (i.e. ' '), dots (i.e. '.'), exclamation marks (i.e. '!') and question marks (i.e. '?'). (**Please note that you are not required to validate the input, we assume that the sentence is well formed.**). The method uses the given sentence to create the corresponding passphrase[2]. The created passphrase should be saved in a suitable instance variable. The passphrase is created using several rules. The rules that your method should use to create the passphrase have been assigned based on the penultimate digit of your student ID number. Please check the *Table 2 Question 2. a. Rules to create the passphrase* for details about the rules assigned to you.
Note that the method should work irrespective of how the user provides the sentence i.e. using 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 letters to be included in the passphrase (i.e. they should be the same as the letter case provided by the user in the initial sentence).

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

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.

(20 marks)

☐ **Rules to create the passphrase**: Use *Table 2 Question 2. a. Rules to create the passphrase* and based on the **penultimate i.e. second to last digit of your student ID** find the rules you have to implement to create the passphrase. **IMPORTANT: This is a submission requirement.** If the incorrect rules are implemented, no marks will be provided for that functionality.

[2] A passphrase is *"a secret phrase, or a combination of letters and numbers, used to get access to a computer or a computer file"* (source: https://dictionary.cambridge.org/dictionary/english/passphrase).

Table 2 Question 2. a. Rules to create the passphrase

Penultimate (i.e. second to last) digit of your student ID	**Rules ID**	**Rules to create the passphrase**	**Examples (sentence and corresponding passphrase)**
0 or 1 or 2 or 3 or 4	RCP1	The passphrase is created using the following rules: • Reverse the characters from the given sentence • Each letter 't' is replaced by number 1 • Each letter 'e' is replaced by number 3 • Each dot (i.e. '.') is replaced by '&' • Each question mark (i.e. '?') is replaced by '%' • All the other characters will remain the same in the passphrase	For example • if the method is passed in the sentence "I ENJOY playing chess and basketball." o then the method should create the passphrase "&llab13ksab dna ss3hc gniyalp YOJN3 I" • if the method is passed in the sentence "Do you enjoy going TO the TheatrE?" o then the method should create the passphrase "%3r1a3h1 3h1 O1 gniog yojn3 uoy oD"
5 or 6 or 7 or 8 or 9	RCP2	The passphrase is created using the following rules: • The passphrase starts with the number of characters in the given sentence (which includes the spaces) • Each space is going to be added twice • Each letter 'a' is replaced by number 4 • Each letter 'g' is replaced by number 9 • Each letter 'i' is replaced by an exclamation mark (i.e. '!') • All the other characters will remain the same in the passphrase	For example • if the method is passed in the sentence "I enjoy progrAmminG in Java." o then the method should create the passphrase "28! enjoy pro9r4mm!n9 !n J4v4." • if the method is passed in the sentence "Do you enjoy rIver rAfting?" o then the method should create the passphrase "27Do you enjoy r!ver r4ft!n9?"

Example: A student with the student ID = 203456**7**8 would implement the rules identified by *RCP2* (because the penultimate digit of that student ID is **7**).

b. Next, develop further the application class *PasswordAnalyser*App (the class previously developed at Question 1. b.) to use the method defined at *Question 2. a*. to create multiple passphrases. The application should allow a user to enter multiple sentences, and for each sentence should create the corresponding passphrase. Please check the *Table 3 Approaches to entering multiple sentences to create passphrases* for details about the approach you have to implement in order for the application to create multiple passphrases. The approach you have to implement has been assigned to you based on the last digit of your student ID number. The application should display on the screen each of the passphrases created by the method implemented at *Question 2. a.* Each of the passphrases should be displayed on a new line.

(15 marks)

☐ **Approach for entering multiple sentences to create passphrases**: Use *Table 3 Approaches to entering multiple sentences to create passphrases* 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 3 Approaches to entering multiple sentences to create passphrases

Last digit of your student ID	Approach ID	Approaches to entering multiple sentences to create passphrases (MSA)
1 or 3 or 5 or 7 or 9	MSA1	Ask the user at the beginning how many passphrases they would like to create, and ensure that the application enables the user to provide that amount of sentences and for each sentence creates its corresponding passphrase.
0 or 2 or 4 or 6 or 8	MSA2	Ask the user to provide a sentence and after the corresponding passphrase is created and displayed on the screen, ask the user if they would like to create another passphrase. 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 passphrases are created.

Example: A student with the student ID = 2034567**8** would implement the approach corresponding to MSA2 (because the last digit of that student ID is **8**).

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.

Please note: The use of IDE's for development **is not** permitted.

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 ***Repeat/Deferred 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
   - 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 ***Repeat/Deferred 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. (15 marks)
  - Question 2. a. (20 marks)
  - Question 2. b. (15 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 functionality of the application is 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 description of 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.