TEAM MEMBERS:

1. Christian hakizimana

2. Kofi Appiah

3. Jojo Fynn Mensah

PROJECT TOPIC: **A MULTIPLE-CHOICE QUIZ APPLICATION**

**Introduction**

**Nowadays, due to population increase, more and more paper is produced and used for many official examinations. Paper production is harmful to the environment. Paper manufacturing uses a staggering forty per cent of the world’s wood, therefore contributes to deforestation. Also, the processes involved in converting wood to paper leads to the release of toxic greenhouse gases** ("Impacts of Paper on the Environment", 2019)**. Furthermore, it is tedious for examiners to grade a large number of paper-based examinations, and manually record the scores of each test taker.**

**Our solution to this problem is a multiple-choice quiz application. As it is a digital solution, it eliminates the use of paper-based systems. Our application has been designed to allow users to set multiple choice questions for quiz takers. The application stores test takers scores as well as their information such as name, age and gender. These features will save time in collecting and accessing information on test-takers.**

Background

Two data structures were used to implement the quiz application; arrays and a map implementation using trees. An array is a linear data structure that serves as a container to hold a constant number of values of the same type ("What is Array in Java? - Definition from Techopedia", 2019). Hence, we used string arrays to store the questions set by the examiner as well as the answers to the questions. Also, a two-dimensional array contained the multiple-choice options for each question. The treemap implementation used in our application was the java TreeMap class, which is a red-black tree-based implementation, that provides an efficient means of storing key-value pairs ("TreeMap in Java - javatpoint", 2019). For our application, the treemap stored the information of the test-takers as values and their ID numbers as keys. The reason for choosing the treemap to perform this task was because we do not want to restrict the number of people that can take the examination and, the runtime for searching for test-taker information is O(log(n)), that is acceptable compared to most data structures.

Approach

Our program comprises of six classes; a multiple-choice class, a person class, a database class, and the main class to run the entire program.

**Multiple choice class**

Multiple choice class is the blueprint from which a multiple-choice quiz is created. There is a multiple-choice tester class which enables the methods specified in the multiple-choice class to be implemented. The class has string array instance variables to store the questions, answers and multiple-choice options. Other instance variables were created to store the number of questions, the number of options for each question and the pass mark for the test as an int data type. A parameterized constructor was created to give values to the instance variables of the multiple-choice class, and setter methods were created for the array instance variables of the class. A check answer method was created to grade the student answers, and the load method was used to obtain test taker information that was saved in a .sav file. The multiple-choice tester class guides the examiner to create a multiple-choice quiz. It uses a scanner object to take values from the test creator for the pass mark, the number of questions and number of options and passes them as parameters into the parameterized constructor of the multiple-choice class that has been instantiated. The scanner object also takes the examiner’s questions, answers and options for each question and stores in three different arrays. Then using the setter methods of the multiple-choice class, the arrays are assigned to their respective instance variables of the multiple-choice object. The multiple-choice object created by the user is stored in a .sav file in the Questions folder.

**Person class**

Person class is the framework from which a test taker’s information is stored. This class has instance variables for the name, age, gender, email, score and pass/fail status for a test taker. A parameterized constructor which accepts all six instance variables was created in addition to a default constructor. Setter and accessor methods were created for the name, age, email and gender instance variables. A toString method was created to show the information stored in a person object as opposed to the memory address of the object. The person class has a main file called PersonMain.java that uses a scanner class to take information about a test taker that is their name, age, gender and email. Also, it generates and prints out an index number. The main class then loads the questions created by the examiner and prints them for the test taker to begin answering. Immediately after the test is completed, the score of the test taker is computed by the checkAnswer() method of the multiple-choice class and his or her pass/fail status is determined. The name, age, gender, email, score and pass/fail status are assigned to a person object and saved in a .sav file in the Testtaker folder.

