# CRC cards

Shape

Description automatically generated with medium confidence

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

Game / GameMaster

*GameMaster makes sure the game is played according to the rules and has two players. The class is responsible for stopping the game when it's over.*

Responsibilities: Initialize the game and ensure the correct execution of the game rules. Determine the starting player and make sure the game ends when a player has lost.

Collaborations: Mainly interact with the **Players** of which the game consists of. Interact directly with the Players **Grid** when the GameMaster must print the state of the game.

Player (Subclass PlayerHuman, PlayerComputer)

*A Player has ONE Grid and ONE Fleet. He collaborates with the GameMaster by declaring and receiving shots. He collaborates with his Fleet and Grid during placement of Boats and while declaring and receiving shots.*

*Note: Difference between Human and Computer is only how they place their Fleet and how they call shots. Computer is randomized (or any strategy really), Human is via user input.*

Responsibilities: Must place his Fleet, which leads to Boats being placed. Must be able to call valid shots for the other player as well as receive them from the other player (via GameMaster). Simultaneously, whenever the Fleet is changed or shots are recorded, the player must update his Fleet and Grid accordingly. Furthermore, the player must be able to provide his up-to-date Grid to the GameMaster and he must be able to declare whether he has lost the game.

Collaborations: With the **GameMaster** when supplying the Grid. With the **Fleet** in all actions regarding placement and shots receiving. With the **Grid** during all the actions which lead to the player updating his Grid.

Fleet

*A Fleet is a collection of Boats and keeps track of placed and destroyed Boats. The specification of the collection is set in the game rules.*

Responsibilities: At any point declare if the whole Fleet is destroyed. The Fleet can receive a shot and declare if any Boat is hit. Must be able to place a Boat when given a list of Coordinates and must be able to tell for a given list of Coordinates if there is an overlap in boats (e.g. if there is a boat already).

Collaborations: With the **Player** when the Boats are placed, or shots are received. With the **Boat** when a Boat is placed and when the status (destroyed) is determined. With **FleetSpecification** to receive the specific boats it consists of.

Grid

*A Grid is a collection of Blocks, given the size of the game rules. It must be able to print itself as either target or ocean grid and needs to update its Blocks according to different events.*

Responsibilities: The Grid must provide a printing method as either a target or ocean grid. It must be able to record and remember shots taken at it and boats placed on it.

Collaborations: With the **GameMaster** when supplying its printing functions. With the **Player** and the **Block** whenever an event leads to an update of the Blocks. The Player will ask the Grid to update for different events. The Grid will pass on this Information to the correct Block.

Block

*A Block is the elementary unit of the grid. Each block has a position in the grid in form of a Coordinate. Furthermore, it knows its state regarding if it has been shot at, it has a boat placed on itself, if it should act as destroyed and what the printing character is.*

*Note: Not all states are important for both printing states.*

Responsibilities: Know and update the status regarding shot at, boat placed on, destroyed status and what kind of character to print if a boat is on there. Need to be able to be called to change any state.

Collaborations: With the **Grid** when changing the states and when the Grid prints itself.

Coordinate

*The Coordinate is the fundamental building block of the whole game and is used through all classes. It provides a common language for positions which make up Blocks, shots, and anything else in the game. It can also convert between internal integer representation of rows and columns and the external representation (e.g. A2)*

Responsibilities: Be able to tell the position in a 2D plane with row and columns numbers. Must be able to print itself either in integer representation or in external representation

Collaborations: With **Boat** for remembering position and shot recording, with **Player** for shot calling and receiving as well as Fleet placement, with **Block** for remembering position.

FleetSpecification

*This class is a lookup artifact which specifies what number of different BoatTypes a Fleet consists of.*

Responsibilities: Know the exact number of BoatTypes which make up a Fleet. Give that Information to Fleet.

Collaborations: With **Fleet** when the Fleet is initialized.

BoatType

*This class describes the different types of Boats that exist in terms of name, description, and length.*

Responsibilities: Know what kind of Boats exist and provide this information.

Collaborations: With **Boat** when a Boat instance is initialized.

# 1.3) 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.

OceanGrid, TargetGrid

We realized that the distinction between two different grids is superfluous. It's enough that each Player only has one Grid and the Grid itself must be able to present itself as either target or ocean grid.

Row / Column Types

We decided to not use enum types to depict rows and columns and use simple integers instead. Reasoning: The most natural way to describe rows and columns as coordinates are in fact integers, not enum types.

FleetSpecification

The FleetSpecificaiton could also be set in the game rules, but we decided to make it a dedicated enum type. We think that it makes the organization of the code cleaner.