Patrick Griffin
Programming Assignment 2 Report
Battleship Game

First, the Game script must be ran. This is your server side of the process and controls the flow of the game. Then you can run up to two Player scripts. The server is started and waits for clients (players).

```
Starting server...
New player request received : Socket[addr=/127.0.0.1,port=63176,localport=5000]
Creating client handler
```

```
//while less than 2 players, loop for client request
while(playerList.size() < 2){
    //if(playerList.size() < 2) {
        //accept client request
        socket = myServer.accept();
    //}
```

(Game.java; lines 23 – 26)

When the player client is ran, the user is asked to enter a username. Once the username has been entered the player is then asked for 3 coordinates for their ships. A board is printed to the user as a reference on where they would like to place their ships.

```
Enter a username:
Pata
```

```
          ***Radar***
|A1|A2|A3|A4|A5|A6|A7|A8|A9|A10
-------------------------------
|B1|B2|B3|B4|B5|B6|B7|B8|B9|B10
-------------------------------
|C1|C2|C3|C4|C5|C6|C7|C8|C9|C10
-------------------------------
|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10
-------------------------------
|E1|E2|E3|E4|E5|E6|E7|E8|E9|E10
-------------------------------
|F1|F2|F3|F4|F5|F6|F7|F8|F9|F10
-------------------------------
|G1|G2|G3|G4|G5|G6|G7|G8|G9|G10
-------------------------------
|H1|H2|H3|H4|H5|H6|H7|H8|H9|H10
-------------------------------
|I1|I2|I3|I4|I5|I6|I7|I8|I9|I10
-------------------------------
|J1|J2|J3|J4|J5|J6|J7|J8|J9|J10
-------------------------------
X = HIT M = MISS

Small Ship:
<]

Where would you like the Small Ship of size 2?
Give a coordinate. Ex: A2, F6, d9, etc...
>
```

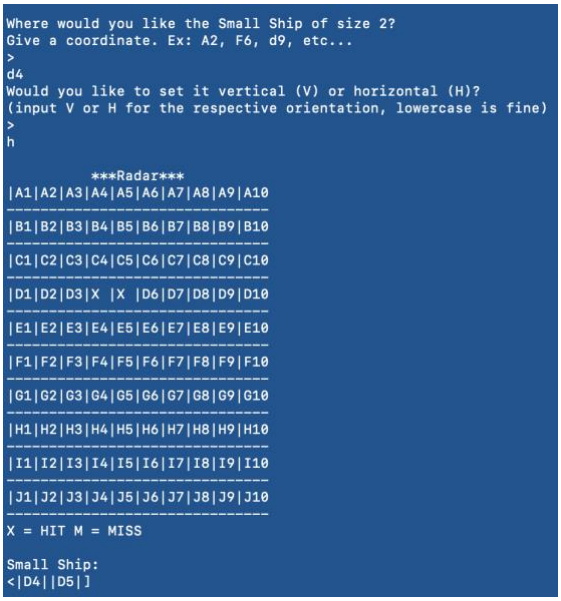
```java
//creates a board for the player
playerBoard = new Board();
//ask the player to set ships on their board
playerBoard.setShips();

//send ship info to server
Ship temp = playerBoard.smallShip;
for(int i = 0; i < 3; i++) {

    for(String coord : temp.getCoordinates()) {
        try{
            //write to the output stream
            dataOut.writeUTF(coord);
        } catch(IOException e){
            e.printStackTrace();
        }
    }
    if(i == 0)
        temp = playerBoard.mediumShip;
    else if(i == 1)
        temp = playerBoard.largeShip;
}
```

(Player.java; lines 57-77)
Code description: Builds a board for the player. Calls Board setShips function. Then sends the coordinates to the game server.

