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_JAN20_O

Semester One Terminal Assignment-based Assessment (TABA) – 2020/21

Release Date on Moodle: 14th of December 2020
Online Moodle Submission Deadline: 11th of January 2021 @23:55

Software Development

Assignment Description

For this assignment, you should develop an application that allows 2 users to play the "*Find Computer Words Game*". "*Find Computer Words Game*" is a turn-based game, where the players take turns to provide one word at a time according to the rules of the game.

The "*Find Computer Words Game*" works as follows:

- At the start of the game, the game should inform the players how they can acquire points (note that details for awarding the points are presented later in this document). The game is formed from multiple rounds. Please see below the section named ***Instructions and Checklist*** for details about the approach you have to implement in order to provide multiple rounds for the game.
  - Per round:
    - At the start of each round, the application (i.e. computer) randomly selects 12 letters from the English alphabet and displays the 12 letters to the players (note that there are 26 letters in the English alphabet). The 12 letters do not have to be unique (i.e. the same letter can be randomly selected multiple times at the beginning of a certain round).
    - Next, each player takes turn to try to make a word using as many of the 12 letters as the player can but using each letter only once (note that in the case that a randomly selected letter has multiple occurrences then the players can use that letter as many times as occurrences the letter has). Input (i.e. word) validation is required according to the rules of the game, including the following rules:
      - The word has to be formed only from the 12 letters randomly selected for that round (i.e. no other letters can be used).
      - The words provided by the players have to be valid computer-related words in English. Each time a player enters a word the game must check the word against an array of valid words. The word the player entered should be displayed back on the screen with the message whether that word is valid or not. For the purposes of this game, use the following 100 computer-related words, store them in an array and use that array to validate the words provided by the players: *algorithm, application, backup, bit, buffer, bandwidth, broadband, bug, binary, browser, bus, cache, command, computer, cookie, compiler, cyberspace, compress, configure, database, digital, data, debug, desktop, disk, domain, decompress, development, download, dynamic, email, encryption, firewall, flowchart, file, folder, graphics, hyperlink, host, hardware, icon, inbox, internet, kernel, keyword, keyboard, laptop, login, logic, malware, motherboard, mouse, mainframe, memory, monitor, multimedia, network, node, offline, online, path, process, protocol, password, phishing, platform, program, portal, privacy, programmer, queue, resolution, root, restore, router, reboot, runtime, screen, security, shell, snapshot, spam, screenshot, server, script, software, spreadsheet, storage, syntax, table, template, thread, terminal, username, virtual, virus, web, website, window, wireless.* In this game, only the words stored in the array of words are considered valid words.
      - Per round a word can be entered only once (i.e. the two users cannot provide the very same word in the same round).
    - Each time a player provides a valid word, the player receives points according to the rule for awarding points. Please see below the section named ***Instructions and Checklist*** for details about the rule you have to implement. The points received for that word should be displayed on the screen. Also, the total points received should be displayed on the screen.
- At the end of the game, the game displays who the winner is, and the number of points received by each player. The winner of the game is the player who has received more points.

Note that the application should work irrespective of how the players provide the words i.e. using upper case letters, lower case letters or a combination of both upper case and lower case letters.

Next, is presented a short example for the players taking turns and providing words according to the rules of the game. Note that for simplicity of exposition the example below does not present the validations you are required to implement such as checking that the word is a valid word. For a game, let's assume:

- An example for a round:
  - At the beginning of the round the application (i.e. computer) randomly selects 12 letters, and then displays them. Let's assume the randomly selected letters are t, o, e, i, b, l, f, d, e, n, f, s
  - Next, each player takes turn to try to make a word using as many of the 12 letters (i.e. t, o, e, i, b, l, f, d, e, n, f, s) as the player can but using each letter only once (note that in the case that a randomly selected letter has multiple occurrences then the players can use that letter as many times as occurrences the letter has; for instance, in this example, letter 'f' has two occurrences, therefore, it can be used up to two times in a word). Input (i.e. word) validation is required according to the rules of the game.
    - *Player 1*: The first player enters a word formed from as many of the 12 letters as the player can. Let's assume the player enters the word "offline". The player receives points, if any, according to the rule you have been assigned to implement. The points received for that word should be displayed on the screen. Also, the total points received should be displayed on the screen.
    - *Player 2*: Next, the second player enters another word formed from as many of the 12 letters as the player can. Let's assume the player enters the word "node". The player receives points, if any, according to the rule you have been assigned to implement. The points received for that word should be displayed on the screen. Also, the total points received should be displayed on the screen.

- Another round is started
  - At the beginning of the round the application (i.e. computer) randomly selects 12 letters, and then displays them. Let's assume the randomly selected letters are y, k, w, o, e, l, r, n, d, r, n, e
  - Next, each player takes turn to try to make a word using as many of the 12 letters (i.e. y, k, w, o, e, l, r, n, d, r, n, e) as the player can but using each letter only once (note that in the case that a randomly selected letter has multiple occurrences then the players can use that letter as many times as occurrences the letter has; for instance, in this example, letter 'e' has two occurrences, therefore, it can be used up to two times in a word). Input (i.e. word) validation is required according to the rules of the game.
    - *Player 1:* The first player enters a word formed from as many of the 12 letters as the player can. Let's assume the player enters the word "keyword". The player receives points, if any, according to the rule you have been assigned to implement. The points received for that word should be displayed on the screen. Also, the total points received should be displayed on the screen.
    - *Player 2:* Next, the second player enters another word formed from as many of the 12 letters as the player can. Let's assume the player enters "kernel". The player receives points, if any, according to the rule you have been assigned to implement. The points received for that word should be displayed on the screen. Also, the total points received should be displayed on the screen.

