QUIZ APPLICATION SYSTEM

HARD COPY

AddQues.java

```
import java.sql.*;
public class AddQues
{
   //Class.forName("com.mysql.cj.jdbc.Driver");
   public void creating(){
      Connection con;
      try{
         //Class.forName("com.mysql.cj.jdbc.Driver");
         con=DriverManager.getConnection("jdbc:mysql://localhost:3306/quiz1","root","dakshu");
         Statement stmt=con.createStatement();
         String sql = "CREATE TABLE quiz_questions (" +
                     "id INTEGER PRIMARY KEY," +
                     "question varchar(400) NOT NULL," +
                     "option1 varchar(100) NOT NULL," +
                     "option2 varchar(100) NOT NULL," +
                     "option3 varchar(100) NOT NULL," +
                     "option4 varchar(100) NOT NULL," +
                     "correct_option INTEGER NOT NULL)";
         stmt.executeUpdate(sql);
         stmt.close();
         con.close();
      }
      catch (SQLException e)
      {
         e.printStackTrace();
      }
   }
   public void inserting(){
      Connection con;
```

```
try{
   con=DriverManager.getConnection("jdbc:mysql://localhost:3306/quiz1","root","dakshu");
   Statement stmt=con.createStatement();
   String query1 = "INSERT INTO quiz_questions (id, question, option1, option2, option3, option4,
correct_option) VALUES " +
                   "(1, 'JDK stands for ____.', " +
                   "'Java development kit', " +
                   "'Java deployment kit', " +
                   "'JavaScript deployment kit', " +
                   "'None of these', 1), " +
                   "(2, 'JRE stands for ___.', " +
                   "'Java run ecosystem', " +
                   "'JDK runtime Environment', " +
                   "'Java Runtime Environment', " +
                   "'None of these', 3), " +
                   "(3, 'What makes the Java platform independent?', " +
                   "'Advanced programming language', " +
                   "'It uses bytecode for execution', " +
                   "'Class compilation', " +
                   "'All of these', 2), " +
                   "(4, 'Can we keep a different name for the java class name and java file name?', " +
                   "'Yes', " +
                   "'No', " +
                   "'Both A and B', " +
                   "'None of these', 1), " +
                   "(5, 'What are the types of memory allocated in memory in java?', " +
                   "'Heap memory', " +
                   "'Stack memory', " +
                   "'Both A and B', " +
                   "'None of these', 3)";
   stmt.executeUpdate(query1);
   stmt.close();
   con.close();
}
catch (SQLException e)
```

```
        {
            e.printStackTrace();
        }
    }
    public void result_Data(){
        Connection con;
        try{
            //Class.forName("com.mysql.cj.jdbc.Driver");
            con=DriverManager.getConnection("jdbc:mysql://localhost:3306/quiz1","root","dakshu");
            Statement stmt=con.createStatement();
            String sql = "CREATE TABLE quiz_results (" +
                        "id VARCHAR(20) PRIMARY KEY NOT NULL," +
                        "Name varchar(20) NOT NULL," +
                        "Score int NOT NULL)";
            stmt.executeUpdate(sql);
            stmt.close();
            con.close();
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        AddQues q=new AddQues();
        q.creating();
        q.inserting();
        q.result_Data();
    }
}
```

DatabaseConnection.java

```
import java.sql.*;
class DatabaseConnection {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/quiz1";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "dakshu";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
    }
}
```

QuizQuestion.java

```
import java.util.*;
class QuizQuestion {
    private int question_id;
    private String question_text;
    private List<String> options;
    private int correct_option;
    private volatile boolean attempted;
    public boolean isAttempted() {
        return attempted;
    }


    // Setter for attempted
    public void setAttempted(boolean attempted) {
        this.attempted = attempted;
    }

    // Constructor without questionId
```

```
public QuizQuestion(String question_text, List<String> options, int correct_option) {
    this.question_text = question_text;
    this.options = options;
    this.correct_option = correct_option;
}

// Constructor with questionId
public QuizQuestion(int question_id, String question_text, List<String> options, int correct_option) {
    this.question_id = question_id;
    this.question_text = question_text;
    this.options = options;
    this.correct_option = correct_option;
    this.attempted = false;
}

// Getter for questionId
public int getQuestionId() {
    return question_id;
}

// Setter for questionId
public void setQuestionId(int question_id) {
    this.question_id = question_id;
}

// Getter for questionText
public String getQuestionText() {
    return question_text;
}

// Setter for questionText
public void setQuestionText(String question_text) {
    this.question_text = question_text;
}
```

```
    // Getter for options
    public List<String> getOptions() {
        return options;
    }

    // Setter for options
    public void setOptions(List<String> options) {
        this.options = options;
    }

    // Getter for correctOption
    public int getCorrectOption() {
        return correct_option;
    }

    // Setter for correctOption
    public void setCorrectOption(int correct_option) {
        this.correct_option = correct_option;
    }
}
```

