FINAL PROJECT #CSA | *BATTLESHIP*

# Computer System and Networking LAB

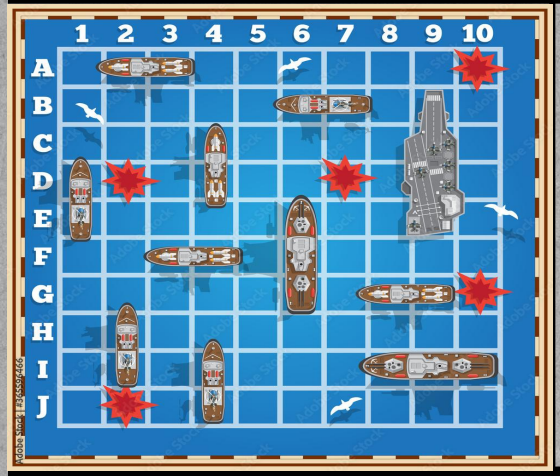
[CSA]

# *BATTLESHIP*

## *GROUP 1*

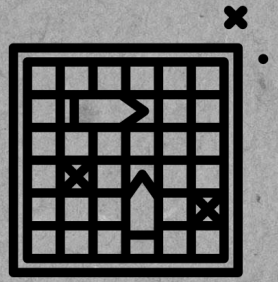
Date_ 23/11/2022


Reynard Pradhitya - 21/472680/PA/20321
Reza Aurelio Brilliansah - 21/475039/PA/20515
Ahmad Fadhil Bukhori - 21/475083/PA/20525



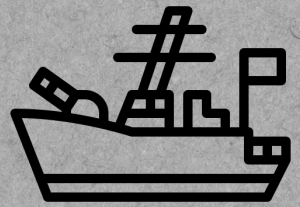

# PROJECT *BATTLESHIP*

Battleship is already widely known by people an the rules are considered quite simple. We believe that most of us are already familiar with this game. We want our project to be enjoyed by people  in general. That is why we chose battleship to be the main theme of our project.



The main goal of our project is to recreate the Battleship board game using socket programming. The objective of the game is to sink the enemy’s battleships before they sink yours by inputting the correct coordinates of the enemy ships.

This program allows a player to create a battleship game server so that another player could join the game as a client by using the correct IP.  The boards for the battleships are printed to the console and locations of the battleships are randomized.



# HOW DOES *IT WORK*

**01**

Player 1 starts the server and hosts a game, and player 2 establishes a connection to the server

**02**

The program creates two grids for each player, one for their own board, and one for the enemy board

**03**

The program randomizes the locations of the battleships based on how many battleships are selected and displays them on the grid displaying their own grid. At the start, the enemy board grid is empty

**04**

Players take turns guessing the locations of the enemy battleships by inputting coordinates until one side has no battleships remaining. When a ship is hit, the number of boats for that player decreases while the attacking player increases his score

**05**

When a player’s board has no more battleships remaining, the user is presented with a ‘defeat’ screen. A player that has a score equal to the amount of boats  will display a ‘victory’ screen

# INSPIRATIONS

We are aware that Battleship is a very popular game and there are a lot of people who have coded their own versions of battleships.

Our main reference was:
***https://github.com/AndreiGhenoiu/Java-Battleships***

As the general mechanic mimics the real game, there are many similarities to other versions available online, but we made sure that our program is purely made from scratch (No ctrl c ctrl v)

# OUR OWN *MODIFICATIONS*

| | | | |
|---|---|---|---|
| 01 | Boats are randomly generated by the program instead of the player inputting the location of the ships | 02 | We use our own logic to implement checking if a coordinate is a hit as the player board used a matrix of boolean values |
| 03 | Modified the losing condition from checking the board for remaining ships to keeping track of the number of ships alive | 04 | Different turn based system between players and IP usage for device-to-device connection in receiving and sending messages |