- Another round is started, and the game is played as described above until all the rounds have been played. The multiple rounds functionality has to work according to the multiple rounds approach you have been assigned to implement.
- At the end of the game, the game displays who the winner is, and the number of points received by each player. The winner of the game is the player who has received more points.

Instructions and Checklist

Please follow the instructions. Ensure that you complete each of the following checklist items:

☐ Multiple rounds approach: The game is formed from multiple rounds. Use *Table 1 Approaches to multiple rounds per game* and based on the last digit of your student ID find the approach you have to implement in your game. Note that you must only implement the rule according to the aforementioned requirement.

Table 1 Approaches to multiple rounds per game

Last digit of your student ID	Approach ID	Multiple rounds approach (MRA)
0 or 2 or 4 or 6 or 8	MRA1	Ask the players after each round whether they would like to play another round. If they answer yes, another round should start, otherwise the game ends.
1 or 3 or 5 or 7 or 9	MRA2	Ask the players at the beginning of the game how many rounds they would like to play, and allow the players to play that amount of rounds.

Important: each student has to implement the approach based on Table 1.
Example: student ID = 20345678 would implement the approach corresponding to MRA1 (because the last digit of the student ID is 8). This is a submission requirement. If the incorrect approach is chosen, no marks will be provided for that functionality.

☐ Rule to award points: Each time a player provides a valid word, the player receives points according to a rule. Use *Table 2 Rules to award points* and based on the penultimate i.e. second to last digit of your student ID find the rule you have to implement to award points in your game. Note that you must only implement the rule according to the aforementioned requirement.

Table 2 Rules to award points

Penultimate digit of your student ID	Rule ID	Rule to award points (RP)	Examples (words and corresponding points)
0	RP0	The player receives 4 points for words longer than 5 characters, otherwise 1 point.	“bit” – 1 point “binary” – 4 points
1	RP1	The player receives double the amount of points as the number of vowels in the word.	“bit” – 2 points “binary” – 4 points
2	RP2	The player receives the same amount of points as the number of characters in the word for words with less than 7 characters, or double the amount of characters in the word for words with at least 7 characters.	“bit” – 3 points “digital” – 14 points
3	RP3	The player receives the same amount of points as the number of occurrences of letters ‘e’ and ‘t’ in the word.	“bit” – 1 point “restore” – 3 points
4	RP4	The player receives 3 points for a word starting with a vowel and 1 point for a word starting with a consonant.	“algorithm” – 3 points “bit” – 1 point
5	RP5	The player receives the same amount of points as the number of duplicated vowels in the word.	“bit” – 0 points “restore” – 2 points
6	RP6	The player receives the same amount of points as the number of characters in the word when the word does not contain both the letters ‘a’ and ‘e’; otherwise 2.5 points.	"application" – 11 points “screen” – 6 points “keyboard” – 2.5 points
7	RP7	The player receives the same amount of points as the number of characters in the word excluding the letters ‘i’, ‘u’ and ‘s’.	“bit” – 2 points “security” – 5 points
8	RP8	The player receives the same amount of points as either the number of characters in the word for words longer than 5 characters, or the amount of characters times 0.75 for words that have at most 5 characters.	“bit” – 2.25 points “shell” – 3.75 points “restore” – 7 points
9	RP9	The player receives triple the amount of points as the number of consonants in the word.	“bit” – 6 points “shell” – 12 points

Important: each student has to implement the rule based on Table 2.
Example: student ID = 20345678 would implement the rule corresponding to the rule identified by RP7 (because the penultimate digit of that student ID is 7). <u>This is a submission requirement</u>. If the incorrect rule is chosen, no marks will be provided for that functionality.

Application Development

The applications should be developed, compiled, and run using a text editor such as TextPad (or similar alternative) 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 Citrix Portal via Moodle Virtual Desktop (i.e. NCI Citrix Virtual Desktop).

Your application **should make use of instantiable classes** and demonstrate your ability to use other programming concepts, which we have covered in the Software Development module.

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) of the application

   Submit the complete Java source code of the application 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)
   - The class diagram for your application
   - Any decisions you take in designing and implementing your application should be specified in the report
   - Screenshots of the application's screens/output
     - Play the game – note that you will play the role of both players – and take some screenshots while playing the game (for example, the output of your application when the game starts, when the two players provides words one after another, the output with the points allocated, the output after the first round of the game, the output of your application after the third round, the output of your application at the end of the game, etc.).
     - You can take a screenshot by using the Snipping Tool. Alternatively, you can take a screenshot by using the key "Print Screen" of the keyboard. On the keyboard, the key may have one of the following labels Print Scrn, Prnt Scrn, Prt Scn, Prt Scr, Prt Sc or Pr Sc.
   - *Note that the entire java source code of your application must also be included as an appendix in your report*! 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 and Game Implementation (55 marks)
    - All requirements have been implemented. Game is functional.
    - All required validations have been implemented
    - Multiple rounds can be played.
- Compiling and Running Code (10 marks)
    - Fully compiling and executing application with no syntax or logical errors which addresses the full requirements of the application (10 marks)
- Object Oriented Design (10 marks)
    - The application should make use of instantiable classes (For example, declare classes where you implement constructors, setter methods, getter methods, processing methods; reuse these classes in your application; etc.) (10 marks)
- Good coding practices and code understanding (15 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 (5 marks)
- Report (10 marks)
    - Evidence of designing and planning of the application prior to coding (For example, IPO diagram, class diagram, flow chart, pseudocode, etc.) (5 marks)
    - Evidence that the application has been manually tested (i.e. relevant screenshots included and documented in the report) (5 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.