# Key of what each color means

Unchanged / nothing new.

A ToDo

Movement of writeup, changed wording, new.

Comment.

# Analysis

## Aim:

The aim of my computing project is to create an Golf application which can record scores from matches following various golf match types, i.e Stableford, matchplay and strokeplay. The golf application should be for casual players who just want to have fun, record their matches and see what their golfing friends are up to.

## Identification of project:

Currently at school match level the teams record the scores on score cards which are pieces of cards. These can be a problem because they are easily lost, damaged and can’t cope with rain. Personally, I believe that an online version will be much better as a lot more data can be stored. This is the same belief as Mr. H, who is a professional golf coach at Stowe. He will be acting as a client for this project. In addition, as a casual player it’s very hard to see where certain people might be playing. This problem shall also be incorporated into the new application so that you can have “Friends” and if they are currently in a match you can see which club they are playing at, the current score in that match and who they are playing against, if they are part of the new application. Another aspect of the old card-based system is the score calculation. A lot of the golf match types can be confusing and involve a lot of confusing math which can lead to mistakes, with an online computer-based system this means that the computer will accurately calculate the score for each hole without failure, therefore speeding up the game allowing more people to play per day. Finally, a way in which the new system shall help golfers is easy access to the clubs with a hyperlink to their websites, you can see how many holes there are, how long each hole is, a rating into how good other golfers think the golf course is and a short description on the golf course. These are all possible data to add as long with tons of other information.

## What’s currently out there?

Within my research I have gone on to the app store, downloaded and tested multiple apps to see what ideas they have and what makes a strong application as a user perspective. Personally, I am not a golfer meaning I have asked for help in judging these applications from Golfers.

### Golf application 1: “Distance Calculator” creator, “Qi Chen”.

Although this isn’t the golfing application as previously mentioned it proposes a very interesting design idea. It works by taking a photo of the golf flag you then crop the photo so that it’s the height of the bottom of the flag to the top and uses the size of the flag to calculate the distance till the hole. This application requires a calibration to set up the correct distance and I’m unsure of the accuracy of application, but it is still a very interesting idea and one to take into consideration when thinking of what to add to really make the product stand out.

### Golf application 2: “Golf Game Book” creator, “GameBook Oy”.

This is more like the design of project as I originally started with and contains much of what I initially thought up of. It contains a strong social media section where you can post photos videos, golf tips and much more. This is much more advanced, but it might be interesting to add a few 5-minute golfing videos for people to improve their game. Personally, I feel that the capability to add photos and videos personally might be unnecessary for my application, but I can speak with my client and see what he thinks. This application also includes an ability to see which friends are on a course practicing or playing a match/tournament this a very beneficial concept to the application and is one of the most valuable credits. The scoring section of the game is very in depth with thousands of clubs and courses on its database each with various amounts of data. Upon selecting a course, you then have a choice of which type of match play you are playing out of the extensive list you then add your players including members and temporary players who don’t have an account. Finally, you add the scores for each player at each hole and when you finish it shows the winner. The only problem is the lack of data in match and on the club. This is something that I shall aim to improve when creating my version of a similar application.

### Golfer application 3: “Golf Weather” creator, “GolfWeather”.

Does exactly what the name suggest. It tells you the weather at that certain course for a week. It only tells you the weather for 6:00, 9:00, 12:00, 15:00, 18:00, the information is the condition at that time the temperature wind speed and whether there is going to be rain. It also tells you Sunrise and Sunset. These are all very important sets of information that are important in deciding whether to play a game. This could be an interesting to incorporate into the project although I’m not sure how I would be able to keep it up to date without doing it manually. There is a also a function called course near me which takes your location and finds all the courses near that location.

### Golfer application 4: “Golf Breaks Sunshine” creator, “Sunshine Golf Breaks Ltd”

This is another interesting idea that could be incorporated. It is a travel app which has a selection of pre-packaged holidays which vary in duration, and amount of people, which are all based at Golf club and containing rounds of golfs. It also has a selection of Green Fees at famous clubs with buggies or walking. This aspect could be added to the object: Course.

