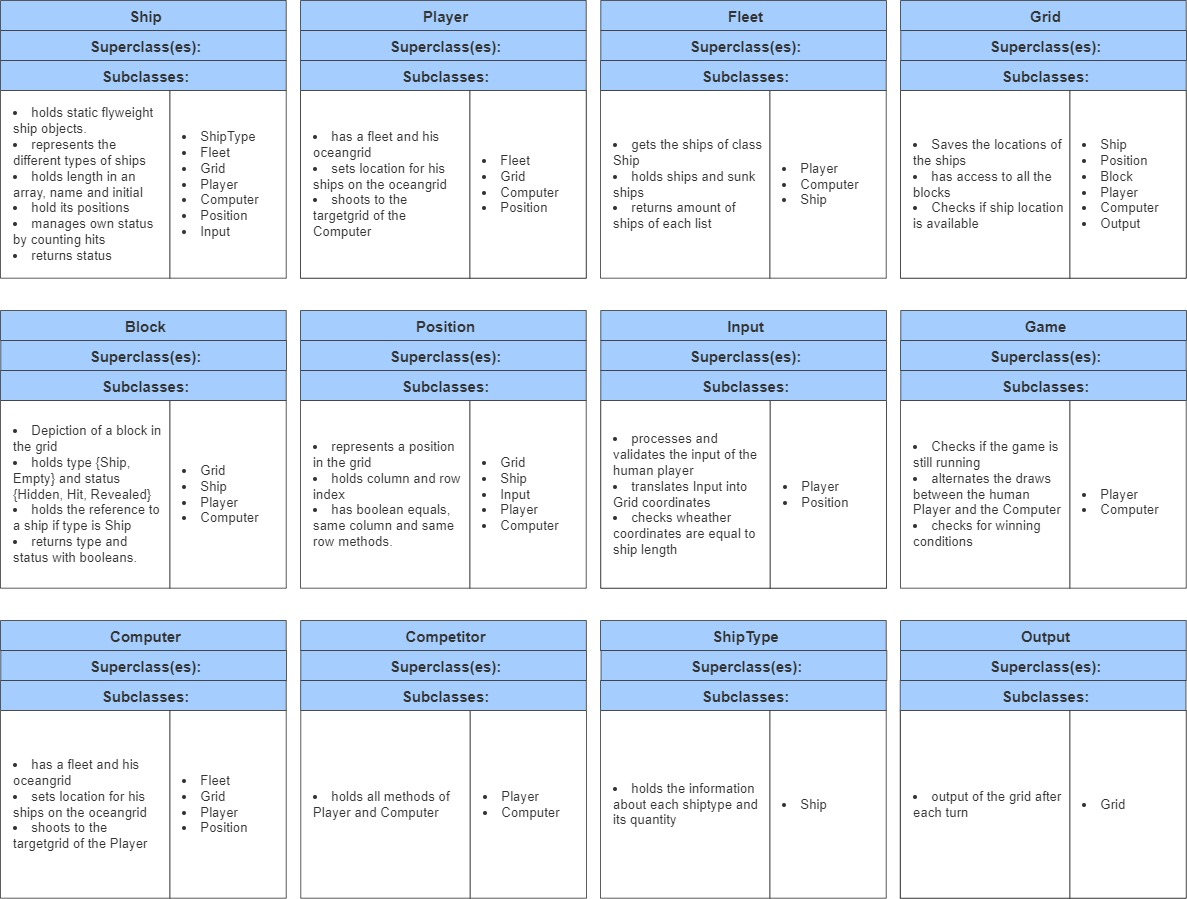
**Exercice 1**

1. **Following the Responsibility Driven Design, start from the game’s requirements and rules and derive classes, responsibilities, and collaborations (use CRC cards). Describe each step you make and store the final cards in your answer.**

Firstly, we want to develop the core elements from the game’s requirements. As one of the key elements we recognize the ships. We think the best way of implementation will be to create objects for each of the ships. Therefore, we will implement a Class Ships, which holds the ships as static flyweight objects. Its main responsibilities are to represents the objects ships, which therefore will have attributes like name, length, position etc. This brings us to two other classes that will be required. Ship Type and Position. The Positions will represent hold the starting and end position of a ship on the grid by having a row and column Index. The ShipType is an enumeration that hold all the information regarding each individual ship type as well as their quantity in the game.

In the before created Class Position we just mentioned the grid. The grid is an essential Part of the game. It is representing the current state of the game. It represents where the ships are placed, where hits are registered in the grid all the location of the ships will be stored. The grid will also be a key Element of the Class Output, which handles the display of the grid after each turn. Each Element of the Grid will be of the Type Block. Block is therefore the next class that needs to be created in order to model the battleship game. The Class block depicts each single block in the Grid and contains the information if it is empty or if there is a ship placed. If the block is hidden, hit, or revealed.

These named Classes should build the foundation for the game, depicting the board and providing the required objects and methods. However, the game needs to be played. Therefore, we will require a Class Player as well as a Class Computer. The Player Class and Computer Class are quite similarly with the difference that the “actions” of the Player are being handled through the input, whereas the computer does this randomly. They both have a fleet, Ocean grid and target Grid. Their Methods are held in the Class Competitor. The afore mentioned Input is the last required Class. It handles, processes, and validates all the input and closely interacts with the Player and Position to do so.



