# Object Oriented Programming Project

## Battleship Game

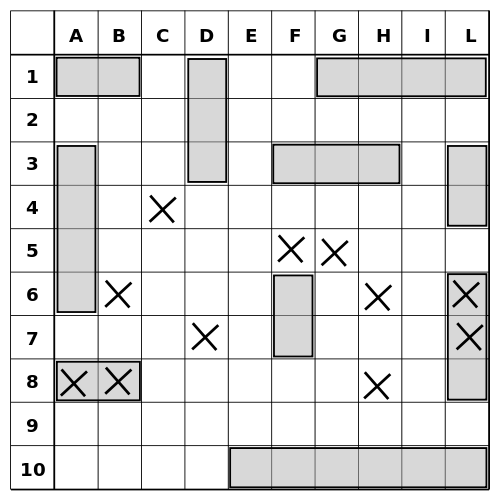
### Team name : "Tigger"

#### Team members: Lina Chalakova, Vladimir Georgiev, Petar Sabinski, Mario Sarbinov

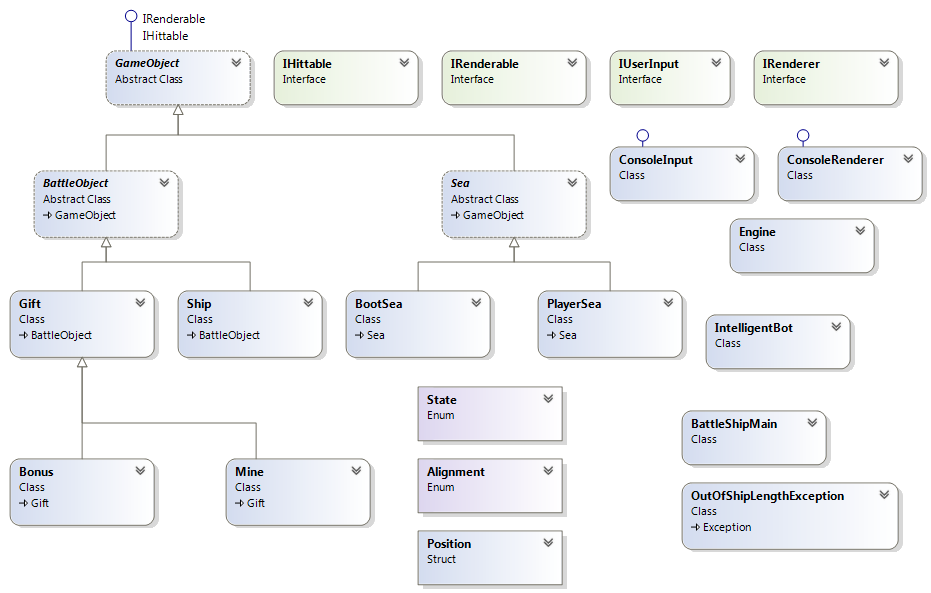
### Project purpose:

The purpose of the project is to implement the well known game "BattleShip". This is a two-player game in which each player has a battlefield(a sea) which could be either rectangle or square with certain dimensions (rows and columns). Players also have a fleet consisting of ships with different length. Each ship is situated on sequence of coordinates, depending on the ship`s length. Therefore the ship forms a vertical or horizontal line inside the body of the sea. On game run each player shoots at opposite player`s sea by naming coordinates in the sea. If the shot is on a coordinate where a ship is situated the ship becomes partially damaged. The aim of the game is to destroy all ships of the opponent before he/she destroys yours. The game could be differently implemented according to the number of ships included, the dimensions of the field, as well as additional objects in the sea that implement additional functionality. In our application we`ve implemented game between a human and the computer.

Look at the following figure that illustrates the game.



### Class diagram:



Briefly explained we`ve determined the following elements:

Interfaces:

* IHittable- having method RespondToHit() differently implemented on different game objects. Saying how an object responds to a shoot command from the different players.
* IRenderable- having method GetBody() that returns an array(might be one-dimensional or two dimensional) containing positions.
* IRenderer() having methods AddSeaToRender(), AddMessageToRender(), ClearRenderObjects(), RenderAll().
* IUserInput interface with methods ReadInputShips(), ReadShootingCommand() that will proceed information entered on the console from reader.

Classes:

* GameObject an abstract class- a parent class of all game objects that implements IHittable and IRenderable interfaces.
* BattleObject an abstract class- a direct derivable of GameObject class- uniting all objects that are direct participants in the battle itself.
* Ship class- A participant in the battle-a direct derivable of BattleObject class. Represents all ships that participate in the battle. The number of its instances may vary according to a const parameter in BattleShipMain class.
* Gift class - A participant in the battle-a direct derivable of BattleObject class. Has two derivable classes- Mine and Bonus. The idea is that they will be a single coordinate class that will add functionality to the game rules. Mine will end the game with a losing player the one that hit it. Bonus will add an additional ship to the sea of the player that shoots on it.
* Sea- an abstract class- representing the game field of each player. Has two derivable classes- PlayerSea and BootSea
* ConsoleInput class that implements the IUserInput interface. It takes care of forwarding the input information from the console to the engine of the game
* ConsoleRenderer- implements IRenderer interface, renders all the information to the console.
* OutOfShipLengthException class- a class that defines an exception to be thrown when a ship with an exceeding length is to be created.
* Engine- core class that runs the game.
* Intelligent robot class- generates intelligent shots.

Enumerations:

* Alignement- vertical and horizontal(determines different possible alignments for the ships).
* State- Empty, Ship, TargetHit, MissedHit. Represents the state for cells of the sea.

Structures:

* Position-keeps X and Y coordinates.

### TFS Repository URL

<https://csharp-exercises.visualstudio.com/DefaultCollection/Tigger-BatleShipGame/_versionControl>