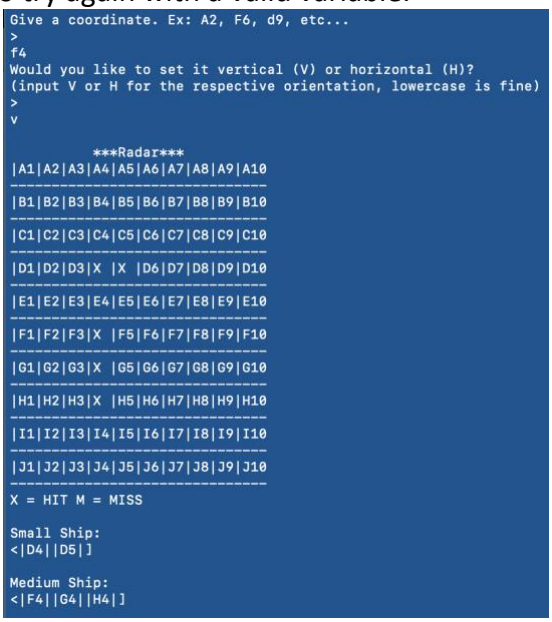
```java
/**
 * Ask player for the coordinates where they want their ships.
 * @param currentShip       the ship that is currently being assigned coordinates
 */
private void setOnBoard(Ship currentShip) {
    char letter;
    boolean validInput = false;
    int letterValue;
    int num;
    String coord;
    String orient;
    String[][] backup = radar.clone();
    String[][] loopBackup = new String[10][10];
    //System.out.println("loop backup: " + loopBackup.toString());

    while(!validInput) {
        try {
            System.out.println(this);
            System.out.println("Where would you like the " + currentShip.getShipType() + " of size " + currentShip.getSize() + "?");
            System.out.println("Give a coordinate. Ex: A2, F6, d9, etc...");
            System.out.println(">");
            coord = input.nextLine();
            System.out.println("Would you like to set it vertical (V) or horizontal (H)?");
            System.out.println("(input V or H for the respective orientation, lowercase is fine)");
            System.out.println(">");
```

(Board.java)
Code description: ask players for ship coordinates

The players must type in a coordinate (a1, a2, b1, b2, j9, j10, etc.), then follow the next prompt with a ‘V’ or a ‘H’ whether the player want to set the ship vertical or horizontal from the initial coordinate given.



The placement of the player's ship is represented on the board as X's, and a ship is printed for the user to see exact coordinates. This continues for the medium and large ships. Ships cannot overlap or be set out of bounds. Player will be instructed to try again with a valid variable.



```
Where would you like the Large Ship of size 4?
Give a coordinate. Ex: A2, F6, d9, etc...
>
i5
Would you like to set it vertical (V) or horizontal (H)?
(input V or H for the respective orientation, lowercase is fine)
>
h

          ***Radar***
|A1|A2|A3|A4|A5|A6|A7|A8|A9|A10
-------------------------------
|B1|B2|B3|B4|B5|B6|B7|B8|B9|B10
-------------------------------
|C1|C2|C3|C4|C5|C6|C7|C8|C9|C10
-------------------------------
|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10
-------------------------------
|E1|E2|E3|E4|E5|E6|E7|E8|E9|E10
-------------------------------
|F1|F2|F3|F4|F5|F6|F7|F8|F9|F10
-------------------------------
|G1|G2|G3|G4|G5|G6|G7|G8|G9|G10
-------------------------------
|H1|H2|H3|H4|H5|H6|H7|H8|H9|H10
-------------------------------
|I1|I2|I3|I4|I5|I6|I7|I8|I9|I10
-------------------------------
|J1|J2|J3|J4|J5|J6|J7|J8|J9|J10
-------------------------------
X = HIT M = MISS

Small Ship:
<|D4||D5|]

Medium Ship:
<|F4||G4||H4|]

Large Ship:
<|I5||I6||I7||I8|]

Please Wait for other player...
```

