University of HUDDERSFIELD

CFS2160: Programming Stream

Tutorial/Practical Week 14

Introduction

There is no Log Book this term.

The exercises in the lab will gradually build up a library of code that you will be able to draw on when you start work on the second assignment.

The purpose of this exercise is to introduce the concept of **properly** refactoring existing code **if suitable**.

Activities

Examine the Super League table:

http://www.bbc.co.uk/sport/rugby-league/super-league/table

Construct a program that could be used to manage this table when the season starts.

Your program should allow for the creation of teams, and the recording of their results in the league. It should be able to print out a league table, sorted in order of the number of league points.

League Rules

A match is played between two teams. The team that scores the most points wins the match, and is awarded two points in the league. The losing team gets no points. In the event of a draw both teams take one point each.

In the league, the teams are ordered first by the number of league points. If this is equal, the ordering is determined by "Points Difference", which is the difference between the number of points scored in matches and the number conceded. In the unlikely event that Points Difference is equal, the teams are ordered randomly[1].

[1] This is not true, but is a sensible simplification here. In real life, the teams would be sorted first on number of wins, then number of points scored in matches, and so on. Feel free to program a more accurate way of sorting; it's not that complicated to do.

Hints

You will probably need three classes: one represents a team, one the league, and one generates the league table. You have seen this pattern before, so refer back to your notes from the lecture.

Think about the instance variables needed for a team. The number of games played by a team is just the sum of the number won, drawn, and lost, for example. The number of league points is just twice the number of wins, plus the number of draws.

Think about which getters and setters are needed. Most of the instance variables for a team are changed in only one way - when they play a match. So there will need to be a method that adjusts the various values given the result of a match.

The table will need to be formatted neatly, in columns.

Illustration

Suppose there have been four results:

- Leeds have beaten Huddersfield 22-12.
- Wigan have beaten Huddersfield 18-12.
- Wigan have beaten Hull FC 34-0.
- Leeds and Wigan have drawn 10-10.

The final table (and the output of the program) should resemble:

```
Super League Table

Wigan Warriors             3    2    1    0    62   22    40   5
Leeds Rhinos               2    1    1    0    32   22    10   3
Huddersfield Giants        2    0    0    2    24   40   -16   0
Hull FC                    1    0    0    1    0    34   -34   0
```

Each result is recorded using two method calls, one for each team, like so:

```
RugbyTeam hudds = new RugbyTeam ("Huddersfield Giants");
RugbyTeam leeds = new RugbyTeam ("Leeds Rhinos");
.
.
hudds.playMatch (12, 22); //note that the points are reversed
leeds.playMatch (22, 12);
```

If you need any more hints - Ask!

Extra Activities

Add the functionality to sort by for / against difference if more than one team's points are the same.