Assignment 1: He Sunk My Battleship Postmortem

# Pre-production (5 hours):

Much of pre-production was handled in class by our discussions and in-class challenges. This covered the following classes and methods therein:

* Gameboard
* Game
* Program

In class, we designed the gameboard class, how to add characters to the grid, how to display the grid and all general formatting therein. We also designed the Program class that holds main, and the beginning of the Game class that held a gameboard, a run method, and a player turn.

From this point, I designed the ship class. I knew that we would need to use random number generation to find the stern, we could use this point to find the bow, then store these values in ship so a ship would know where it is on the board. The ship would also have a length field that would be used to set the bow. Based on the length, a ship could also determine what type it is.

If changes are necessary at all, it would be useful to not have to change every instance of a field. I designed a Constants class that would store things that could be changes such as the length of ships, or the size of the gameboard.

Lastly, I designed the game class with the following tasks in mind:

* Determine if a shot was a hit or miss
* Determine if a shot was valid
* Place the ships

# Production (6 hours):

My production was focused on one task at a time. The first obstacle I tacked was how to put the ships on the board. I first wrote a simple case, placing one ship horizontally on the board. After I made this work, I expanded it to multiple ships. After this was working, I introduced a variable that would determine if the ship would be vertical or horizontal. Initially this did not work and was my first hurdle; I made two sections. If vertical, the bow array index [0] (the rows) would need to change. Otherwise, the bow[1] (columns) would need to change. To validate the placement, I would see if those coordinates had a ship yet, and if they did, it would re-roll and re-place the ship.

After I had developed how the ships would be placed, and confirmed they were working, I needed to develop a way for the ship to know if all its parts had been hit. I used an idea stated in class about having a ship object have an amount of help, and iteratively determined what ship was hit and decremented its health. Once the health was all depleted, it would display that object’s type and that it was sunk to the player. This led to the advent of the Health and Type variables in the Ship class. I also used a successfully registered hit to determine how many hits have been registered total and if total hits equal the total health, the game will end and prompt the user if they would like to play again. Implementing this proved to be a challenge as I needed to store all the ships in something I could iterate to look at the coordinates of each ship. To do this, I initially thought to use an array list, but on Microsoft’s documentation for Array Lists, it stated that List is recommended hence the “fleet” List in the game class.

After this stage of development, I entered my final phase of making a cheat mode and a non-cheat mode. To do this, I used a player board and a shipboard as stated in class. Based on the user’s decision at the beginning of the game, one or the other would show on the console window.