Assignment 3

Assignment Date: Week 3
Due Date: Week 5 (2 weeks)

Assignment Objective

The purpose of this assignment is to become more comfortable with the C# language and the Visual Studio IDE (Integrated Development Environment). It will also give you a chance to use some of the object-oriented concepts that were taught in the previous course (COMP 1451).

Note that this is the last console-based assignment—all remaining assignments will be Windows Forms projects.

Your Task

Write a console application that 1) reads a file containing data about the results of sports games and 2) displays the data in a visually pleasing way. The program must do just these two things and then quit—that is all.

The program should be able to handle two sports: tennis and hockey. However, your design should make it relatively easy to add more sports in the future. For this assignment, we will be expecting you to use *polymorphism* when you design the classes related to a sport. You may wish to review the object-oriented concept of polymorphism before reading further…

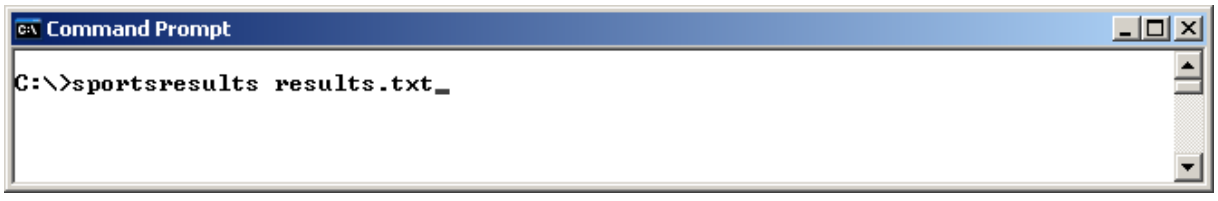
~~~

The format of the file is described below, along with some design requirements that you must follow for this exercise. To keep the assignment from getting too large, your program can assume that the file is always formatted correctly.

Details

Obtaining the Data File's Path and Filename

When you launch your program, specify the path and filename of the data file on the command line. For instance, if the name of your program is `sportsresults.exe` and your data file is `results.txt` then run your program from the command line like this:



In order to pick up the filename from the command line, use the `args` parameter in your `Main` method, like this:

```
static void Main(string[] args)
```
~~~

`args` is an array of type `string` containing any arguments entered on the command line when the program is launched.

When the program first starts, make sure that a command line argument is supplied by checking `args.Length` (it should be 1) and then use `args[0]` as the path and filename of the data file. If no filename is supplied on the command line then inform the user that one must be supplied and then quit.

General Tip: To specify command line arguments in the debugger, from the main menu select: Project | Properties | Debug tab | Command line arguments.

The Data File

The text file should contain outcomes of matches for sporting events. Each match has the following data:

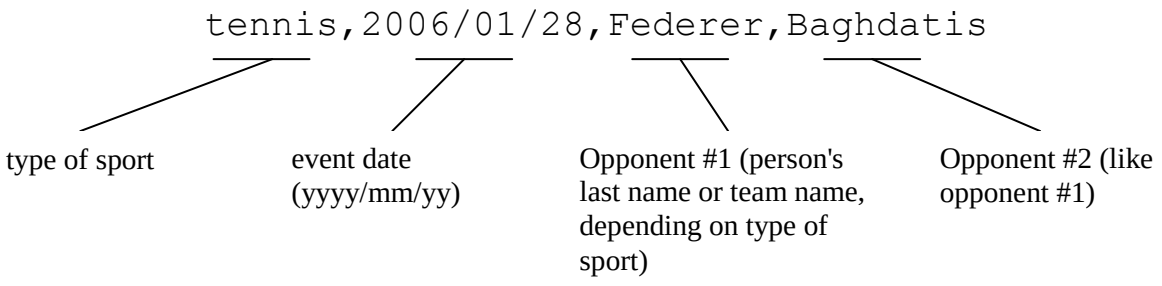
1) type of sport (for example: tennis, hockey)
2) date of the match
3) the names of the two opposing teams or two opposing players, depending on the sport
4) the outcome of the match, which could be a different amount of data depending on the type of sport