QuizResult.java

```
import java.sql.*;
class QuizResult {
    private String id;
    private String username;
    private int score;

    public QuizResult(String id, String username, int score) {
        this.id = id;
        this.username = username;
        this.score = score;
```

```
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public int getScore() {
        return score;
    }

    public void setScore(int score) {
        this.score = score;
    }
}
```

QuizResultDAO.java

```
class QuizResultDAO {
    public void saveQuizResult(QuizResult quizResult) {
        try{
            Connection conn = DatabaseConnection.getConnection();
```

```
        PreparedStatement stmt = conn.prepareStatement("INSERT INTO quiz_results (id,Name,Score)
VALUES (?,?,?)");
            stmt.setString(1, quizResult.getId());
            stmt.setString(2, quizResult.getUsername());
            stmt.setInt(3, quizResult.getScore());
            stmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public QuizResult getQuizResultById(String id)
    {
        try {
            Connection conn = DatabaseConnection.getConnection();
            PreparedStatement stmt = conn.prepareStatement("SELECT * FROM quiz_results WHERE id =
?");
            stmt.setString(1, id);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                String username = rs.getString("Name");
                int score = rs.getInt("Score");
                return new QuizResult(id, username, score);
            }
        }
        catch (SQLException e) {
            e.printStackTrace();
        }

        return null; // Return null if no previous result is found
    }
}
```

QuizDAO.java

```
import java.sql.*;
import java.util.*;
```

```
class QuizDAO {
    public List<QuizQuestion> getQuizQuestions() {
        List<QuizQuestion> quizQuestions = new ArrayList<>();

        try{
            Connection conn = DatabaseConnection.getConnection();
            Statement stmt = conn.createStatement();
            String query = "SELECT id, question,option1, option2, option3, option4, correct_option " +
                    "FROM quiz_questions";

            ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                int question_id = rs.getInt("id");
                String question_text = rs.getString("question");
                List<String> options = new ArrayList<>();
                options.add(rs.getString("option1"));
                options.add(rs.getString("option2"));
                options.add(rs.getString("option3"));
                options.add(rs.getString("option4"));
                int correct_option = rs.getInt("correct_option");

                QuizQuestion quizQuestion = new QuizQuestion(question_id, question_text, options,
correct_option);
                quizQuestions.add(quizQuestion);
            }
            Collections.shuffle(quizQuestions);
        }
        catch (SQLException e) {
            e.printStackTrace();
        }

        return quizQuestions;
    }
}
```

