Higher Diploma in Science in Computing (HDCOMP)
HDCOMP, Year 1, HDSDEV_JAN
HDCOMP, Year 1, HDSDEV_JANOL_Y1

Software Development
Terminal Assignment-based Assessment (50%)

The end-of semester terminal exam assessment for the Software Development module is being replaced with a terminal assignment-based assessment. The terminal assignment-based assessment represents 50% of the overall module mark.

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 ON ANY WEBSITES. YOU ARE NOT ALLOWED TO PUBLISH/SHARE YOUR SOLUTION WITH OTHERS.

Key Submission Details

Submission Deadline: Friday 8th of May 2020, 23:55

This is a terminal assessment. Consequently, **NO late submission or extension are allowed**.

Assignment Description

For this assignment, you should develop an application that allows 2 users to play the "Java Words Game". "Java 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.

A game of "Java Words Game" works as follows:

- At the start of a game, each of the two users is given *N* lives, and each of them starts the game with zero points. The game should display the number of lives the players received. In addition, the game should inform the players how they can acquire points (note that details for awarding the points are presented later in this document).
- A game is formed from multiple rounds. Per round:
    - At the start of each round, the game (i.e. computer) randomly selects one letter from the English alphabet and displays the letter to the players (note that there are 26 letters in the English alphabet).
    - First, the first player enters a word that begins with the letter which has been randomly selected at the beginning of that round. The word has to be formed from at least 3 letters. Input validation is required according to the rules.

- Second, the second player enters a word that begins with the last two letters of the last word provided by the other player. The word has to be formed from at least 3 letters. Input validation is required according to the rules.
- Next, the first player enters a word that begins with the last two letters of the last word provided by the other player. The word has to be formed from at least 3 letters. Input validation is required according to the rules.
- The round continues with the players taking turns and entering a word according to the rules specified above. 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.
- A round ends when a player is not able to provide a word that begins with the last two letters of the word previously entered by the other player. A player shows that he/she cannot provide such a word when he/she enters a "-". Note that the player may not be able to provide a word either because he/she does not know a word that begins with the last two letters of the last word provided by the other player or because such a word may not exist (i.e. there is no word that begins with those 2 letters). At that time, the player loses one life, and the other player will start the next round.

- The game is played as described above until one of the players loses all the lives allocated by the game. The winner of the game is the user who still has lives left at the end of the game.
- At the end of the game, the game displays who the winner is, and the number of points received by each player.

Note that the words provided by the players have to be valid 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, we use a limited vocabulary. The Java source code of a limited vocabulary comprised of 1024 English words encoded as an array of words is provided on our Moodle page (see *LimitedVocabulary.java* file), and you should use it as the vocabulary for your application. Please note that you are not required to add additional words to the vocabulary, but should you wish to add extra words you can do so. In this game, only the words stored in the array of words are considered valid words.

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.

Your application should allow a user to play a game multiple times. Please see below the section named ***Instructions and Checklist*** for details about the approach you have to implement.

Instructions and Checklist

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

☐ **Player number of lives**: at the start of a game, each of the two users is given *N* lives. Define a constant in your program for the number of lives, and assign a value of your choice that is not smaller than 3.

☐ **Rule to award points**: Each time a player provides a word, the player receives points according to a rule. Use Table 1 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 previous requirement.

Table 1 Rules to award points

Penultimate digit of your student ID	Rule ID	Rule to award points	Examples (words and corresponding points)
0	R0	The player receives 5 points when he/she provides two consecutive words longer than 7 characters; otherwise the player receives 1 point.	Note that the next example assume that one player provides all the words when his/her turn comes. Let's assume that the sequence of provided words are: "aria" – 1 point "solution" – 1 point "xylophone" – 5 points "world" – 1 point "fraction" – 1 point
1	R1	The player receives the same amount of points as either the number of characters in the word or double the amount of characters in the word for words with at least 9 characters.	"continent" – 18 points "air" – 3 points
2	R2	The player receives the same amount of points as the number of vowels in the word.	"freedom" – 3 points "gym" – 0 points "air" – 2 points
3	R3	The player receives 1 point for a word starting with a vowel and 2 points for a word starting with a consonant.	"air" – 1 point "continent" – 2 points
4	R4	The player receives the same amount of points as the number of duplicated vowels in the word.	"moon" – 2 points "cheerleader" – 4 points "answer" – 0 points
5	R5	The player receives the same amount of points as the number of occurrences of letters 'u' and 'j' in the word.	"jump" – 2 points "language" – 1 point "page" – 0 points