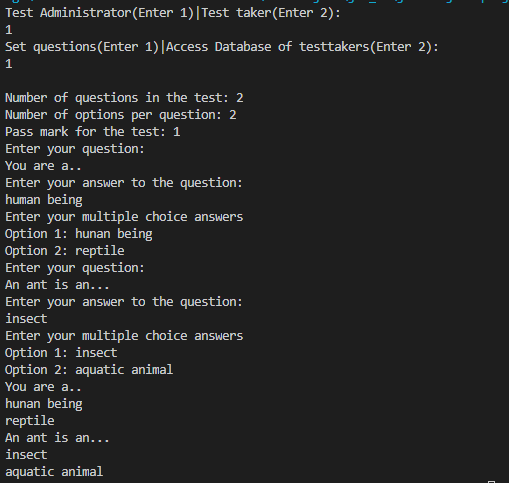
**TreeMap**

The Database.java file was used to implement the treemap for storing the user’s information. The class had a main method which looped through the Testtaker folder containing the .sav files of the test-takers. For each person, their data was loaded into a person object and placed as a value in the treemap with the key being their iD number. In addition, the test taker can specify the number of queries in obtaining test takers information from the treemap. To obtain the test takers information, the administrator enters the iD number of the particular test taker, and his or her information is printed.

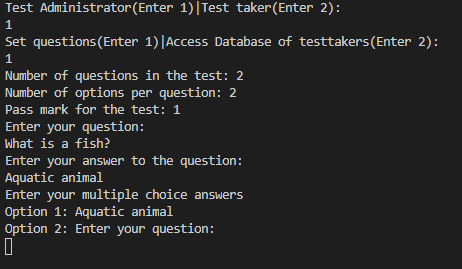
**Main class**

Results

One output of our code is below



The code was tested by running the main class (Main.java) and entering generic questions to see if it will run smoothly. Problems were experienced with our scanner variables. The nextLine() method was skipping certain inputs to be read given by the user. This issue occurred because when the nextInt() method of Scanner class is reading the number of questions given by the test taker it returns the value as expected, but the cursor remains just after it hence it makes it difficult to read the next scanner object as seen in the picture below.



The issue was later fixed by parsing all int scanner variables into Integers whiles simultaneously reading them as string e.g. int var = Integer.parseInt(sc.nextLine()) and consuming leftover newline using nextLine() method e.g. sc.nextLine(). Our code also manages to print questions, accept user answers and compute their score after they take the test. Also, it allows test administrators to get test-taker information, which is stored in a tree map

Limitations of our program include:

* Test administrator can not include diagrams in his examination.
* No time duration can be set for an examination.

Conclusion

The goal of this project was to create a quiz application to save examiners the hassle of using paper-systems for examinations. We learnt that to create an effective application, we need mastery of java, in order to understand all its syntax and in-built classes and functions. Possible improvements for our work in the future include adding a graphical user interface to our application. Quiz makers would be able to include diagrams and pictures in their quizzes and also set a time limit. In addition, they will be able to define requirements for eligibility to write the examinations they create.

Acknowledgements

Our person class, Person.java, was taken from the solutions for Homework 1 sent by Emmanuel Jojoe Ainoo, Faculty Intern for the Data Structures and Algorithms course.

The code to save the Person and MultipleChoice1 objects as .sav files was adapted from a stack overflow response and modified to suit our code. Retrieved from Stack overflow thread: https://stackoverflow.com/questions/30086741/how-to-use-variables-from-main-method-in-another-class-serialization?rq=1

References

Impacts of Paper on the Environment. (2019). Retrieved 6 December 2019, from

https://community.aiim.org/blogs/dennis-kempner/2016/12/06/impacts-of-paper-on-

the-environment

TreeMap in Java - javatpoint. (2019). Retrieved 6 December 2019, from https://www.javatpoint.com/java-treemap

What is Array in Java? - Definition from Techopedia. (2019). Retrieved 6 December 2019,

from https://www.techopedia.com/definition/1143/array-java