# Documentation for Battleship game.

## Simple UML design

## Program instructions and guide

To start the game, double click execute\_server\_program in the project directory and after the successful execution and it starts to listen for the client, then double click on the execute\_client\_program in the same directory. If both execution is successful the ClientManager class will be called by Server class and the ClientManager receive messages (String) from the client, passes the message to Server’s handleMessagesFromClient() method. Works in a loop until the Boolean flag to stop connection is set to true. The connection/Interaction between the server and the client block waits until it reads a message from the client and then sends it for handling by the server, thread indefinitely waits at the following statement until something is received from the client. If there is an error, while the connection is not stopped, it closes all. And it Performs the function of sending a message from Server to remote Client Using ObectOutputStream, and Closes all connections for the client.

The Client class manages:

opens a connection to the server

setup Object IO streams for the socket.

Handles sending a message to server. In this case, it is a String.

Handle message from the server. In this case, simply display them.

Simply display a String message in the terminal.

handles user inputs from the terminal.

This should run as a separate thread. In this case, main thread.

Can perform any pre-processing or checking of the user input before sending it to server.

Close all connections

The thread that communicates with the server. receives a message from the server, passes it to handleMessageFromServer().

While the Server class manages:

Initializes the ThreadGroup.

Use of a ThreadGroup is easier when handling multiple clients, although it is not a must.

handles messages from each client.

Modified to prepare a response and send back to the same client. If shared among multiple clients, make it synchronized.

Initializes the server. Takes port number, creates a new serversocket instance.

Starts the server's listening thread.

Represents the thread that listens to the port, and creates client connections.

Here, each client connection is treated as a separate thread.

Can perform any pre-processing or checking of the user input before sending it to server.

In this case, the same message is sent to all the clients.

Handle the console user input in the server.

i.e., commands to be send to the client, stop command to the server.

Game Rules:

Both players place their ships on their grid according to the chart above. Whoever goes first calls out a position i.e. 2,-5). The other player says either “Hit” or “Miss” depending upon whether one of his ships is in the position called out. The person calling out should mark a hit or a miss on the “opponent grid” to keep track of the shots. The other person should mark the shot on the “defensive grid”. If the shot is a “Hit”, the player goes again-otherwise the other player takes a turn. Once the opposing player has scored a hit on all of the spaces for a particular ship, you must call out Hit...you sunk my Cruiser“(or whatever type of ship it was). Once a player has sunk all the opponents’ ships, he is declared the winner.

Goal:

Sink all opponents ships.

## References:

Shaukat, A., 2020. *Ayeshashaukat/Project-Battle-Ships-Game*. [online] GitHub. Available at: <https://github.com/AyeshaShaukat/Project-Battle-Ships-Game/blob/master/BattleShips.java> [Accessed 13 March 2018].

Ashjaee, N., 2020. *Learn The Basics Of Battleship With This Easy Guide*. [online] The Spruce Crafts. Available at: <https://www.thesprucecrafts.com/the-basic-rules-of-battleship-411069> [Accessed 11 May 2020].

Sampletemplates.com. 2020. *Battleship Game Template*. [online] Available at: <https://www.sampletemplates.com/business-templates/sample-battleship-game.html> [Accessed 11 May 2020].