6	R6	The player receives the same amount of points as either the number of characters in the word or the amount of characters times 0.5 for words shorter than 6 characters.	“motion” – 6 points “learn” – 2.5 points
7	R7	The player receives the same amount of points as the number of consonants in the word.	“freedom” – 4 points “protect” – 5 points
8	R8	The player receives the same amount of points as the number of characters in the word excluding the letters ‘a’, ‘e’ and ‘o’.	“temperature” – 7 points “voice” – 3 points "rhythm" – 6 points
9	R9	The player receives the same amount of points as the number of characters in the word when the word contains no vowels; otherwise 1 point.	“gym” – 3 points "rhythm" – 6 points “song” – 1 point

Important: each student has to implement the rule based on Table 1. Example: student ID = 1234567**7**8 would implement the rule corresponding to the rule identified by **R7** (because the penultimate digit of that student ID is 7). This is a submission requirement. If the incorrect rule is chosen, no marks will be provided for that functionality.

☐ **Play multiple games approach**: Your application should allow a user to play a game multiple times. Use Table 2 and based on the **antepenultimate i.e. third from 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 previous requirement.

Table 2 Approaches to play multiple games

Antepenultimate digit of your student ID	Approach ID	Multiple games approach
0 or 2 or 4 or 6 or 8	A1	Ask the players at the beginning of the application how many games they would like to play, and allow the players to play that amount of games.
1 or 3 or 5 or 7 or 9	A2	Ask the players after each game whether they would like to play another game. If they answer yes, the game should start again, otherwise the application ends.

Important: each student has to implement the approach based on Table 2. Example: student ID = 1234567**8** would implement the rule corresponding to the rule identified by **A1** (because the antepenultimate digit of the student ID is 6). This is a submission requirement. If the incorrect approach is chosen, no marks will be provided for that functionality.

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

- An example for round 1:
    - At the beginning of the round the game randomly selects the letter '**c**', and then displays it
    - *Player 1*: First, the first player enters a word that begins with the letter which has been randomly selected at the beginning of the round i.e '**c**'. The word has to be formed from at least 3 letters.
        - Let's assume the player enters the word "**c**at**ch"**. The player receives points, if any, according to the rule you have to implement.
    - *Player 2*: Second, the second player enters a word that begins with the last two letters of the last word provided by the other player. The word has to be formed from at least 3 letters.
        - In this example the last 2 letters of the words are *"ch"*, therefore let's assume the player enters the word "**ch**an**ge**". The player receives points, if any, according to the rule you have to implement.
    - *Player 1:* Next, the first player enters a word that begins with the last two letters of the last word provided by the other player. The word has to be formed from at least 3 letters.
        - In this example the last 2 letters of the words are *"ge"*, therefore let's assume the player enters the word "**ge**ner**al**". The player receives points, if any, according to the rule you have to implement.
    - *Player 2:* In this example the last 2 letters of the words are *"al"*, therefore let's assume the player enters "**all**". The player receives points, if any, according to the rule you have to implement.
    - *Player 1:* In this example the last 2 letters of the words are *"ll"*, the player does not know any word that starts with "ll", therefore the player enters "**-**" consequently, the round finishes and the player loses 1 life.

- Another round is started, and this time player 2 is going to provide the first word because player 1 lost the previous round
    - At the beginning of the round the game randomly selects the letter **'d'**
    - *Player 2*: The player enters a word that begins with the letter which has been randomly selected at the beginning of the round i.e '**d**'. The word has to be formed from at least 3 letters.
        - Let's assume the player enters "**d**re**am"**. The player receives points, if any, according to the rule you have to implement.

- *Player 1*: The player enters a word that begins with the last two letters of the last word provided by the other player. The word has to be formed from at least 3 letters.
  - In this example the last 2 letters of the words are *"am"*, therefore let's assume the player enters "**am**o**ng**". The player receives points, if any, according to the rule you have to implement.
- *Player 2:* In this example the last 2 letters of the words are *"ng"*, the player does not know any word that starts with "ng", therefore the player enters "-" consequently, the round finishes and the player loses 1 life.

- Another round is started, and this time player 1 is going to provide the first word because player 2 lost the previous round. The game is played as described above until one of the players loses all the lives allocated by the game. The winner of the game is the user who still has lives left at the end of the game. At the end of the game, the game displays who the winner is, and the number of points received by each player.

Deliverables

The Project deliverables **must be submitted via Moodle**, they cannot be submitted by email. You will submit a .zip file on Moodle containing the following:

- Complete Java source code (.java files) of the application
- A report 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 project 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, output after the first round of the first game, the output of your program after the third round, the output of your program at the end of a 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.

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 games can be played.
- **Compiling and Running Code (10 marks)**
  - Fully compiling and executing application with no syntax or logical errors which addresses the main parts of the game logic (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 and objects 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.

Acknowledgment

The rules for providing the words in the above game have been adapted from a Romanian words game.