***Part – 01.***

**Attributes.**

The following extra attributes were chosen for the KarateCompetitor class to store the competitor's details

private int competitorID;

private Name competitorName;

private String level;

private String country;

Extra Attribute,

**country**: A string that holds the country of the competitor, helping to identify the competitor's nationality.

These attributes help define the identity and rank of each competitor, essential for the management of the competition

**Overall Method Explenation.**

Currently, the **getOverallScore()** method is implemented as a placeholder that returns a fixed value of 5 for all competitors. This method is designed to be expanded in Part 2, where the overall score will be dynamically calculated based on scores from 5 different rounds.

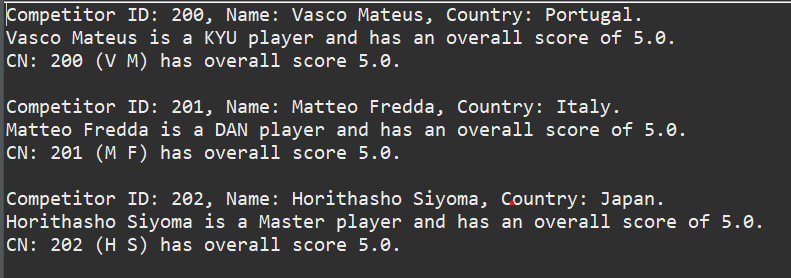
**Known Bugs and Limitations**

* **Fixed Overall Score**: The current **getOverallScore()** method is not functional for actual score calculation. It returns a fixed value of 5 for every competitor, which is unrealistic for evaluating performance in a real competition.
* **Missing Round Scores**: The program does not yet include attributes or methods for handling the scores across the 5 rounds. This will be a limitation until Part 2 is implemented.
* **No Validation for Name**: The Name object is assumed to be valid, but there is no validation or handling for cases such as empty names or improperly formatted names (e.g., missing first name or last name).
* **Exception Handling for Level**: While the setLevel method validates that the level is either "KYU", "DAN", or "Master", it throws an IllegalArgumentException if the value is incorrect. In part 2 it will mange referring overallScore .

Testing Method:



Output:



***Part – 02.***

**Extra Attributes Chosen**

The following attributes were introduced for managing the competitor's performance across the 5 rounds and their ranking:

* **scores (int[])**: An array of 5 integers, each representing the score of a competitor for a particular round (with scores ranging from 0 to 10).

This array allows for easy tracking of scores across all rounds. Scores are validated to ensure they are within the valid range (0 to 10).

* **level (String)**: The competitor's level is calculated based on their overall score. The possible levels are:
  + **KYU** (1-4 score range)
  + **DAN** (5-7 score range)
  + **Master** (8-10 score range)

The level is dynamically assigned based on the competitor's overall score, which is calculated using a weighted average formula.

**Overall Score Calculation Method**

The **getOverallScore()** method calculates the competitor's overall score using a **WEIGHTED AVERAGE** formula, where each round's score is multiplied by a weight corresponding to its round number (1 for the first round, 2 for the second round, etc.).

**Explanation**:

1. The method computes the sum of scores multiplied by their respective weights.
2. The total weight is the sum of the round numbers (1 through 5).
3. The overall score is the weighted average, rounded to two decimal places.

**Formula**:

( Scores1 x 1 ) + ( Scores2 x 2 ) +( Scores3 x 3 ) +( Scores4 x 4 ) +( Scores5 x 5 )

1 + 2 + 3 + 4 + 5

Overall Score =

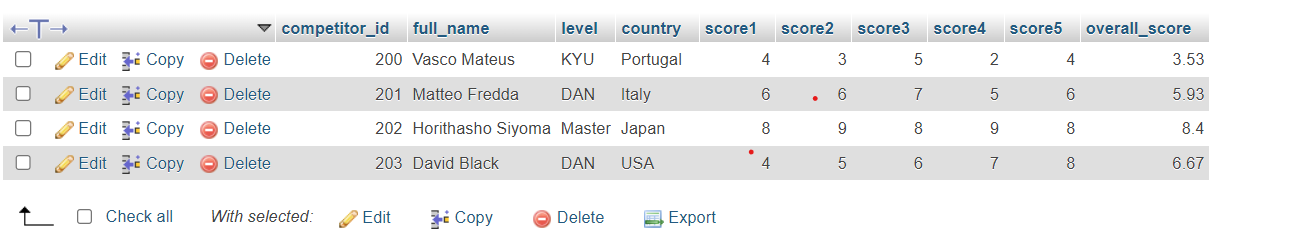
This approach gives more importance to later rounds.



Output:

A screenshot of a computer program

Description automatically generated



***Part – 03.***

I have added two classes: **Manager** and **CompetitorList**. The Manager class handles the main program flow, where it loads competitor data from the database, allows the user to interact via a menu, and provides options to view competitor details, add a new competitor, or generate statistics. The CompetitorList class stores and manages a list of competitors, allowing functionalities like loading data from the database, finding competitors by ID, and generating statistics based on their scores.