# 00!

# BATTLESHIP *CODE*

*SCREENSHOT*

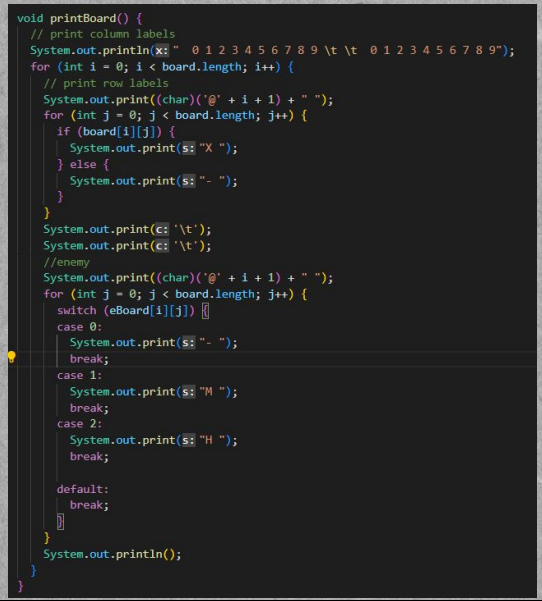
*BRIEF EXPLANATION*

Date_ 23/11/2022

# CODE *EXPLANATIONS*

Print Player Board using a simple for loop. We used an if statement to check if there is a boat

Print Enemy Board using a for loop at the side of the player board. We used a switch to print a hit, miss, or untouched



```
void printBoard() {
  // print column labels
  System.out.println(x: "  0 1 2 3 4 5 6 7 8 9 \t \t  0 1 2 3 4 5 6 7 8 9");
  for (int i = 0; i < board.length; i++) {
    // print row labels
    System.out.print((char)('@' + i + 1) + " ");
    for (int j = 0; j < board.length; j++) {
      if (board[i][j]) {
        System.out.print(s: "X ");
      } else {
        System.out.print(s: "- ");
      }
    }
    System.out.print(c: '\t');
    System.out.print(c: '\t');
    //enemy
    System.out.print((char)('@' + i + 1) + " ");
    for (int j = 0; j < board.length; j++) {
      switch (eBoard[i][j]) {
      case 0:
        System.out.print(s: "- ");
        break;
      case 1:
        System.out.print(s: "M ");
        break;
      case 2:
        System.out.print(s: "H ");
        break;

      default:
        break;
      }
    }
    System.out.println();
  }
}
```

# CODE *EXPLANATIONS*

```
// create random boats
void generateBoats(int num) {
  boats = num;
  int curNum = 0;
  while (curNum < num) {
    // get random location
    Random random = new Random();
    int posx = random.nextInt(bound: 10);
    int posy = random.nextInt(bound: 10);
    // check if its empty
    if (!board[posx][posy]) {
      // if yes, add ship there
      board[posx][posy] = true;
      curNum++;
    }
    // if its not, repeat without increasing ship number
  }
}
```

Randomized the ships location by generating random numbers for x and y axis. We made sure that the boats are not in the same place

# CODE *EXPLANATIONS*

Check if enemy coordinates are a hit . Since the boat locations are stored in a boolean matrix, we can check the truth value of the coordinates

Modify the enemy board matrix based on whether our coordinates are a hit or not

```
boolean checkhit(int x, int y) {
  // function to check if coordinates are a hit
  boolean isHit = board[x][y];
  if (isHit) {
    boats--;
    System.out.println(x: "\nYour ship has been Hit");
  } else {
    System.out.println(x: "\nYour ship has been Missed");
  }
  return (isHit);
}

void returnHit(int x, int y, boolean isHit) {
  if (isHit) {
    eBoard[x][y] = 2;
  } else {
    eBoard[x][y] = 1;
  }
}
```