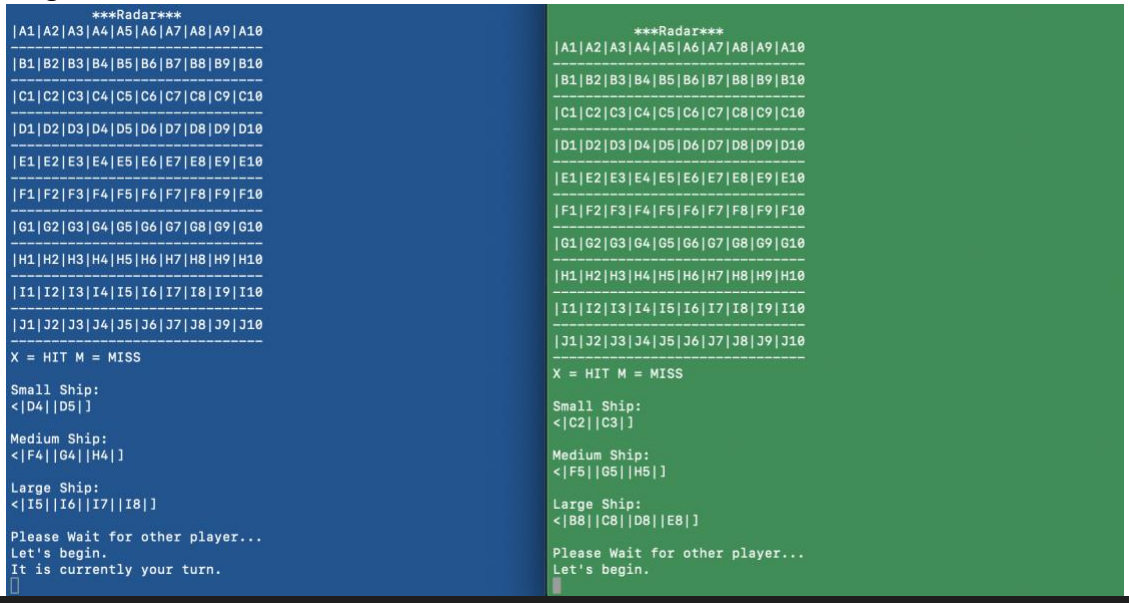
After the final ship is placed the board is refreshed for the beginning of the game. The first player to finish inputting coordinates is told to wait for the second player.

```
58      System.out.println("Creating PlayerHandler");
59      PlayerHandler currentClient = null;
60
61      if(playerList.size() == 0) {
62          //Instantiate player 1 object for request
63          currentClient = new PlayerHandler(socket, name, input, output, Players.PLAYER1, shipCoordinates);
64      } else if(playerList.size() == 1) {
65          //Instantiate player 2 object for request
66          currentClient = new PlayerHandler(socket, name, input, output, Players.PLAYER2, shipCoordinates);
67      }
68
69      //Instantiate a new Thread with the current client
70      Thread thread = new Thread(currentClient);
71
72      //add client to client list
73      playerList.add(currentClient);
74
75      //start the thread
76      thread.start();
77
78      //add players to player list
79      if(playerList.size() == 2)
80      for(int i = 0; i < playerList.size(); i++){
81              PlayerHandler temp = Game.playerList.get(i);
82              temp.dataOut.writeUTF("ready");
83      }
84
85  }
86  //picker player to start game
87  System.out.println("The current player has been choosen.");
88  currentPlayer = Players.PLAYER1;
89  playerList.get(0).dataOut.writeUTF("It is currently your turn.");
```

(Game.java)

Code Description: Once ships have been properly entered the PlayerHandler objects are instantiated, threads are created, and the PlayerHandlers are added to a list for the server to reference.

When the game begins, the server sends a message to the player clients. Then the players are informed and the player that goes first is told that it is their turn.



```
          ***Radar***
|A1|A2|A3|A4|A5|A6|A7|A8|A9|A10
-------------------------------
|B1|B2|B3|B4|B5|B6|B7|B8|B9|B10
-------------------------------
|C1|C2|C3|C4|C5|C6|C7|C8|C9|C10
-------------------------------
|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10
-------------------------------
|E1|E2|E3|E4|E5|E6|E7|E8|E9|E10
-------------------------------
|F1|F2|F3|F4|F5|F6|F7|F8|F9|F10
-------------------------------
|G1|G2|G3|G4|G5|G6|G7|G8|G9|G10
-------------------------------
|H1|H2|H3|H4|H5|H6|H7|H8|H9|H10
-------------------------------
|I1|I2|I3|I4|I5|I6|I7|I8|I9|I10
-------------------------------
|J1|J2|J3|J4|J5|J6|J7|J8|J9|J10
-------------------------------
X = HIT M = MISS

Small Ship:
<|D4||D5|]

Medium Ship:
<|F4||G4||H4|]

Large Ship:
<|I5||I6||I7||I8|]

Please Wait for other player...
Let's begin.
It is currently your turn.
```

```
          ***Radar***
|A1|A2|A3|A4|A5|A6|A7|A8|A9|A10
-------------------------------
|B1|B2|B3|B4|B5|B6|B7|B8|B9|B10
-------------------------------
|C1|C2|C3|C4|C5|C6|C7|C8|C9|C10
-------------------------------
|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10
-------------------------------
|E1|E2|E3|E4|E5|E6|E7|E8|E9|E10
-------------------------------
|F1|F2|F3|F4|F5|F6|F7|F8|F9|F10
-------------------------------
|G1|G2|G3|G4|G5|G6|G7|G8|G9|G10
-------------------------------
|H1|H2|H3|H4|H5|H6|H7|H8|H9|H10
-------------------------------
|I1|I2|I3|I4|I5|I6|I7|I8|I9|I10
-------------------------------
|J1|J2|J3|J4|J5|J6|J7|J8|J9|J10
-------------------------------
X = HIT M = MISS

Small Ship:
<|C2||C3|]

Medium Ship:
<|F5||G5||H5|]

Large Ship:
<|B8||C8||D8||E8|]

Please Wait for other player...
Let's begin.
```

```
136   //start game loop
137   while(!gameOver){
138
139       try{
140           received = dataIn.readUTF();
141           received = received.toUpperCase();
142
143           miss = true;
144           //only the current player can send coordinates, all other input is ignored by the server
145           if(Game.currentPlayer == this.player) {
146               for(PlayerHandler c : Game.playerList) {
147                   c.dataOut.writeUTF(this.player + " fires torpedo to coordinate " + received);
148
149                   if(Game.currentPlayer != c.player) {
150                       //check the coordinate against the player's ship coordinates
151                       for(int i = 0; i < c.shipCoordinates.size(); i++) {
152                           //if we get a match > hit
153
154                           if(c.shipCoordinates.get(i).equals(received)) {
155                               c.shipCoordinates.remove(i);
```

