Battleships

Stage 1

Battleships is a classic 2 player game. Each player has a 10 by 10 grid that represents the ocean their fleet is in. Each fleet consists of a number of ships of varying length (specified below). Each player positions their fleet on their own grid. These positions are kept secret. Each player should see two grids, one with their own fleet and where the opponent has fired. The other grid shows what is known so far about the opponent's fleet. That is, where hits and misses have been made so far. The players take turns to shoot at their opponent's fleet by telling them the coordinates they are attacking. An attack result is either of “miss”, “hit”, or “sunk X” (where X is the type of ship). A ship is sunk once every space it occupies on the grid has been hit. The game continues until one player's entire fleet has been sunk. Each player has the following 3 ships in their fleet, which take up the specified number of adjacent spaces on the grid:

- **Minesweeper**
- **Destroyer**
- **Battleship**

Ships can be placed horizontally or vertically on the grid, but not diagonally.

You are part of the team of developers working on this game. The specific task that has been assigned to you is calculating the result of an enemy attack on the fleet: given a set of coordinates of the attacked position, deduce the result as one of the options:

- MISS / HIT
- SUNK (and determine the type of ship sunk)
- SURRENDER (all ships have been sunk).

You should present the UML class and sequence diagrams of your solution.

Try to make your design flexible so that future changes in the specifications will cause as few changes to the existing code as possible.
Battleships
stage 2

For the future release of our best selling game “Battleships”, the following changes have been made to the game’s specification:

- First, a new concept has been introduced - the “captain’s quarters”. If the captain’s quarters are “hit”, the entire ship sinks, regardless of the status of its other elements. In addition, the captains quarters for battleships and destroyers (but not for minesweepers!) are armored, meaning that it takes two attacks on the same square in order to “hit” it (i.e. the result of the first attack always counts as “missed”). The captain’s quarters are located as follows:

```
Minesweeper

Destroyer

Battleship
```

- Second, a new type of special “weapon” was introduced: the “sonar pulse”. The sonar pulse allows a player to reveal a portion of the map, as suggested in the figure below. The sonar pulse merely reveals the status of the cell as being free (grey) or occupied (black), but it does not reveal the type of ship or the location of the captain’s quarters. A player can use a total of two (2) sonar pulses in a game, and only AFTER successfully sinking the first enemy ship.

```
Modify the design of the game, in order to accommodate these changes.
```
Battleships
stage 3

Our product Battleships has a steadily growing fan base. In order to maintain and strengthen our competitive edge, we need to add new features to the game, while at the same time improving the internal design. These transformations are to occur gradually, in a series of increments.

1. Features:
   - A new type of ship will be introduced: the submarine. A submarine can be placed on the surface or it can be submerged, and thus be placed on the grid under any other type of surface ship. It takes up five blocks on the grid, as follows:

   ![Submarine grid]

   - A new type of weapon is introduced: “the space laser”. It is fired from a network of geostationary satellites, and unlike the conventional bombs that we have until now, they penetrate the water, being able to hit both a surface ship, and a sub that is placed below it at the same time (i.e. in contrast the bomb can only hit the surface). The player receives the activation codes for the space laser only after sinking the first enemy ship (i.e. this weapon is an upgrade, and replaces the conventional bomb in the player’s arsenal).

2. Design Improvements:
   Restructure your design to allow interaction with the game through a graphical user interface (GUI). Your design should be based on the “Model-View-Controller” architectural pattern. The first step should be a clear separation between the model component on one side, and the view and controller components on the other side.

A good article from Microsoft explaining MVC: