Batting Average

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Run: Yes **Evaluate**: Yes
Automatic grade: Yes **Maximum execution time**: 16 s

National Cricket Academy (NCA) wants to monitor the performance of the players registered with them based on the average runs scored in the matches played.

As a software consultant of NCA, help them by writing a java program to meet their requirements.

The application needs to store the runs scored by a player and calculate the average score.

Component Specification: Player Class

Type(Class)	**Attributes**	**Methods**	**Responsibilities**
Player	List<Integer> scoreList	Include the getter and setter methods for the attribute.	

Requirement 1: Store the runs scored by a player

Whenever a match is over, NCA needs to store the runs scored by the players for future reference.

The addScoreDetails method accepts runScored as argument and adds it to the scoreList.

Component Specification: Player Class

Component Name	Type (Class)	Methods	Responsibilities
Add runs scored to the scoreList	Player	public void addScoreDetails(int runScored)	This method takes the runScored as an argument, and adds it to the scoreList.

Requirement 2: Calculate the average run scored

NCA Academy needs to monitor the performance of a player periodically based on the average runs scored.

The method getAverageRunScored should calculate the average runs scored based on the scoreList.

Component Specification: Player Class

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Fetch the runs scored in each match from the list and calculate the average runs scored.	Player	public double getAverageRunScored()	This method needs to calculate the average runs scored based on the scoreList and return the result. If the scoreList is empty, the method should return 0.

The average runs scored needs to be calculated as:

Average runs scored = sum of all runs available in the scoreList / size of the scoreList

Create a Main class with the main method.

Design the menu as described in the Sample Input and Output as:

1. Add runs scored

2. Calculate average runs scored

3. Exit

Enter your choice

When the choice is 1, get the runs scored from the user and add it to the scoreList.

When the choice is 2, display the average runs scored.

The entire program should be executed within a loop. It should not terminate after the completion of a functionality. When the choice provided is 3, it should terminate with a message "Thank you for using the Application".

Please do not use System.exit(0). Instead use break to terminate the program.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object oriented specifications provided in the question description.
Ensure to provide the names for the classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

Sample Input / Output 1:

1. Add Runs Scored

2. Calculate average runs scored

3. Exit

Enter your choice

1

Enter the runs scored

150

1. Add Runs Scored

2. Calculate average runs scored

3. Exit

Enter your choice

1

Enter the runs scored

50

1. Add Runs Scored

2. Calculate average runs scored

3. Exit

Enter your choice

1

Enter the runs scored

50

1. Add Runs Scored

2. Calculate average runs scored

3. Exit

Enter your choice

2

Average runs secured

83.33333333333333

1. Add Runs Scored

2. Calculate average runs scored

3. Exit

Enter your choice

3

Thank you for using the Application