Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 Milestone 1
Student: Oseghale A. (oka)
Status: Submitted | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 7/24/2025 2:30:20 AM
Updated: 7/24/2025 7:23:17 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-1/grading/oka
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-milestone-1/view/oka

Instructions

- Overview Link: https://youtu.be/9dZPFwi76ak

1. Refer to Milestone1 of any of these docs:
    2. Rock Paper Scissors
    3. Basic Battleship
    4. Hangman / Word guess
    5. Trivia
    6. Go Fish
    7. Pictionary / Drawing
2. Ensure you read all instructions and objectives before starting.
3. Ensure you've gone through each lesson related to this Milestone
4. Switch to the Milestone1 branch
    1. `git checkout Milestone1` (ensure proper starting branch)
    2. `git pull origin Milestone1` (ensure history is up to date)
5. Copy `Part5` and rename the copy as `Project` (this new folder should be in the root of your repo)
6. Organize the files into their respective packages `Client`, `Common`, `Server`, `Exceptions`
    1. Hint: If it's open, you can refer to the Milestone 2 Prep lesson
7. Fill out the below worksheet
    1. Ensure there's a comment with your UCID, date, and brief summary of the snippet in each screenshot
    2. Since this Milestone was majorly done via lessons, the required comments should be placed in areas of analysis of the requirements in this worksheet. There shouldn't need to be any actual code changes beyond the restructure.
8. Once finished, click "Submit and Export"
9. 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 Milestone1`
    4. On Github merge the pull request from `Milestone1` to `main`
10. Upload the same PDF to Canvas
11. Sync Local

1. git checkout main
2. git pull origin main

Section #1: (1 pt.) Feature: Server Can Be Started Via Command Line And Listen To Connections

Progress: 100%

Task #1 (1 pt.) - Evidence

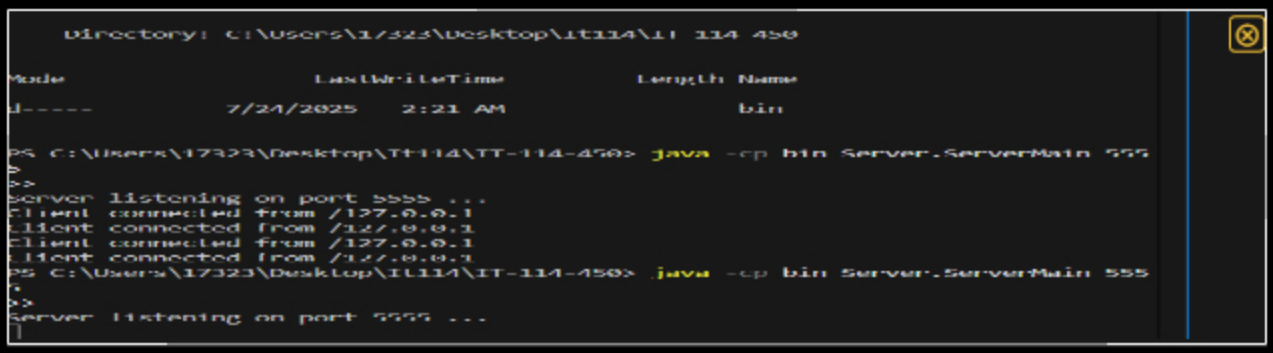
Progress: 100%

Part 1:

Progress: 100%

Details:

- Show the terminal output of the server started and listening
- Show the relevant snippet of the code that waits for incoming connections



The server creates a ServerSocket on the given port and blocks on accept(). Each time a client connects, we spawn a new ServerTh

```
ic void startServer() {
try (ServerSocket serverSocket = new ServerSocket(port)) {
    System.out.println("Server listening on port " + port + " ...");
    while (true) {
        Socket clientSocket = serverSocket.accept();
        ServerThread st = new ServerThread(clientSocket, this, roomManage
        clients.add(st);
        st.start();
```

The server creates a ServerSocket on the given port and blocks on accept(). Each time a client connects, we spawn a new ServerTh

Saved: 7/24/2025 2:41:21 AM

Part 2:

Progress: 100%

Details:

- Briefly explain how the server-side waits for and accepts/handles connections

Your Response:

The server creates a ServerSocket on the given port and blocks on accept(). Each time a client connects, we spawn a new ServerThread to handle I/O for that socket. The main loop keeps listening so additional clients can still connect.