### Golfer application 5: “Golf GPS Range Finder Simple” creator, “Bryan Thornbury”

This range finder works very different to the first one but is still interesting. It works by presenting a google earth view. From this view you can see the golf course and your current position. The next step is to press a position on the map and it will mark the line from you to where you press and marks the length in yards you can then press another position and it will calculate the distance from the new position marked and the previous one. This can be done as many times as wanted and in the top right it has the total distance. This is perfect for recording the length of multiple shots. When your done you can press the rubber and it would delete all the lines you have made.

## User Needs

• To create a profile as a golfer.

• To be able to add more data to the system.

• To be able to create a match and send invites to offer golfers to join the match.

• To be able to add a score for each hole in the match.

• The system must be able to work out the winner of the tournament.

• To be able to add a “friend”.

• To be able to send a “friend” a message.

## Possible Additions

# Design

## General Design of Solution The overall product shall be presented as an application that can be accessed on any device but mainly a phone which then can be used while playing golf as mentioned in the design. The application must be as simple as possible and therefore shall be made from a login screen which then can be used to see whether the user is a casual player or a manager and therefore what this user can see within the application. This is going to help with authorization. The login aspect will also allow the application to identify the user to check which matches they are in and not allow that user to affect a match that they are not a part of. For the login I shall be using Auth0 which will allow the user to use their Facebook and google accounts. This reduces the frustration of creating a new account just for the golfing app. This will also force validation upon the user. Using a pre-created login system allows me to focus down in the technical section of my product without reducing the amount I can portray within my project.

The second section will be the homepage and consists of all the functionality which will be split up into separate categories, for example; Matches will be a heading and there you can create a match. The use of naked objects and objection-oriented code will aid me in the design for the interface of the application and therefore will make the application far easier for the user. This is the most important aspect of the program otherwise some users will not be able to use the program too its full potential.

## Why I chose an Object-Oriented approach?

Object Orienting programing is a very important concept of computer science which allows for the use of many other concepts for example, encapsulation, inheritance, polymorphism and also worked well with functional programing. Object Oriented Programing is where the world is viewed as “Objects” this object could be physical like a Golfer or it can be abstract like a data structure. This is very important for my program where there are many physical objects which need to be incorporated for example; course, hole and golfer but there are also multiple abstract objects, match, score and friends. Each of these objects can interact with each other allowing them to be treated as if they were physical. Each of these objects must have attributes. These are values assigned to an object just like a name is assigned to a golfer or a hole is assigned to course. This allows me to treat a golfer in the code just like a real person and helps me to explain it to my client who is not a computer scientist and have a very limited experience of coding. Finally, an object has behaviors. These are the methods for the objects for example a golfer plays a shot. Both aspects of object orientated programing are very influential on the database. The objects relate to a table with the attributes being the field. Finally, the use of OOP works for the interface. Rather than dragging out each value using procedure programing all the data is together within the object allowing easy extraction from the database.

Objects have relationships just like a database does. These relationships work on a “has a “bases and are called associations. This is another aspect where Object Oriented programing is useful in design of the database. Where I can see which objects need to connect.

Finally, Object-oriented Programing allows for inheritance. Inheritance works by a “is a “relationship between two objects this means that you can create a super class with all the common functions and then build down like a tree network increasing the specificity of the method to the object. This will be useful when creating the different types of matches and also for the golfers.

