Technical Guide

Quiz Manager

By: Rellu L Naga Durga Venkata Sai Mahesh Babu

July 04, 2019

1. Introduction

1.1. ProjectBackground

This project is a console application quiz - preparation and execution. This document is prepared by Rellu L Naga Durga Venkata Sai Mahesh Babu for his Fundamental Java Project in the Computer Science Master's Program at L'École Pour l'Informatique et les Techniques Avancées (EPITA).

1.2. ProjectOverview

In this project, user can login as a student, take a test and get their grades at the end of the quiz. In addition, user can login as a teacher, create questions and quiz.

1.3. ProjectScope

In Scope :

1. Automatically assembles quizzes using open questions and multiple-choice questions
2. Auto-grading for multiple choice questions
3. Creating data access object with CRUD methods
4. Export quiz to plain text and it's answers in a text file with score
5. Two Admin were created in database so that can login and do operations

Out of Scope :

6. Incorporating Associative questions
7. Exporting quiz to PDF
8. Using an algorithm to arrange quiz
9. Creating and Stylizing the GUI
10. Creating different user accounts other than Student
11. Providing real questions for the Quiz Manager application

1.4. ProjectDependencies

In order to run the program, the following steps are required:

- Install h2 JDBC
- Install Java 8
- Create tables in database
- Add questions
- Add Adimin Users and their passwords

1.5. Acronyms and abbreviations

Acronyms	Meaning
CRUD	(Create, Read, Update, Delete)Functions that are implemented in relational database applications
DAO	(Data Access Object) Service class for the application to communicate with the H2 database
UML	(Unified Modeling Language)a standard way to visualize the design of a system
MCQ	(Multiple Choice Question) Question with enumerated possible answers, implemented with radial buttons in our application
OQ	(Open question) Question with a free-text field response and no predefined structured answer
JDBC	(Java Database Connectivity) application programming interface (API) for the programming language Java

2. Classes

2.1.Launcher

Launcher class is one of the main classes of the program. It is a class used to navigate to other classes, this class contains main method. It holds the main information about the processes. It gives to options to user to select, user can be a student who takes the quiz or Admin who creates quiz.

2.2.Admin

Admin class is the main responsible for admin behavior of admin who logs into it by giving credentials of login. It displays the following options:

Quiz Creation
Create Question
update Question
delete Question
search Question

Admin can select one option from above list. User can work according but using it.

2.3. Student

Student class is the main responsible for student behavior of student who takes quiz or exam. He can select the
subject which they want to take based on the topics given by the Admin. The Student can select Open Questions or Multiple choice questions based on their interest, At last score will be displayed to user and the questions displayed for user and with corresponding correct answers will be shared in text file.
Student can get their result inn text file.

2.4. Question

Question class is one of the main classes of the program. It is an abstract class used to derive other specific type questions. But it holds the most information (common attributes of questions) and most of the processes are done through this class. The common attributes includes content, topics, difficulty.

2.5. Answer

Answer is a helper container that holds an answer for a specific question. It is an abstract class to derive question type specific answers out of it, since the correct answer containers for questions are different for different type of questions.

- **Multiple Choice Answer**

Since there is only one correct answer for multiple choice question, Multiple Choice Answer holds only answer given by the user. To correct it, only checking if it matches with the correct answer of associated question will be enough.

- **Open Answer**

Open answer only holds the input given by user. Since there is a hint, the given answer should match the correct answer or else it will consider as false, partial answers are not awarded with marks.

3. Business Requirements

BR#	Requirement Description
1	Create a quiz based on user's specifications (topics)
2	Use questions stored in database (using h2 database)
3	Allow a student to take a quiz
4	Correct MCQ questions and display result at the end of the evaluation
5	Export this quiz under a plain text format to text file
6	Search questions by topic
7	Use CRUD operations on a question (create, read, update, or delete a question)
8	Use existing credentials to from data base to create the quiz
9	3 types of questions: associative, multiple choice, and open questions.

4. System Specifications

4.1. Functional Requirements

BR#	Implementation

1	**Admin can customize their quiz when instantiating the Quiz object, which takes a list of topics as one of the parameters.** `public List<Quiz> search(Quiz quizCriterion) throws SearchFailedException {` `String searchQuery = ConfigurationService.getInstance()` `.getConfigurationValue(ConfigEntry.DB_QUERIES_QUIZ_SEARCHQUERY, "");` `List<Quiz> quizList = new ArrayList<>();` `try {Connection connection = getConnection();` `PreparedStatement pstmt = connection.prepareStatement(searchQuery);` `pstmt.setInt(1, quizCriterion.getId());` `pstmt.setString(2, "%" + quizCriterion.getTitle() + "%");` `ResultSet rs = pstmt.executeQuery();` `while (rs.next()) {` `int id = rs.getInt("ID");` `String topic = rs.getString("NAME");` `Quiz quiz = new Quiz(topic);` `quiz.setId(id);` `quizList.add(quiz);` `}` `rs.close();` `} catch (SQLException e) {` `throw new SearchFailedException(quizCriterion);` `}` `return quizList;` `}`
2	**Admin can use the questions that are already used and stored in the database.** `public List<Question> searchQ(String tpc) {` `List<Question> queList = new ArrayList<Question>();` `Connection connection;` `try {` `connection = getConnection();` `PreparedStatement pstmt = connection.prepareStatement(SEARCH_QUES);` `pstmt.setString(1, tpc);` `ResultSet rs = pstmt.executeQuery();` `while (rs.next()) {` `String topic = rs.getString("QUESTION");` `Question ques = new Question(topic);` `queList.add(ques);` `}` `rs.close();` `} catch (SQLException e) {` `System.err.checkError();` `}` `return queList;` `}`