Student.java

```
import java.util.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

class Student
{
   public void startQuiz(Scanner scanner) {
      String password = "student";
      System.out.print("Enter Password: ");
      String inputPassword = scanner.next();

      if (password.equals(inputPassword)) {
         QuizDAO quizDAO = new QuizDAO();
         List<QuizQuestion> quizQuestions = quizDAO.getQuizQuestions();
         int timeLimitSeconds = 60; // 2 minutes

         System.out.print("Enter your ID: ");
         String id = scanner.next();

         System.out.print("Enter your username: ");
         String username = scanner.next();

         QuizResultDAO quizResultDAO = new QuizResultDAO(); // Update the variable name here
         QuizResult previousResult = quizResultDAO.getQuizResultById(id); // Update the variable name
here

         if (previousResult != null) {
            System.out.println("You have already attempted the quiz.");
            System.out.println("Your previous score: " + previousResult.getScore());
       // Exit the application since the user has already attempted the quiz.
         }
```

```
        int score = 0;
        for (QuizQuestion question : quizQuestions) {
            System.out.println(question.getQuestionText());
            List<String> options = question.getOptions();
            for (int i = 0; i < options.size(); i++) {
                System.out.println((i + 1) + ". " + options.get(i));
            }
            System.out.print("Enter your answer (1/2/3/4): ");
            int userAnswerIndex = scanner.nextInt() - 1;
            int correctAnswer = question.getCorrectOption();
            // Get the user's answer using the user's answer index
            if (userAnswerIndex >= 0 && userAnswerIndex < options.size()) {
                if (userAnswerIndex + 1 == correctAnswer) {
                    System.out.println("Correct!");
                    score++;
                    TimeLimitThread timeLimitThread = new TimeLimitThread(timeLimitSeconds, id,
username, score, quizQuestions, Thread.currentThread());
                    timeLimitThread.start();
                } else {
                    System.out.println("Wrong!");
                }
                question.setAttempted(true);
                System.out.println("");
            } else {
                System.out.println("Invalid answer choice. Please choose a valid option (1/2/3/4).");
            }
        }

        TimeLimitThread timeLimitThread = new TimeLimitThread(timeLimitSeconds, id, username,
score, quizQuestions, Thread.currentThread());
        timeLimitThread.start();
        QuizResult quizResult = new QuizResult(id, username, score);
        quizResultDAO.saveQuizResult(quizResult);
        System.out.printf("Your score is: %d/%d", score, quizQuestions.size());
        System.out.println("");
```

```
            System.out.println("Quiz Summary:");
            System.out.println("------------");
            for (QuizQuestion question : quizQuestions) {
                System.out.println(question.getQuestionText());
                System.out.println("Correct Option: " + question.getCorrectOption() + "\n");
            }
            createCertificate(id, username, score, quizQuestions.size());
            timeLimitThread.interrupt();
        } else {
            System.out.println("Invalid password. Access denied.");
        }
    }

    public void createCertificate(String studentId, String username, int score, int totalQuestions)
    {
        QuizDAO quizDAO = new QuizDAO();
        List<QuizQuestion> quizQuestions = quizDAO.getQuizQuestions();
        double percentile = (double) score / totalQuestions * 100;

        // Calculate grade based on percentile
        char grade;
        if (percentile >= 90) {
            grade = 'A';
        } else if (percentile >= 80) {
            grade = 'B';
        } else if (percentile >= 70) {
            grade = 'C';
        } else if (percentile >= 60) {
            grade = 'D';
        } else {
            grade = 'F';
        }
        StringBuilder certificateContent = new StringBuilder();

certificateContent.append("==================================================\n");
```

```
        certificateContent.append("\t\t\t  Congratulations!\n");

certificateContent.append("==================================================\n");
        certificateContent.append("Student ID: ").append(studentId).append("\n");
        certificateContent.append("Name: ").append(username).append("\n");
        certificateContent.append("Marks Scored:
").append(score).append("/").append(totalQuestions).append("\n");
        certificateContent.append(String.format("Percentile: %.2f%%\n", percentile));
        certificateContent.append("Grade: ").append(grade).append("\n");

certificateContent.append("==================================================\n\n");


        // Write the content to a text file
        String fileName = "QuizCertificate_" + studentId + ".txt";
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))) {
            writer.write(certificateContent.toString());
            System.out.println("Certificate generated and saved as: " + fileName);
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Teacher.java

```
import java.util.List;
import java.sql.*;


class Teacher {
    public void addQuizQuestion(QuizQuestion quizQuestion) {
        try {
            Connection conn = DatabaseConnection.getConnection();
            Statement stmt1 = conn.createStatement();


            String query = "SELECT COUNT(*) FROM quiz_questions";
            int questionCount = 0;
```

