Assignment 04 – Sports match generator

Comp 123 – Programming 2

DUE: Friday, 20 August 2021, 23:59:59, Eastern time

VALUE: 10%

Overview

You may use the **Sports team trader GUI application from assignment 03** as a base for this assignment, or recreate the application from scratch. Only the new features and updates outlined below will be considered during grading.

Existing features that must still be present are a GUI app to display a set of sports teams, and their rosters. All the starting data (teams, and player data) are provided in **assign04_teamData.json**. The main view will list all the teams in the league and the selected team's roster, changing the roster as the team changes. The trade player functionality (button, second view, and updating the main GUI) from assignment 03 must also be included.

You will introduce the following new features and updates. Update your `Player` and `Team` constructors to ensure IDs are not reused; this should account for IDs loaded from a json file. In new code, rather than `for` loops you should use `foreach` loops; where possible, use appropriate LINQ expressions instead (such as `.FirstOrDefault()`, `.Any()`, `.Select()`, etc). Add two buttons to the main form to allow adding a `Player` or a `Team`; when adding a `Player`, you must put them onto an existing `Team`, whereas a `Team` is added without `Player`s. Added players and teams should be visible without reloading the application (like when trading players).

The other **new feature** to add is a season match viewer. This is a form to generate and view seasonal matchups. The matchups will be handled by a repository that generates season matchups based on user interactions. These matchups will be serialized to a json file. You must generate matches per the following rules:

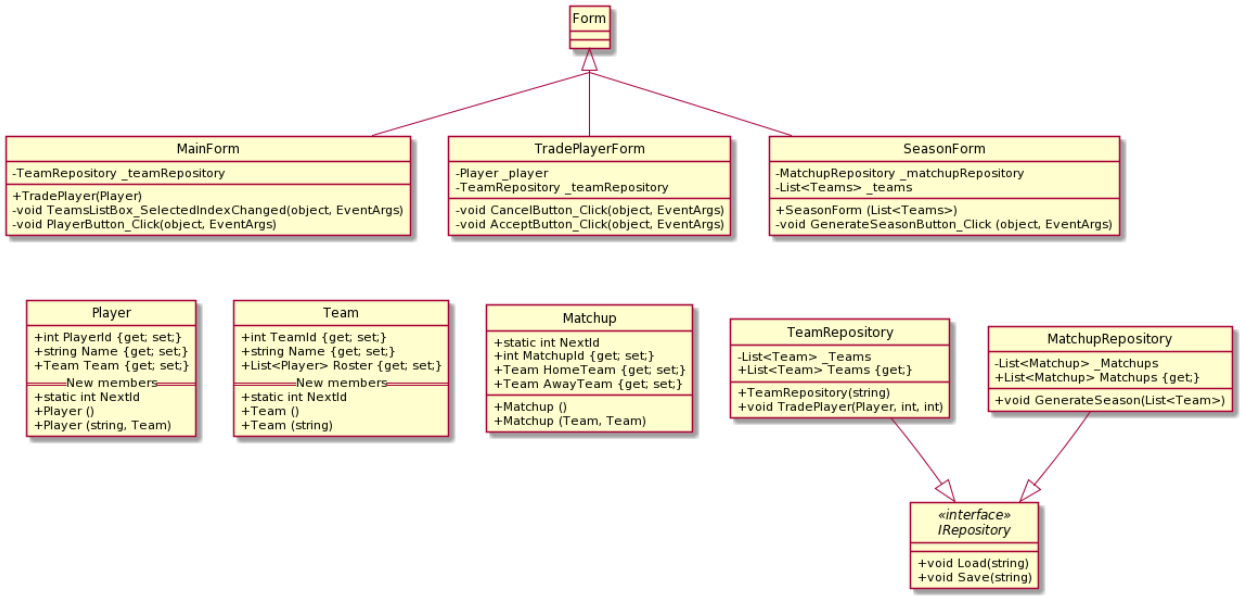
- Each team plays a home game against each other team.
- Each team plays one away game against each other team.
- Teams without Players do not participate in a season.

Your solution will verify those rules using a **test project**. Each rule must have a unit test to show it works correctly.

You can find a mock-up of all screens in **assign04_wireframe.png**; your design can be different from the mock-up if all functionality is included.

Requirements

1. Use the following class diagrams as a guide for your solution. Existing members from assignment 03 are not described in detail.



2. `Matchup` class member descriptions
    a. `MatchupId`: a unique value for each `Matchup`
    b. `HomeTeam`: the hosting team
    c. `AwayTeam`: the visiting team
3. `MatchupRepository` class member descriptions
    a. `_Matchups`: a private collection of all the Matchups in the loaded season
    b. `Matchups`: a public property to give access to the _Teams field
    c. `GenerateSeason`: Using the teams provided, generates a season based on the rules outlined in the Overview section.
    d. `Load`: get the json formatted collection data from a file and re-connect missing values
    e. `Save`: persist the collection data to a file in json format
4. `SeasonForm` represents the code-behind for the `TradePlayerForm` view
    a. `_matchupRepository`: a private field to reference a `MatchupRepository` object
    b. `Constructor`: Accepts the teams currently available to generate a season.
    c. `SeasonComboBox_SelectionChanged`: event handler for when the `SeasonComboBox`'s selected item changes; this will load a season and display it
    d. `GenerateSeasonButton_Click`: event handler for when the `GenerateSeasonButton` is clicked; this will generate a season using the teams passed into the constructor

Submitting your work

Code will be submitted via the eCentennial assignment folders **Assignment 04**. All project files must be contained in a zip file named following the schema below, replacing **mmacdonn** with your Centennial College user ID (your email address, without **@my.centennialcollege.ca**).

> Schema: **Comp123-mmacdonn-Assign04.zip**

Only one file in a folder will be accepted for grading. In the event more files are submitted, **the top-most zip file matching the appropriate naming schema** will be the only file checked for grading **regardless of additional notes in the file name and submission time**. It is **highly recommended** you remove any files other than the desired submission.

Please be aware that you may be required to demonstrate your submission at any time.

Late submissions

The folder will close on a timer set for the above time. No materials, nor changes, will be accepted after that time.

In the event of extenuating circumstances, extensions will be considered on a case-by-case basis. Additional information will be requested in such a situation.

External code and tutoring

Software engineering is an inherently collaborative and open discipline. However, as an academic institution it is important to identify the individual's own knowledge and understanding, as well as adhering to Academic Honesty and Integrity. For this reason, the use of sources outside the lecture and course materials are not permitted.

Additionally, discussion of assignments with your peers or tutors is permitted. However, all materials must be created by the individual student, and not through the actions or aid of another person, whether they are associated with Centennial College or not.

Should you be interested in alternative solutions, you are encouraged to research them and discuss them with the professor. You must do so **before** including such materials in any assignments, graded work, or tests. Failure to do so may result in a violation of Centennial College's Academic Honesty and Plagiarism Policy.