**University of Wolverhampton**

**Faculty of Science and Engineering**

**Department of Mathematics and Computer Science**

**Module Assessment**

|  |  |
| --- | --- |
| **Module** | 5CS019 – Object-Oriented Programming |
| **Module Leader** | Hiran Patel |
| **Semester** | 1 |
| **Year** | 2023/24 |
|  |  |
| **Assessment** | Portfolio |
| **% of module mark** | 100% |
| **Due Date** | See canvas portfolio submission gateway |
| **Hand-in – what?** | **Portfolio as specified in this document** |
| **Hand-in- where?** | Canvas |
|  |  |
| **Pass mark** | 40% |
| **Method of retrieval** | Submit the resit assessment (will be distributed at the end of the module) by end of resit week (July) |
| **Feedback** | Individual feedback via Canvas, in addition verbal feedback is available in class. |
| **Collection of marked work** | N/A |

For this portfolio, you will be creating a “modified” version of the famous game trivial pursuit. Trivial pursuit is a quiz-based board game where you answer questions from specific categories. For this assessment however, you will be simulating a rolling dice (generating a random number between 1 to 6) to determine which category the question is from. You will then present that question to the player, if the player submits the correct answer, they will be asked to roll again and so on until either the player completes a total of 10 questions or if the player gets 3 wrong answers before the 10 has been reached (max 3 lives). **The number of questions and lives should be a variable and set before the game has been launched, 10 questions and 3 lives are just an example.**

The assessment will be split up into multiple sections, each section builds from the previous one, so it is best to start in a chronological order.

The first part (40%) will test your knowledge of Object-Oriented Programming, file handling and Unit Testing using Java. For this part, you will create a set of classes to produce a simple quiz game, where you will quiz the user from a list of questions from a series of CSV files (6 in total (6 different categories)) which hold questions and answers. For each method you create, you should aim to have at least 2 tests for it to prove the correct output has been produced.

The functionality for the part one submission should contain the following:

1. You can add a question into the CSV file using a question and answer as a parameter (both strings). For this function, you will need to aim to mitigate as many problems as possible such as empty strings passed or empty strings for either the question or answer etc.
2. You can add multiple questions to the CSV using a LinkedList of questions and answers (both strings). For this function, you should check for errors before adding questions such as blank entries for questions and/or answers. You should also check if the number of questions match the number of answers supplied within the parameters.
3. You can remove a question and answer via the row ID.
4. You can read questions and answers (functions should return Linked Lists) from the CSV to be used within the game.
5. Finally, you should have a function which starts the game. You can add your own functionality, but the following basic rules should be implemented:
   1. When the startGame function is called in the main method, the game begins. This function will be stored in a class called GameSetup and proceed to call the appropriate functions for the game to be executed. The GameSetup class structure is explained in this brief later on.
   2. The program should then ask the user to roll a dice. For this section, the player will need to type “roll” when prompted to. This should generate a random number between 1 and 6 and present the number that has been rolled. This will determine the category of the question which will be displayed, the category picked is up to you so you should assign a number to a category. You should then automatically randomly pick a question from the table you are using.
   3. Once the question has been displayed, the game should prompt the player to type their answer. If the player guesses correctly, they should be prompted to roll again, and the question-and-answer sequence should be repeated. If the guess is incorrect, a life is lost, and the user is prompted to roll again.
   4. Step C should be repeated until either; the specified “lives” amount has been lost in which case the game ends showing the user how many answers they got correct, or, if the specific max correct answers have been given before max “lives” have been lost. For example, if the admin has set the maxQuestions to 10 in the constructor of the GameSetup class then the user should only be able to answer 10 questions correctly as 10 is the maximum score.

