University of HUDDERSFIELD
Inspiring tomorrow's professionals

Software Design & Development
CFS2160

Week 14 – Understanding a Question

Session Plan

University of HUDDERSFIELD
Inspiring tomorrow's professionals

- An important message!
- Look at how we can understand Tony's tutorial questions.
- Questions?
- Continue to work on any outstanding practical work.

Important Message!

University of HUDDERSFIELD
Inspiring tomorrow's professionals

Model before programming

For coursework 2, you are required to develop a full system. The assignment states you must complete the model BEFORE starting coding. You HAVE TO create a model for the assessment, do it first and it will save you time!

The information in this presentation will help you start to think in a way to aid the following tutorial work and assignment.

Understanding a question

University of HUDDERSFIELD
Inspiring tomorrow's professionals

You may have noticed that Tony poses questions that are vague, there is a reason for this. We want you to think about the work you are doing before you start coding (modelling).

Understanding what you need to do is crucial to answering the question.

Re-using Tony's code without the knowledge of what it does will most likely end up taking more time than writing it from scratch. Take the time to understand what the code examples do and how they work.

The following slides give hints on how to identify what you are required to do (a design pattern).

Carefully read the tutorial question

University of HUDDERSFIELD
Inspiring tomorrow's professionals

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].

Look at the website suggested, it will show detail of what is expected

Gives us an indication of how the scores need to be calculated

Tells us how to order the teams in the league

Check the hints

University of HUDDERSFIELD
Inspiring tomorrow's professionals

Tony puts hints in for a reason, read and understand what they are and why we might need them?

It might not be necessary to have getters and setters for all attributes of a class, don't just put them all in for the sake of it.

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.

Further clues?

University of HUDDERSFIELD
Inspiring tomorrow's professionals

This is not a full set of results, perhaps you could add more?

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:

Note it says should resemble, this doesn't mean your output must be the same as Tony's?

```
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
```

Will all of these value will be required as attributes in our team class? Some of these values can be calculated from others. This is mentioned on the tutorial worksheet.

Even more clues!

University of HUDDERSFIELD
Inspiring tomorrow's professionals

This tells us that the constructer of a teams only require one value, the team name. Create an instance for each real team in the Super League

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);
leeds.playMatch (22, 12);
```

We can also see that that the team has a method to deal with playing matches, put such a method in the Team class

Each match requires a match entry for the home and away team, see that the values 12 and 22 are reversed.

Other Classes

University of HUDDERSFIELD
Inspiring tomorrow's professionals

> 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.

Tony gives an indication as to the other classes needed for the programme as shown before.

The programme will need a single `main` method to be run, this should be in a class that does not represent the Team or the League.

All of the creating of `Teams` and the `League` should happen in the `main` method (exactly the same as the Poppleton Dogs home example).

Putting it all together

University of HUDDERSFIELD
Inspiring tomorrow's professionals

At this point you should have a good understanding the requirements of the tutorial question, have identified the necessary classes and the relationships between them.

The league will have an `ArrayList` which contains all of the `Team` Objects, the printing of the league table requires a `loop` and is the same as `Christmas Club` and `Dogs Home` examples.

The techniques mentioned in this presentation can be viewed as modelling, combining this with UML can give you a very good understanding of the problem you are trying to solve.

Finally

University of HUDDERSFIELD
Inspiring tomorrow's professionals

- Any questions about today's session?
- Continue with any unfinished tutorial work.