University of Wolverhampton

5CS019 Object Oriented Design and Programming

Quiz App Report

Name: Piyush Rauniyar
Group: L5CG2
Student ID: 2505494
Tutor: Mr. Sandip Tiwari

Table of Contents

List of Figure

1.Decisions:

In the design of the "Competitor" class, the addition of the attribute "Country" for each competitor has been considered. This allows the competitors to be distinguished based on their countries. This might be used to add features in the future that allow the data to be shown in a leaderboard. This attribute is a "String" type field in the database. It also gets shown in the detail view of the competitor's information and the aggregated managers' view.

The algorithm used was developed to minimize major fluctuations in any given competitor's performance score. The getOverallScore method is passed an array containing integer scores from events. The highest and lowest points are removed from this selection, and the score is determined by computing the arithmetic mean among these remaining three score points. This system, which was also used in sporting contests, was aimed at preventing extreme scores from negatively affecting performance, irrespective of what brought about these extreme results.

2. Status Report

It can also be noted that the application developed is completely functional and strictly adheres to the specifications provided in the assignment brief. The application also establishes a good connection to the MySQL database, known as the CompetitionDB database. This ensures easy storage, retrieval, and updation of the competitors. Moreover, it can also be noted that the Graphical User Interface helps in interacting with the application. Registration, authentication, and a quiz session are possible at different difficulty levels.

Additionally, the management aspect of the program is effective in creating the necessary reports. The class correctly retrieves the needed information from the database to generate a nicely formatted table of all the competitors. It also correctly uses algorithms to identify the best-performing person based on the calculated overall score. Finally, it successfully creates a statistical report on frequency. All the unique features of the assignment have successfully been achieved.

3. Known Bugs and Limitations:

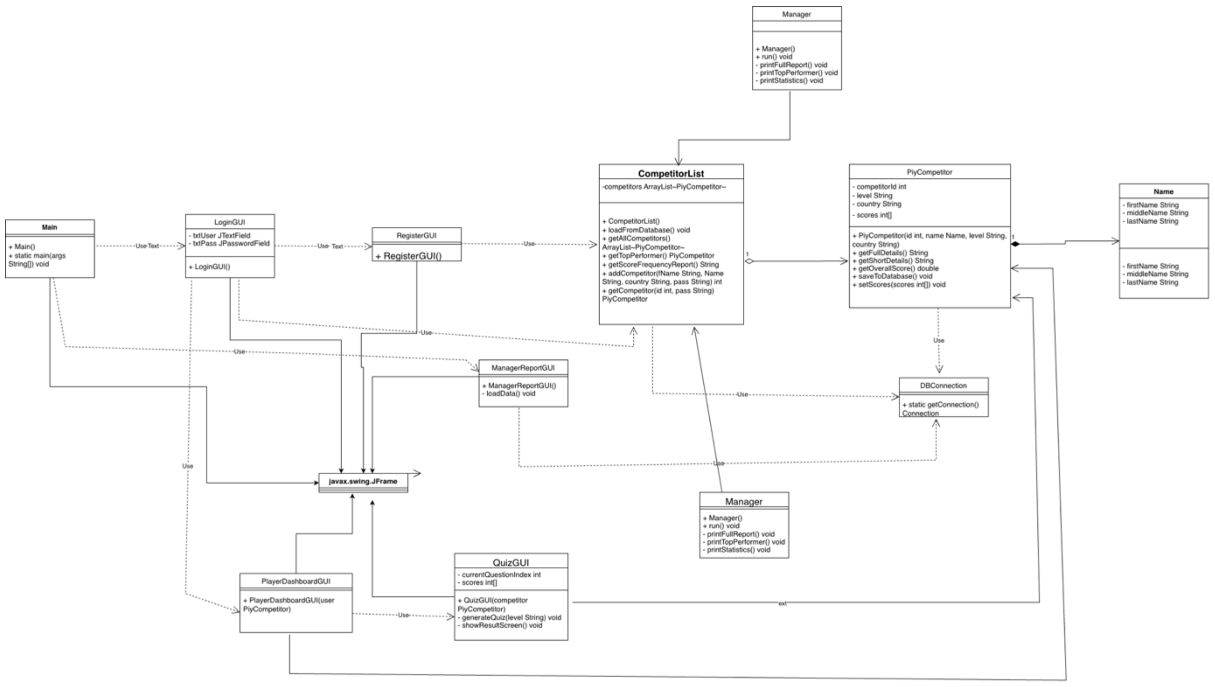
The application has undergone extensive testing and, at present, has no major known defects. The system has very good error handling: it handles invalid users' input very well, never crashing the system. For example, the authentication module correctly rejects invalid credentials, and data entry forms make use of exception handling to block non-numeric input in integer fields. As a result, the core functionality around database connectivity and user session management is stable under standard operating conditions.