# CODE *EXPLANATIONS*

```
try {
  // establishing connection
  System.out.println(x: "Waiting for Other Player...");
  ServerSocket ss = new ServerSocket(port: 9806);
  Socket soc = ss.accept();
  System.out.println(x: "Connection established");

  BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in)); // get user input
  PrintWriter out = new PrintWriter(soc.getOutputStream(), autoFlush: true); // output to other player
  BufferedReader in = new BufferedReader(new InputStreamReader(soc.getInputStream())); // input from other player

  //get number of boats
  System.out.println(x: "Enter the number of ships");
  int boatsNumber =  Integer.parseInt(userInput.readLine());
  out.println(boatsNumber);

  playerBoard.generateBoats(boatsNumber);

  System.out.println(x: "\nPlayer Board \t\t\tEnemyBoard: ");
  playerBoard.printBoard();
```

Creating a connection by opening a server socket. We also made a buffered readers and print writers to communicate with the other client

Inputting number of ships and then sending it to the client. Then generate the player board.

# CODE *EXPLANATIONS*

Connecting with the host by inputting their IP address.

The client receives a message from the server containing the amount of boats. The client parses it into a string and generates the required boats

```
Scanner input = new Scanner(System.in);
System.out.println(x: "Enter the Server IP Address");
String ipaddress = input.nextLine();

try {
  //connect to host
  System.out.println(x: "Client Started");
  Socket soc = new Socket(ipaddress, port: 9806);

  BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
  PrintWriter out = new PrintWriter(soc.getOutputStream(), autoFlush: true);
  BufferedReader in = new BufferedReader(new InputStreamReader(soc.getInputStream()));

  //get number of boats
  System.out.println(x: "Waiting for host to set-up game");
  int boatsNumber = Integer.parseInt(in.readLine());
  System.out.println(boatsNumber);
  turn = 2;

  playerBoard.generateBoats(boatsNumber);

  //print boards
  System.out.println(x: "\nPlayer Board \t\t\tEnemyBoard: ");
  playerBoard.printBoard();
```

# CODE *EXPLANATIONS*

```
while (turn > 0) {
  // turn 0 = win/lose, turn 1 = player 1, turn 2 = player 2
  if (turn == 1) {
    // input coordinate
    System.out.println(x: "Enter Coordinates:");
```

The program runs in a while loop that is true as long as the turn is over 0. When turn = 1, the player attacks, and when the turn=2, the player receives an attack. When the game ends, the score is set to 0, ending the loop. The host is set to always have the first turn

# CODE *EXPLANATIONS*

Checking the input value whether it is valid or not by checking the coordinates value

```
while (!validInput) {
  //2 characters only
  if (coords.length() == 2 &&
    (coords.charAt(index: 0) >= 65 && coords.charAt(index: 0) <= 74) &&
    (coords.charAt(index: 1) >= 48 && coords.charAt(index: 1) <= 57)) {
    String y = coords.charAt(index: 1) + "";
    xcoord = (int) coords.charAt(index: 0) - 65;
    ycoord = Integer.parseInt(y);
    validInput = true;
  } else {
    coords = userInput.readLine();
  }
}
```

# CODE *EXPLANATIONS*

```
//send coordinates to client
System.out.println("Shooting at: (" + coords.charAt(index: 0) + "," + ycoord + ")");
out.println(coords);

//receive response
String hitMiss = in.readLine();
boolean isHit = true;
clearScreen();

//convert string to boolean
if (hitMiss.equals(anObject: "false")) {
  isHit = false;
} else if (hitMiss.equals(anObject: "true")) {
  isHit = true;
  playerBoard.addscore();
}
```

Output the coordinates to the other player and wait for a response containing hit or miss, then convert the message to a boolean

# CODE *EXPLANATIONS*

Using the returnHit function we update the enemy board with a hit or miss on the coordinate. Then we check for our win condition, and if so, we show the victory match result. Otherwise, we pass the turn

```
playerBoard.returnHit(xcoord, ycoord, isHit);

//check if win
if (playerBoard.score == boatsNumber) {
  clearScreen();
  playerBoard.matchResult(hasLost: false);
  turn = 0;
} else {
  //print enemy boards again
  System.out.println(x: "\nPlayer Board \t\t\tEnemyBoard: ");
  playerBoard.printBoard();

  //end turn
  turn = 2;
}
```