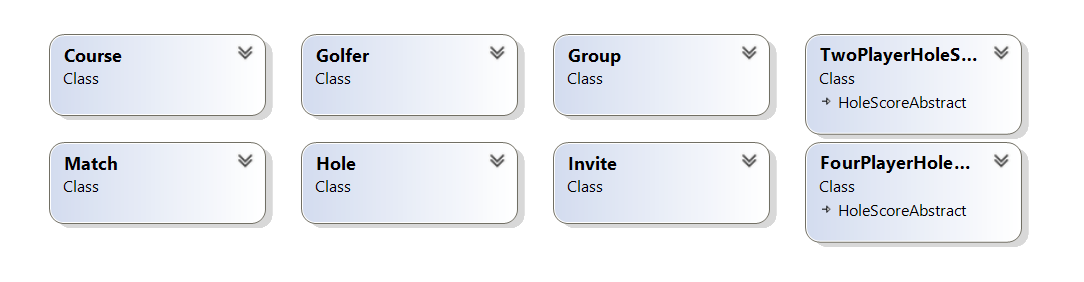
## Why I chose to Naked Objects.

Naked objects is a architecture used in the business world for the creation of various projects. It acts like a framework forcing me into true object-oriented programing. This is because the interface is almost completely consistent of domain objects. The main aspect of Naked Objects which will really help me in the creation of this project is the encapsulation that Naked Object offers. In the business world encapsulation is an object-oriented principle which allows large companies to let two different programmers to work on separate objects and not interfere with one another. This shall teach me many principles used in the business world. In addition, encapsulation also gives birth to another principle called Information Hiding where you are literally hiding the details of an object’s instance variables. This prevents users from seeing unwanted data like birthdays but also prevents objects from see other variables on different objects.

Naked Objects includes restful api…

Naked Object also contains a lot of files which create the initial interface for you. These files could take a long time to create and, in a time pressured project most of the time is centered towards the main functionality of the program and therefore having a foundation to build on is very useful. This has allowed me to change fonts, to authorize values and classes, and change the color of objects along with multiple different little things. Being able to change the color of objects is very useful for the user as they can easily differentiate between objects and now what they are doing. The interface also has a menu made for it. Allowing for easy navigation between objects and pages.

## Class Diagram showing my objects



Objects:

Course: This is just as the name suggests it’s a course. Each course has a few properties that are just data about that course for example a name location address and so on. Course also has a collection of holes this means there is a one to many relationships between course and holes. I have made Hole a separate object because that its self has data about it and its own functions.

Golfer: A golfer is the account that you as a user will create when joining the application. You will need to add handicap but will not have to add properties like emails are optional.

Hole: As mentioned before holes interact with course but other than that they have no method on them self just optional data. This will effective be the exact some rows that are on a paper score card.

HoleScores: this score for each hole and portrayed as a table within the interface. This is the actual score sheet that you can see. There is two objects since match play only has 2 players but stableford and strokeplay has 4 players.

Match: Match is the super class for the other match types it contains the collection of golfers showing another many to one relationship as a golfer can only be in one match at a time. It also ahas the one to one relationship for course.

Group: is a collection of golfers with a group owner and a collection of members this just allows quick access to your friends or people you play with on a regular basis, there can also be a collection of messages that everyone in the group can see.

Invite: This is just a Boolean that says whether to accept to be in that match.

## Class Diagram showing inheritance for matches.

### What Inheritance allows me to do

### Inheritance inside my program.

This class diagram shows that I have deicded that MatchPlay,MatchStableFord and MatchStrokePlay all inherit data and behavior from parent class. This parent class is “Match” and is also known as the super class. The written terms this effectively means MatchPlay “is a “Match and the same for MatchStableFord and MatchStrokePlay. This is helpful in when they all have a certain same method and there for I can reduce the code duplication.

Add Golfer inheritance

For each of the main objects there is also a Configuration class which creates new instances and or methods like show all golfers.

## Class Diagram showing associations.

### What is an association?

## Structure of Database.

# Technical Solution

## Stableford

### What is Stableford

Stableford is a way of scoring a game of golf invented by Dr. Stableford, hence the capital S. Instead of counting the number of strokes like in stroke play you score points based on each hole. This turns up becoming a complicated algorithm as you have to adjust the score based on the Handicap, gender and stroke index of that hole and golfer.

