Tennis Tournament Assignment

Introduction

You must create a C# program that can handle a tennis tournament such as Wimbledon or US Open. A tournament consists of a number of games. In each round of the tournament, there is a winner of each game. This winner participates in the next round of the tournament. The loser of the game is no longer a part of the tournament.

A game is played best of 3 or 5 sets. Details are listed below. As an example, a men's single between the players Anders and Bent can have the following result (where Anders wins all three games).

- 6-2, 6-1, 6-5 (3 sets)
- 2-6, 6-5, 6-0, 6-0 (4 sets)
- 6-4, 2-6, 6-0, 5-6, 6-2 (5 sets)

There are both single and double games in a tournament. Each game has exactly one referee that over sees that the game is played correctly.

There are a lot of additional rules for a real-world tennis tournament. The code you write only has to fulfill the requirements listed in this document.

Program Structure

There must be classes for the following in the code.

- Player, referee, and a game master
- A tennis game
- A tournament

You are free to add additional classes or interfaces and make usage of inheritance where you find it appropriately. You are free to name the classes. However, readability of the code is important.

In the following, there is a listing of the requirements to each of the program structures listed above. These requirements are split into data requirements, i.e., what should be stored for an object and functionality, i.e., which methods should there be on each of the classes.

Data Foundation

The following two text files will be provided to you later in the course via Moodle.

- players.txt
- referees.txt

Each line of the text files will contain a single player (or referee). You must be able to read these two text files and create a tournament based on this data.

Requirements to Tournament

There are the following data requirements to the tournament.

- A tournament has a name, e.g., Wimbledon.
- A tournament has a year, e.g., 2014.
- A tournament has a from date and a to date, e.g., from 2014-06-01 to 2014-06-05.
- A tournament consists can consist of 8, 16, 32, or 64 players (or sets of players for doubles). ○ The code you write should work with any of these tournament sizes.
- A tournament consists of 1 or more referees.
- A tournament is always a cup tournament, i.e., after each game there is a winner and a loser. The winner participates in the next round of the tournament the loser is no longer a part of the tournament. As an example, if there are 32 players in the tournament there are
    - 16 matches in round 1
    - 8 matches in round 2, between the 16 winners from round 1
    - 4 matches in round 3, between the 8 winners from round 2
    - 2 matches in round 4, between the 4 winners from round 3
    - 1 match (the final) between the 2 winners from round 4
- In a tournament there are **no** games played for secondary positions, e.g., for the third place.
- A tournament has exactly one referee that is the game master and responsible for the entire tournament.

There are the following functionality requirements to the tournament.

- There **must** be methods for adding removing players to/from a tournament.
- There **must** be methods for adding/removing the referees to/from a tournament.
- There **must** be methods for adding/removing the game master to/from a tournament.
- There **must** be methods for setting the game master.

- There **must** be methods for listing all the players in the tournament. One method must list the players by first name, e.g., Anders before Bent. It must also be possible to list the players by their last name, e.g., Zorro Christensen before Anders Petersen.
- There **must** be a method that can determine if the tournament is over or not.
- There **must** be a method that returns the winner of the tournament.

You may add any methods that are convenient for you to implement the tournament.

To check that the software works it must be possible to simulation the results of all the games in a tournament. There are the following requirements to this simulation.

- There **must** be a method that can create a tournament with 8 players and put these players into 4 games for the round 1 of the tournament.
- There **must** be a method that can create the result of the four games in the round 1.
- There **must** be a method that can pick the four winners of the four games in round 1 and make the two new games in round 2.
- There **must** be a method that can create the result of the two games in round 2.
- There **must** be a method that can pick the two winners of the two games in round 2.
- There **must** be a method that can pick the two winners of the two games in round 2 and create the final game.
- There **must** be a method that can create the result of the final game in round 3.
- There **must** be a method that can show the result of each of the round 1, 2, and 3.

You may implement the simulation of the tournament in the number of methods/classes that you find is appropriate. You may add additional methods that can assist you in the implementation of the simulation. You may place the method for simulating a tennis tournament on the classes that you find most appropriate.

Requirement to Game (or Tennis Game)

There are the following data requirements to game.

- A game has one of the following types
  - A women's single
  - A men's single
  - A women's double
  - A men's double
  - A mix double (one male and one female player on each team)
- You play best of 3 or 5 sets. The result of these sets must be stored.
  - Women's single and double are best of three sets.
  - Men's single and double plus mix-double is best of five sets.
- The winner of a set has 6 points and the loser of the set has between 0 and 5 points.
  - We do **not** care about playing longer than to 6 points in a set.

- A game has exactly one referee.

There are the following functionality requirements to the game.

- There **must** be methods for getting the winner or loser of the game based on the result of the game.
- There **must** be methods for getting the number of sets played in the game.
- There **must** be methods for adding/removing a referee to the game.
- There **must** be methods for validating that it is a legal single, double, and mix-double, i.e.,
  - Two women/men are playing women's/men's single.
  - Four women/men are playing women's/men's double.
  - One woman and one man on each of the teams in a mix-double.

Requirements to Player (a Tennis Player)

There are the following data requirements to a player.

- First name (exactly one name)
- Middle names (zero or many names)
- Last name (exactly one name)
- Date-of-birth, e.g., 1985-12-25
- Nationality, e.g., Danish or Canadian
- Gender, i.e., male or female

There are the following functional requirements to the player.

- There **must** be a method that returns the age of the player in years.

Requirements to Referee

There are the following data requirements to a referee.

- First name (exactly one name)
- Middle names (zero or many names)
- Last name (exactly one name)
- Date-of-birth, e.g., 1985-12-25
- Nationality, e.g., Danish or Canadian
- Gender, i.e., male or female
- Date when referee license acquired
- Date when license most recently renewed.

There are no functionality requirements to the referee.

Additional Information

There are the following additional overall requirements to the code base.

- The code **must** follow the Microsoft C# Coding Convention.
- The code **must** be well-structured and highly readable by humans.
- Each class **must** have appropriate constructors.
- Each class **must** have appropriate properties.
- There **must** be an appropriate use of the basic data types such as int and string.
- There **must** be an appropriate use of access modifiers, e.g., private, protected, and public.
- There **must** be an appropriate use of parameters and return values from the methods.
- There **must** be at least one method that raises an exception.
- There **must** be an appropriate use of the collection classes this includes the use of generics.

You **can** consider the following additional requirements.

- You **can** add variables, properties, and methods that can assist you in the implementation.
- There **can** be usage of abstract classes and interfaces.
- Parts of the public interface **can** be tested using the C# Unit-Test Framework.
- Parts of the public interface **can** be document using the C# XML Documentation features.
- You **can** do simple error checking of for example actual parameters. We will overall assume that the user of the program behaves appropriately.
- You **can** store data in a file or a database if relevant for your implementation.