Jonathan Wright – OOPD Assignment (19779085)

I implemented a basic layout that is output to the user to allow simple navigation within a loop, I did this as it allows for a decent navigation menu to be output consistently. The navigation menu is then looped using a do-while loop, I chose a do-while loop as the menu will be called once or more times which a while loop does not do (it does zero or more) and a for-loop makes no sense as I don't know how many times it needs to be looped. The validation for the selection integer for the menu was done through the use of a try catch with the exception being caught being a InputMismatchException which will be thrown if something other than a integer was being caught, if a invalid integer is entered it will loop if it is not 7 (Exit) and will be output "Invalid Selection" if it is invalid from the default of switch statement. I can easily add more menu options (I had to add/remove quite a few) by changing the EXIT constant and adding another case statement which is why I chose to add a constant to the UserInterface class. I tried to originally use a modular approach to the menu (allowing easy addition to the displayed menu from a submodule) but I was unable to get this to work logically, I believe that if I could've gotten this approach to work it would have been a much better approach.

Data validation happens through separate submodules within my classes, this is done for neater looking code and better cohesion of the overall submodules (Rather than having the setCylinders having both validation and the setting of cylinders it purely handles setting the cylinders while the submodule for cylinder validation handles validation for cylinders.). The input validation happens through separate submodules to the validation occurring in Ship class, Sub class or Fighter class (it happens through submodules in UserInterface) this is because if an error does occur from the validation in UserInterface (hopefully) the validation in the container classes will pick up and throw an exception to tell us what happened. If the input data is invalid an IllegalArgumentException is thrown with a generally meaningful error message, there is a few cases where there is no message but that is because rather then returning a IllegalArgumentException.message it returns a string message from the validation method about what went wrong, I chose to do this because it felt better than doing it with nested if statements.

Container classes have functionality to do with calculations and setting/getting of the class fields. This is because from a design decision it was decided that all output/input would be handled by the UserInterface class. So, for instance the Ship Class had calc Travel which would be the formula associated with ship type to calculate the time for ship calcTravel and return purely the time and no output occurs. The functionality for the output of this time is then handled by UserInterface class, the reason this design decision was reached was to avoid class responsibility violation.

Inheritance mainly simplified my code by cutting down a lot of repetition of code, however, a big part of my original UserInterface was modifying ships, after I introduced inheritance I was no longer able to get modifyShips to work so I had to comment it all out, I would now have to use toFileString and then use type to figure out which ship to modify, overall it is a lot harder than it was, however, the rest of my code is a lot nicer and easier to understand so I decided to include inheritance for the reason of it being a lot nicer to look at and easier to understand.

Down casting occurred in my code for equals methods. This is necessary to first check that the object being compared to the class object is actually the same e.g. in Object is actually a Ship when being compared to a ship, if it is then we Downcast it to a Ship object, this is to make it, so we can access the ship classes accessors/setters for the equality check.
One of the challenges I had for many days was getting Decimal Format to work for me, it was not declaring and I didn't understand why, I figured out that it was because I was setting the variable name as 2dp which you cant do in java I remedied this by naming it twodp.
Another challenge I had was getting file manager to read CSV (UTF-8), I have been unable to get this to work so I have decided to lose marks for this.
A original design issue I had was not understanding that only UserInterface and Error should be outputting to the user, I had originally had ShipStorage and Submarine outputting to user. Likewise I also had UserInterface having class violation by performing logic that should be handled by container classes, I had it checking whether there were ships or not in shipStorage which I remedied both of these issues to make sure class violations do not occur.