The scoring system works by:

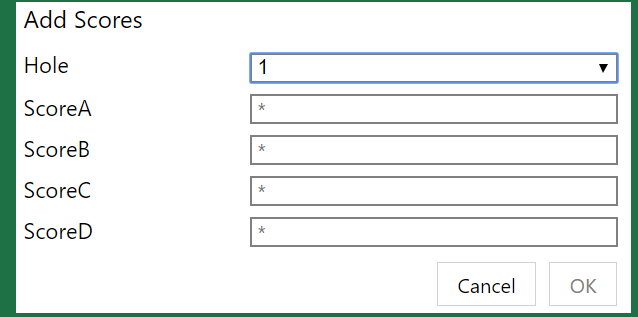
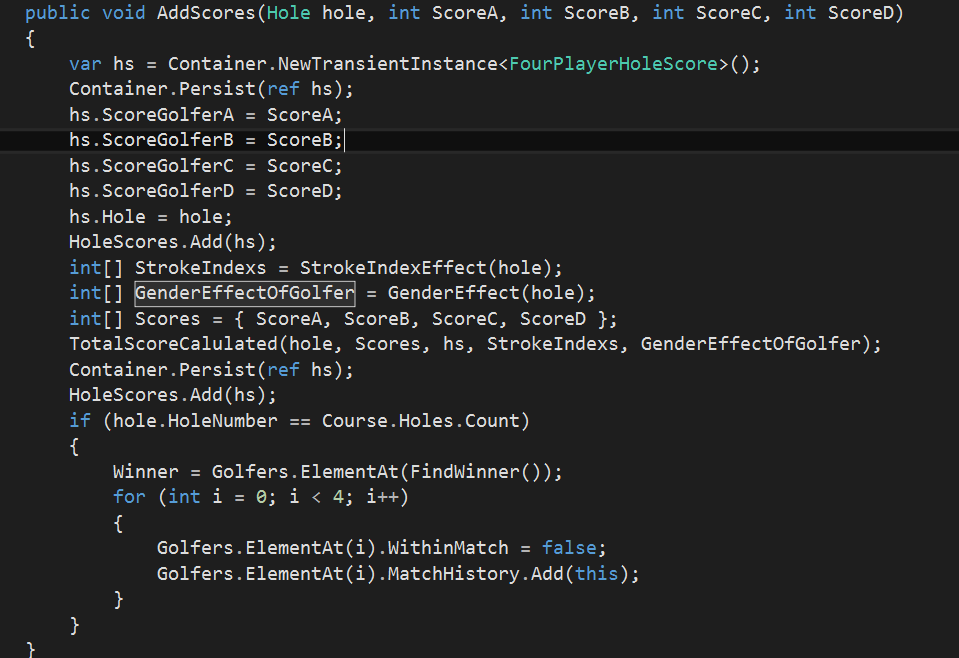
1 point for taking 1 more shot than the par, this is often called a “bogey”.

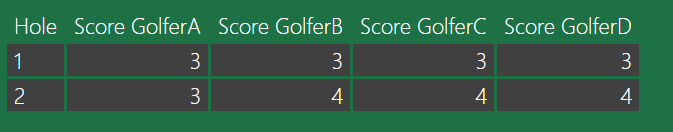
2 points for taking the same number of shots as the same as the par, this is just called “par”.