(Game.java; inside the PlayerHandler class)

The game server threads listen for messages from the player clients.

```
159   //hit flag for player class
160   if(this.player == Game.Players.PLAYER1) {
161       Game.playerList.get(1).dataOut.writeUTF("HIT" + received); //inform player that has been hit
162       Game.playerList.get(0).dataOut.writeUTF("SET_TO_RADAR" + received); //inform player that hit their opponent to mark it on the radar
163   } else {
164       Game.playerList.get(0).dataOut.writeUTF("HIT" + received);
165       Game.playerList.get(1).dataOut.writeUTF("SET_TO_RADAR" + received);
166   }
167
168   for(PlayerHandler p : Game.playerList) {
169       p.dataOut.writeUTF("PRINT_BOARD");
170       p.dataOut.writeUTF("BOOOM!! " + this.name + " got a hit.");
171   }
172
173   //check for game over
174   if(c.shipCoordinates.isEmpty())
175       gameOver = true;
176
```

(Game.java)

Series of messages that can be sent to the player clients depending on what coordinates are received from the player clients.

```
123         //create readMessage thread
124         Thread readMessage = new Thread(new Runnable(){
125             @Override
126             public void run(){
127                 //while the game isn't over
128                 while(!gameOver){
129 
130                     try{
131                         //read the message sent to this client
132                         String msg = dataIn.readUTF();
133 
134                         if(msg.length() > 3 && msg.substring(0, 3).equals("HIT")) {
135                             //player took a hit. Update the ship that was hit.
136                             //check the rest of the string for the coordinate to update/remove
137                             playerBoard.updateShips(msg.substring(3));
138 
139                         } else if(msg.equals("GAMEOVER")){
140                             //end of game, close socket
141                             gameOver = true;
142                             System.out.println("Game over!");
143                             msg = dataIn.readUTF();
144                             if(msg.equals("WINNER"))
145                                 System.out.println(name + " is the winner!!");
146                             else
147                                 System.out.println(name + " is the loser...");
148                         } else if(msg.length() > 12 && msg.substring(0, 12).equals("SET_TO_RADAR")) {
149                             //updates board where the hit took place
150                             playerBoard.updateBoard(msg.substring(12));
151                         } else if(msg.equals("PRINT_BOARD")) {
152                             //informs player client to print board
153                             System.out.println(playerBoard);
154                         } else if(msg.length() > 4 && msg.substring(0, 4).equals("MISS")) {
155                             //updates board where the miss occured
156                             playerBoard.updateMiss(msg.substring(4));
157                         } else
158                             System.out.println(msg);
```

(Player.java)

Code description: Player client has a number of messages that it can read from the server. Depending on the message, the player client can update the board, ships, declare the winner or loser, and print information to the players.

