5CS019 Object Oriented Design and Programming – Portfolio Report

Java Swing Quiz Application with MySQL (CompetitionDB)

Student Name: Shreejak Subedi

Student Number: 2509984

Module: 5CS019 Object Oriented Design and Programming

Module Leader: Dr Adeel Rafiq

Semester / Year: Semester 1, 2024–25

IDE: Eclipse

Java Version: JavaSE-23

Build Tool: Maven

Database: MySQL (XAMPP)

UI Technology: Java Swing

Contents

1. APPLICATION OVERVIEW

This portfolio comprises a Java Swing Quiz Application, written to take the coursework requirements of dealing with competitors in a competition system. A Competitor in my implementation is a quiz player. Every competitor contains a set of personal information (with the help of a specific Name class), a level of competition, and five scores in an array. The system stores competitor data in MySQL database (CompetitionDB) through JDBC, retrieves it on demand and creates reports which present performance and statistics. Correlation of the coursework brief with the Quiz Application.

Mapping the coursework brief to the Quiz Application

- Competitor → Quiz player
- Competition Level → Difficulty level
  (BEGINNER, INTERMEDIATE, ADVANCED)
- Score1..Score5 → Results for 5 quiz questions (0 = incorrect, 1 = correct)
- Overall score → Calculated from the score array using a level-based rule
- Report → Scoreboard table + top performer + summary statistics

What the user can do (GUI features)

1. Register
    - Create competitor profile (ID, name, level, age)
    - Create login account (username + password)
    - Saves to database (Competitors + Users)
2. Login
    - Authenticates user and opens the main menu
3. Start Quiz (5 questions)
    - Question set is selected based on competitor level
    - Stores 5 scores (0/1) and updates score1..score5 in MySQL
4. Scoreboard / Report

- Displays a table of competitors with ID, Name, Level, and Score1..Score5
- Identifies the top performer
- Shows summary statistics including frequency of scores (0 and 1)

5. Search Competitor by ID
    - If valid, displays short details using getShortDetails()
    - Optional: also displays full details using getFullDetails()

How the design demonstrates OOP (briefly)

- Encapsulation & data hiding: private fields with getters/setters
- Classes & objects: Competitor, Name, Question, etc.
- Collections & arrays: int[5] scores + ArrayList<Competitor> in CompetitorList
- Separation of concerns: UI layer, service layer, DAO layer, model layer

2. DECISIONS

2.1 Extra competitor attribute

Selected additional characteristic: age (int).

Rationale: Age is realistic, valid, and well exemplifies the extension of the basic model with the minimum required attributes.

2.2 Score design

- Every quiz attempt generates 5 scores which is precisely saved in an int[5].

Each score is 0 or 1, which is suitable to the context of the quiz and ease frequency statistics.

2.3 Computation of overall score (level-based)

The coursework is that which demands total points calculated on the basis of scores and level. A level-based rule that I used was:

- BEGINNER: best 3 scores out of 5
- INTERmediate: 4 maximum out of 5.
- ADVANCED: best 5 scores out of 5

Then normalize the result to 05 scale:

Average of top N scores yields the value between 0 and 1.

- Overall Score = averageTopN × 5

This is simple to describe, involves the use of arrays and uses of computations and a significant difference in levels.

2.4 Level-based questions

The level of the competitors is also entered into the database and is utilized when launching the quiz to enable the difficulty to actually affect the gameplay rather than being a title.

2.5 Password handling decision

The passwords are not kept as plain text. They are hashed in a SHA 256 (PasswordUtil.sha256()) and stored in the database.

3. DATABASE DESIGN (MYSQL / XAMPP)

3.1 Tools and setup

- Database server: MySQL via XAMPP
- Database name: CompetitionDB
- Connection: JDBC (MySQL connector dependency managed by Maven)

3.2 Tables

Table: Competitors

Stores competitor details and five scores:

- competitorId (INT, Primary Key)
- firstName (VARCHAR)
- middleName (VARCHAR, nullable)
- lastName (VARCHAR)
- level (VARCHAR) — stored as BEGINNER, INTERMEDIATE, ADVANCED
- age (INT)
- score1..score5 (INT) — values 0 or 1

Table: Users

Stores login credentials and links them to a competitor:

- userId (INT, Primary Key, Auto Increment)
- username (VARCHAR, Unique)
- passwordHash (VARCHAR)
- competitorId (INT, Foreign Key → Competitors.competitorId)

3.3 CRUD operations implemented

- Create: registration inserts into Competitors and Users
- Read: retrieve competitor by ID (search feature and quiz initialisation)
- Read: retrieve all competitors (report generation)

- Update: update score1..score5 after quiz completion

4. STATUS REPORT

Status: Complete

The system addresses the coursework requirements and consists of:

- Java Swing GUI (no console UI)
- OOP design required classes and methods.
- Tables and computations (array of scores, total score, totals, frequencies)
- JDBC CRUD storage in MySQL database.
- Manager + CompetitorList report and statistics.
- Output of the report: table of scoreboard and top performer and summary statistics.
- User interaction: competitive search by ID and display brief information.
- Bad input and DB error processing.
- CompetitorList and generated HTML javadoc.
- UML class diagram included

5. KNOWN BUGS AND LIMITATIONS

1. Quiz questions are not stored in the database but coded.
2. The scores are binary (0/1), and therefore with frequency statistics, there exist counts of 0 and 1 only.
3. Password hashing takes the hash algorithm of SHA 256 without salt (security vulnerability).
4. No gui screen to edit competitor information once registered.
5. It uses the MySQL (XAMPP) application; it does not have offline.