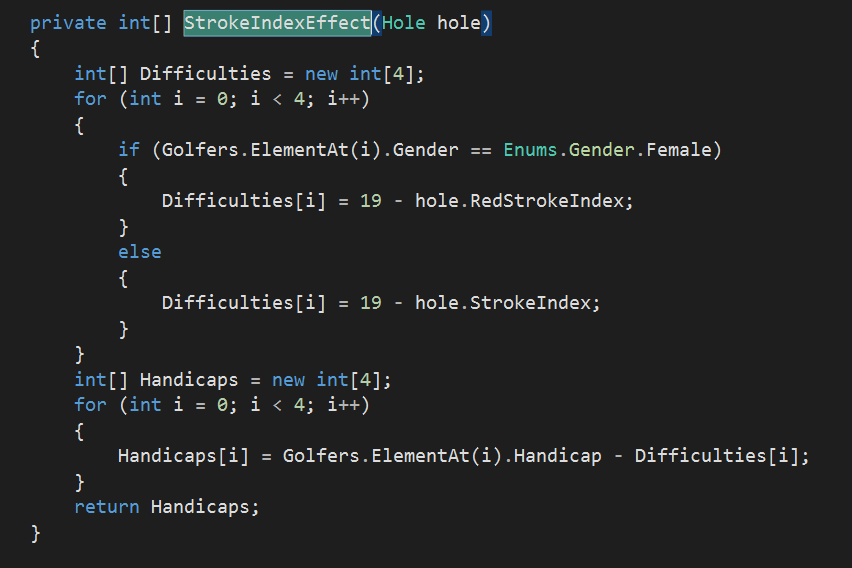
3 points for taking 1 less shot than the par, this is often called an “eagle”.

4 + points for any more shots under the par for example 3 shots under par would be 5 points.

### *How my Algorithm Works:*

Once there is 4 golfers in the match the add score action becomes available to the golfers within that match. Upon clicking this method a table will show with the first hole and 4 empty boxes named ScoreA, ScoreB … This is where one of the golfers will input the raw scores which is the number of strokes each player took. This is where an electronic scoring system is better than a human since it takes out human error as you don’t need to do the complicated algorithm within your head where a mistake could be made. A photo of the table is shown beside.

The next screen shot shows the main function for stableford. As you can see this function delegates to many other functions this was originally all one block of code but was changed since there was a lot of code repetition and it became hard to read what was going on. The First Step that this method takes is that it obtains the values obtained from what you input in the table so that a second table can be added as shown below.

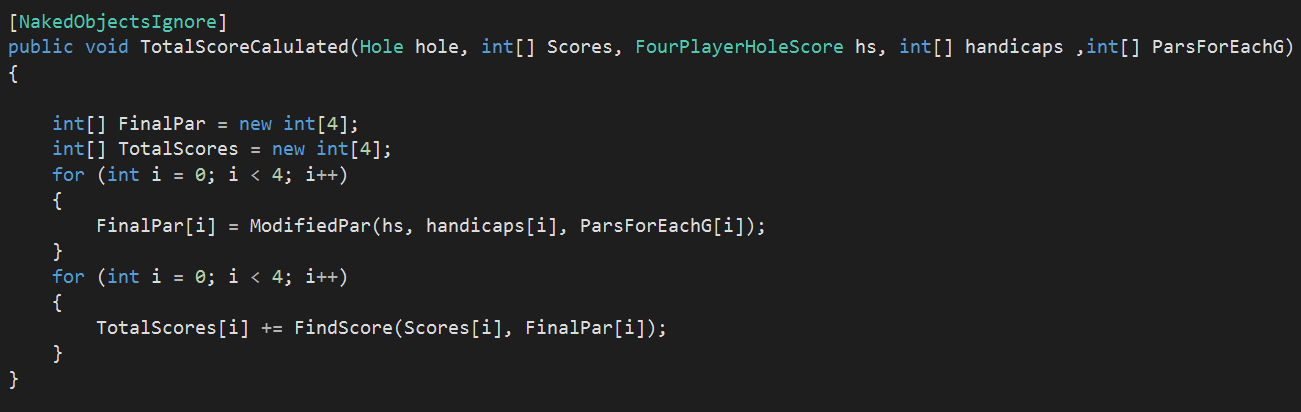
Secondly 3 array of type integers are created the first array is called StrokeIndexs this is because each hole has a different stroke index which effects the handicap. This is calculated using a separated function which takes in the object hole. The function can be seen beside this text.