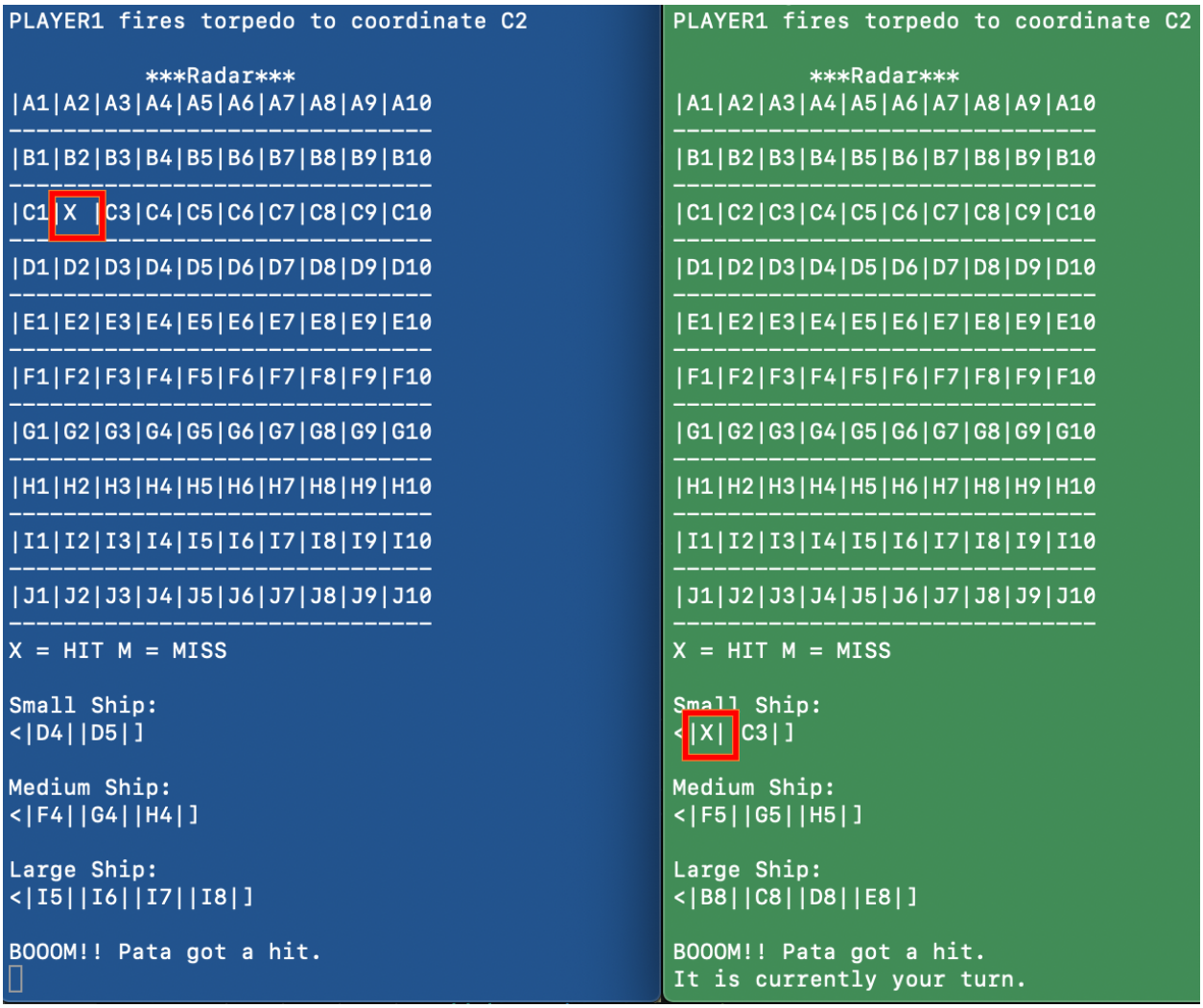
```
192         //informing player that is not currently picking the coordinates to chill
193     } else if(this.player == Game.Players.PLAYER1){
194         Game.playerList.get(0).dataOut.writeUTF("It is not currently your turn. Please wait.");
195     } else if(this.player == Game.Players.PLAYER2) {
196         Game.playerList.get(1).dataOut.writeUTF("It is not currently your turn. Please wait.");
197     }
```

(Game.java)

If a player client sends a message to the game server when it is not their turn that player is sent a message informing them it is not their turn.