```
        ResultSet rs = stmt1.executeQuery(query);
        if (rs.next()) {
            questionCount = rs.getInt(1);
        }
        PreparedStatement stmt = conn.prepareStatement("INSERT INTO quiz_questions (id,question,
option1, option2, option3, option4, correct_option) VALUES (?,?, ?, ?, ?, ?, ?)");
        stmt.setInt(1,questionCount+1);
        stmt.setString(2, quizQuestion.getQuestionText());
        List<String> options = quizQuestion.getOptions();
        for (int i = 0; i < 4; i++) {
            stmt.setString(i + 3, options.get(i));
        }
        stmt.setInt(7, quizQuestion.getCorrectOption());

        stmt.executeUpdate();
        System.out.println("Quiz question added successfully!");
      }
      catch (SQLException e) {
        e.printStackTrace();
      }
    }
    public void updateQuizQuestion(int questionId, QuizQuestion updatedQuestion) {
      try {
        Connection conn = DatabaseConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement("UPDATE quiz_questions SET question=?,
option1=?, option2=?, option3=?, option4=?, correct_option=? WHERE id=?");

        stmt.setString(1, updatedQuestion.getQuestionText());
        List<String> options = updatedQuestion.getOptions();
        for (int i = 0; i < 4; i++) {
            stmt.setString(i + 2, options.get(i));
        }
        stmt.setInt(6, updatedQuestion.getCorrectOption());
        stmt.setInt(7, questionId);
```

```
            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Quiz question updated successfully!");
            } else {
                System.out.println("No quiz question found with ID: " + questionId);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void deleteQuizQuestion(int questionId) {
        try {
            Connection conn = DatabaseConnection.getConnection();
            PreparedStatement stmt = conn.prepareStatement("DELETE FROM quiz_questions WHERE
id=?");
            stmt.setInt(1, questionId);

            int rowsAffected = stmt.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Quiz question deleted successfully!");
            } else {
                System.out.println("No quiz question found with ID: " + questionId);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    public void getStudentMarksById(String studentId) {
        try {
            QuizDAO quizDAO = new QuizDAO();
            List<QuizQuestion> quizQuestions = quizDAO.getQuizQuestions();
            Connection conn = DatabaseConnection.getConnection();
            PreparedStatement stmt = conn.prepareStatement("SELECT Name, Score FROM quiz_results
WHERE id=?");
            stmt.setString(1, studentId);
```

```
            ResultSet rs = stmt.executeQuery();


            if (rs.next()) {
                String studentName = rs.getString("Name");
                int score = rs.getInt("Score");
                System.out.printf("Student ID: %s\n", studentId);
                System.out.printf("Student Name: %s\n", studentName);
                System.out.printf("Marks Scored: %d/%d\n", score, quizQuestions.size());
            } else {
                System.out.println("No quiz result found for ID: " + studentId);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }


    }
    public void printStudentsOrderByMarks() {
        try {
            Connection conn = DatabaseConnection.getConnection();
            PreparedStatement stmt = conn.prepareStatement("SELECT id, Name, Score FROM quiz_results
ORDER BY Score DESC");
            ResultSet rs = stmt.executeQuery();


            System.out.println("Students in order of highest marks:");
            System.out.println("-----------------------------------");
            while (rs.next()) {
                String studentId = rs.getString("id");
                String studentName = rs.getString("Name");
                int score = rs.getInt("Score");
                System.out.printf("Student ID: %s, Name: %s, Marks Scored: %d\n", studentId, studentName,
score);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
```

```
    }
}
```

TimeLimitThread.java

```
import java.util.*;
class TimeLimitThread extends Thread {
    private int timeLimitSeconds;
    private String studentId;
    private String username;
    private int score;
    private List<QuizQuestion> quizQuestions;
    private volatile Thread mainThread;

    public TimeLimitThread(int timeLimitSeconds, String studentId, String username, int score,
List<QuizQuestion> quizQuestions, Thread mainThread) {
        this.timeLimitSeconds = timeLimitSeconds;
        this.studentId = studentId;
        this.username = username;
        this.score = score;
        this.quizQuestions = quizQuestions;
        this.mainThread = mainThread;
    }

    @Override
    public void run() {
        try {
            Thread.sleep(timeLimitSeconds * 800);
            System.out.println("\nTime's up! Quiz completed.");
            mainThread.interrupt();
            printScoreAndExit();
        } catch (InterruptedException e) {

        }
    }
```

```
    private void printScoreAndExit() {
        Student student = new Student();
        student.createCertificate(studentId, username, score, quizQuestions.size());
        System.out.printf("Score:%d/%d",score,quizQuestions.size());
        QuizResultDAO quizResultDAO = new QuizResultDAO();
        QuizResult quizResult = new QuizResult(studentId, username, score);
        quizResultDAO.saveQuizResult(quizResult);
        System.exit(0);
    }
}
```

QuizApplication.java

