Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 Module 4 Sockets Part3 Challenge
Student: Mukaddis I. (mi348)
Status: Submitted | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 6/23/2025 10:00:45 PM
Updated: 6/23/2025 11:56:44 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/grading/mi348
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/view/mi348

Instructions

- Overview Link: https://youtu.be/_029E_aBTFo

1. Ensure you read all instructions and objectives before starting.
2. Create a new branch from `main` called `M4-Homework`
    1. `git checkout main` (ensure proper starting branch)
    2. `git pull origin main` (ensure history is up to date)
    3. `git checkout -b M4-Homework` (create and switch to branch)
3. Copy the template code from here: GitHub Repository - M4 Homework
    - It includes Sockets `Part1`, `Part2`, and `Part3`. Put all into an `M4` folder or similar if you don't have them yet (adjust `package` reference at the top if you chose a different folder name).
    - Make a copy of `Part3` and call it `Part3HW`
        - Fix the `package` and `import` references at the top of each file in this new folder (Note: you'll only be editing files in `Part3HW`)
    - Immediately record to history
        - `git add .`
        - `git commit -m "adding M4 HW baseline files"`
        - `git push origin M4-Homework`
        - Create a Pull Request from `M4-Homework` to `main` and keep it open
4. Fill out the below worksheet
    - Each Problem requires the following as you work
        - Ensure there's a comment with your UCID, date, and brief summary of how the problem was solved
        - Code solution (add/commit periodically as needed)
        - Hint: Note how `/reverse` is handled
5. Once finished, click "Submit and Export"
6. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
    1. `git add .`
    2. `git commit -m "adding PDF"`

3. `git push origin M4-Homework`
4. On Github merge the pull request from `M4-Homework` to `main`

7. Upload the same PDF to Canvas
8. Sync Local
    1. `git checkout main`
    2. `git pull origin main`

Section #1: (3 pts.) Challenge 1 - Coin Flip

Progress: 100%

Task #1 (3 pts.) - Implement a Coin Flip Command

Progress: 100%

Details:

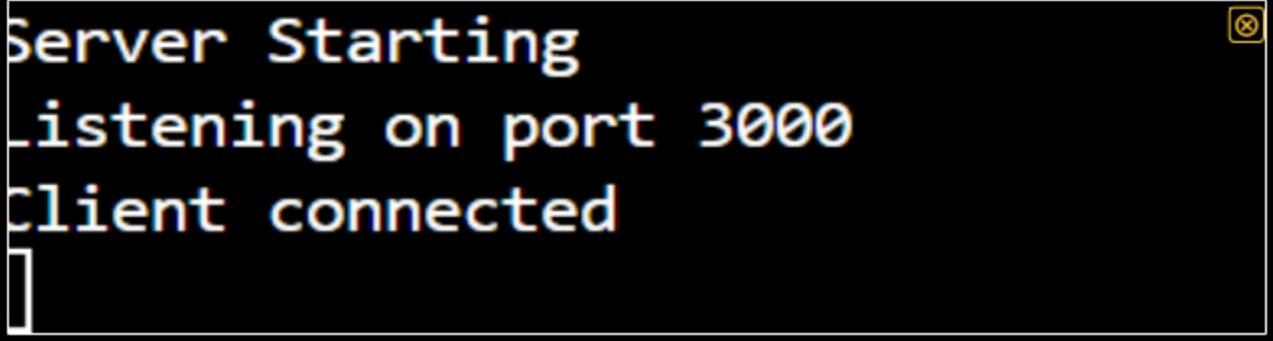
- `Client` must capture the user entry and generate a valid command per the lesson details
    - Command format must be `/flip`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic and send the result to everyone
    - The message must be in the format of `<who> flipped a coin and got <result>` and be from the Server
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
    - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
    - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
    - Should only need to create a new method and pass the result message to `relay()`
4. Show 5 examples of the command being seen across all terminals (2+ Clients and 1 Server)
    1. This can be captured in one screenshot if you split the terminals side by side



Server

```
Welcome, Mukaddis
flip
Server: Mukaddis flipped a coin and got Tails
flip
Server: Mukaddis flipped a coin and got Tails
flip
Server: Mukaddis flipped a coin and got Heads
```

Client

```
/MI348 6/23/2025
    private boolean processClientCommand(String text, Scanner si) {
        if (isConnection(text)) {
            String[] parts = text.trim().replaceAll(regex:" +", replacement:" ").split(regex:" ")[1].split(regex:":");
            connect(parts[0].trim(), Integer.parseInt(parts[1].trim()), si);
            return true;
        } else if ("/quit".equalsIgnoreCase(text)) {
```

Client.java

```
            out.println("Welcome, " + clientName);
//MI348 6/23/2025

            String input;
            while ((input = in.readLine()) != null) {
                if (!processCommand(input.trim())) {
                    server.relay(clientName, input);
                }
```

Server.java

```
/UCID MI348 Date 6/23/2025
    private boolean processClientCommand(String text, Scanner si) {
        if (isConnection(text)) {
            String[] parts = text.trim().replaceAll(regex:" +", replacement:" ").split(regex:" ")[1].split(regex:":");
            connect(parts[0].trim(), Integer.parseInt(parts[1].trim()), si);
            return true;
        } else if ("/quit".equalsIgnoreCase(text)) {
            isRunning = false;
```

Serverthread.java

Saved: 6/23/2025 11:53:25 PM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/MIsmail215/mi348-IT114-450/commits/eee8486be084d634624ee725a519861872e6621b

URL
https://github.com/MIsmail215/m

Saved: 6/23/2025 11:53:25 PM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenge (note: this isn't the same as `what` the code does)

Your Response:

The code solves the challenge by allowing clients to connect to the server and send commands like /flip. When the /flip command is received, the server generates a random coin flip result and broadcasts a formatted message to all clients. This setup uses multithreading to handle multiple clients at once and ensures real-time communication through sockets.

Saved: 6/23/2025 11:53:25 PM

Section #2: (3 pts.) Challenge 2 - Private Message

Progress: 100%

Task #1 (3 pts.) - Implement a Private Message Command

Progress: 100%

Details:

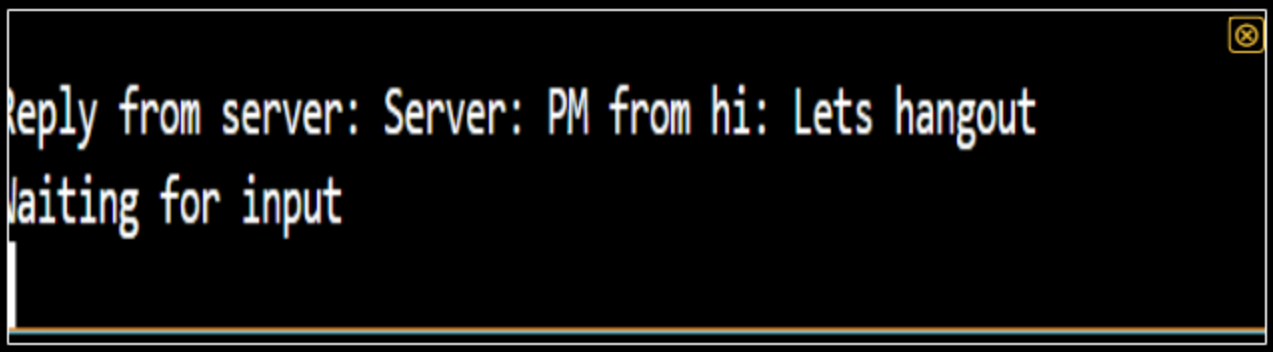
- `Client` must capture the user entry and generate a valid command per the lesson details
  - Command format must be `/pm <target id> <message>`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic
  - The message must be in the format of `PM from <who>: <message>` and be from the Server
  - The result must only be sent to the original sender and to the receiver/target
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
   - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
   - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
   - Should only need to create a new method and send the result message to just the sender and receiver
4. Show 3 examples of the command being seen across all terminals (3+ Clients and 1 Server)
   1. This can be captured in one screenshot if you split the terminals side by side
   2. Note: Only the sender and the receiver should see the private message (show variations across different users)



Shows the client

```
//mi348 6/23/2025
    public void pm(int fromId, int toId, String message) {
        ServerThread sender = clients.get(fromId);
        ServerThread receiver = clients.get(toId);
        if (receiver != null && sender != null) {
            String formatted = "PM from " + sender.getClientName() + ": " + message;
            receiver.sendMessage("Server: " + formatted);
            sender.sendMessage("Server: " + formatted);
        } else if (sender != null) {
            sender.sendMessage(message:"Server: PM failed. User ID not found.");
        }
    }
```

Server.java

```
//MI348 6/23/2025
            String input;
            while ((input = in.readLine()) != null) {
                processCommand(input.trim());
            }
        } catch (IOException e) {
            System.out.println("Connection to client " + clientId + " lost.");
        } finally {
            server.removeClient(clientId);
            try {
                socket.close();
            } catch (IOException e) {
                System.out.println("Error closing socket for client " + clientId);
            }
        }
    }

    private void processCommand(String input) {
        if (input.startsWith(prefix:"/pm")) {
```

ServerThread.java

```
Client 1 connected.
Client 2 connected.
```

Shows the server

```java
private boolean processClientCommand(String text) {
    if (isConnection(text)) {
        // replaces multiple spaces with single space
        // splits on the space after connect (gives us host and port)
        // splits on : to get host as index 0 and port as index 1
        String[] parts = text.trim().replaceAll(regex:" +", replacement:" ").split(regex:" ")[1].split(regex:":");
        connect(parts[0].trim(), Integer.parseInt(parts[1].trim()));
        return true;
    } else if ("/quit".equalsIgnoreCase(text)) {
        isRunning = false;
        return true;
    }
    return false;
}
```

Client.Java

```
Reply from server: Server: PM from hi: Hi there OK, bit of a weird name
Waiting for input
```

Shows the Client

```
/pm 2 YOOO
Reply from server: Server: Unknown command.
Waiting for input
/pm 2 hello
Reply from server: Server: PM from Mukaddis: YOOO
Waiting for input
```

Shows client

Saved: 6/23/2025 11:53:43 PM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/MIsmail215/mi348-IT114-450/commits/420f66d960d9496832188214a36a8f9830aeb86c

URL
https://github.com/MIsmail215/m

Saved: 6/23/2025 11:53:43 PM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenges (note: this isn't the same as `what` the code does)

Your Response:

The code allows clients to connect to the server and send messages, including private ones using the /pm command. Each client is assigned a unique ID, and the server uses that to route private messages to only the sender and intended recipient. This solves the challenge by implementing personalized, bi-directional communication using sockets and multithreading.

Saved: 6/23/2025 11:53:43 PM

Section #3: (3 pts.) Challenge 3 - Shuffle Message

Progress: 100%

Task #1 (3 pts.) - Implement a Shuffle Message Command

Progress: 100%

Details:

- `Client` must capture the user entry and generate a valid command per the lesson details
  - Command format must be `/shuffle <message>`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic and send the result to everyone
  - The message must be in the format of `Shuffled from <who>: <shuffled_message>` and be from the Server
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
    - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
    - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
    - Should only need to create a new method and do similar logic to `relay()`
4. Show 3 examples of the command being seen across all terminals (2+ Clients and 1 Server)
    1. This can be captured in one screenshot if you split the terminals side by side

```
}
//UCID mi348 6/23/2025
private void onServerThreadInitialized(ServerThread serverThread) {
    connectedClients.put(serverThread.getClientId(), serverThread);
    relay(sender:null, String.format(format:"*User[%s] connected*", serverThread.getClientId()));
}

private synchronized void disconnect(ServerThread serverThread) {
    serverThread.disconnect();
    ServerThread removed = connectedClients.remove(serverThread.getClientId());
    if (removed != null) {
        relay(sender:null, "User[" + removed.getClientId() + "] disconnected");
    }
}
```

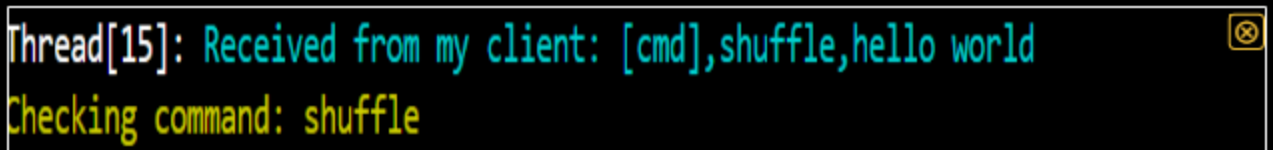
Server.java relay

```
//UCID mi348 6/23/2025
private boolean processCommand(String message) {
    boolean wasCommand = false;

    if (message.startsWith(Constants.COMMAND_TRIGGER)) {
        String[] commandData = message.split(regex:",");
        if (commandData.length >= 2) {
            final String command = commandData[1].trim();
            System.out.println(TextFX.colorize("Checking command: " + command, Color.YELLOW));

            switch (command) {
                case "disconnect":
                    server.handleDisconnect(this);
                    wasCommand = true;
                    break;
                case "reverse":
                    String revText = String.join(delimiter:" ", Arrays.copyOfRange(commandData, from:2, commandData.length));
                    server.handleReverseText(this, revText);
                    wasCommand = true;
                    break;
                case "shuffle":
                    String shuffleText = String.join(delimiter:" ", Arrays.copyOfRange(commandData, from:2, commandData.length));
                    server.handleShuffleText(this, shuffleText);
                    wasCommand = true;
                    break;
                default:
```

Serverthread Processcommand

```
//UCID mi348 6/23/2025
private boolean processClientCommand(String text) throws IOException {
    boolean wasCommand = false;
    if (isConnection(text)) {
        String[] parts = text.trim().replaceAll(regex:" +", replacement:" ").split(regex:" ")[1].split(regex:":");
        connect(parts[0].trim(), Integer.parseInt(parts[1].trim()));
        wasCommand = true;
    } else if ("/quit".equalsIgnoreCase(text)) {
        close();
        wasCommand = true;
    } else if ("/disconnect".equalsIgnoreCase(text)) {
        String[] commandData = { Constants.COMMAND_TRIGGER, "disconnect" };
        sendToServer(String.join(delimiter:",", commandData));
        wasCommand = true;
    } else if (text.startsWith(prefix:"/reverse")) {
        text = text.replace(target:"/reverse", replacement:"").trim();
        String[] commandData = { Constants.COMMAND_TRIGGER, "reverse", text };
        sendToServer(String.join(delimiter:",", commandData));
```

Client Processcommand



```
Thread[15]: Received from my client: [cmd],shuffle,hello world
Checking command: shuffle
```

```
Thread[15]: Received from my client: [cmd],shuffle,hello there my friend
Checking command: shuffle
```

Server Side

```
Server: Shuffled from User[15]: hl roellwdo
/shuffle hello there my friend
Server: Shuffled from User[15]: el er  nirhoemtefyldh
```

Client Test 1

```
Client connected
Server: *User[15] connected*
/shuffle hello world
Server: Shuffled from User[15]: hl roellwdo
```

Client test 2

Saved: 6/23/2025 11:53:57 PM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/MIsmail215/mi348-IT114-450/commits/39b9b63afbe32a5655cafa5b8668cd52aca51699

URL
https://github.com/MIsmail215/m

Saved: 6/23/2025 11:53:57 PM

Part 3:

Progress: 100%

Details:
Briefly explain how the code solves the challenges (note: this isn't the same as what the code

does)

Your Response:

The code enables a client to send special commands like /shuffle or /reverse, which are detected and processed by the server using a predefined trigger ([cmd]). The ServerThread parses these commands, invokes the appropriate logic (e.g., handleShuffleText), and broadcasts the result to all connected clients.

Saved: 6/23/2025 11:53:57 PM

Section #4: (1 pt.) Misc

Progress: 100%

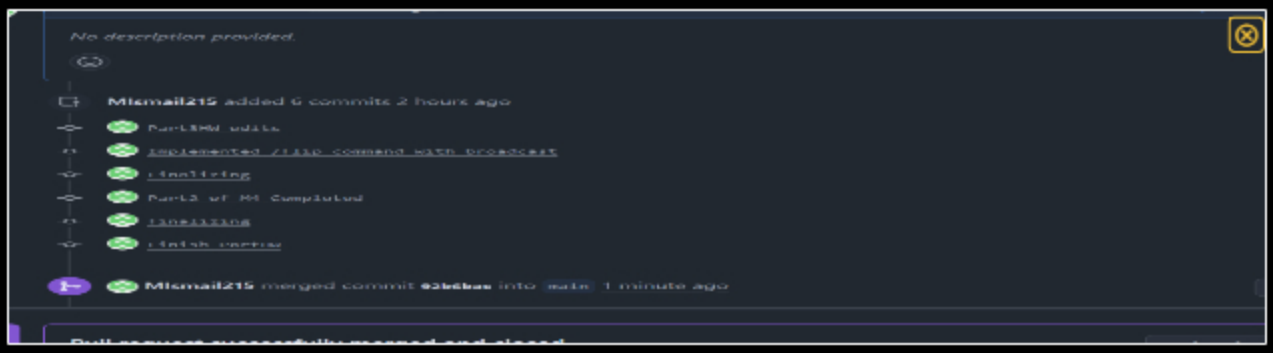
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

Details:
From the Commits tab of the Pull Request screenshot the commit history Following minimum should be present



Github Pull8

Saved: 6/23/2025 11:56:44 PM

Part 2:

Progress: 100%

Details:
Include the link to the Pull Request (should end in `/pull/#`)

URL #1
https://github.com/MIsmail215/mi348-

URL
https://github.com/MIsmail215/n

Saved: 6/23/2025 11:56:44 PM

Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

Details:

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary

File Bottom page

Top WakaTime

Saved: 6/23/2025 11:55:25 PM

Task #3 (0.33 pts.) - Reflection

Progress: 100%

Task #1 (0.33 pts.) - What did you learn?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

I learned how to implement client-server communication using Java sockets, including how to handle multiple clients with threads. I also learned how to process and broadcast custom commands like /shuffle and /reverse. The project helped reinforce object serialization and concurrency concepts.

Saved: 6/23/2025 11:56:24 PM

Task #2 (0.33 pts.) - What was the easiest part of the assignment?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

Setting up the basic client-server connection and sending simple messages was straightforward. Writing the logic for reading and writing objects using ObjectInputStream and ObjectOutputStream was also relatively easy. Using predefined methods and constants helped simplify command parsing.

Saved: 6/23/2025 11:56:31 PM

Task #3 (0.33 pts.) - What was the hardest part of the assignment?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The hardest part was debugging SocketException errors, especially those related to stream order and flushing. Ensuring that commands like /shuffle were correctly parsed and broadcasted to all clients also required careful attention. Managing multiple threads and handling cleanup properly after disconnection was also a bit tricky.

Saved: 6/23/2025 11:56:35 PM