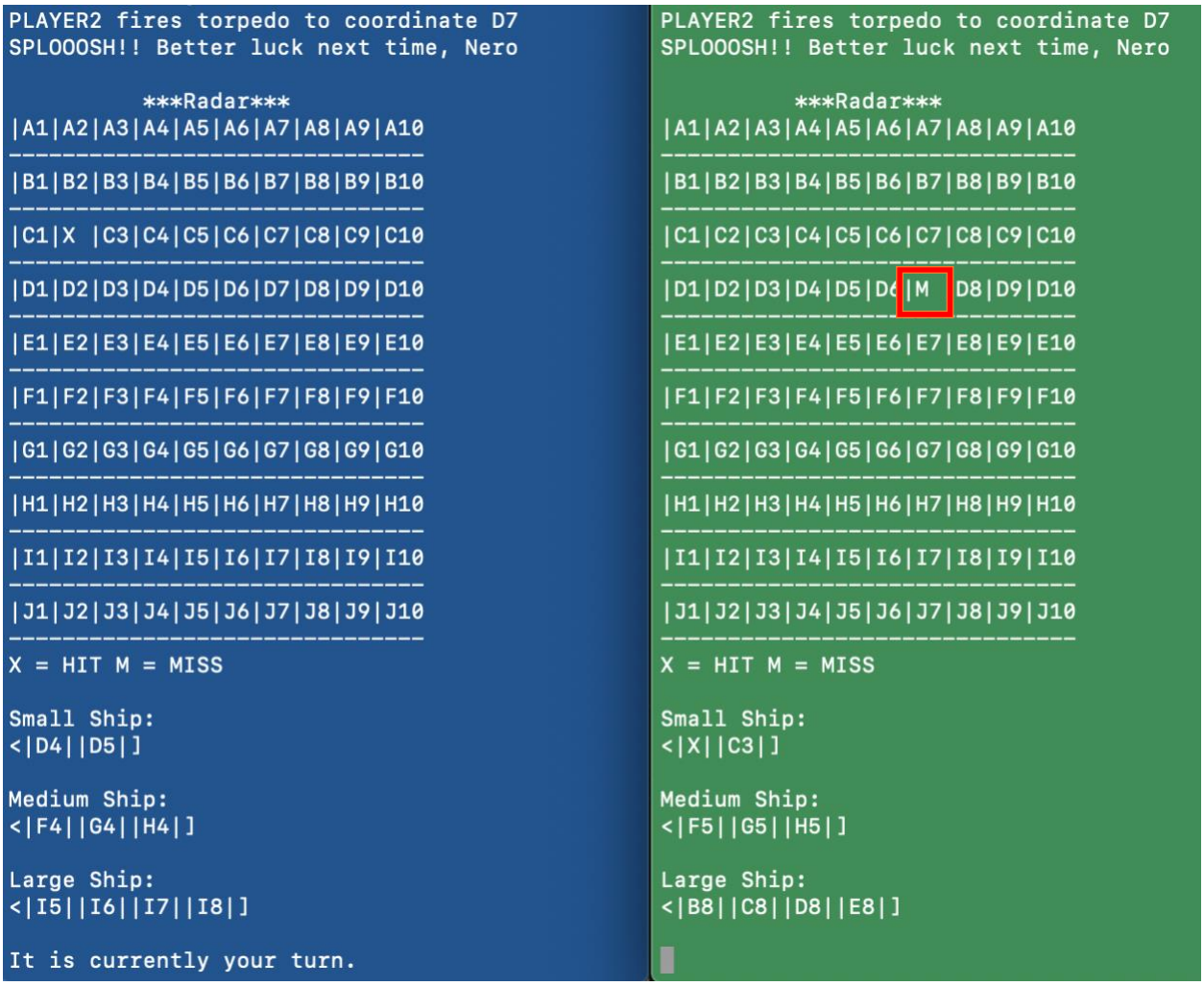
Players take turns trying to torpedo each other's ships.

If there is a hit:



The player that got the hit updates their board to keep track of the hits. The player that was hit has their ships updated to keep track of the ship that was hit.

