My solution met all my original objectives except for being able to add different types of boat. My solution met all these objectives well as my code is understandable and passed all the test cases that I gave to it during the testing phase. The testing phase covered all types of data that could have been given to the program: valid, invalid, erroneous and boundary cases – it passed all of these.

The table below shows what my objectives were and how I met them:

|  |  |  |
| --- | --- | --- |
| Objective | Did I meet it? | How? |
| Provide the user with a menu to choose to play a new game, resume, read instructions or quit the game` | Yes | All aspects of the menu are fully functional as I will go into in more detail in this table |
| When new game is selected, prompt player to enter coordinates for each boat, display the boats on the grid | Yes | Using the Generate Game method, I allowed the user to enter the coordinates for 5 boats. If the same coordinate was entered twice, or if the coordinate was invalid, the user is able to enter a valid coordinate again – no error is thrown. |
| For the player’s turn, allow the player to enter a target coordinate and display whether it is a hit or a miss | Yes | Using the P1 Turn method, I allowed the user to enter target coordinates, catching invalid coordinates as I did in the generate game method. The method then displays the ‘player tries’ grid where the player can see where they targeted and whether it is a hit or a miss. If it is a hit the o is changed to an H and the colour changed to green. If it is a miss, then the colour is changed to red, and the o is changed to an M. This makes it clear to the player if they have hit or missed. |
| For the computer’s turn, random coordinates should be generated. If these coordinates hit the play’s boat, hit should be displayed. If it is a miss, this should be displayed too | Yes | In the Comp Turn method, I randomly generated coordinates for the computer to target. If these coordinates had already been used, they were generated again. The logic for catching invalid coordinates across all my methods was done using a do while loop with all the possible errors within the while() which makes it quite readable. |
| Once either the player or computer has destroyed all their opponents' boats, the winner should be displayed, and the player sent back to the menu. | Yes | The player’s and computer’s boat coordinates are stored in lists. If a boat is hit, its coordinates are removed from the list. The Winner method checks if either of these lists are empty after each turn. If the player list is empty the computer has won and if the computer list is empty the player has won. This way of checking for a winner is faster than my previous method which was to check if there were any B’s left on the grid. However, a grid was not necessary for this, so I just stored the coordinates in a list which reduced the number of elements to check by a significant amount. |
| If the player selects to resume the game, the game’s previous state should be loaded in so that the player can continue playing | Yes | Every time the player completes a turn, the game state is saved to a file. It is also saved if the used types ‘quit’ before the program quits. If the player re-runs the program, they can resume the game state and continue playing as normal. |
| If the player selects to read the instructions, the instructions should be displayed | Yes | A comprehensive set of instructions is displayed along with an example grid to ensure that the player knows how to play the game. |

To further improve my project, different types of boat could be added to the game. This would introduce more complexity to the program as there is more to handle such as: the orientation of the boats, whether the boat has been sunk, hit, or missed and how to choose which type of boat to place.

I think that I could have done this if I had managed my time better. At the beginning of my project, I did not realise how much time would be spent writing up the report rather than coding: I thought that it would be mostly coding and a little bit of writing on the side; in reality it is more of an even split. This meant that in order to finish my project, I had to focus on getting the report done rather than adding the final objective to my program.