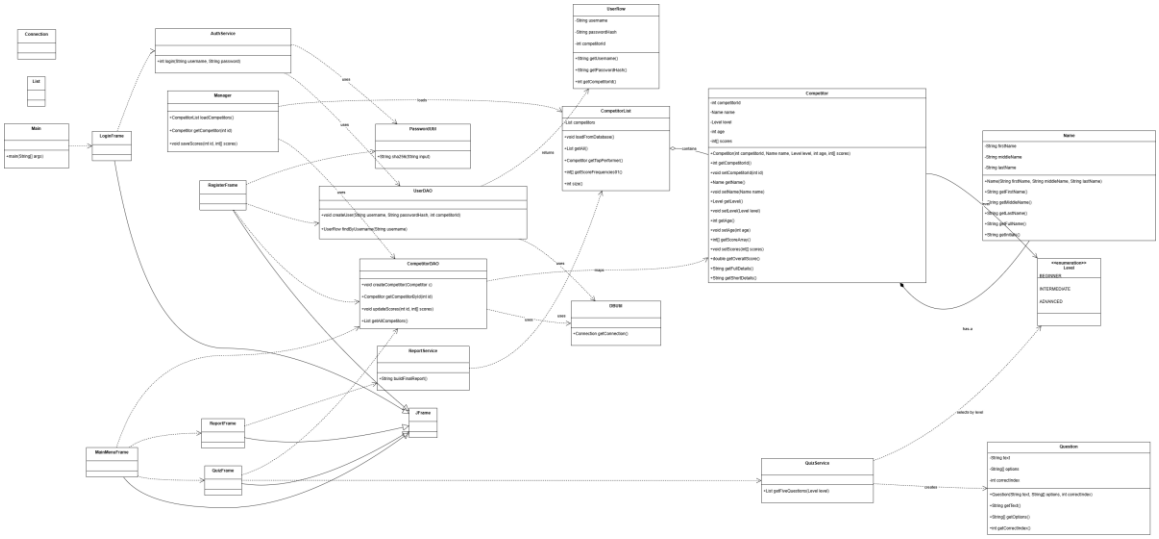
6.TESTS PERFORMED

Test ID	Feature	Steps / Input (exact)	Expected Result	Actual Result (example you can write)	Pass/Fail
T1	Register	Open Register → enter Account A details (ID 501, Shreejak Subedi, BEGINNER, 21, shreejak501, Pass@501) → Create Account	Account created; competitor row + user row inserted in DB	Message shown: "Account created! You can login now." DB rows exist for 501 and shreejak501	Pass
T2	Register duplicate username	Register again using **different competitor ID** (e.g., 503) but **same username** shreejak501	Registration fails; no duplicate username inserted	Message shown similar to: "Register error: Duplicate entry 'shreejak501' for key 'username'"	Pass
T3	Login success	Login with: username shreejak501 password Pass@501	Main Menu opens	Main Menu screen opens successfully	Pass
T4	Login failure	Login with username shreejak501 and password WrongPass	"Invalid login" shown and stay on login screen	Message: "Invalid login."	Pass
T5	Level-based quiz	Login as Account A (BEGINNER) → Start Quiz. Logout. Login as Account B (ADVANCED) → Start Quiz	Different question sets appear / header shows different level	BEGINNER quiz shows beginner questions; ADVANCED quiz shows advanced questions	Pass
T6	Save scores	Login as Account A → Start Quiz → answer all 5 correctly → Finish	DB score1..score5 updated to 1s for competitor 501	Message: "Quiz complete… Saved to database." DB shows score1..score5 = 1,1,1,1,1	Pass
T7	Search by ID valid	Main Menu → Find Competitor (Short Details) → enter 501	Short details displayed	Example: "CN 501 (SS) has overall score 5."	Pass
T8	Search by ID invalid	Main Menu → Find Competitor (Short Details) → enter abc (or empty)	Error message shown	"Competitor ID must be a number." / "cannot be empty."	Pass
T9	Report loads	Main Menu → View Report	Table loads with competitors + summary shown	JTable shows IDs 501/502 and their scores; summary shows total + top performer + frequency	Pass

Test ID	Feature	Steps / Input (exact)	Expected Result	Actual Result (example you can write)	Pass/Fail
T10	DB offline	Stop XAMPP MySQL → open app → View Report (or Register/Login)	Application shows DB connection error	Message like: “Report error: Communications link failure” / “Error: …”	Pass

7. DIAGRAMS

A UML class diagram (draw.io) is provided, showing:



Classes included

- Model: Competitor, Name, Level, Question, CompetitorList
- DAO: CompetitorDAO, DBUtil, UserDAO.
- Service: AuthService, QuizService, Reportservice, Manager.
- UI: UI: LoginFrame, RegisterFrame, MainMenuFrame, QuizFrame, ReportFrame.
- App: Main

Key relationships shown

A Competitor possesses a Name and a Level.

- Competitor has an array of scores[5].
- CompetitorList has several Competitor objects in it.
- UI classes are based on service/DAO classes.
- UI frames extend JFrame

8.CONCLUSION

The coursework requirements are fulfilled by this Quiz Application because it integrates OOP design, array-based scoring, interconnection with MySQL database using JDBC CRUD and a fully-developed Swing-based user interface. MySQL is used to store competitor results and retrieve them, quizzes are level-based and reporting results in a clear scoreboard with top performer and summary statistics. The project is structured in rational Maven packages and contains testing, UML class diagram, and Javadoc documentation necessary to be submitted.