1. **Following the Responsibility Driven Design, describe the main classes you designed to be your project in terms of responsibilities and collaborations.**

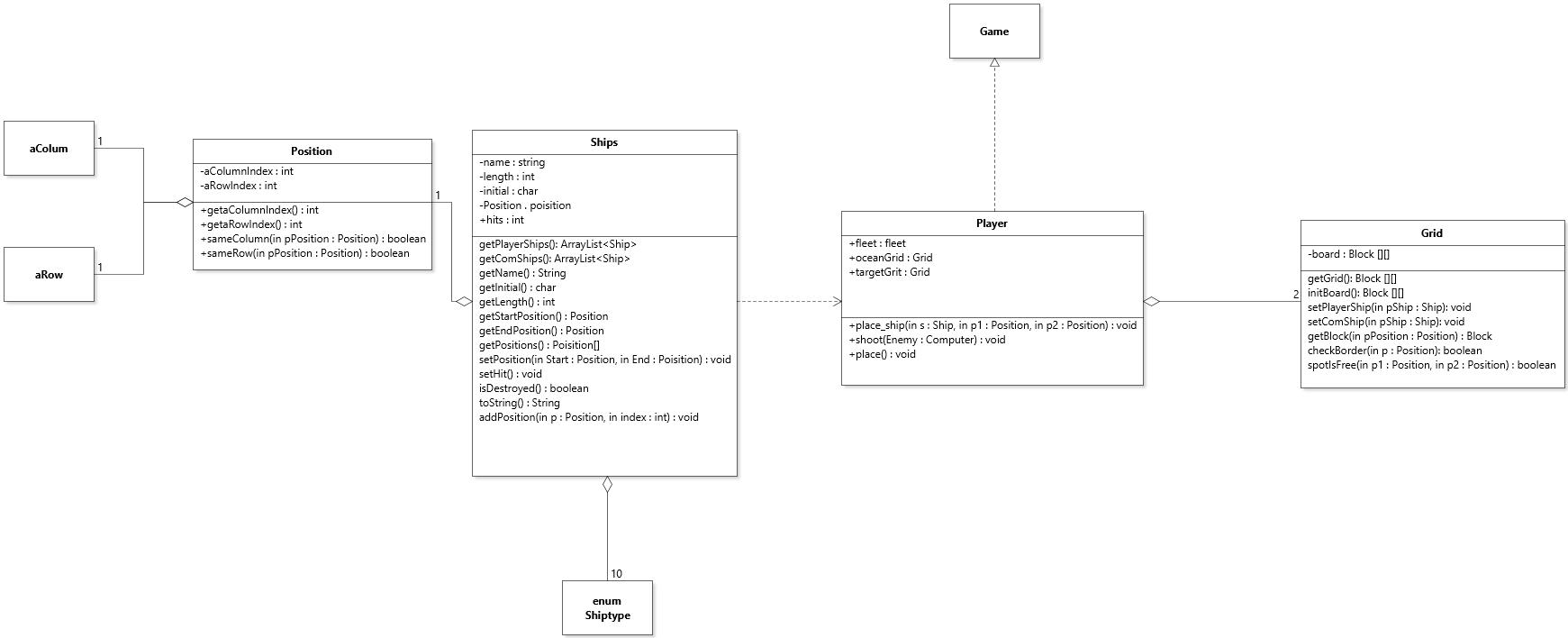
The main classes will consist of the classes “Grid, Player, Ship and Position”, since they will be the most commonly used and therefore are the most fundamental building blocks for the battleship game. Due to that we expect them to have the highest interaction of methods between each other and determine them to be te main classes to be designed.

1. **Why do you consider the other classes as less important? Following the Responsibility Driven Design, reflect if some of those non-main classes have similar/little responsibility and could be changed, merged, or removed.**

Each Class has it´s importance and use, else they could be omitted. However, we identified the aforementioned Classes as key elements of the battleship game. The other Classes are required to provide Objects and methods for the main classes to function for the game to be played.

Regarding which non main classes could be modified we determine, that he Position Class might be merged with the Ship class, since the Position could be assigned as attributes to the ship Objects. Another thought might me to merge the classes of Player and Computer, since they they behave quite similarly. The merge could reduce the amount of code but make it less understandable. The separation of both however might make the implementation easier.

1. **Draw the class diagram of the aforementioned main elements of your game**



1. **Draw an object diagram to show the main elements of your game in a step of the game of your choosing**

**Exercise 2**

The first step was to implement the main Classes.

The Class ship has a private constructor aswell as private Attributes. The Attributes of the Class ship are its name, length, initial, positions and hits. All of the attributes are private and except for the hits final. There are various public methods to return the above named attributes. The Player ships aswell as the computer ships are stored in an array list as flyweight objects.

For the creation of the Objects we require the Class ShipType, which is an enum of all the ships and also contains the exact amount of ships of each type that the player requires.

The Position Class has two attributes, which both are private and final. These Attributes are a column and a row Index. Methods to return these attributes have been implemented aswell as methods to compare.

In accordance with the learned design patterns the equal method has been overwritten.

The next two classes that needed to be implemented were the Grid and Block classes. The Grid is a two dimensional Arraylist of blocks. It contains methods to place the Players ships aswell as the computers ships. Other public methods are the return of an individual block, e check if a spot is free for the placement of a ship aswell as check if a position is withing the boarders of the grid.

Objects of the class block have private attributes. These attributes are type (either empty or a ship), status (hidden, revealed, or hit) a initial and a representation. The initial is received from the assigned ship which gets assigned after the placement on the grid. The methods in the Class Block consist of various public methods that for example Adjust the blocks attributes if a ship is assigned to it (either by the Player or the Computer), change the status to hit if a shot has been placed on a ship or miss if the block has no assigned ship. There are also multiple return methods to return attributes of the block as well as methods checking if a ship is assigned or if the spot on the grid is empty.

Class Player

Class Computer

Class Game

Interface competitior

Class Fleet