If there is a miss:



The server sends a message to the client that missed to update their board. A message is printed informing the players of the miss, and the player that missed has their board updated with a M.

```
182     if(miss) {
183         //if miss. Players are informed and player that miss has their board updated with an M
184         for(PlayerHandler c : Game.playerList) {
185             c.dataOut.writeUTF("SPLOOOSH!! Better luck next time, " + this.name);
186             if(Game.currentPlayer == c.player)
187                 c.dataOut.writeUTF("MISS" + received);
188             c.dataOut.writeUTF("PRINT_BOARD");
189         }
190     }
```

(Game.java)

```java
    /**
     * Updates player's board in the event of a hit ship with a X
     * @param input     the coordinate that was hit
     * @return
     */
    protected boolean updateBoard(String input) {
        for(int i = 0; i < 10; i++) {
            for(int j = 0; j < 10; j++) {
                if(radar[i][j].equals(input)) {
                    radar[i][j] = "X ";
                    //TO DO: update ships
                    return true;
                }
            }
        }
        return false;
    }

    /**
     * Updates player's board in the event of a miss with a M
     * @param input     the coordinate where the miss occurred
     */
    protected void updateMiss(String input) {
        for(int i = 0; i < 10; i++) {
            for(int j = 0; j < 10; j++) {
                if(radar[i][j].equals(input)) {
                    radar[i][j] = "M ";

                }
            }
        }

    }
```

(Board.java)
Board functions that are used to update player client boards when misses and hits occur.

After each turn is taken the server changes the current player. This continues till a player has sunk the other player's battleships.

```
PLAYER1 fires torpedo to coordinate E8

          ***Radar***
|A1|A2|A3|A4|A5|A6|A7|A8|A9|A10
-------------------------------
|B1|B2|B3|B4|B5|B6|B7|X |B9|B10
-------------------------------
|C1|X |X |C4|C5|C6|C7|X |C9|C10
-------------------------------
|D1|D2|D3|D4|D5|D6|D7|X |D9|D10
-------------------------------
|E1|E2|E3|E4|E5|E6|E7|X |E9|E10
-------------------------------
|F1|F2|F3|F4|X |F6|F7|F8|F9|F10
-------------------------------
|G1|G2|G3|G4|X |G6|G7|G8|G9|G10
-------------------------------
|H1|H2|H3|H4|X |H6|H7|H8|H9|H10
-------------------------------
|I1|I2|I3|I4|I5|I6|I7|I8|I9|I10
-------------------------------
|J1|J2|J3|J4|J5|J6|J7|J8|J9|J10
-------------------------------
X = HIT M = MISS

Small Ship:
<|X||D5|]

Medium Ship:
<|F4||X||X|]

Large Ship:
<|I5||I6||I7||I8|]

BOOOM!! Pata got a hit.
Game over!
Pata is the winner!!
```

```
PLAYER1 fires torpedo to coordinate E8

          ***Radar***
|A1|A2|A3|A4|A5|A6|A7|A8|A9|A10
-------------------------------
|B1|B2|B3|B4|B5|B6|B7|B8|B9|B10
-------------------------------
|C1|M |C3|C4|C5|C6|C7|C8|C9|C10
-------------------------------
|D1|D2|D3|X |D5|D6|M |D8|D9|D10
-------------------------------
|E1|E2|E3|E4|E5|E6|E7|E8|E9|E10
-------------------------------
|F1|F2|F3|F4|F5|F6|F7|F8|F9|F10
-------------------------------
|G1|G2|G3|X |G5|G6|G7|G8|G9|G10
-------------------------------
|H1|H2|H3|X |H5|H6|H7|M |H9|H10
-------------------------------
|I1|I2|I3|I4|I5|I6|I7|I8|I9|I10
-------------------------------
|J1|J2|J3|J4|J5|J6|J7|J8|J9|J10
-------------------------------
X = HIT M = MISS

Small Ship:
<|X||X|]

Medium Ship:
<|X||X||X|]

Large Ship:
<|X||X||X||X|]

BOOOM!! Pata got a hit.
It is currently your turn.
Game over!
Nero is the loser...
```

Once a player has all their battleships sunk the game is over. Players are told who won and who lost.

```
217     //send game over flag to player clients
218     try {
219         for(PlayerHandler c : Game.playerList) {
220             c.dataOut.writeUTF("GAMEOVER");
221             if(c.shipCoordinates.isEmpty())
222                 c.dataOut.writeUTF("LOSER");
223             else
224                 c.dataOut.writeUTF("WINNER");
225         }
```

When the server detects a player's ships have all been sunk (removed from shipCoordinates), a "GAMEOVER" message is sent to the player clients along with a "WINNER" OR "LOSER" message. The player clients print the appropriate winner and loser on their respective screens.