3	**Student can take a quiz by selecting their own choice of topics, open questions with a hint , with a choice of enabling/ disabling multiple choice questions.** `public void myStudent() {` `System.out.println("Welcome TO THE Quiz");` `System.out.println("Please Select TOPIC From Below Options");` `List<Quiz> quizList = daoData.getQuiz();` `for(Quiz qui : quizList) {` `System.out.println(qui.getId()+ " --"+qui.getTitle());` `}` `sc = new Scanner(System.in);` `opt = Integer.parseInt(sc.nextLine());` `List<Answer> ansList = daoData.selectQues(opt);` `List<Answer> ansOpenList = daoData.selectOpenQues(opt);` `System.out.println("Select the type of question \n1.Open Questions \n2.Multiple Questions :");` `sc = new Scanner(System.in);` `typeques = Integer.parseInt(sc.nextLine());` `if(typeques==2) {` `int count =0;` `for (Answer ans : ansList) {` `System.out.println("Question :"+ans.getQuestion().getContent());` `System.out.println("Opti-on A :"+ans.getOptA());` `System.out.println("Option B :"+ans.getOptB());` `System.out.println("Option C :"+ans.getOptC());` `System.out.println("Option D :"+ans.getOptD());` `System.out.println("Enter Your Choice: ");` `String crctAns = sc.next();` `if(crctAns.equalsIgnoreCase(ans.getText())) {` `count++;` `}` `}`
4	**Users can have his quiz based on their request by selecting open questions or multiple choice questions.** `public void myStudent() {` `System.out.println("Welcome TO THE Quiz");` `System.out.println("Please Select TOPIC From Below Options");` `List<Quiz> quizList = daoData.getQuiz();` `for(Quiz qui : quizList) {` `System.out.println(qui.getId()+ " --"+qui.getTitle());` `}` `sc = new Scanner(System.in);` `opt = Integer.parseInt(sc.nextLine());` `List<Answer> ansList = daoData.selectQues(opt);` `List<Answer> ansOpenList = daoData.selectOpenQues(opt);` `System.out.println("Select the type of question \n1.Open Questions \n2.Multiple Questions :");`

```
            sc = new Scanner(System.in);
            typeques = Integer.parseInt(sc.nextLine());
            if(typeques==2) {
            int count =0;
            for (Answer ans : ansList) {


System.out.println("Question :"+ans.getQuestion().getContent());
                  System.out.println("Opti-on A :"+ans.getOptA());
                  System.out.println("Option B :"+ans.getOptB());
                  System.out.println("Option C :"+ans.getOptC());
                  System.out.println("Option D :"+ans.getOptD());

                  System.out.println("Enter Your Choice: ");
                  String crctAns = sc.next();
                        if(crctAns.equalsIgnoreCase(ans.getText())) {
                              count++;
                        }
}
            exportFile(ansList, count);

            System.out.println("Your Score is :::"+count +"\n You can
check the score with correct answers in text file \n ");
            }
            else if (typeques==1) {
                  int count =0;
                  for (Answer ans : ansOpenList) {


System.out.println("Question :"+ans.getQuestion().getContent());
                        System.out.println("Hint :"+ans.getOptD());

                        System.out.println("Enter Your Answer: ");
                        String crctAns = sc.next();

      if(crctAns.equalsIgnoreCase(ans.getOptD())) {
                                    count++;
                              }
      }
                  exportFile(ansOpenList, count);

                  System.out.println("Your Score is :::"+count +"\n
You can check the score with correct answers in text file \n ");
            }

      }
```

5	**Users can export their quiz as a plain text.** `private void exportFile(List<Answer> ansList, int count) {` `final String FNAME = "Quiz_result.txt";` `try (BufferedWriter bw = new BufferedWriter(new FileWriter(FNAME))) {` `for (Answer ans : ansList) {` `bw.write("\nQuestion is :" +ans.getQuestion().getContent() + "\nCorrect Anser is :"+ans.getText());` `}` `bw.write("\nTotal Marks you socred in this Quiz is :"+count);` `bw.close();` `} catch (IOException e) {` `System.err.checkError();` `}` `}`
6	**Users can search questions by topics.** `public List<Question> searchQ(String tpc) {` `List<Question> queList = new ArrayList<Question>();` `Connection connection;` `try {` `connection = getConnection();` `PreparedStatement pstmt = connection.prepareStatement(SEARCH_QUES);` `pstmt.setString(1, tpc);` `ResultSet rs = pstmt.executeQuery();` `while (rs.next()) {` `String topic = rs.getString("QUESTION");` `Question ques = new Question(topic);` `queList.add(ques);` `}` `rs.close();` `} catch (SQLException e) {` `System.err.checkError();` `}` `return queList;` `}`

7

Users can create, read, update, and delete a question.