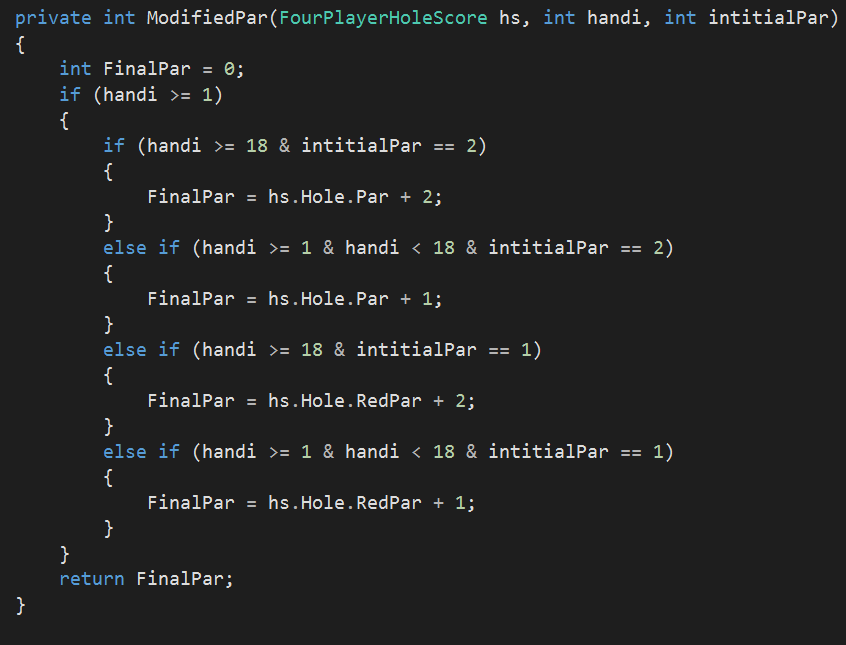
This function starts with an If then else statement that calculates which stroke index to use based on your gender. It then subtracts the value obtained from the Golfers Handicap to create the new value for handicap which is specific to that hole.

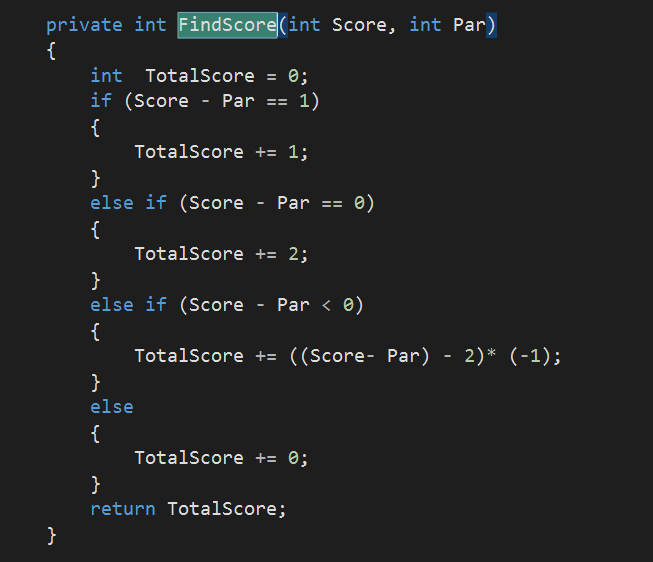
The second array is called Gender Effect which is either a 1 or 2 based on being female or male respectively. This value is also used later for calculating the final score of each golfer on this specific hole.

The third array is called scores which is just the values that you input in the table added into the array.

Once these arrays have been calulcated the scores can be calculated using the method TotalScoreCalculated.

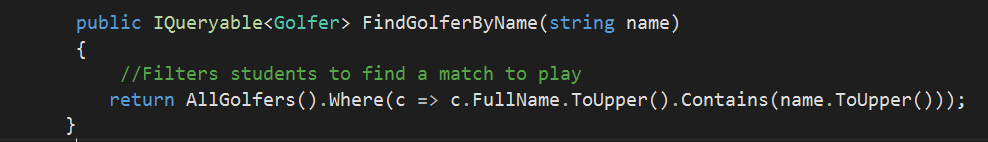
spelling mistake in function name is changed in code need to change screenshot.

