Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-sockets-part-1-3-checkpoint/grade/mbh3

IT114-004-S2024 - [IT114] Sockets Part 1-3-Checkpoint

Submissions:

Submission Selection

1 Submission [active] 2/20/2024 2:59:28 PM

Instructions

COLLAPSE

- Create a new branch for this assignment
- Go through the socket lessons and get each part implemented (parts 1-3)
    - You'll probably want to put them into their own separate folders/packages (i.e., Part1, Part2, Part3) These are for your reference
    - Part 3, below, is what's necessary for this HW
    - https://github.com/MattToegel/IT114/tree/Module4/Module4/Part3
- Create a new folder called Part3HW (copy of Part3)
- Make sure you have all the necessary files from Part3 copied here and fix the package references at the top of each file
    - Add/commit/push the branch
    - Create a pull request to main and keep it open
- Implement **two** of the following **server-side** activities for all connected clients (majority of the logic should be processed server-side and broadcasted/sent to all clients if/when applicable)
    - Simple number guesser where all clients can attempt to guess while the game is active
        - Have a /start command that activates the game allowing guesses to be interpreted
        - Have a /stop command that deactivates the game, guesses will be treated as regular messages (i.e., guess messages are ignored)
        - Have a guess command that include a value that is processed to see if it matches the hidden number (i.e., */guess 5*)
            - Guess should only be considered when the game is active
            - The response should include who guessed, what they guessed, and whether or not it was correct (i.e., Bob guessed 5 but it was not correct)
        - No need to implement complexities like strikes
    - Coin toss command (random heads or tails)
        - Command should be something logical like /flip or /toss or /coin or similar
        - The result should mention *who* did *what* and got what *result* (i.e., Bob Flipped a coin and got heads)
    - Dice roller given a command and text format of "/roll #d#" (i.e., roll 2d6)
        - Command should be in the format of /roll #d# (i.e., roll 1d10)
        - The result should mention *who* did *what* and got what *result* (i.e., Bob rolled 1d10 and got 7)
    - Math game (server outputs a basic equation, first person to guess it correctly gets congratulated and a new equation is given)
        - Have a /start command that activates the game allowing equaiton to be answered
        - Have a /stop command that deactivates the game, answers will be treated as regular messages (i.e., any game related commands when stopped will be ignored)
        - Have an answer command that include a value that is processed to see if it matches the hidden number (i.e., */answer 15*)

the hidden number (i.e., */answer 15*)
- The response should include who answered, what they answered, and whether or not it was correct (i.e., Bob answered 5 but it was not correct)
- Private message (a client can send a message targetting another client where only the two can see the messages)
  - Command can be /pm, /dm followed by the user's name or an @ preceding the users name (clearly note which)
  - The server should properly check the target audience and send the response to the original sender and to the receiver (no one else should get the message)
  - Alternatively (make note if you do this and show evidence) you can add support to private message multiple people at once. Evidence should show a larger number of clients than the target list of the private message to show it works. Note to grader: if this is accomplished add 0.5 to total final grade on Canvas
- Message shuffler (randomizes the order of the characters of the given message)
  - Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)
  - The message should be sent to all clients showing it's from the user but randomized
    - Example: Bob types */command* hello and everyone recevies Bob: lleho

Fill in the below deliverables
Save the submission and generated output PDF
Add the PDF to the Part3HW folder (local)
Add/commit/push your changes
Merge the pull request
Upload the same PDF to Canvas

Branch name: M4-Sockets3-Homework

Tasks: 7 Points: 10.00

● **Baseline** (2 pts.)
^COLLAPSE^

● ^COLLAPSE^

Task #1 - Points: 1

Text: Demonstrate Baseline Code Working

Details:
This can be a single screenshot if everything fits, or can be multiple screenshots

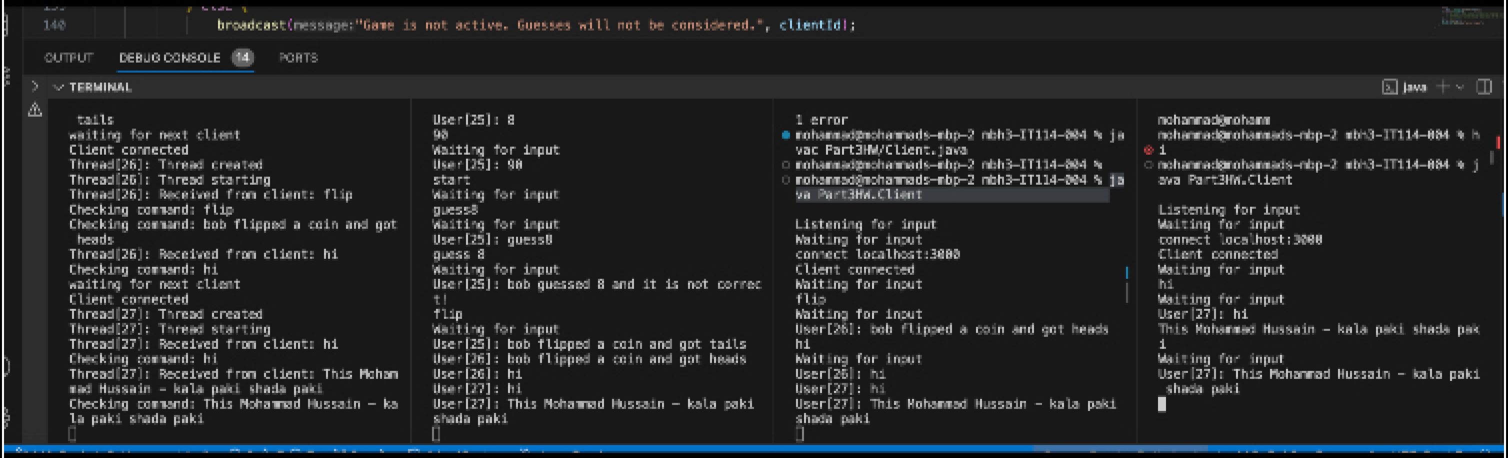
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Server terminal/instance is clearly shown/noted
#2	1	At least 3 client terminals should be visible and noted
#3	1	Each client should correctly receive all broadcasted/shared messages
#4	1	Captions clearly explain what each screenshot is showing
#5	1	Include a screenshot showing you grabbed Parts 1-3 correctly and have them in your repository alongside Part3HW

