Golf Case study: some quick notes

If one applies the noun approach to establishing the classes most of them are fairly obvious. These being a golfer, hole, golf course, round and match. But when drawing the class diagram there are some considerations that need to be taken into account:

- A match and a round are not the same thing. A match is a specialized form of a round but is not the same as a round. A match will have a number of attributes and operations that a round will not (see diagram). So you have to pick up on this difference and model it into the diagram. Also, note that in both round and match there are input operations. That is because the players have to input the score for each hole. Also as players can query their scores there must be a query operation and some form of validation. Now a second note with the scores is how one records the score for each hole as there are 18? A more elegant option is to have an associative class that will record the hole no, the score for the hole and the round. While this is an option you can also just have 18 records in your round class that will store the information.

The relationship between hole and round is a many to many as one round will consist of many holes and one hole will be played by many rounds. So to store the information accurately you need an associative class. This is not the only way though, you could have 18 records in round or you could have a ternary relationship between your round, hole and golfer classes.

- Another point is the option of non-member golfers taking part in matches. It is not stated whether the tournaments are exclusive to members. Therefore, it is a good idea, for extra functionality, to have that option. Hence you can model in a non-member player.

- As only employees can make changes to the records of the match and the rounds, an employee class will be required. This class will possess the ability to access and modify various details; you will show this through the operations for the class. Some may argue that this is not required but how else do you model an employee being able to access and make changes to data regarding matches and rounds?

- Now the tricky part, the ranking and scores. There are a few different ways that you could model this. One way is to include some form of a ranking or scoring class. This class would be responsible for getting the match, round details along
with the player details and calculating the scores and then updating the ranking of the concerned players. If you use this option then it makes sense to place as many of your calculation operations in this class in terms of calculating hole scores and then total round/match scores, calculating the points, updating this information, displaying the information etc. Another way is to disperse the responsibilities amongst the various classes. Hence you may have match and round calculating scores for holes and in total, retrieving the player details and then either updating them or sending the message to a golfer class that will then update the necessary attributes. The question to answer is basically do you have one dedicated class doing it or can you add it to the responsibilities of your existing classes? Now the only problem with dispersing it amongst other classes is that you will have a lot of public attributes and operations which could be a security risk, but that is not a concern at this stage. The bottom line is that as long as you have covered that functionality adequately you are okay.

- Now another note on operations is the usage of your basic generic operations: Create, Read, Update and Delete. These are basic operations and often many classes will usually have at least one or two of these operations. Therefore, always think about using them whenever you are looking at operations.

- Lastly I want to quickly mention patterns in object modeling. If you practice enough models you will notice that certain patterns often come up. In other words, a customer in two different case studies will always share certain similarities, often both in terms of attributes and operations. An order may be for a supplier or it may be for a customer, but the two types will possess commonalities. In OO this happens quite a lot and it is always good practice to look out for those patterns.

The diagram provided is fairly basic. There are a number of different ways to model some of the aspects of the case study, I have mentioned a few. You may have modeled the case study differently, which is fine, as long as you have captured the functionality. In other words, have you captured the difference between a round and a match, do you have some form of calculating scores, updating details etc. I will be putting up some more notes in a few hours, please keep an eye on the website.

Regards,
Osman