**Players and Team-Requirement**

**Phase-1 End Project**

**Pre-Requisites**: Should have good knowledge of Classes, Interfaces, Generic Collections and LINQ(Lambda expressions, Extension methods and Anonymous Function)

**Application Environment:** Console Application using C#.

**Case Study:**

FastPace Cricket Academy has decided to create a solution to maintain information about teams’ players for one day game with the below functionalities:

1. User will be able to add a player to the team with details Player Id, Name and Age.
2. User will be able to remove a player from the team by passing the player id.
3. User will be able to get player details by passing the player’s id.
4. User will be able to get player details by passing the player’s name.
5. User will be able to get all player details.
6. User will not be able to add more than 11 players to the team.

To fulfill the above requirements, follow the instructions given below:

1. Create **Player** class with auto-implemented properties:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Modifier** |
| PlayerId | int | Public |
| PlayerName | string | Public |
| PlayerAge | int | public |

2. Create an interface **ITeam** and declare functions as given below:

|  |  |  |
| --- | --- | --- |
| **Interface Members** | **Member** | **Description** |
| void Add(Player player); | Function | To implement functionality for adding a player to the Team by passing an object of type Player as a parameter. |
| void Remove(int playerId); | Function | Implement functionality for removing the player from the Team by passing Player Id as a parameter. |
| Player GetPlayerById(int playerId); | Function | To implement functionality to get player by passing Player Id as a parameter. |
| Player GetPlayerByName(string playerName); | Function | To implement functionality to get player by Player Name as a parameter. |
| List<Player> GetAllPlayers(); | Function | To implement functionality to get all players from the Team. |

3. Create a derived class **OneDayTeam** to implement ITeam interface functionalities as given below.

|  |  |  |
| --- | --- | --- |
| **Class Members** | **Member** | **Description** |
| public static List<Player> oneDayTeam = new List<Player>(); | Global Field Member | Create a global variable of type List<Player> to collect objects of type Player. |
| public OneDayTeam() | Constructor Function | Write a constructor to set the capacity property to 11 |
| public void Add(Player player) | Function | To implement functionality for adding a player in the Team by passing an object of type Player as a parameter. |
| public void Remove(int playerId) | Function | Implement functionality for removing the player from the Team by passing Player Id as a parameter. |
| public Player GetPlayerById(int playerId) | Function | To implement functionality to get player by passing Player Id as a parameter. |
| public Player GetPlayerByName(string playerName) | Function | To implement functionality to get player by Player Name as a parameter. |
| public List<Player> GetAllPlayers() | Function | To implement functionality to get all players from Team. |

4. After writing all the above functions write the code in the **Main** function inside the default class **Program.**

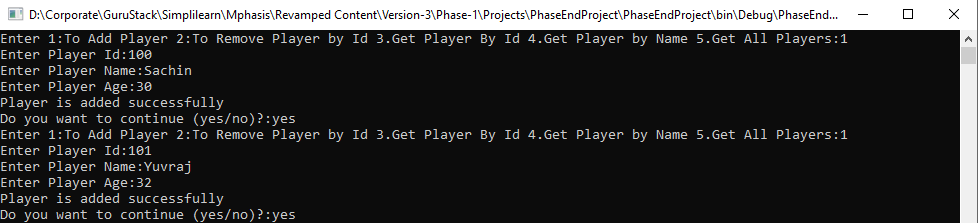
a. Create a menu to prompt the user to perform the required functionality as shown below:

"Enter 1:To Add Player 2:To Remove Player by Id 3.Get Player By Id 4.Get Player by Name 5.Get All Players:"

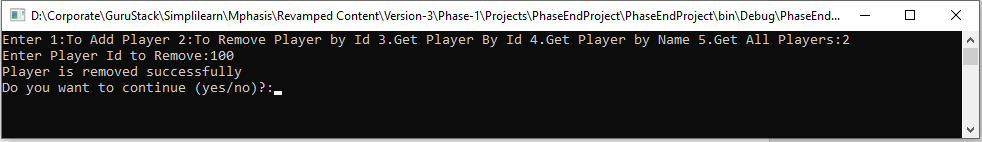
b. Write switch-case construct to call the required function from **OneDayTeam** class as per user choice.

5. Please observe below Console Output Window to meet all the above requirements:

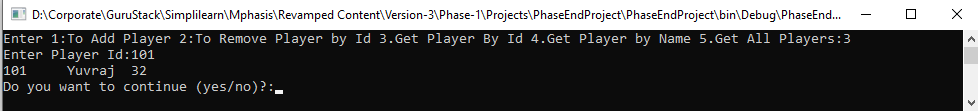
a. To add Player:



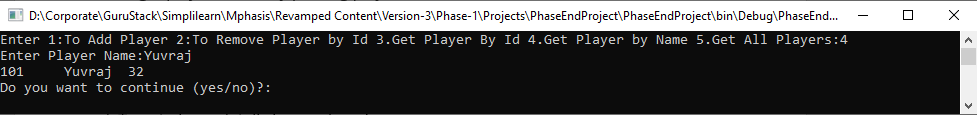
b. To remove a player by passing player id.



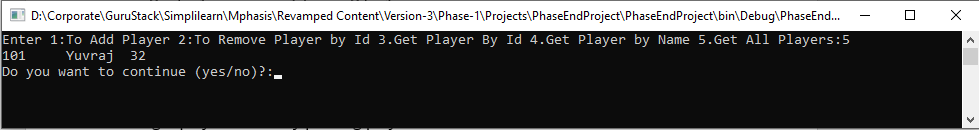
c. To get player details by passing player Id.



d. To get player details by passing player name.



e. To get all player details currently present in the Team.



f. Full output window with all inputs and outputs:

