**Source Code**

## **Program.cs (namespace PlayerAndTeamApp)**

**Source Code:**

using PlayerAndTeamLibrary;

using System;

using System.Collections.Generic;

namespace PlayerAndTeamApp

{

class Program

{

static void Main(string[] args)

{

OneDayTeam oneDayTeam = new OneDayTeam();

bool isContinue = true;

while (isContinue)

{

Console.WriteLine("Enter your choice:\n 1:To Add Player \n 2:To Remove Player by Id \n 3.Get Player By Id \n 4.Get Player by Name \n 5.Get All Players:");

int choice = Convert.ToInt32(Console.ReadLine());

switch (choice)

{

case 1:

var playerToBeAdded = EnterPlayerDetails();

oneDayTeam.Add(playerToBeAdded);

Console.WriteLine("Player is added successfully");

isContinue = Confirmation();

break;

case 2:

int idOfPlayerToBeRemoved = RemovePlayerById();

oneDayTeam.Remove(idOfPlayerToBeRemoved);

Console.WriteLine("Player is removed successfully");

isContinue = Confirmation();

break;

case 3:

GetPlayerById(oneDayTeam);

isContinue = Confirmation();

break;

case 4:

GetPlayerByName(oneDayTeam);

isContinue = Confirmation();

break;

case 5:

var playerList = oneDayTeam.GetAllPlayers();

ShowAllPlayers(playerList);

isContinue = Confirmation();

break;

default:

Console.WriteLine("Invalid choice");

isContinue = Confirmation();

break;

}

}

}

public static Player EnterPlayerDetails()

{

Player player = new Player();

Console.Write("Enter Player ID: ");

player.PlayerId = Convert.ToInt32(Console.ReadLine());

Console.Write("Enter Player Name: ");

player.PlayerName = Console.ReadLine();

Console.Write("Enter Player Age: ");

player.PlayerAge = Convert.ToInt32(Console.ReadLine());

return player;

}

public static int RemovePlayerById()

{

Console.Write("Enter the player ID to Remove: ");

int playerId = Convert.ToInt32(Console.ReadLine());

return playerId;

}

public static void GetPlayerById(OneDayTeam oneDayTeam)

{

Console.Write("Enter the player ID: ");

int playerId = Convert.ToInt32(Console.ReadLine());

Player player = oneDayTeam.GetPlayerById(playerId);

if(player.PlayerId==0 && player.PlayerName == null && player.PlayerAge == 0)

{

Console.WriteLine("Player does not exist");

}

else

{

Console.WriteLine(player.PlayerId + " " + player.PlayerName + " " + player.PlayerAge);

}

}

public static void GetPlayerByName(OneDayTeam oneDayTeam)

{

Console.Write("Enter the player name: ");

string playerName = Console.ReadLine();

// OneDayTeam oneDayTeam = new OneDayTeam();

Player player = oneDayTeam.GetPlayerByName(playerName);

if (player.PlayerId == 0 && player.PlayerName == null && player.PlayerAge == 0)

{

Console.WriteLine("Player does not exist");

}

else

{

Console.WriteLine(player.PlayerId + " " + player.PlayerName + " " + player.PlayerAge);

}

}

public static void ShowAllPlayers(List<Player> playerList)

{

foreach (var item in playerList)

{

Console.WriteLine(item.PlayerId + " " + item.PlayerName + " " + item.PlayerAge);

}

}

public static bool Confirmation()

{

Console.Write("Do you want to Continue (yes/no)?:");

string confirmationResult = Console.ReadLine();

if (confirmationResult == "yes" || confirmationResult == "y" || confirmationResult == "YES" || confirmationResult == "Yes" || confirmationResult == "Y")

{

return true;

}

else if(confirmationResult == "no" || confirmationResult == "n" || confirmationResult == "NO" || confirmationResult == "N" || confirmationResult == "No")

{

return false;

}

else

{

Console.WriteLine("Please enter valid input");

return Confirmation();

}

}

}

}

## **Player.cs (namespace PlayerAndTeamLibrary)**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PlayerAndTeamLibrary

{

public class Player

{

public int PlayerId { get; set; }

public string PlayerName { get; set; }

public int PlayerAge { get; set; }

}

}

## **ITeam.cs (namespace PlayerAndTeamLibrary)**

**Source Code:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PlayerAndTeamLibrary

{

public interface ITeam

{

void Add(Player player);

void Remove(int playerId);

Player GetPlayerById(int playerId);

Player GetPlayerByName(string playerName);

List<Player> GetAllPlayers();

}

}

## **OneDayTeam.cs (namespace PlayerAndTeamLibrary)**

**Source Code:**

using System.Collections.Generic;

namespace PlayerAndTeamLibrary

{

public class OneDayTeam : ITeam // derived class "OneDayTeam" to implement "ITeam" interface functionalities

{

public int Capacity{ get; set; }

public OneDayTeam() // constructor to set the capacity property to 11

{

Capacity = 11;

}

public List<Player> oneDayTeam = new List<Player>();

public void Add(Player player) // adding a player

{

if (oneDayTeam.Count <= Capacity)

{

oneDayTeam.Add(player);

}

else

{

System.Console.WriteLine("Cannot have more than 11 players");

}

}

public void Remove(int playerId) // removing the player

{

for (int i = 0; i < oneDayTeam.Count; i++)

{

if (oneDayTeam[i].PlayerId == playerId)

{

oneDayTeam.RemoveAt(i);

}

}

}

public Player GetPlayerById(int playerId) // get player by passing PlayerId

{

Player playerDetails = new Player();

foreach (var item in oneDayTeam)

{

if (item.PlayerId == playerId)

{

playerDetails = item;

break;

}

}

return playerDetails;

}

public Player GetPlayerByName(string playerName) // get player by PlayerName

{

Player playerDetails = new Player();

foreach (var item in oneDayTeam)

{

if (item.PlayerName == playerName)

{

playerDetails = item;

break;

}

}

return playerDetails;

}

public List<Player> GetAllPlayers() // get all players

{

List<Player> allPlayerDetails = new List<Player>();

foreach (var item in oneDayTeam)

{

allPlayerDetails.Add(item);

}

return allPlayerDetails;

}

}

}