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
broadcast(message:"Game is not active. Guesses will not be considered.", clientId);
```

OUTPUT DEBUG CONSOLE 14 PORTS

TERMINAL

```
 tails
waiting for next client
Client connected
Thread[26]: Thread created
Thread[26]: Thread starting
Thread[26]: Received from client: flip
Checking command: flip
Checking command: bob flipped a coin and got
 heads
Thread[26]: Received from client: hi
Checking command: hi
waiting for next client
Client connected
Thread[27]: Thread created
Thread[27]: Thread starting
Thread[27]: Received from client: hi
Checking command: hi
Thread[27]: Received from client: This Mohan
mad Hussain - kala paki shada paki
Checking command: This Mohammad Hussain - ka
la paki shada paki
```

```
User[25]: 8
90
Waiting for input
User[25]: 90
start
Waiting for input
guess8
Waiting for input
User[25]: guess8
guess 8
Waiting for input
User[25]: bob guessed 8 and it is not correc
t!
flip
Waiting for input
User[25]: bob flipped a coin and got tails
User[26]: bob flipped a coin and got heads
User[26]: hi
User[27]: hi
User[27]: This Mohammad Hussain - kala paki
shada paki
```

```
1 error
mohammad@mohammads-mbp-2 mbh3-IT114-004 % ja
vac Part3HW/Client.java
mohammad@mohammads-mbp-2 mbh3-IT114-004 %
mohammad@mohammads-mbp-2 mbh3-IT114-004 % ja
va Part3HW.Client

Listening for input
Waiting for input
connect localhost:3000
Client connected
Waiting for input
flip
Waiting for input
User[26]: bob flipped a coin and got heads
hi
Waiting for input
User[26]: hi
User[27]: hi
User[27]: This Mohammad Hussain - kala paki
shada paki
```

```
mohammad@mohamm
mohammad@mohammads-mbp-2 mbh3-IT114-004 % h
i
mohammad@mohammads-mbp-2 mbh3-IT114-004 % j
ava Part3HW.Client

