Academic Year	Module	Assessment Number	Assessment Type
2024/27	5CS019/HJ1: OBJECT-ORIENTED DESIGN AND PROGRAMMING	01	Report

Report on Quiz App

Student Id : 2438463

Student Name : Prashansa Hamal
Section : L4CG22

Module Leader : Mr. Subash Bista
Tutor : Mr. Nischal Shakya

Submitted on : 12-02-025

1. Introduction

Java Quiz Application is a complex, interactive application aimed at testing and improving the knowledge of students at three skill levels: Beginner, Intermediate, and Advanced. The application is GUI-based, thoroughly scripted using JFrame components, and has the advantage of utilizing MySQL for secure data storage and effective data management. Its major features are user login, dynamic quiz generation based on difficulty levels, and detailed reporting, providing a simple and interactive user interface. Adherence to object-oriented programming (OOP) principles and Model-View-Controller (MVC) design provides modular organization, data abstraction, and encapsulation, resulting in maintainable, scalable, and reusable code.

1.1.Objectives:

Develop a successful quiz system with support for user participation and result viewing.

Implement a user authentication system to differentiate between the student and administrator roles.

Provide an Admin Dashboard for complete quiz question management and viewing of student reports.

Ensure system dependability with extensive testing through JUnit.

1.2. Expected Outcomes:

A fully functional, user-friendly, and dependable quiz application with mentioned requirements met.

Optimized data management practices for seamless execution and scalability.

Application of software engineering principles for a robust and maintainable system in practice.

A demonstration of Java programming and database management skills.

2. Methodology

The application adheres to the Model-View-Controller (MVC) architectural pattern, enabling modularity, testability, and maintainability.

2.1.Software Development Lifecycle:

A iterative methodology, with aspects of Agile and Waterfall methodologies, was used.

Requirements Gathering: Project specifications definition, features, and technical requirements.

Design: Architecture design, database schema, and GUI components design.

Implementation: Backend logic development, database interactions, and GUI components development.

Testing: Unit tests implementation using JUnit and manual testing.

Deployment: Getting the application ready for local deployment and usage.

2. 2.Tools and Processes:

Programming Language: Java

Database: MySQL

Database Connectivity: JDBC

GUI Framework: Java Swing (JFrame)

Testing Framework: JUnit

IDE: Eclipse

3. Implementation

3.1. Competitor Class Development (Model Classes)

The "Competitor" was represented by the following Model Classes:

Player: Represents a user with attributes player_id, username, password, name, and level.

Question: Represents a quiz question with attributes question_id, question, options, correct_answer, and difficulty.

Report: Represents a quiz report with attributes report_id, player_id, correct_answers, wrong_answers, score, and difficulty.

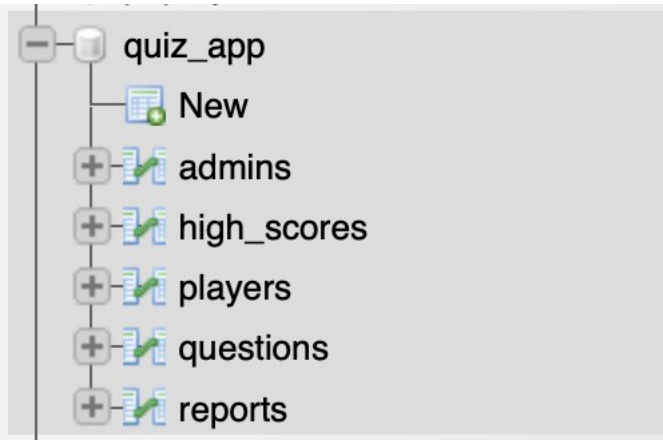
These classes encapsulate the data and the behavior that is associated with it, enabling easy objectoriented modeling. The methods in such classes contain getters and setters to retrieve and alter attributes, as well as any custom logic specific to an entity, e.g., calculation of scores or generation of reports.

3.2 MySQL and Arrays

The MySQL system is used for data storage in the application, and arrays (or, more specifically, lists) are utilized in the Java code to keep collections of questions, options, and results.

Score Handling: Every question can have a score. There are no explicit arrays of scores, but the application does compute a total score as a sum of the number of correct answers.

Database Schema Details:



players:

Extra options

player_id	username	password	name	level	score
1	Praise@gmail.com	#123	Praise Hamal	Beginner	0
2	Prashansa12@gmail.com	#111	Prashansa Hamal	Intermediate	0
3	nischal14@gmail.com	3000	Nischal Shakya	Advanced	0
4	Test1739293315902@gmail.com	#123	Prashansa	Intermediate	0
5	Test1739293316085@gmail.com	#123	Prashansa	Intermediate	0
6	Test1739293316095@gmail.com	#123	Prashansa	Intermediate	0
7	Test1739293316125@gmail.com	#123	Prashansa	Intermediate	0
8	Test1739293316132@gmail.com	#123	Prashansa	Intermediate	0
9	Test1739293750875@gmail.com	#123	Prashansa	Intermediate	0

