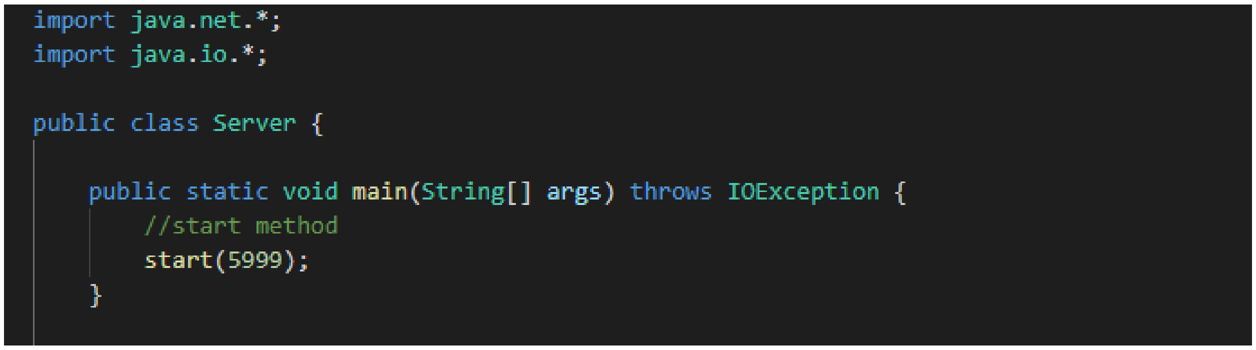
Charlie Le

4311 Networking and Telecomms

2 May 2022

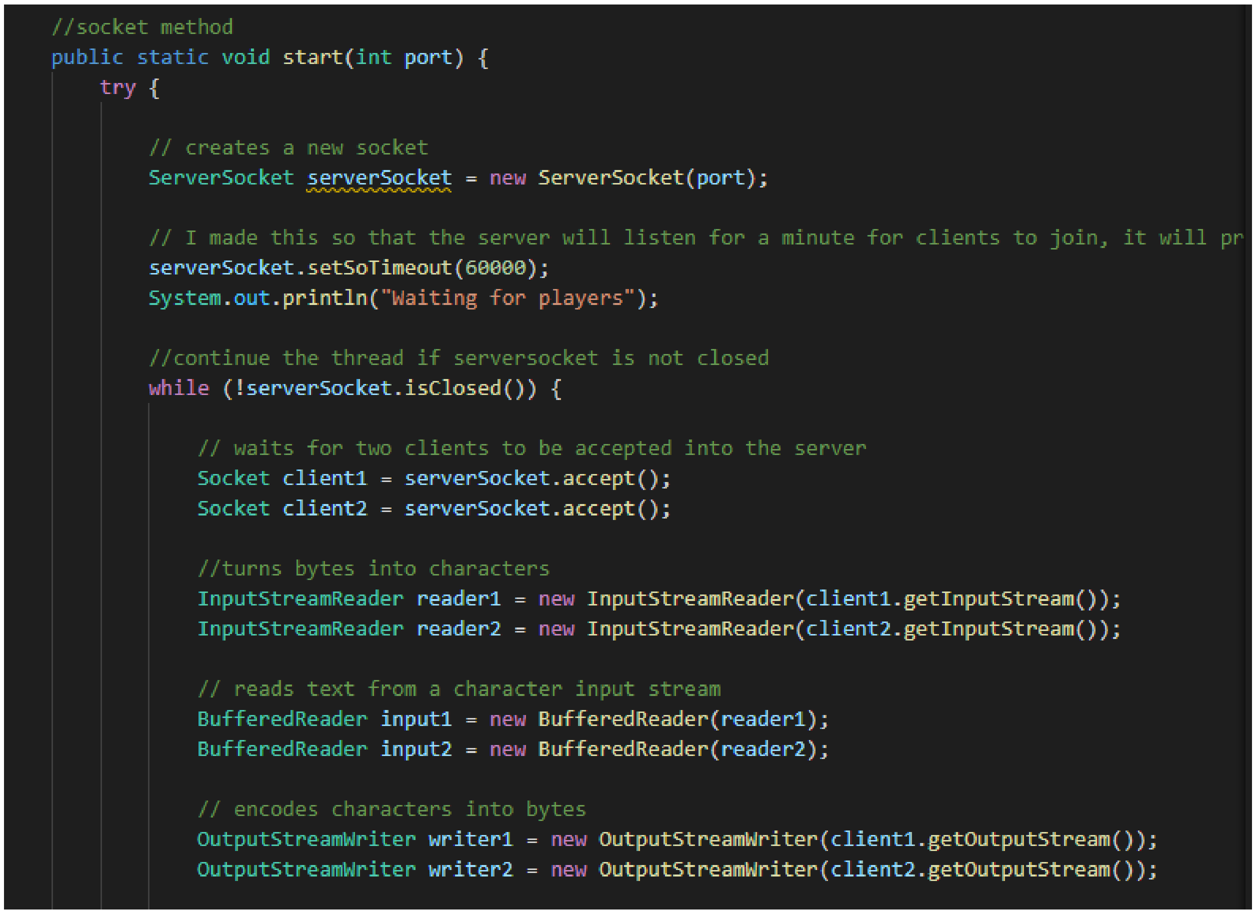
Programming Assignment 2

I created a rock paper scissors game. It is a multithreaded server that is based on server- client architecture.

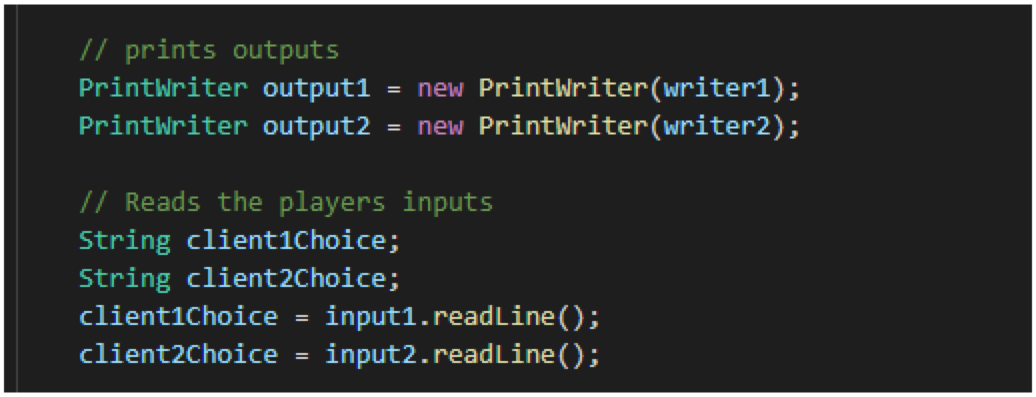