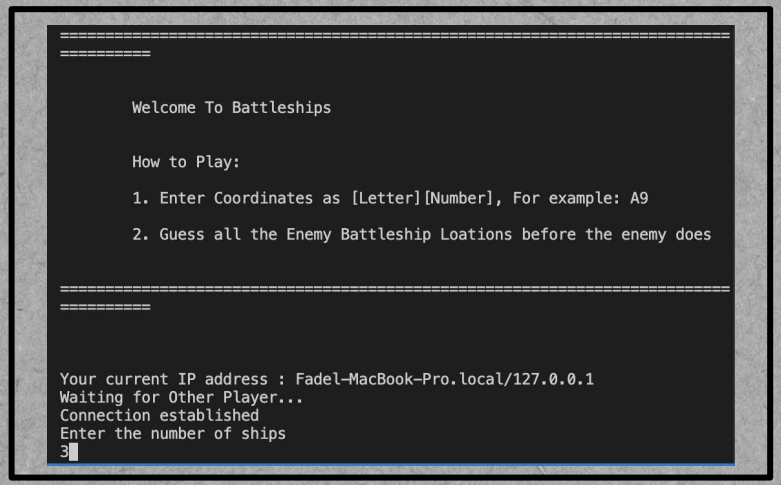
# CODE *EXPLANATIONS*

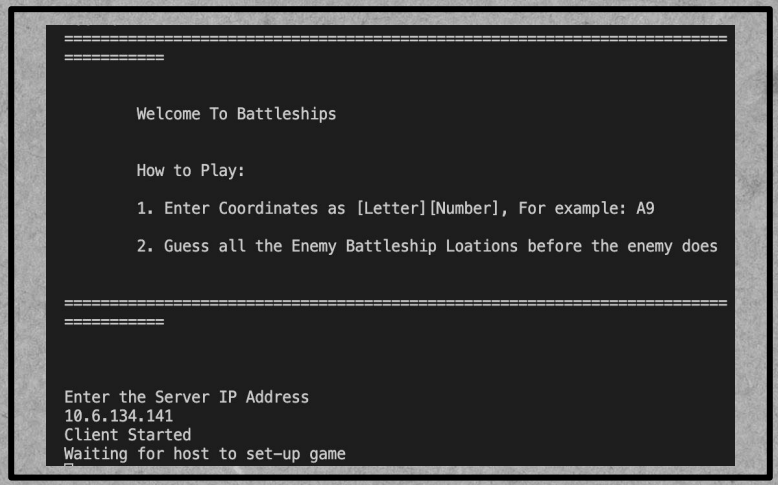
```
} else if (turn == 2) {
  //send waiting message
  System.out.println(x: "Waiting for other player...");

  // receive coordinates from player 1
  String coords = in .readLine();

  //convert string to x,y coordinates
  int xcoord = (int) coords.charAt(index: 0) - 65;
  int ycoord = Integer.parseInt(coords.charAt(index: 1) + "");

  System.out.println("Enemy fired at: (" + coords.charAt(index: 0) + "," + ycoord + ")");
  boolean isHit = playerBoard.checkhit(xcoord, ycoord);

  //send to player 1
  out.println(isHit);

  //check if lose
  if (playerBoard.boats == 0) {
    playerBoard.matchResult(hasLost: true);
    turn = 0;
  } else {
    //end turn
    turn = 1;
  }
```

This part shows the receiving attack part. We receive the coordinates from the other player. We extract the first character and convert the character into an integer and we parse the second character which is a number. Then check if it is a hit and miss. Then we send this to the other player