Listening for input
Waiting for input
connect localhost:3000
Client connected
Waiting for input
hi
Waiting for input
User[27]: hi
This Mohammad Hussain - kala paki shada pak
i
Waiting for input
User[27]: This Mohammad Hussain - kala paki
 shada paki
```

3 Terminal

Checklist Items (5)

#1 Server terminal/instance is clearly shown/noted

#2 At least 3 client terminals should be visible and noted

#3 Each client should correctly receive all broadcasted/shared messages

#4 Captions clearly explain what each screenshot is showing

#5 Include a screenshot showing you grabbed Parts 1-3 correctly and have them in your repository alongside Part3HW

Feature 1 (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: What feature did you pick? Briefly explain how you implemented it

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Feature is clearly stated (best to copy/paste it from above)
#2	1	Explanation sufficiently and concisely describes implementation (should be aligned with code snippets in related task)

Response:

I chose the coin toss feature in my game server. I've chosen to handle three commands: "flip", "toss", and "coin", each capable of initiating a virtual coin flip. Whenever I receive one of these commands, I first identify the user with the getClientName function to personalize the response. To determine the outcome of the flip—heads or tails—I use the Random class to generate a boolean value randomly; true corresponds to heads, and false to tails. This simple yet effective method ensures that each toss is entirely left to chance, mirroring the unpredictability of a real coin toss. Once the result is determined, I craft a message announcing the user's name along with the outcome of their coin flip and broadcast this message to the client. This feature adds an element of chance and fun, engaging users in a straightforward yet entertaining interaction with my server.

COLLAPSE

Task #2 - Points: 1

Text: Add screenshot(s) showing the implemented feature working (code and output)

Details:

Add screenshots of the relevant code changes AND relevant output during runtime

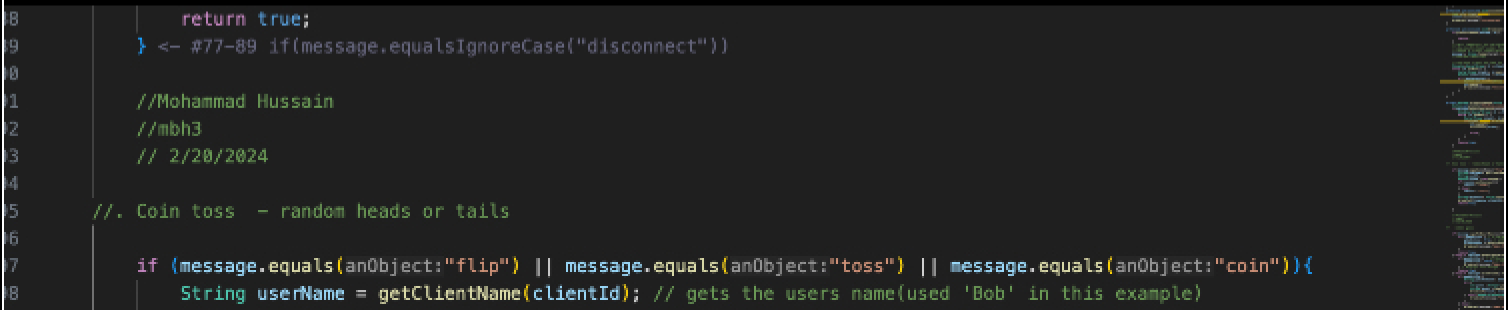
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Output is clearly shown and captioned
#2	1	Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

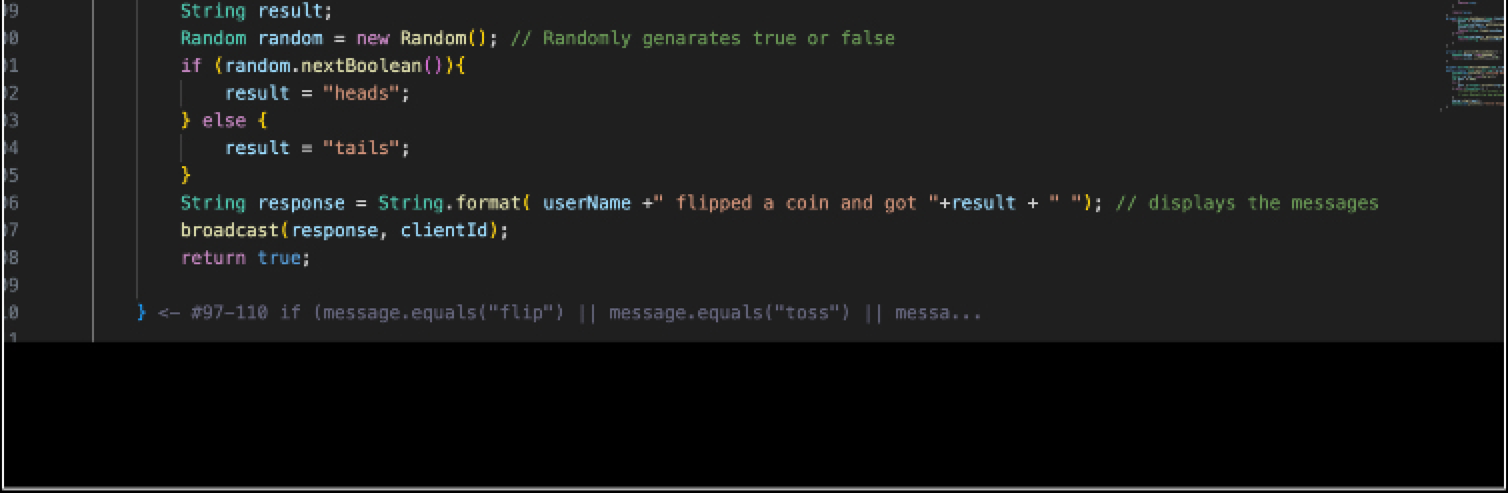
Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
            return true;
        } <- #77-89 if(message.equalsIgnoreCase("disconnect"))

        //Mohammad Hussain
        //mbh3
        // 2/20/2024

    //. Coin toss  - random heads or tails

        if (message.equals(anObject:"flip") || message.equals(anObject:"toss") || message.equals(anObject:"coin")){
            String userName = getClientName(clientId); // gets the users name(used 'Bob' in this example)
```