**You should also make a note that when a question is displayed, it must not be used again throughout that specific game so once a random question has been chosen, you will need to delete it from the list variable, so it is not chosen again.**

The structure of your program is overall up to you as this assessment tests your knowledge of the SOLID principles and general software engineering principles. The following classes however are a good template to use (but not limited to):

1. Read.java – this class reads data from a table, for this section, just the CSVs. This class should contain functions to read the questions and answers from CSV files and return them in a suitable format. The template given has four functions; one to return the questions from a single file, one to return the answers from a single file, one to return a whole set of questions from multiple files and the other to return a whole set of answers from multiple files. You can modify this class to accommodate your needs but make sure to follow the Single responsibility principle (should only have read functionality).
2. Write.java – this class writes data to a table, for this section, just the CSVs. This class should contain functions to write new questions and answers to a specific CSV. The template given has two functions which both contains 3 parameters; the first function takes in the question, the answer, and the path to the CSV file you want to write to. The second function does the same but allows multiple questions and answers to be added. The new data should be appended and not replace the previous data. Your function should also check the previous rowID so that the new row will have the previous rowID + 1.
3. Delete.java – this class should contain delete functions. The template given has a single function which takes two parameters; the rowID and the csv you want to modify. You should specify a specific Row ID and this should remove that row from the CSV file specified.
4. GameFunctions.java – This class should contain functions that the game will use in for it to run. The template given has various functions such as:
   1. pickQuestionAndAnswer – this function returns a LinkedList which will contain two String values, the question and answer. The question is chosen from a random number which is the third parameter. Parameters 1 and 2 are the list you want to choose from
   2. rollDice – this function should generate a number between 1 and 6. The template function has 2 parameters; int min and int max. These values set the lower and upper limits. For this game, you will call the function using 1 and 6 as a dice can only roll a number between these two numbers. In the future however, if you’d like to add more categories, these numbers can be modified hence why it is a good idea to allow the admin to change these numbers and not hard code them.
   3. checkAnswer – this function should compare the user inputted answer to the real answer. The template function returns a Boolean and has two parameters: String userInput and String realAnswer. You should do a comparison and return true if the answer matches or false if it’s incorrect.
5. GameSetup.java – this class should call all the relevant function to create the game. The template code gives a constructor which takes three parameters: the players name, max number of lives and the number of questions. The only other function is “void start.” This function should contain all the steps to not only set the game up but play the game. For example, here you will read all the data from the csv files into appropriate variables, you will present the user with questions, get user inputs for the answers, increment scores and lives lost etc. This will most likely be inside a loop of your choice. How the game operates is up to you, but you must use functions from different classes to operate the game.
6. Main.java – this function calls the start function from GameSetup to run the game. You should only ask for the players name, number of lives and max correct questions to input into the constructor of GameSetup.

The above class structure is only a template, there maybe some functions missing, functions which need additional functionality and some which you may not use but are important as in the future, you may use them.

**For all functions that you create, you must create at least 2 relevant tests. Each class should have its own testing class. Some are going to require you to test multiple features such as when you load the CSV file data into a variable, you may check specific indexes to see if the data has loaded properly.**

For this assessment, you will be given 5 CSV files with questions and answers. You should create an additional set of questions of your choice into a 6th CSV file. You do not need to use the ones provided but you can if you want. This data can be found in the assessment section (in modules) on canvas. The data file will be called Questions.zip.

Part 2 – Adding a Graphical User Interface (25%)

For part 2, you will create GUI for the quiz game you created for part 1. You should have at least the following functionality:

1. As the GUI opens, it will allow the user to type in their name at the beginning of the game. You should have a button or some activation in place to submit this information and store within a variable.
2. Once the name has been entered there should be menu options (these can be buttons), the first should read “admin” and the second should read “play game.” You should then create the following functions for the following options:
   1. Admin Menu:
      1. Add single question:
         1. This option should present the user with 3 text fields for the question, answer and table name they wish to store the question in. If using CSV’s, this is the name of the CSV file, when using MySQL, this should be the table name.
         2. Once all fields have been populated, you should allow the user to submit (via a button) to store that data in the correct destination. You should create a test function to read the last line of the table you added the new question to and see if it has been successfully appended.
      2. Add multiple questions:
         1. This option should present the user with 2 buttons and a text field; one button to upload a CSV file full of questions and answers, the other to submit the questions and the text field to specify which table they wish to store the questions and answers in. If using CSV’s, this is the name of the CSV file, when using MySQL, this should be the table name.
      3. Remove a question:
         1. This option should present the user with 2 text fields and submit button. One text field is for the row number and the other is for which table/file you want to remove from. The submit button to submit the request.
      4. (For all the above features, you should write test functions to see if the correct processes have been outputted from the submit buttons).
   2. Play Game:
      1. (You should follow the same instructions as part one, but this time utilise GUI components to print out the questions such as text labels (show score and dice roll), buttons (to roll dice and submit answers) etc. You can either use window builder or code the GUI from scratch, this is up to you.

Part 3 – Add a high score table (5%)

For part 3, you will need to implement a high score feature within the software. To do this, you will need to create either a CSV file or a MySQL table to hold the data. Each time a user plays the game, there name and score should be stored appropriately once the game has been finished. You should include an option when the game starts to view the high score table. This section should also include a “clear high scores” feature where all the data is cleared from the file/table. Your table should show the highest score achieved so far from all players so you will need to think about sorting algorithms.

Part 4 – Integration with MySQL (20%) –Details to be added in Week 7

Part 5 – Demonstration (10%) – Details to be added in Week 7

Your completed work for assignments must be handed in on or before the due date. ***You must keep a copy or backup of any assessed work that you submit.  Failure to do so may result in your having to repeat that piece of work.***

**Penalties for late submission of coursework**

Standard Faculty of Science and Technology arrangements apply.

**ANY late submission (without valid cause) will result in 0 marks being allocated** **to the coursework**.

**Procedure for requesting extensions**

If you have a valid reason for requiring an extension you must request an extension using e:vision. **Requests for extension to assignment deadlines should normally be submitted at least one week before the submission deadline and may be granted for a maximum of seven days (one calendar week).**

**Retrieval of Failure**

A pass of 40% or above must be obtained overall for the module (but not necessarily in each assessment task).

**Where a student fails a module they have the right to attempt the failed assessment(s) once, at the next resit opportunity (normally July resit period).  If a student fails assessment for a second time they have a right to repeat (i.e. RETAKE) the module.**

**NOTE: STUDENTS WHO DO NOT TAKE THEIR RESIT AT THE NEXT AVAILABLE RESIT OPPORTUNITY WILL BE REQUIRED TO REPEAT THE MODULE.**

**Mitigating Circumstances (also called Extenuating Circumstances).**

If you are unable to meet a deadline or attend an examination, and you have a valid reason, then you will need to request via e:vision **Extenuating Circumstances.**

**Feedback of assignments**

You will be given feedback when you demonstrate your work.

You normally have **two working weeks** from the date you receive your grade and feedback to contact and discuss the matter with your lecturer. See the Student’s Union advice page <http://www.wolvesunion.org/adviceandsupport/> for more details.

**Registration**

Please ensure that you are registered on the module. You can check your module registrations via e:Vision You should see your personal tutor or the Student Support Officer if you are unsure about your programme of study. The fact that you are attending module classes does not mean that you are necessarily registered. A grade may not be given if you are not registered.

**Cheating**Cheating is any attempt to gain unfair advantage by dishonest means and includes **plagiarism** and **collusion.** Cheating is a serious offence. You are advised to check the nature of each assessment. You must work individually unless it is a group assessment.

**Cheating** is defined as any attempt by a candidate to gain unfair advantage in an assessment by dishonest means, and includes e.g. all breaches of examination room rules, impersonating another candidate, falsifying data, and obtaining an examination paper in advance of its authorised release.

**Plagiarism** is defined as incorporating a significant amount of un-attributed direct quotation from, or un-attributed substantial paraphrasing of, the work of another.

**Collusion** occurs when two or more students collaborate to produce a piece of work to be submitted (in whole or part) for assessment and the work is presented as the work of one student alone.