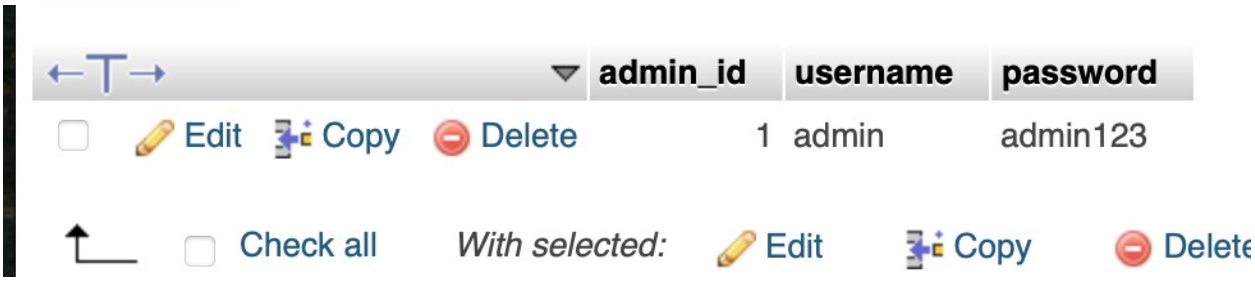
questions:

question_id	question	option1	option2	option3	option4	correct_answer	difficulty
1	Which OOP principle restricts direct access to obj...	Encapsulation	Inheritance	Polymorphism	Abstraction	1	Beginner
2	Which of the following is NOT an OOP principle?	Abstraction	Inheritance	Compilation	Polymorphism	1	Beginner
3	Which feature allows a class to acquire properties...	Abstraction	Inheritance	Encapsulation	Overloading	2	Beginner
4	What does the Single Responsibility Principle (SRP...	A class should handle all tasks	A class should have only one reason to change	One function should do multiple tasks	A class must have a single instance	2	Intermediate
5	What is method overriding?	Changing the method's return type	Defining a method in a subclass with the same sign...	Writing multiple methods with the same name but di...	Hiding a parent class method	2	Intermediate
6	Which design principle is followed	Liskov Substitution Principle	Dependency Inversion Principle	Open/Closed Principle	Encapsulation	3	Intermediate

reports:

report_id	player_id	correct_answers	score	difficulty
1	1	6	60	Intermediate
2	1	6	60	Intermediate
3	1	6	60	Intermediate
4	1	6	60	Intermediate
5	1	6	60	Intermediate
6	1	6	60	Intermediate
7	1	1	10	Beginner
8	1	1	10	Beginner
9	1	1	10	Beginner
10	1	2	20	Beginner

admins:



JDBC Connection: The JDBConnection class creates a connection to the MySQL database via JDBC.

3.3. MySQL Integration and Reports

ReportController and ReportDAO facilitate report creation and retrieval.

ReportController communicates with ReportDAO to save player reports to the database and fetch them for viewing.

HighScoreController and HighScoreDAO provide retrieval and viewing of high scores, divided by difficulty level.

PlayerReportFrame displays the reports in an organized manner, showing important values like player name, score, and difficulty level.

3.4. Error Handling

The program has minimal error handling through try-catch blocks for catching exceptions that may be thrown when working with the database.

Invalid Input: Partially implemented; needs to be supplemented with user input validation.

MySQL Connection Problems: The DBConnection class deals with connection problems using a try-catch block, but it needs to be supplemented with better logging and retrying.

3.5. Testing:

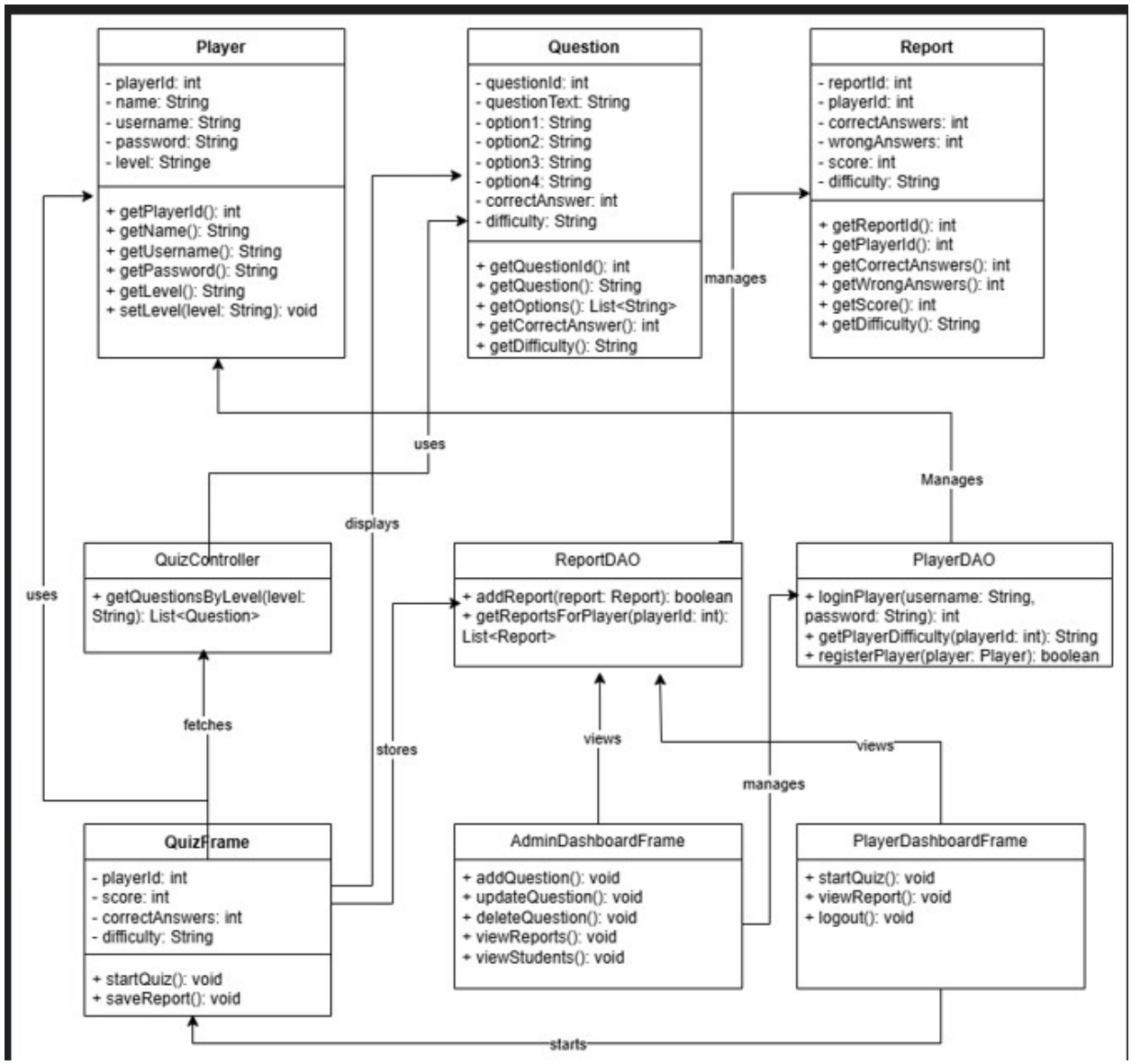
To ensure system reliability, JUnit tests were conducted on key functionalities. The following test cases were executed:

Player Login: Ensuring valid users can authenticate successfully.
Question Retrieval: Verifying that questions are fetched based on diAiculty levels.

Quiz Completion: Ensuring correct score calculation and storage.
Report Generation: Validating that quiz results are correctly recorded in the database.

```
<terminated> TestRunner [Java Application]
All tests passed: true
```

3.6. Class Diagram:



The UML class diagram for a quiz application represents the key components and their relationships. At the core, there are several classes: `Player` stores the player's details and level, `Question` holds individual quiz questions and answers, and `Report` tracks the results of a player's quiz attempt. The `QuizController` manages the logic for selecting questions based on the player's level, while `QuizFrame` represents the interface where the player interacts with the quiz. The `AdminDashboardFrame` allows administrators to manage quiz questions and view reports, and the `PlayerDashboardFrame` provides players with access to the quiz and their results. The `ReportDAO` and `PlayerDAO` are responsible for saving and retrieving player data and quiz results. The diagram also shows the relationships between these classes: for example, `QuizController` uses `Question` to get questions for the quiz, `QuizFrame` fetches `QuizController` to present the quiz, and `AdminDashboardFrame` views both `ReportDAO` and `PlayerDAO` to manage and access data. This structure forms the blueprint for how the quiz application is organized, ensuring the different parts of the app work together seamlessly.

3.7. Test Cases:

TestCaseID-Description-InputData-ExpectedOutput-Result

Test Case ID	Description	Input Data	Expected Output	Result
TC_AdminLogin	Test successful admin login	username = "admin", password = "password"	Login successful (returns true)	PASS
TC_AddQuestion	Test adding a new question	Question object	Question added successfully (returns true)	PASS
TC_GetHighScore	Test retrieving high scores for "Beginner" level	level = "Beginner"	List of high scores sorted by score	PASS
TC_PlayerRegister	Test registering a new player	Player object	Player registered successfully (returns true)	PASS
TC_PlayerLogin	Test successful player login	username = "testuser", password = "testpass"	Returns player_id > 0	PASS
TC_InvalidLogin	Test login with incorrect credentials	username = "invalid", password = "wrong"	Returns -1	PASS

3.8. Status Report

The application is completed and structured for the most part, with core features implemented and unit tests completed. There are some limitations with regards to exception handling that must be resolved.

9. Conclusion

Java Quiz Application is a good start to a stable educational application. It has basic features, follows MVC architecture, and has unit tests for verification. Nevertheless, tackling the observed shortcomings, specifically on security, functionality, and exception handling, is essential in making the overall quality and dependability of the application better. Future improvements include the implementation of password hashing, completion of statistical reports, enhancement of error handling using custom exceptions and logging, and the addition of comprehensive input validation. The utilization of Javadoc comments would greatly enhance maintainability.. By making these improvements, the quiz application can be a more secure, feature-rich, and reliable testing and knowledge augmentation platform for students