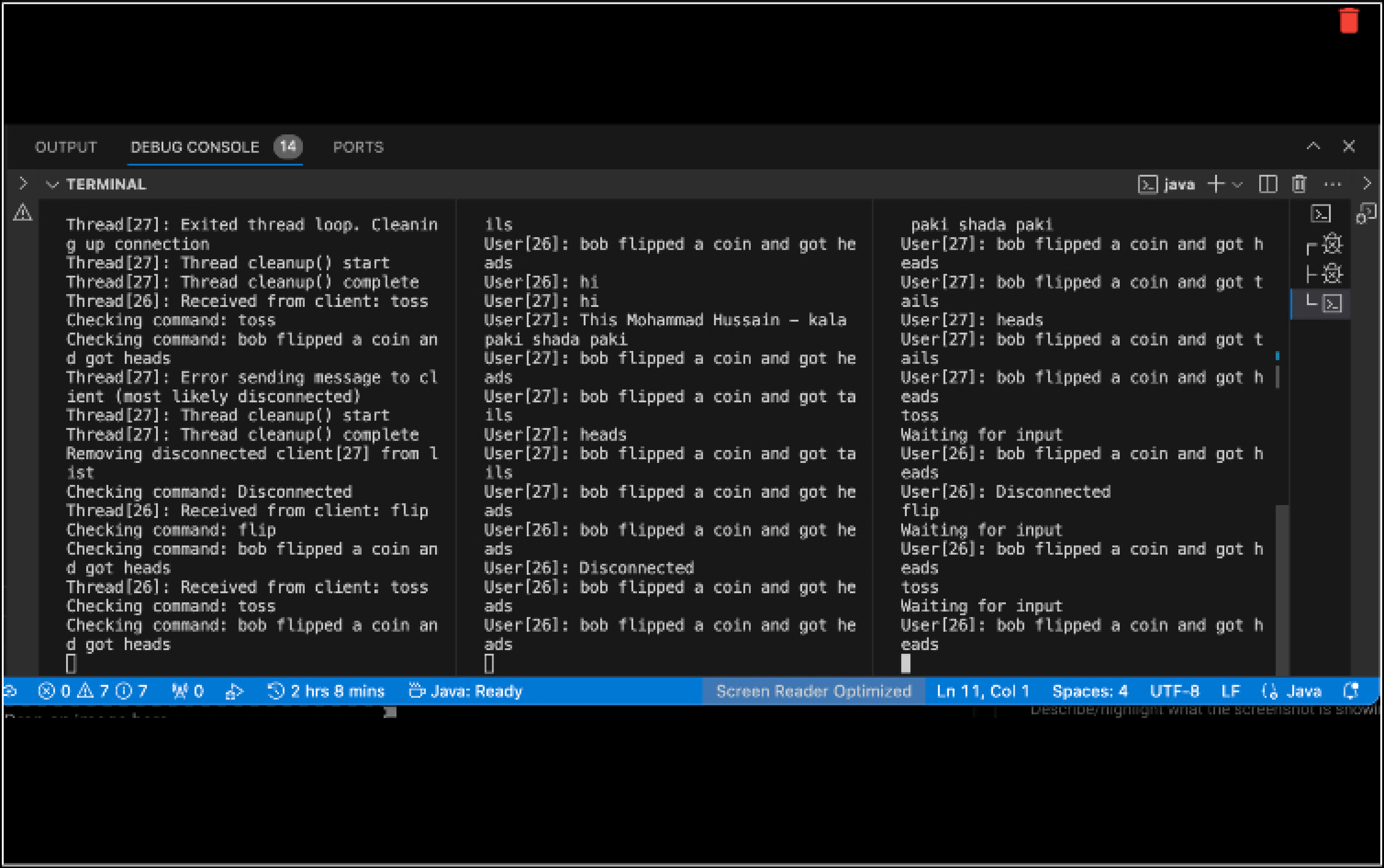


```java
        String result;
        Random random = new Random(); // Randomly genarates true or false
        if (random.nextBoolean()){
            result = "heads";
        } else {
            result = "tails";
        }
        String response = String.format( userName +" flipped a coin and got "+result + " "); // displays the messages
        broadcast(response, clientId);
        return true;

    } <- #97-110 if (message.equals("flip") || message.equals("toss") || messa...
```

Coin toss - random heads or tails

Checklist Items (1)

#2 Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.



```
Thread[27]: Exited thread loop. Cleanin
g up connection
Thread[27]: Thread cleanup() start
Thread[27]: Thread cleanup() complete
Thread[26]: Received from client: toss
Checking command: toss
Checking command: bob flipped a coin an
d got heads
Thread[27]: Error sending message to cl
ient (most likely disconnected)
Thread[27]: Thread cleanup() start
Thread[27]: Thread cleanup() complete
Removing disconnected client[27] from l
ist
Checking command: Disconnected
Thread[26]: Received from client: flip
Checking command: flip
Checking command: bob flipped a coin an
d got heads
Thread[26]: Received from client: toss
Checking command: toss
Checking command: bob flipped a coin an
d got heads
```

```
ils
User[26]: bob flipped a coin and got he
ads
User[26]: hi
User[27]: hi
User[27]: This Mohammad Hussain - kala
paki shada paki
User[27]: bob flipped a coin and got he
ads
User[27]: bob flipped a coin and got ta
ils
User[27]: heads
User[27]: bob flipped a coin and got ta
ils
User[27]: bob flipped a coin and got he
ads
User[26]: bob flipped a coin and got he
ads
User[26]: Disconnected
User[26]: bob flipped a coin and got he
ads
User[26]: bob flipped a coin and got he
ads
```

```
 paki shada paki
User[27]: bob flipped a coin and got h
eads
User[27]: bob flipped a coin and got t
ails
User[27]: heads
User[27]: bob flipped a coin and got t
ails
User[27]: bob flipped a coin and got h
eads
toss
Waiting for input
User[26]: bob flipped a coin and got h
eads
User[26]: Disconnected
flip
Waiting for input
User[26]: bob flipped a coin and got h
eads
toss
Waiting for input
User[26]: bob flipped a coin and got h
eads
```

output of the coin toss game

Checklist Items (1)