//Create Topic

```
public boolean createTopic(String quiz) throws CreateFailedException
{
            boolean var = false;
            try {
                  Connection connection = getConnection();
                  PreparedStatement pstmt =
connection.prepareStatement(INSERT_TOPIC);
                  pstmt.setString(1, quiz);
                  pstmt.execute();
                  var = true;
            } catch (SQLException sqle) {

            }
            return var;
      }
```

/Create Question

```
      public List<Quiz> getQuiz()  {
            List<Quiz> qList = new ArrayList<>();
            try {
                  Connection connection = getConnection();
                  PreparedStatement pstmt =
connection.prepareStatement(SELECT_QUIZ);
                  ResultSet rs = pstmt.executeQuery();
                  while (rs.next()) {
                        int id = rs.getInt("ID");
                        String topic = rs.getString("NAME");
                        Quiz quiz = new Quiz(topic);
                        quiz.setId(id);
                        qList.add(quiz);
                        }
                  rs.close();
            } catch (SQLException sqlEx) {
              SQLClientInfoException clientInfoEx = new
SQLClientInfoException();
              clientInfoEx.initCause(sqlEx);
          }
            return qList;
      }
```

//Read

```
      public List<Answer> selectQues(int opt) {
            Connection connection;
            List<Answer> ansList = new ArrayList<Answer>();
            try {
                  connection = getConnection();
                  PreparedStatement pstmt =
connection.prepareStatement(SELECT_QUESTN);
                  pstmt.setInt(1, opt);
                  pstmt.execute();
                  ResultSet rs = pstmt.executeQuery();
```

```
                  while (rs.next()) {
                        String ques = rs.getString(1);
                        Question questn = new Question(ques);
                        Answer ans = new Answer(rs.getString(2),
rs.getString(3), rs.getString(4), rs.getString(5),
                                    rs.getString(6));
                        ans.setQuestion(questn);
                        ansList.add(ans);
                        }
                  rs.close();

            } catch (SQLException e) {
                  System.err.checkError();
            }

            return ansList;
      }
```

//Delete

```
      public boolean delQuestion(HashMap<String, String> delMap) {
            boolean isExec = false;
            Connection connection;
            try {
                  connection = getConnection();
                  PreparedStatement pstmt =
connection.prepareStatement(DELETE_QUES);
                  pstmt.setString(1, delMap.get("tpc"));
                  pstmt.setString(2, delMap.get("question"));
                  pstmt.execute();

                  isExec = true;

            } catch (SQLException e) {
                  System.err.checkError();
            }
            return isExec;
      }
```

8 **Admin can use credentials saved from database take create a quiz**

```
private static boolean authenticate(Scanner
scanner) {
            boolean b=false;
            System.out.println("Please enter your login : \n");
            LGN = scanner.next();
            LGN=LGN.toUpperCase();
            System.out.println("Please enter your password : \n");
            PASS = scanner.next();
            String isExec = daoData.selectUser(LGN);
            if(isExec.equals(PASS)) {
                  System.out.println("Login Sucessfull \n");
                  b=true;
            }
            else {
                  System.out.println("Invalid Credentilas \n");
                  b=false;
```

	`` } return b; } ```
10	**User is able to run the application using Eclipse.**
	`` Welcome to the epita Quiz Please select the option 1.Admin 2.Student 1 You Are Admin Please enter your login : mahesh Please enter your password : M@hesh1992 Login Sucessfull You're authenticated Welcome to Admininstrator Console 1. Create Quiz 2. Create Question 3. Update Question 4. Delete Question 5. Search Question q. Quit What is your choice ? (1|2|q) : ```

4.2. Non-functional Requirements

3.2.1. The program shall be able to run on PC/Laptop with java..

3.2.2. The program shall be able to read a configuration property set from a file on the file system which will avoid hard coded parameters.

3.2.3. The data is stored in h2databases.

5. User Interface Design

Refer to the User Guide.

6. HardwareRequirements

#	**Hardware**	**Requirement**
1	Operating System	Compatible with Windows, Mac OS X, Linux
2	RAM	Minimum required 124MB
3	Disk Space	Minimum required 124MB
4	Processor	64-bit, four-core, 2.5 GHz minimum per core

7. Appendix

UML Class Diagram