Hands-On for Java SE using JDBC

Quiz Management System

Aim

- To design and develop a **Quiz Management System** using Java SE.
- To implement a **console-based application** and a **Swing-based GUI application** integrated with MySQL using JDBC.
- To allow **Admins** to manage quizzes and **Students** to attempt quizzes with a restriction of **one attempt per quiz** and view their scores.

Technologies used

- Java Programming:
  - Classes & Objects, Encapsulation, Inheritance & Polymorphism, Composition / Association, Interfaces & Abstraction
  - Exception Handling (Custom Exceptions)
  - Java Collections Framework, File Handling (Text File Parsing), Serialization / Deserialization, JDBC (MySQL Connectivity), Swing GUI Components.

Duration

- Console Based Implementation: 30 Hours
- GUI & Integration with classes: 8 Hours

Description

- A training institute wants to create a **Quiz Management System** for conducting online tests for students.
  - The system should support two types of users:
    - **Admin** is responsible for managing quizzes and overseeing student performance..
    - **Students** can register, login, take quizzes, and view their scores.
- System Functional Requirements
  - ADMIN FUNCTIONALITY:

    Admin should be able to:
    - Login using admin credentials
    - Create a new quiz by providing:
      - Quiz Title
      - Importing questions from a .txt file
    - View all available quizzes.
    - View results of any quiz (student-wise score report).
    - Delete quizzes (including related questions and attempts).
    - Logout
  - STUDENT FUNCTIONALITY:

    Student should be able to:
    - Register with name, email, and password.
    - Login using credentials.
    - View all quizzes.
    - Attempt quiz (only once per quiz).
    - View their previous scores.

- Logout

Note- Once a student attempts a quiz, the same quiz should NOT appear again in their available quiz list.

Console Application Flow

When the application starts:

```
=== Quiz App Console ===
1) Admin Login
2) Student Register
3) Student Login
4) Exit
```

```
Admin Menu:
1) Create Quiz
2) List Quizzes
3) View Results
4) Delete Quiz
5) Logout
```

```
Student Menu:
1) View Quizzes
2) Take Quiz
3) View Scores
4) Logout
```

Questions must be provided in a text file format as:

```
Which company created Java?
A) Sun Microsystems
B) Microsoft
C) Apple
D) Google
ANSWER: A
```

The system should parse this file and insert questions into the database.

Database Design:

Tables Used:

- users
- quizzes
- questions
- quiz_attempts

Each quiz is linked to questions, and each student attempt is stored with score and timestamp.

Note :

- Ensure that none of the classes (except Main / UI classes) contain code related to console input/output.
  This will make the system more portable to GUI or Web-based applications.
- Quiz Management System should maintain list of quizzes as:
  Set (java.util.HashSet<Quiz>) to avoid duplicate quizzes.

- Quiz Management System should maintain list of quiz attempts as: List (java.util.LinkedList<Attempt>) so that attempts are stored in chronological order.
- Each Quiz should contain list of Questions as: List (java.util.LinkedList<Question>)
- Each Student should maintain record of attempted quizzes using: Map (java.util.HashMap<Integer, Attempt>)
  - Key: Quiz ID
  - Value: Attempt object
    (This ensures one quiz can be attempted only once.)
- The Admin should manage all quizzes using: Collection framework + DAO Pattern for better flexibility and database interaction.
- Use Serialization and Deserialization to optionally store and restore:
  - Quiz metadata
  - Student session data
  - Offline backup of quizzes
- Implement equals() and hashCode() in the following classes for proper searching and comparison:
  - User
  - Quiz
  - Question
  - Attempt
- Design and utilize appropriate Custom Exception classes, such as:
  - QuizAlreadyAttemptedException
  - QuizNotFoundException
  - InvalidUserException
  - NoQuestionsFoundException
  - AuthenticationFailedException
- Implement Layered Architecture:
  - Model Layer (POJO classes)
  - DAO Layer (Database Operations)
  - Utility Layer (DBUtil, Validation, File Parser)
  - UI Layer (Console / Swing)
- Use JDBC with PreparedStatement for:
  - Secure database access
  - Prevention of SQL Injection
  - Efficient query execution
- Design GUI application which allows:
  - Admin to manage quizzes dynamically
  - Students to attempt quizzes without conflicts
  - Simultaneous handling of multiple users
- The Swing GUI should follow:
  - MVC principle
  - Event-driven programming
  - Real-time quiz filtering (hide already attempted quizzes)
- Implement proper validation for:
  - Email format
  - Password strength
  - File format
  - Quiz availability