#1 Output is clearly shown and captioned

● Feature 2 (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: What feature did you pick? Briefly explain how you implemented it

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Feature is clearly stated (best to copy/paste it from above)
#2	1	Explanation sufficiently and concisely describes implementation (should be aligned with code snippets in related task)

Response:

I implemented a number-guessing feature to add an interactive and fun element. When I receive a "start" message and the game isn't already running, I activate it by setting a flag, gameActive, to true and generate a random number between 1 and 10 as the target for players to guess. This process is kickstarted by my generateRandomNumber method. I then announce the start of the game, inviting players to guess the number. If a player sends a guess by including the word "guess" followed by their number, I check if the game is still active. If it is, I parse their guess and compare it with the hidden number. I've crafted a checkGuess method that handles this comparison and constructs a response based on whether the guess is correct or not. For correct guesses, I congratulate the player by name, deactivate the game, and broadcast the success. For incorrect guesses, I notify the player, encouraging further attempts.

Should a "stop" message be received, I immediately deactivate the game, signaling the end of guessing and transitioning guesses back to regular messages. This feature is designed to be straightforward yet engaging, encouraging players to interact not just with the server but also with each other, creating a dynamic gaming experience within my server environment.

COLLAPSE

Task #2 - Points: 1

Text: Add screenshot(s) showing the implemented feature working (code and output)

Details:
Add screenshots of the relevant code changes AND relevant output during runtime

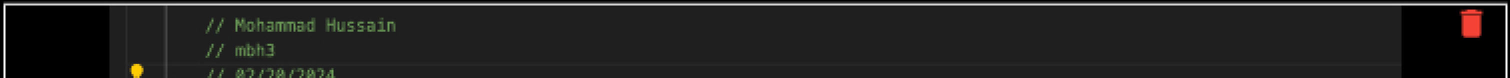
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Output is clearly shown and captioned
#2	1	Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