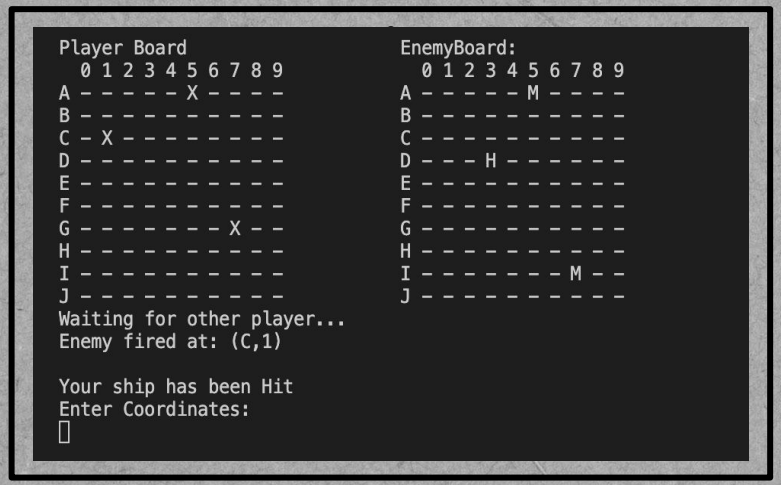
Check if we have any boats remaining, if we don’t then it is the player’s defeat, otherwise, pass the turn

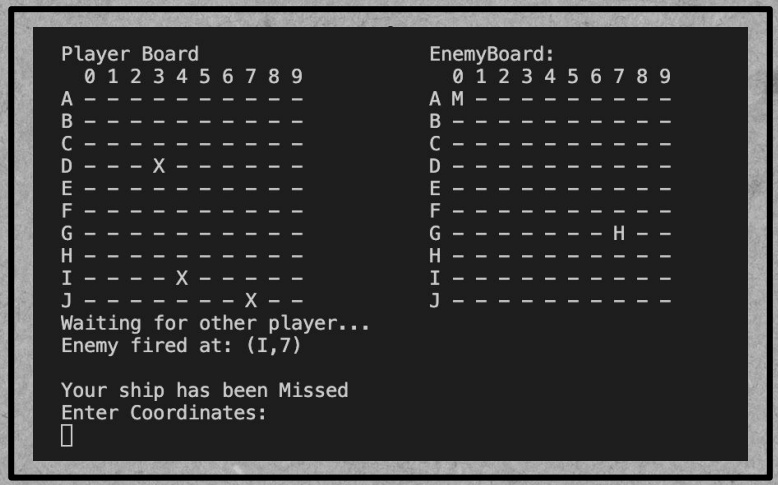
# RUNNING *THE CODE*



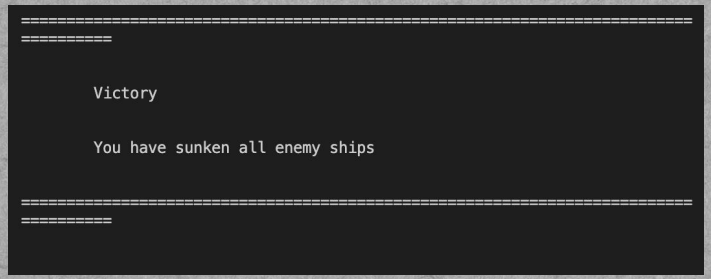



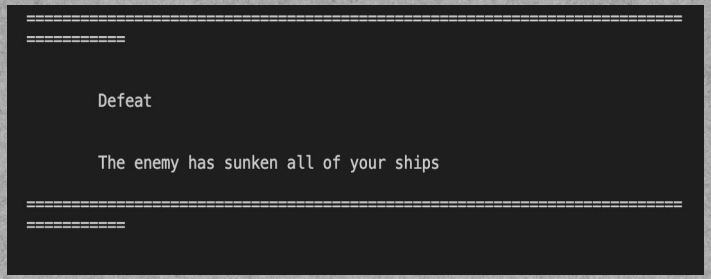
# RUNNING *THE CODE*



# RUNNING *THE CODE*




THANKS !