```
import java.util.*;
public class QuizApplication {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        while(true){
            System.out.println("");
            System.out.println("");
            System.out.println("**************************");
            System.out.println("WELCOME TO QUIZ APPLICATION");
            System.out.println("**************************");
            System.out.println("");
            System.out.println("==================================");
            System.out.println("Are you a Teacher or a Student?");
            System.out.println("==================================");
            System.out.println("---------------------------------");
            System.out.println("1. Teacher");
            System.out.println("2. Student");
            System.out.println("3. Exit Quiz Application");
            System.out.println("---------------------------------");
            System.out.print("Enter your choice (1/2/3): ");
            int choice = scanner.nextInt();
```

```
        if (choice == 1) {
            System.out.println("Enter the password:");
            String password=scanner.next();
            Teacher teacher = new Teacher();
            if (password.equals("root"))
            {
              while(true)
              {
                System.out.println("");
                System.out.println("Choose an option:");
                System.out.println("1. Add a quiz question");
                System.out.println("2. Update a quiz question");
                System.out.println("3. Delete a quiz question");
                System.out.println("4. See student marks");
                System.out.println("5. All Students Progress");
                System.out.println("6.Exit Teacher Field");
                System.out.print("Enter your choice (1/2/3/4/5/6): ");
                int option = scanner.nextInt();
                switch (option)
                {
                  case 1:
                    scanner.nextLine(); // Consume the newline left by nextInt()
                    System.out.print("Enter the question text: ");
                    String questionText = scanner.nextLine();
                    List<String> options = new ArrayList<>();
                    for (int i = 1; i <= 4; i++) {
                      System.out.print("Enter option " + i + ": ");
                      String optionText = scanner.nextLine();
                      options.add(optionText);
                    }
                    System.out.print("Enter the correct option index (1/2/3/4): ");
                    int correctOption = scanner.nextInt();
                    QuizQuestion newQuestion = new QuizQuestion(questionText, options,
correctOption);
                    teacher.addQuizQuestion(newQuestion);
```

```
                    break;

                case 2:
                    System.out.print("Enter the question ID to update: ");
                    int questionIdToUpdate = scanner.nextInt();
                    scanner.nextLine(); // Consume the newline left by nextInt()
                    System.out.print("Enter the updated question text: ");
                    String updatedQuestionText = scanner.nextLine();
                    List<String> updatedOptions = new ArrayList<>();
                    for (int i = 1; i <= 4; i++)
                    {
                        System.out.print("Enter updated option " + i + ": ");
                        String updatedOptionText = scanner.nextLine();
                        updatedOptions.add(updatedOptionText);
                    }
                    System.out.print("Enter the updated correct option index (1/2/3/4): ");
                    int updatedCorrectOption = scanner.nextInt();
                    QuizQuestion updatedQuestion = new QuizQuestion(questionIdToUpdate,
updatedQuestionText, updatedOptions, updatedCorrectOption);
                    teacher.updateQuizQuestion(questionIdToUpdate,updatedQuestion);
                    break;

                case 3:
                    System.out.print("Enter the question ID to delete: ");
                    int questionIdToDelete = scanner.nextInt();
                    teacher.deleteQuizQuestion(questionIdToDelete);
                    break;

                case 4:
                    System.out.print("Enter the student's ID to see marks: ");
                    String studentId = scanner.next();
                    teacher.getStudentMarksById(studentId);
                    break;
                case 5:
                    System.out.println("Displaying all Students marks who attempted");
```

```
                    teacher.printStudentsOrderByMarks();
                    break;
                case 6:
                    System.out.println("Exiting... Goodbye!");
                    break;

                default:
                    System.out.println("Invalid option. Please enter a valid choice (1/2/3/4).");
                    break;
            }
            if (option == 6) {
                break; // Exit the teacher menu loop and go back to the main loop
            }
        }
    }
    else {
        System.out.println("Invalid password. Access denied.");
    }
}
else if (choice == 2)
{
    Student student = new Student();
    student.startQuiz(scanner);

}
else if(choice ==3){
    System.out.println("Exiting");
    System.exit(0);
}
else{
    System.out.println("Invalid choice!!");
}

}
```

```
    }
}
```