```java
import java.net.*;
import java.io.*;

public class Server {

    public static void main(String[] args) throws IOException {
        //start method
        start(5999);
    }
```

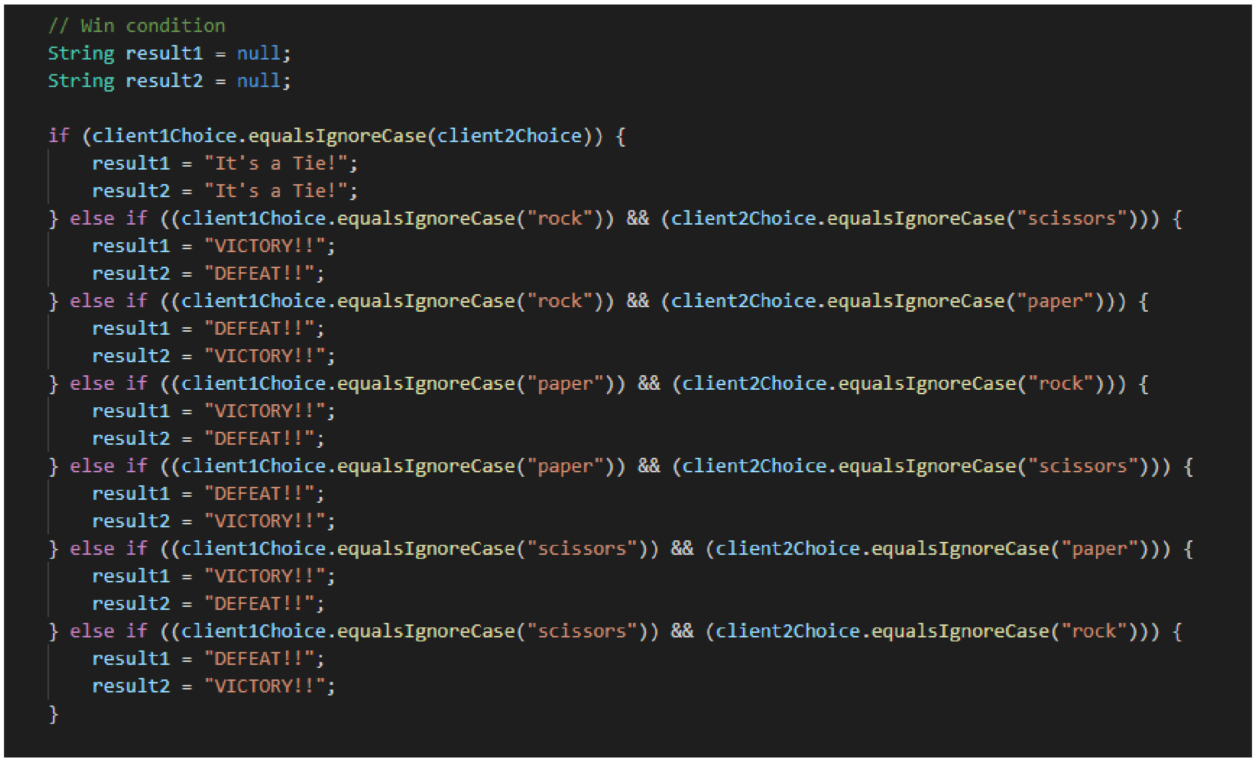
In the beginning of my Server class I have a main method that hosts a port for clients.



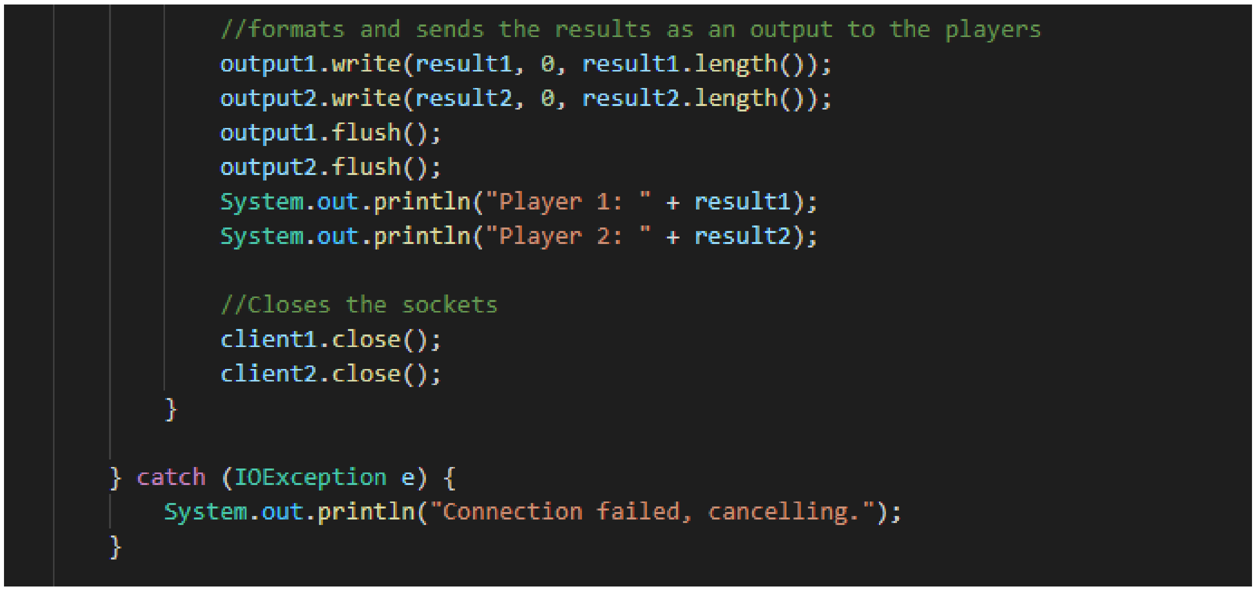
```java
    //socket method
    public static void start(int port) {
        try {

            // creates a new socket
            ServerSocket serverSocket = new ServerSocket(port);

            // I made this so that the server will listen for a minute for clients to join, it will pr
            serverSocket.setSoTimeout(60000);
            System.out.println("Waiting for players");

            //continue the thread if serversocket is not closed
            while (!serverSocket.isClosed()) {

                // waits for two clients to be accepted into the server
                Socket client1 = serverSocket.accept();
                Socket client2 = serverSocket.accept();

                //turns bytes into characters
                InputStreamReader reader1 = new InputStreamReader(client1.getInputStream());
                InputStreamReader reader2 = new InputStreamReader(client2.getInputStream());

                // reads text from a character input stream
                BufferedReader input1 = new BufferedReader(reader1);
                BufferedReader input2 = new BufferedReader(reader2);

                // encodes characters into bytes
                OutputStreamWriter writer1 = new OutputStreamWriter(client1.getOutputStream());
                OutputStreamWriter writer2 = new OutputStreamWriter(client2.getOutputStream());
```


```java
// prints outputs
PrintWriter output1 = new PrintWriter(writer1);
PrintWriter output2 = new PrintWriter(writer2);

// Reads the players inputs
String client1Choice;
String client2Choice;
client1Choice = input1.readLine();
client2Choice = input2.readLine();
```

The start method creates a multi thread. It creates new socket and sets timer so for players to join. While the serverSocket is open, it will accept two clients. I used an InputStreamReader to turn bytes into characters, a BufferedReader to read texts from input streams, an OutputStreamWriter to turn characters into bytes, and a print writer to print outputs.

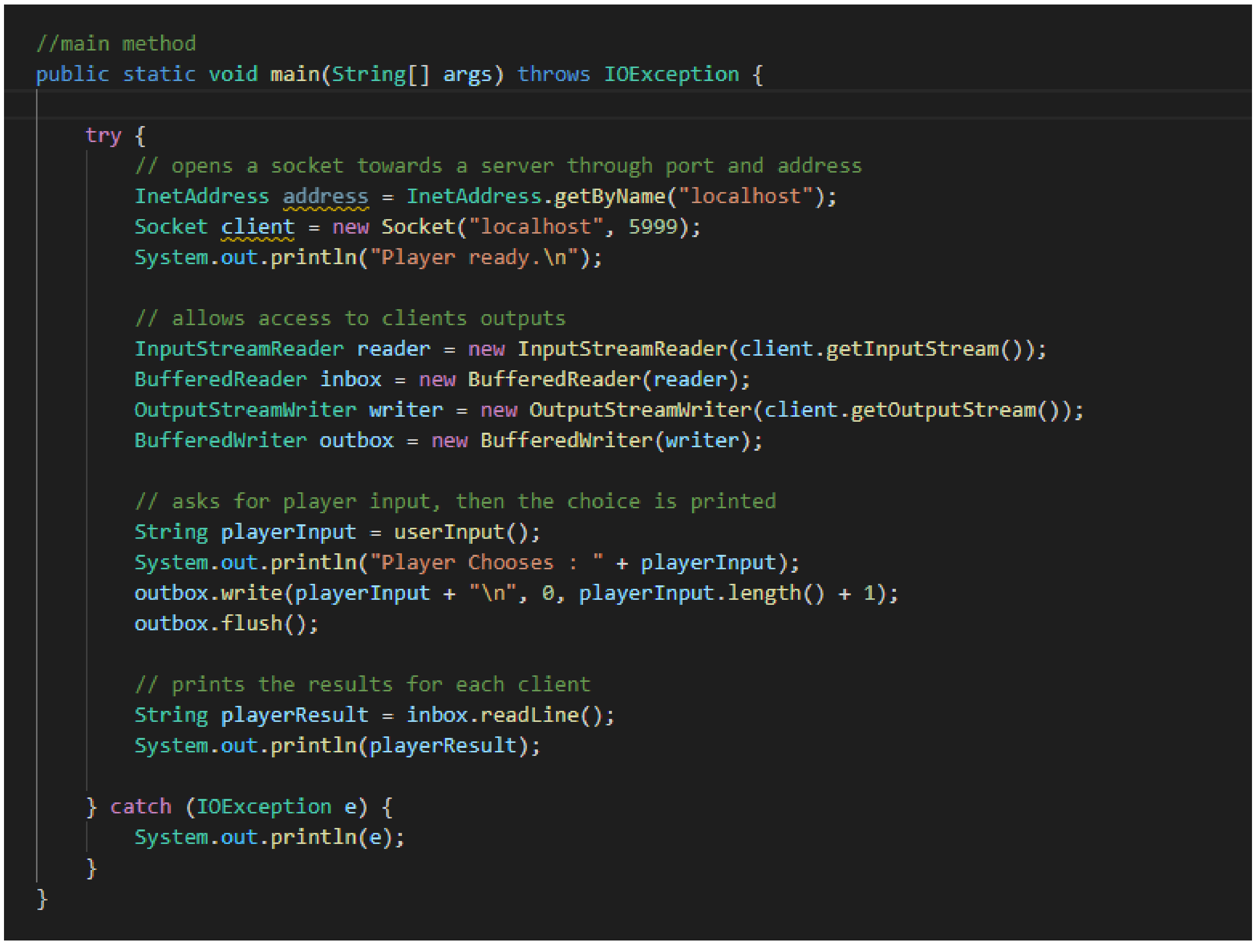

```java
// Win condition
String result1 = null;
String result2 = null;

if (client1Choice.equalsIgnoreCase(client2Choice)) {
    result1 = "It's a Tie!";
    result2 = "It's a Tie!";
} else if ((client1Choice.equalsIgnoreCase("rock")) && (client2Choice.equalsIgnoreCase("scissors"))) {
    result1 = "VICTORY!!";
    result2 = "DEFEAT!!";
} else if ((client1Choice.equalsIgnoreCase("rock")) && (client2Choice.equalsIgnoreCase("paper"))) {
    result1 = "DEFEAT!!";
    result2 = "VICTORY!!";
} else if ((client1Choice.equalsIgnoreCase("paper")) && (client2Choice.equalsIgnoreCase("rock"))) {
    result1 = "VICTORY!!";
    result2 = "DEFEAT!!";
} else if ((client1Choice.equalsIgnoreCase("paper")) && (client2Choice.equalsIgnoreCase("scissors"))) {
    result1 = "DEFEAT!!";
    result2 = "VICTORY!!";
} else if ((client1Choice.equalsIgnoreCase("scissors")) && (client2Choice.equalsIgnoreCase("paper"))) {
    result1 = "VICTORY!!";
    result2 = "DEFEAT!!";
} else if ((client1Choice.equalsIgnoreCase("scissors")) && (client2Choice.equalsIgnoreCase("rock"))) {
    result1 = "DEFEAT!!";
    result2 = "VICTORY!!";
}
```

I created a win condition for all the scenarios in the rock, paper, scissors game. There is also tie conditions. I used .equalsIgnoreCase so that client's inputs won't be case sensitive.



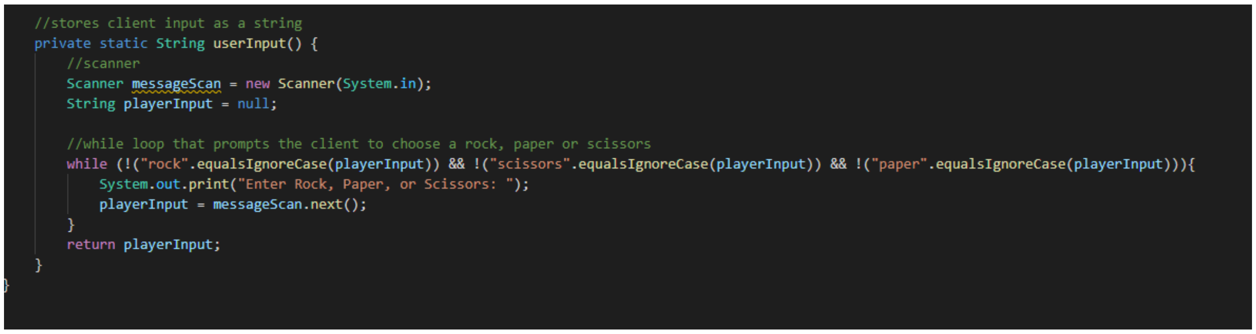
```java
                //formats and sends the results as an output to the players
                output1.write(result1, 0, result1.length());
                output2.write(result2, 0, result2.length());
                output1.flush();
                output2.flush();
                System.out.println("Player 1: " + result1);
                System.out.println("Player 2: " + result2);

                //Closes the sockets
                client1.close();
                client2.close();
            }

        } catch (IOException e) {
            System.out.println("Connection failed, cancelling.");
        }
```

This shows the clients the results of the game and closes the sockets.



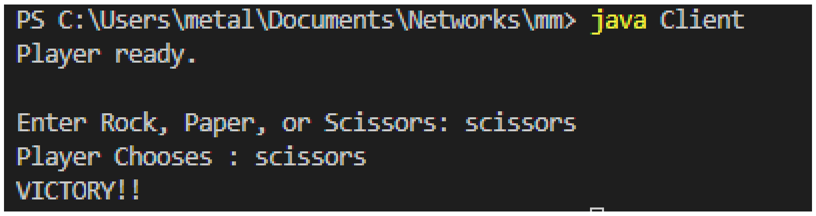
```java
//main method
public static void main(String[] args) throws IOException {

    try {
        // opens a socket towards a server through port and address
        InetAddress address = InetAddress.getByName("localhost");
        Socket client = new Socket("localhost", 5999);
        System.out.println("Player ready.\n");

        // allows access to clients outputs
        InputStreamReader reader = new InputStreamReader(client.getInputStream());
        BufferedReader inbox = new BufferedReader(reader);
        OutputStreamWriter writer = new OutputStreamWriter(client.getOutputStream());
        BufferedWriter outbox = new BufferedWriter(writer);

        // asks for player input, then the choice is printed
        String playerInput = userInput();
        System.out.println("Player Chooses : " + playerInput);
        outbox.write(playerInput + "\n", 0, playerInput.length() + 1);
        outbox.flush();

        // prints the results for each client
        String playerResult = inbox.readLine();
        System.out.println(playerResult);

    } catch (IOException e) {
        System.out.println(e);
    }
}
```

I created a Client class that has a main method that allow clients to connect to a server through address and port. Clients are prompted for their input and the choices are sent off to the server. Then results are printed to the clients.

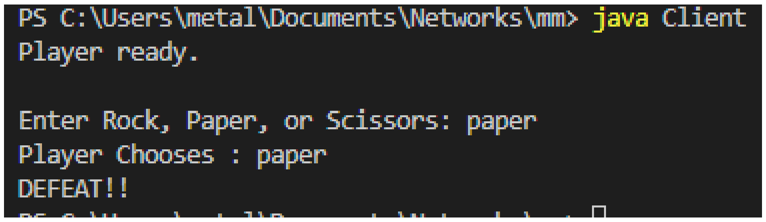


```java
    //stores client input as a string
    private static String userInput() {
        //scanner
        Scanner messageScan = new Scanner(System.in);
        String playerInput = null;

        //while loop that prompts the client to choose a rock, paper or scissors
        while (!("rock".equalsIgnoreCase(playerInput)) && !("scissors".equalsIgnoreCase(playerInput)) && !("paper".equalsIgnoreCase(playerInput))){
            System.out.print("Enter Rock, Paper, or Scissors: ");
            playerInput = messageScan.next();
        }
        return playerInput;
    }
}
```

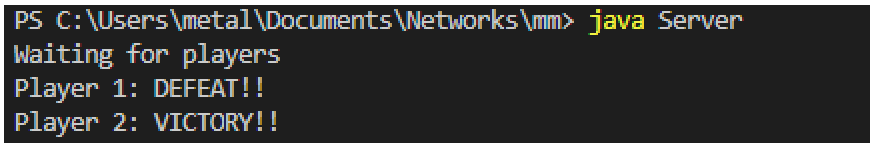
Lastly, I created a userInput method so that inputs can be stored as a string.



```
PS C:\Users\metal\Documents\Networks\mm> java Client
Player ready.

Enter Rock, Paper, or Scissors: scissors
Player Chooses : scissors
VICTORY!!
```



```
PS C:\Users\metal\Documents\Networks\mm> java Client
Player ready.

Enter Rock, Paper, or Scissors: paper
Player Chooses : paper
DEFEAT!!
```



```
PS C:\Users\metal\Documents\Networks\mm> java Server
Waiting for players
Player 1: DEFEAT!!
Player 2: VICTORY!!
```

Discussion

I would have liked to make it to where you can see usernames and maybe even add a tournament system for more than 2 clients. This was only made for two people. I could have also created another loop so that you could play again or do best out of multiple rounds.