This function once again delegates to more method the first method being Modified Par. This uses the first two arrays to calculate whether the par for that hole needs to be increased or not.

The second method actually calculates the score for that golfer using the following method and the scoring system mentions above of scoring par being 2 points , 1 under being 3 points and so on.

These scores are then saved and the program checks whether that was the final hole. If that was the final hole it calculates the winner based on the person with the highest points and saves the golfer object as the winner and adds the match to all of the golfer’s Match History and turns their within match to false so they can join another match.

## Function Programing

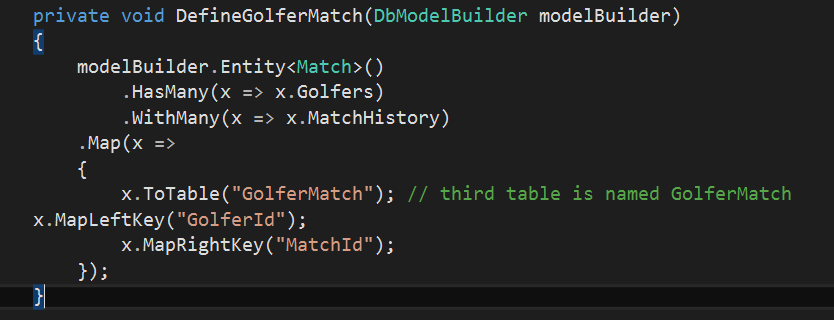
Throughout multiple stages of my programing I have used C# multi paradigm ability to add “LINQ” statements to my code whether deemed useful an example of this is within the class GolfeConfig.

This statement reads: I’ve forgotten how to read it in English.

The use of functional programing here is useful as it is easy to read, and you know exactly the result that you should obtain, therefore its very reliable. Finally, it reduces the amount of statements of code significantly as function program requires no sequential statements. This can be shown in the line of code above without a Linq statement this could have taken multiple lines of code.

## DataBase Mapping

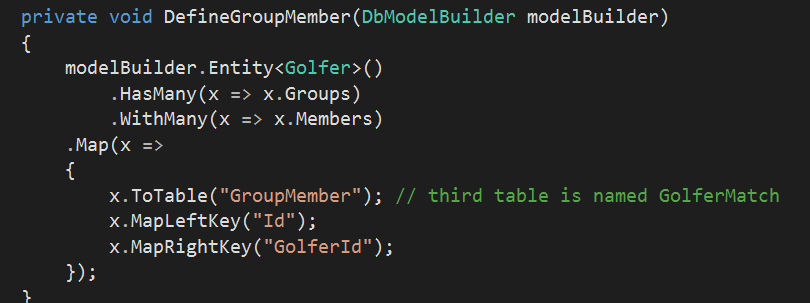
### Manual Many To Many

When my client mentioned to importance of adding match history to the program I made a first attempt but came across a program with my database where a link table wasn’t being a created to cope for the many to many relationship between Match History and Golfers therefore I needed to create a table myself. The code used to create this new table can be seen below.

This creates a third table called “GolferMatch” and uses a compound Foreign key as the primary key for the new table there primary keys used as forign keys are “GolferID” and “MatchID”. The code reads a math has many golfers with many match historys therefore create a GolferMatchTable using GolferID and MatchID. Could this be functional programing?

### Manual many to self.

In another meeting with a client a system of groups was mentioned so that you could have all the people to usually play with in one group and then you could send a message to all of them arranging a time to play rather than sending individual messages. Once again when attempting to create this there was a problem with the database. This relationship was a many to self-relationship and therefore the client was getting confused therefore I manually created a table which allowed for the groups to happen. A Golfer has many groups with many members within the groups and each group has an ID with many Golfer ID’s



# Testing

# Evaluation