* The Manager class controlling the main menu and different user choices.
* The CompetitorList class managing the competitors' data, including adding and searching competitors.

**Known Bugs and Limitations**

* **Missing Round Handling :**

Currently, there is no explicit data validation for the individual rounds' scores in terms of when the scores are entered. It is assumed that scores are provided correctly for all 5 rounds. Future enhancements can handle edge cases for missing scores.

* **Database Connection Assumptions**:

The database connection uses default credentials (e.g., root with no password), which may not be suitable for all environments. It's important to modify these configurations before deploying the system in production.

* **Top Performer in Case of Tie**:

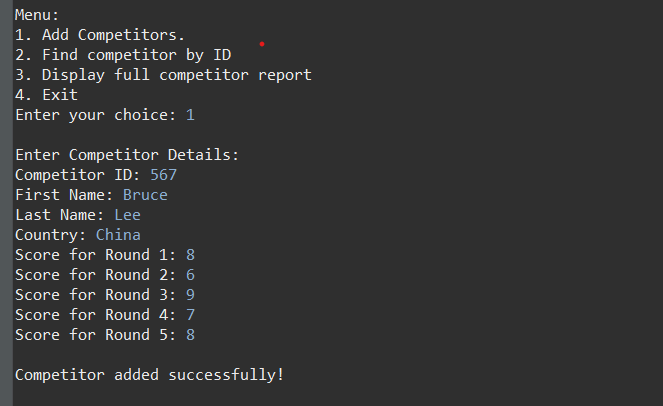
When two or more competitors have the same overall score, the program simply selects the first competitor in the list as the top performer.

* **Limited Attributes**:

No timestamp or competition-specific attributes are currently stored, which could be useful for future enhancements.

Testing Method

Option 01: Add Competitor

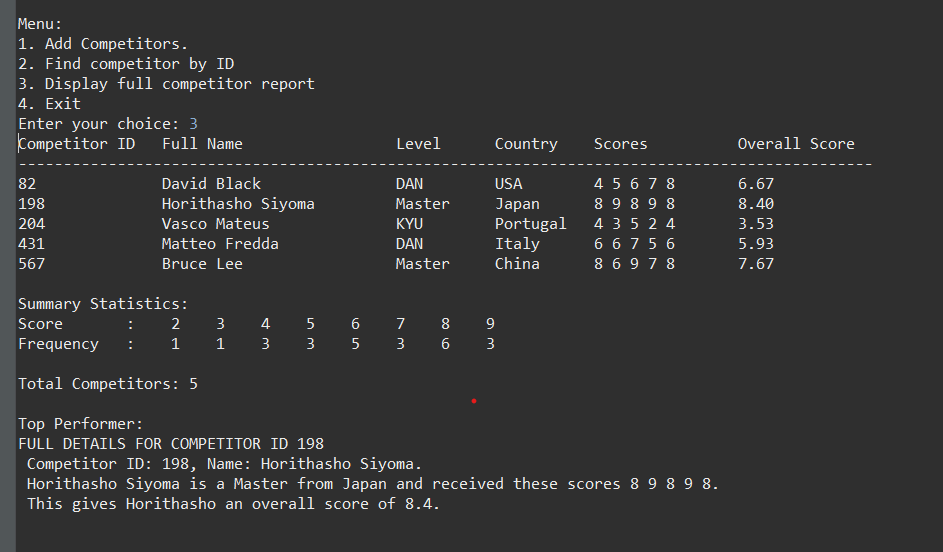


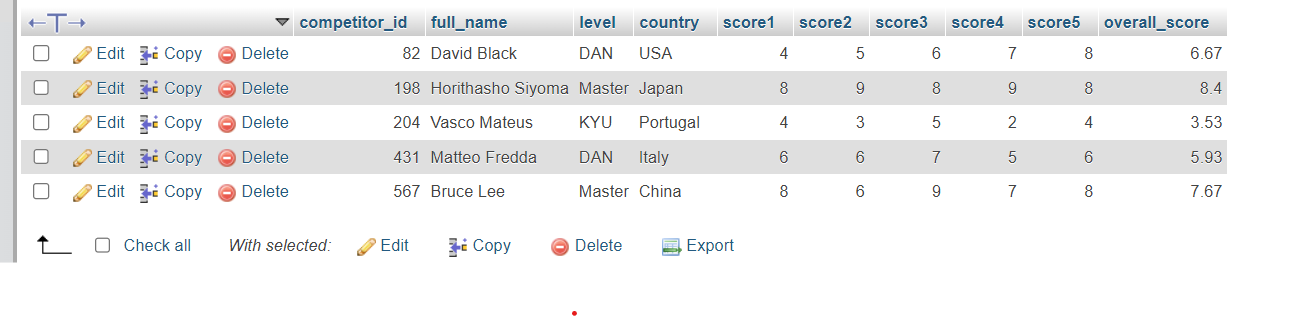
Option 02: Find Competitor By ID

A screen shot of a computer

Description automatically generated

Option 03: Summery Report





ThankYou