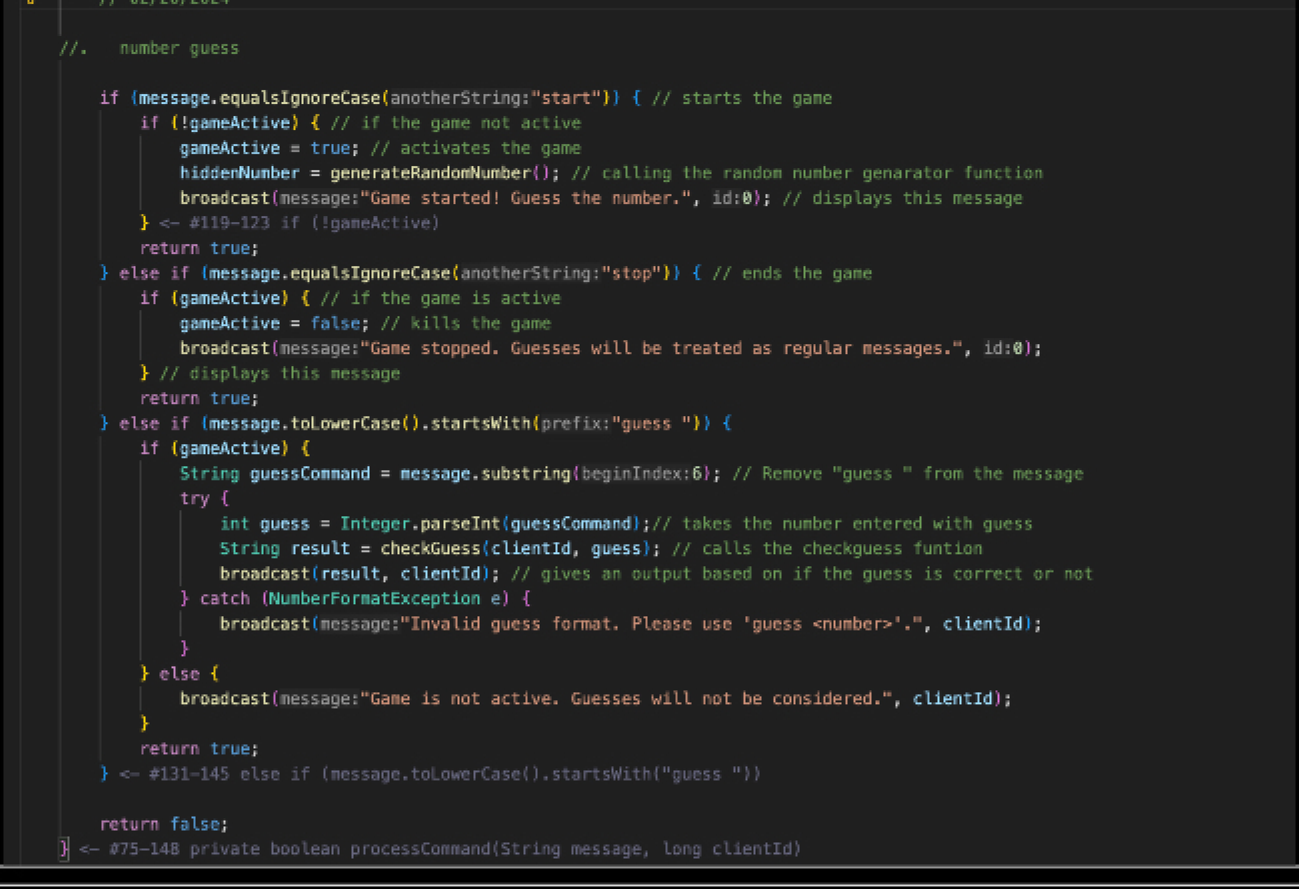
Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
// Mohammad Hussain
// mbh3
// 02/20/2024
```

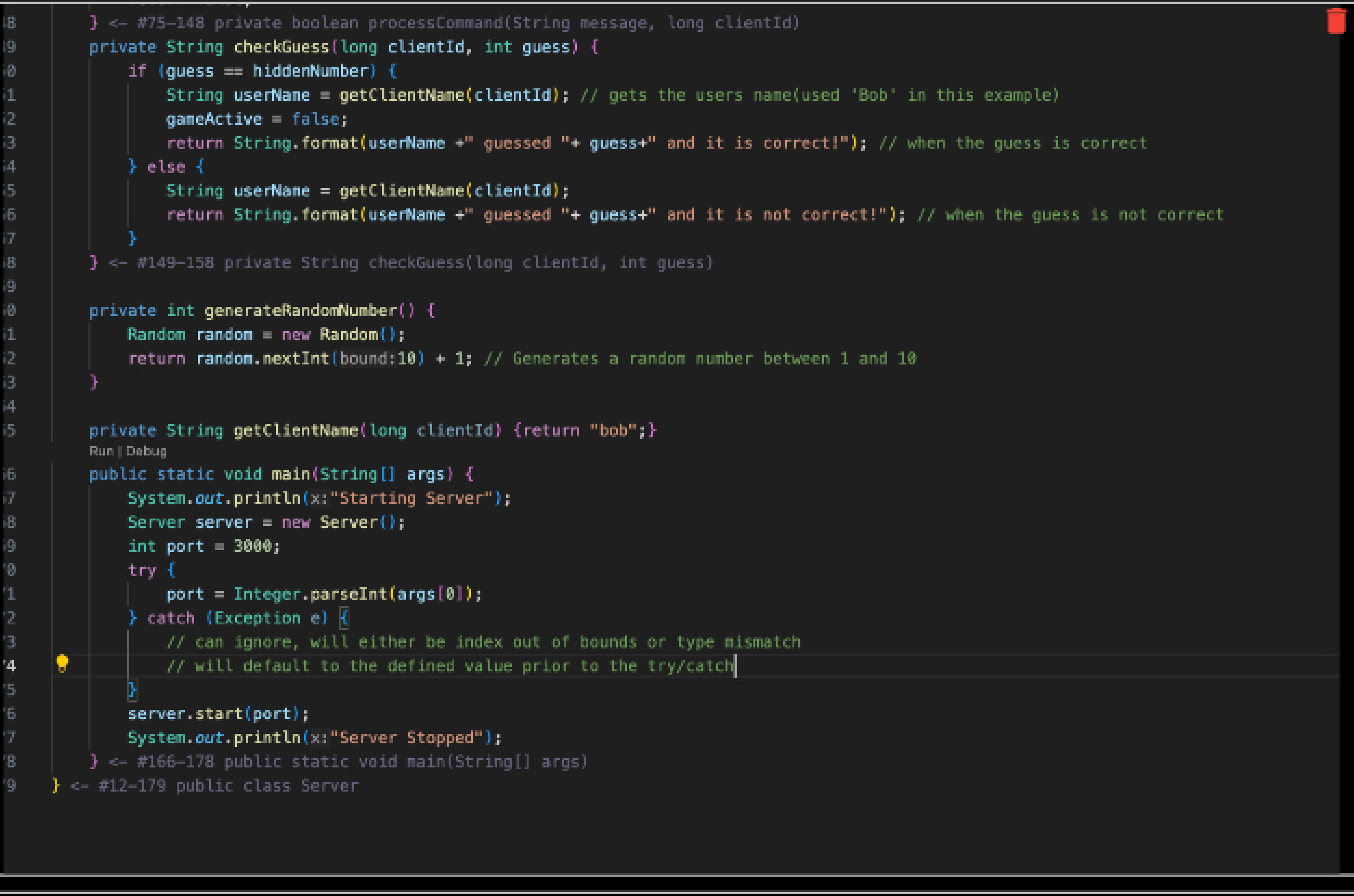


