Object Oriented Programming (OOP) Lab Tasks 02 E1

<u>Task 1:</u>

The `SafeArray` class is designed to provide a safe way to access and modify elements in an array of integers. It has the following attributes:

- **`data`:** An array of integers to store the elements.

- **`length`:** The length of the array.

The class offers the following functionalities:

1. Default Constructor:
   The default constructor initializes an instance of the SafeArray class without any parameters. It creates an empty array with a default size of 100. This constructor can be useful when you want to create an instance of SafeArray without specifying the initial size or providing any initial values.

2. Parameterized Constructor with Size:
   The parameterized constructor with size is used to create an instance of the SafeArray class with a specified size. It takes an integer parameter representing the size of the array. This constructor allows you to define the initial size of the array upon object creation. The array created will have the specified size, and all elements will be initialized to default values (0 for integer arrays).

3. `getIndex(int index)`: This method allows you to safely retrieve the value at a specified index in the array. It takes an `index` parameter representing the index of the element you want to retrieve. If the `index` is within the valid range (from 0 to `length - 1`), the method returns the value at that index. If the `index` is outside the valid range, it displays an error message and returns 0 instead.

4. `setIndex(int index, int value)`: This method allows you to safely set the value at a specified index in the array. It takes an `index` parameter representing the index of the element you want to modify and a `value` parameter representing the new value to set. If the `index` is within the valid range, the method updates the element at that index with the provided `value`. If the `index` is outside the valid range, it displays an error message and does not modify the array.

5. `maxValue()`: This method finds and returns the maximum value present in the array. It iterates through the array and keeps track of the maximum value encountered.

6. \`minValue()\`: This method finds and returns the minimum value present in the array. It iterates through the array and keeps track of the minimum value encountered.

7. \`sum()\`: This method calculates and returns the sum of all elements in the array. It iterates through the array, adding each element to a running sum.

The \`SafeArray\` class provides a safe way to access and modify elements in the array by performing index range checks and displaying error messages when accessing or modifying elements outside the valid range.

Task 3:

You are required to implement a \`CricketPlayer\` class in Java that represents a cricket player. The class should have the following attributes:

- **\`playerName\` (String):** Represents the name of the player.

- **\`score\` (int):** Represents the total score of the player.

- **\`ballsPlayed\` (int):** Represents the total number of balls played by the player.

- **\`numFours\` (int):** Represents the number of fours hit by the player.

- **\`numSixes\` (int):** Represents the number of sixes hit by the player.

The \`CricketPlayer\` class should provide the following functionalities:

1. Parameterized Constructor: A constructor that initializes the \`playerName\`, \`score\`, \`ballsPlayed\`, \`numFours\`, and \`numSixes\` attributes of the player.
2. Accessor Methods: Getter methods to access the \`playerName\`, \`score\`, \`ballsPlayed\`, \`numFours\`, and \`numSixes\` attributes of the player.
3. Get Strike Rate: A **private** method that calculates and returns the strike rate of the player. The strike rate is calculated as (score / ballsPlayed) * 100.
4. Get Boundary Percentage: A **private** method that calculates and returns the percentage of boundaries (fours and sixes) scored by the player. The boundary percentage is calculated as ((numFours + numSixes) / ballsPlayed) * 100.
5. A public print method that displays the information of Player

Sample Output:

```
Player Name: Babar Azam
Total Score: 78
Balls Played: 54
Number of Fours: 8
Number of Sixes: 3
Strike Rate: 144.44
Boundary Percentage: 25.92
```