Dot Net Phase 1: Player and Team Project

Phase-End Project - Writeup

- ## Introduction

  FastPace Cricket Academy is a C# Console Application that provides a solution to maintain information about the teams' players for one day game. It was developed as a part of the course "Agile, Git and basics of C# programming".

- ## Application overview

  The main functionality of the application is to enable users to register and maintain information about Players and the Team. The basic functionality is as follows:

  ➔ User will be able to add a player to the team with details Player Id, Name, and Age.
  ➔ User will be able to remove a player from the team by passing the player's Id.
  ➔ User will be able to get player details by passing the player's Id.
  ➔ User will be able to get player details by passing the player's name.
  ➔ User will be able to get all player details.
  ➔ User will not be able to add more than 11 players to the team.

- ## Project Structure

  ➔ **Interfaces**

  - ➔ **ITeam Interface**

    The ITeam interface outlines a set of methods that define the basic operations like adding, removing players and managing a team of players.

  ➔ **Classes**

  - ➔ **Player**

    The Player class represents a player and their associated attributes.
    It includes properties for the player's ID, name, and age.

  - ➔ **OneDayTeam**

The OneDayTeam class implements the ITeam interface. The class provides methods to:

- ◆ Add a player to the team.
- ◆ Remove a player from the team using their ID.
- ◆ Retrieve a player from the team using their ID.
- ◆ Retrieve a player from the team using their name.
- ◆ Retrieve a list of all players in the team.

➔ **TeamManager**

The TeamManager class is responsible for managing player operations. The class has the following methods:

- ◆ AddPlayerRemovePlayer
- ◆ GetPlayerById
- ◆ GetPlayerByName
- ◆ DisplayAllPlayers
- ◆ GetValidInt
- ◆ GetValidString

This class was created to separate the user interaction logic from the Program class.

➔ **Program**

The Program class serves as the entry point for the console application. It initializes the TeamManager instance and provides a menu driven user interface for interacting with the cricket team management functionality.

● Functionality

“Input validation is provided for user inputs.”

➔ **OneDayTeam Class**

1. Static List of Players:

The class contains a static List<Player> called oneDayTeam that holds instances of the Player class. This list serves as the storage for the players in the team.

2. Team Capacity:

The class has a property named Capacity, which is set to a constant value of 11 players. This represents the maximum number of players the team can have.

3. Constructor:

- The class has a constructor that sets the team's capacity to 11.

4. Add Player:
The Add method allows you to add a player to the team by providing a Player object as a parameter. It simply adds the player to the oneDayTeam list.

5. Remove Player:
The Remove method takes a player's ID as a parameter.
It checks if a player with the given ID exists in the team's list.
If the player is found, it is removed from the list using LINQ's RemoveAll method. It then prints a success message. If the player is not found, an error message is printed.

6. Get Player by ID:
The GetPlayerById method takes a player's ID as a parameter.
It searches the oneDayTeam list for a player with the provided ID using LINQ's Find method and returns the player if found.

7. Get Player by Name:
The GetPlayerByName method takes a player's name as a parameter.
It searches the oneDayTeam list for a player with the provided name using LINQ's Find method and returns the player if found.

8. Get All Players:
The GetAllPlayers method returns the entire oneDayTeam list, providing a way to access all players in the team.

➔ **TeamManager Class**
1. Initialization:
The OneDayTeam instance is created as a private member within the TeamManager class. This instance will be used to manage the team's data.

2. AddPlayer Method:
The AddPlayer method allows the user to add a new player to the team.
It checks if the team is already at full capacity (11 players) before allowing addition.
The user is prompted to enter the player's ID, name, and age using the GetValidInt utility method.
A new Player instance is created with the provided information.

The new player is then added to the team using the Add method from the OneDayTeam instance.
A success message is displayed.

3. RemovePlayer Method:
The RemovePlayer method lets the user remove a player from the team by providing their ID.
The method checks if a player with the provided ID exists in the team using the Exists method.
If the player exists, they are removed using the Remove method from the OneDayTeam instance, and a success message is displayed.
If the player doesn't exist, an error message is shown.

4. GetPlayerById Method:
The GetPlayerById method retrieves and displays information about a player using their ID.
The method calls the GetPlayerById method from the OneDayTeam instance to fetch the player.
If the player exists, their information (ID, name, age) is displayed.
If the player doesn't exist, an error message is shown.

5. GetPlayerByName Method:
Similar to GetPlayerById, the GetPlayerByName method retrieves and displays player information using their name.
The method calls the GetPlayerByName method from the OneDayTeam instance to fetch the player.
If the player exists, their information is displayed.
If the player doesn't exist, an error message is shown.

6. DisplayAllPlayers Method:
The DisplayAllPlayers method shows information for all players in the team.
It fetches the list of all players using the GetAllPlayers method from the OneDayTeam instance.
If there are players, their information is displayed.
If the team is empty, a message is shown indicating the need to add players.

7. GetValidInt Utility Method:
The GetValidInt method ensures that the user enters a valid integer input.
It repeatedly prompts the user until a valid integer is provided.

This method helps ensure accurate user inputs throughout the class's functionality.

8. GetValidString Utility Method:
The GetValidString method ensures that the user enters a valid string input.
It repeatedly prompts the user until a valid string is provided.
This method helps ensure accurate user inputs throughout the class's functionality.

➔ **Program Class**

1. Initialization:
The Program class begins by instantiating a TeamManager object named teamManager, which will be responsible for managing player-related operations.

2. Menu-Driven Interaction:
Within a while loop, the program displays a menu to the user, presenting a set of options to choose from.

3. User Choice Handling:
The user's choice is stored in the choice variable.
The program then uses a switch statement to execute different actions based on the chosen option.

4. Action Methods:
The switch statement directs the flow to different methods within the teamManager object, based on the user's choice.

5. User Interaction Loop:
After the chosen action is executed, the program prompts the user with the question "Do you want to continue? (yes/no): "
The user's response is obtained through Console.ReadLine(). If the user's response is "yes," the loop continues and the menu is displayed again. If the response is anything else, the loop exits.

6. Termination:
Once the loop ends (either by user choice or by reaching the end of the program), the application terminates.