```
//.   number guess

    if (message.equalsIgnoreCase(anotherString:"start")) { // starts the game
        if (!gameActive) { // if the game not active
            gameActive = true; // activates the game
            hiddenNumber = generateRandomNumber(); // calling the random number genarator function
            broadcast(message:"Game started! Guess the number.", id:0); // displays this message
        } <- #119-123 if (!gameActive)
        return true;
    } else if (message.equalsIgnoreCase(anotherString:"stop")) { // ends the game
        if (gameActive) { // if the game is active
            gameActive = false; // kills the game
            broadcast(message:"Game stopped. Guesses will be treated as regular messages.", id:0);
        } // displays this message
        return true;
    } else if (message.toLowerCase().startsWith(prefix:"guess ")) {
        if (gameActive) {
            String guessCommand = message.substring(beginIndex:6); // Remove "guess " from the message
            try {
                int guess = Integer.parseInt(guessCommand);// takes the number entered with guess
                String result = checkGuess(clientId, guess); // calls the checkguess funtion
                broadcast(result, clientId); // gives an output based on if the guess is correct or not
            } catch (NumberFormatException e) {
                broadcast(message:"Invalid guess format. Please use 'guess <number>'.", clientId);
            }
        } else {
            broadcast(message:"Game is not active. Guesses will not be considered.", clientId);
        }
        return true;
    } <- #131-145 else if (message.toLowerCase().startsWith("guess "))

    return false;
} <- #75-148 private boolean processCommand(String message, long clientId)
```

code of the number guess game

Checklist Items (1)

#2 Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

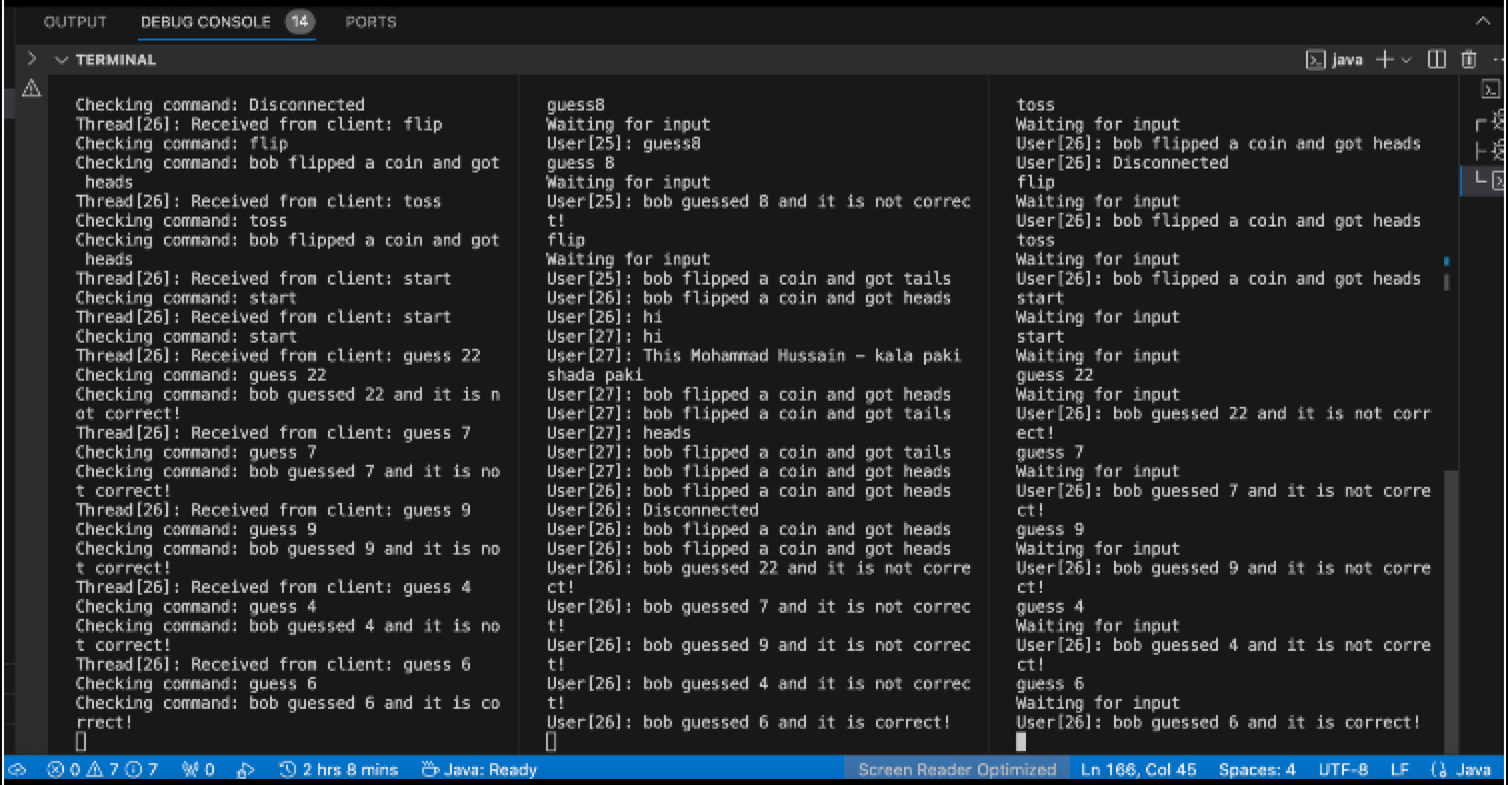