Despite this stability, the existing architecture has some drawbacks, the most significant of which is the static construction of the assessment module, wherein the quiz questions are hardwired into the QuizGUI class. Such an architecture mandates changes in source code in order to make changes in the content; this can be improved in future versions with the inclusion of a separate database table for question management, enabling dynamic question management. Another weakness is that the database supports all the CRUD operations, while 'Delete' functionality is yet to be exposed from the interface of the Manager; this would be a logical extension of the system.

4. Tests:

Test Class	Test Method	Description	Input Data	Status
CompetitorTest	testGetOverallScore	Validates the core scoring algorithm (drop min/max, average remaining)	Scores: [4, 3, 5, 2, 4]	Pass
CompetitorTest	testGetOverallScoreAllSame	Tests edge case where all scores are identical.	Scores: [5, 5, 5, 5, 5]	Pass
CompetitorTest	testGetFullDetails	Checks if the full details string contains all required attributes.	User: ID 200, Name "Alice Green"	Pass
CompetitorTest	testGetShortDetails	Checks compliance with the specific short format requirement.	User: ID 200, Name "Alice Green"	Pass
CompetitorTest	testGetScoreArray	Verifies that the score array getter returns correct data.	Input Array: [4, 3, 5, 2, 4]	Pass
DBTest	testConnection	Verifies connectivity to the MySQL database.	Call DBConnection.getConnection()	Pass
DBTest	testRegisterAndLogin	Tests the full workflow of creating a user and then retrieving them.	Register unique user → Login with password	Pass
DBTest	testScoreUpdate	Tests data persistence (CRUD Update).	Set Scores to [5,5,5,5,5] Save Reload	Pass

5. Diagrams:



Manager
+ Manager()
+ run() void
- printFullReport() void
- printTopPerformer() void
- printStatistics() void
CompetitorList
-competitors ArrayList~PiyCompetitor~
+ CompetitorList()
+ loadFromDatabase() void
+ getAllCompetitors() ArrayList~PiyCompetitor~
+ getTopPerformer() PiyCompetitor
+ getScoreFrequencyReport() String
+ addCompetitor(fName String, Name String, country String, pass String) int
+ getCompetitor(id int, pass String) PiyCompetitor
PiyCompetitor
- competitorId int
- level String
- country String
- scores int[]
+ PiyCompetitor(id int, name Name, level String, country String)
+ getFullDetails() String
+ getShortDetails() String
+ getOverallScore() double
+ saveToDatabase() void
+ setScores(scores int[]) void
Name
- firstName String
- middleName String
- lastName String
- firstName String
- middleName String
- lastName String
Main
+ Main()
+ static main(args String[]) void
LoginGUI
- txtUser JTextField
- txtPass JPasswordField
+ LoginGUI()
RegisterGUI
+ RegisterGUI()
ManagerReportGUI
+ ManagerReportGUI()
- loadData() void
DBConnection
+ static getConnection() Connection
javax.swing.JFrame
Manager
+ Manager()
+ run() void
- printFullReport() void
- printTopPerformer() void
- printStatistics() void
PlayerDashboardGUI
+ PlayerDashboardGUI(user PiyCompetitor)
QuizGUI
- currentQuestionIndex int
- scores int[]
+ QuizGUI(competitor PiyCompetitor)
- generateQuiz(level String) void
- showResultScreen() void
Use Text
Use Text
Use
Use
Use
Use
Use
Use
Use
Use
ext
1
1

Figure 1 class diagram of Quiz app

6. Javadoc Comments:

The CompetitorList class is fully documented using standards set forth by Javadoc. The HTML documentation generated is provided in the attached zip file . This will explain what each method and its respective parameters or return types do, which will give future developers an explicit understanding of the code.