The program should expect the data file to be formatted with the so-called Comma Separated Value (CSV) format. We could use the XML format instead but let's keep it simple. CSV files store each "entity" on a single line. In this program, an entity is a sporting event outcome as described above in 1 to 4. All of the values comprising an outcome are separated by a comma character. For this particular CSV file, your program should expect the order of the data for each sporting event to be the same as the order listed above in 1 to 4.

Write the data file yourself using Visual Studio, Notepad or any text editor, and invent some data for it. The file should contain zero or more tennis matches and zero or more hockey matches, in any order. Here is an example of a CSV file that contains two tennis matches and one hockey match:

```
tennis,2006/01/28,Federer,Baghdatis,5,7,7,5,6,0,6,2
hockey,2005/04/22,Canucks,Oilers,6,1
tennis,2006/01/24,Clijsters,Hingis,6,3,6,3
```

No matter what the sport, the first four fields will always contain the same kind of data. Example:



After the first four fields, each sport has different data for the outcome of the match:

Tennis matches consist of at least two mini matches called *sets*. The winner of a set is the player with the highest score in that set. The winner of the match is the player who wins the most sets. Therefore, after the first four fields—sport, date, opponent #1 and opponent #2—should be at least two pairs of numbers, where each pair is the outcome of a set. From these numbers, the program should be able to determine who won the match (the player who won the most sets) or if there was a tie. Here is an example of a tennis match where the second opponent won two sets out of three and therefore won the match:

```
tennis,2006/01/28,Federer, Hernych,1,6,6,5,4,6
```

Hockey matches have three periods but the file will only store the final score. So, after the first four fields—sport, date, team #1 and team #2—should be one pair of numbers for the outcome of the match. From these numbers, the program should be able to determine who won the match or if there was a tie. Here is an example of a hockey match where the first team won:

```
hockey,2006/03/14,Flyers,Rangers,3,2
```

For simplicity, your program can assume that the file is always formatted correctly.

General Tip: Make use of the `String.Split` method.

Displaying the Information

The term d*ata* is slightly different from the term *information* in the following way: Data is a collection of values that do *not* have any *obvious* meaning to a human trying to read it, whereas information is data that has been presented in a way that is meaningful and clear to humans. All of the data from the file must be displayed to the screen as information—I will leave it up to you how to display it. I am *not* looking for anything fancy but the output should be clean, clear and complete. If you simply output the contents of the file without changing the format at all then marks will be lost. Don't forget to inform the user who won or if there was a tie; as part of the information displayed about each match, indicate which team or person won, even though this information is not stored in the file as a separate piece of data.

Design Requirements

Follow these design requirements in your program:

1) use an object-oriented approach
2) read the data from the file *into objects*
3) design for the possibility of adding more sports in the future, like baseball (this speaks to the class design)
4) to make things simpler, players and teams have no "intelligence" so just make them strings
5) design for high cohesion and loose coupling (practice good encapsulation and abstraction)
6) minimize code duplication and logic duplication

I realize that some of these requirements might be quite challenging for you at this point so I am allotting only about 10% of the mark to design. In other words, even with a weak design, you can get around 90% for this assignment. The main purpose is to get your hands dirty in C# code.

~~ Please turn over ~~

For all your code, be sure to:

- Use good tabbing in your code
- Use good spacing in your code
- Comment your code intelligently
- Use XML comments for each class summary and for each of its *public members*

Please Hand In:

1. A copy your project files copied to your directory in ShareIN.
2. A sample data file that you used for testing.

Please DO NOT Hand In:

1. The `bin` and `obj` folders that come with the project. They can be quite large and I do not need them. Marks will be taken off if you include these folders!