```
    } <- #75-148 private boolean processCommand(String message, long clientId)
    private String checkGuess(long clientId, int guess) {
        if (guess == hiddenNumber) {
            String userName = getClientName(clientId); // gets the users name(used 'Bob' in this example)
            gameActive = false;
            return String.format(userName +" guessed "+ guess+" and it is correct!"); // when the guess is correct
        } else {
            String userName = getClientName(clientId);
            return String.format(userName +" guessed "+ guess+" and it is not correct!"); // when the guess is not correct
        }
    } <- #149-158 private String checkGuess(long clientId, int guess)

    private int generateRandomNumber() {
        Random random = new Random();
        return random.nextInt(bound:10) + 1; // Generates a random number between 1 and 10
    }

    private String getClientName(long clientId) {return "bob";}
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"Starting Server");
        Server server = new Server();
        int port = 3000;
        try {
            port = Integer.parseInt(args[0]);
        } catch (Exception e) {
            // can ignore, will either be index out of bounds or type mismatch
            // will default to the defined value prior to the try/catch
        }
        server.start(port);
        System.out.println(x:"Server Stopped");
    } <- #166-178 public static void main(String[] args)
} <- #12-179 public class Server
```

other half of the code of number guess

Checklist Items (1)

#2 Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.



```
Checking command: Disconnected
Thread[26]: Received from client: flip
Checking command: flip
Checking command: bob flipped a coin and got
 heads
Thread[26]: Received from client: toss
Checking command: toss
Checking command: bob flipped a coin and got
 heads
Thread[26]: Received from client: start
Checking command: start
Thread[26]: Received from client: start
Checking command: start
Thread[26]: Received from client: guess 22
Checking command: guess 22
Checking command: bob guessed 22 and it is n
ot correct!
Thread[26]: Received from client: guess 7
Checking command: guess 7
Checking command: bob guessed 7 and it is no
t correct!
Thread[26]: Received from client: guess 9
Checking command: guess 9
Checking command: bob guessed 9 and it is no
t correct!
Thread[26]: Received from client: guess 4
Checking command: guess 4
Checking command: bob guessed 4 and it is no
t correct!
Thread[26]: Received from client: guess 6
Checking command: guess 6
Checking command: bob guessed 6 and it is co
rrect!
```

```
guess8
Waiting for input
User[25]: guess8
guess 8
Waiting for input
User[25]: bob guessed 8 and it is not correc
t!
flip
Waiting for input
User[25]: bob flipped a coin and got tails
User[26]: bob flipped a coin and got heads
User[26]: hi
User[27]: hi
User[27]: This Mohammad Hussain - kala paki
shada paki
User[27]: bob flipped a coin and got heads
User[27]: bob flipped a coin and got tails
User[27]: heads
User[27]: bob flipped a coin and got tails
User[27]: bob flipped a coin and got heads
User[26]: bob flipped a coin and got heads
User[26]: Disconnected
User[26]: bob flipped a coin and got heads
User[26]: bob flipped a coin and got heads
User[26]: bob guessed 22 and it is not corre
ct!
User[26]: bob guessed 7 and it is not correc
t!
User[26]: bob guessed 9 and it is not correc
t!
User[26]: bob guessed 4 and it is not correc
t!
User[26]: bob guessed 6 and it is correct!
```

```
toss
Waiting for input
User[26]: bob flipped a coin and got heads
User[26]: Disconnected
flip
Waiting for input
User[26]: bob flipped a coin and got heads
toss
Waiting for input
User[26]: bob flipped a coin and got heads
start
Waiting for input
start
Waiting for input
guess 22
Waiting for input
User[26]: bob guessed 22 and it is not corr
ect!
guess 7
Waiting for input
User[26]: bob guessed 7 and it is not corre
ct!
guess 9
Waiting for input
User[26]: bob guessed 9 and it is not corre
ct!
guess 4
Waiting for input
User[26]: bob guessed 4 and it is not corre
ct!
guess 6
Waiting for input
User[26]: bob guessed 6 and it is correct!
```

the output of the code

Checklist Items (1)

#1 Output is clearly shown and captioned

Misc (2 pts.)

^COLLAPSE ^

^COLLAPSE ^

Task #1 - Points: 1

Text: Reflection: Did you have an issues and how did you resolve them? If no issues, what did you learn during this assignment that you found interesting?

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	An issue or learning is clearly stated
#2	1	Response is a few reasonable sentences

Response:

During the implementation of the number guessing feature in my game server, I encountered challenges, particularly with parsing user guesses and implementing the game logic to compare these guesses against the hidden number. Initially, I was confused about how to extract and use the guessed number from the user's message. To overcome this, I broke down the problem: I learned to parse the guess by removing the prefix and converting the string to an integer, and implemented error handling to manage non-numeric inputs gracefully. Refining the checkGuess method to provide immediate and meaningful feedback on whether the guess was correct or incorrect was crucial. This experience taught me the value of breaking complex issues into smaller, manageable tasks and reinforced the importance of user feedback in interactive applications. It was a rewarding process that enhanced my understanding of game logic, error handling, and user engagement.

●
COLLAPSE

Task #2 - Points: 1

Text: Pull request link

Details:
URL should end with /pull/# and be related to this assignment

URL #1
https://github.com/Mohammadh222/mbh3-IT114-004/compare/M4-Sockets3-Homework?expand=1

End of Assignment