Saved: 7/24/2025 2:41:21 AM

Section #2: (1 pt.) Feature: Server Should Be Able To Allow More Than One Client To Be Connected At Once

Progress: 100%

Task #1 (1 pt.) - Evidence

Progress: 100%

Part 1:

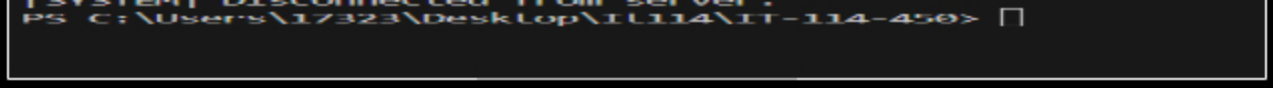
Progress: 100%

Details:

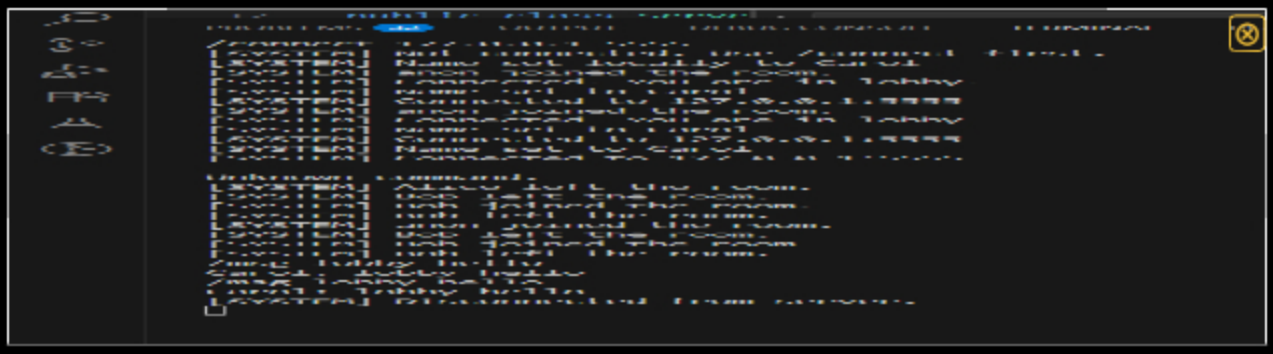
- Show the terminal output of the server receiving multiple connections
- Show at least 3 Clients connected (best to use the split terminal feature)
- Show the relevant snippets of code that handle logic for multiple connections

Three clients connected concurrently; server prints each connection.

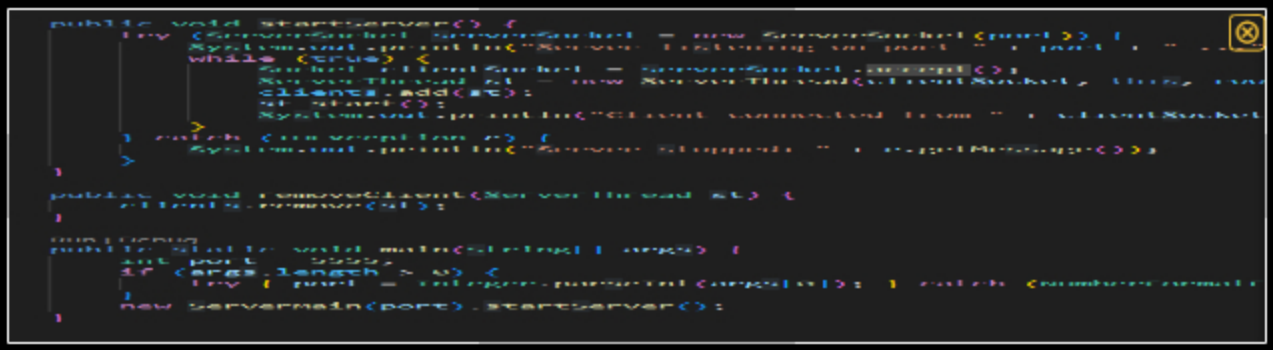
```
t
/name Bob
/connect 127.0.0.1 5555
[SYSTEM] Not connected. Use /connect first.
[SYSTEM] Name set locally to Bob
[SYSTEM] anon joined the room.
[SYSTEM] Connected. You are in lobby.
[SYSTEM] Name set to Bob
[SYSTEM] Connected to 127.0.0.1:5555
/joinroom games
/msg hey Alice
/leave
/quit
```



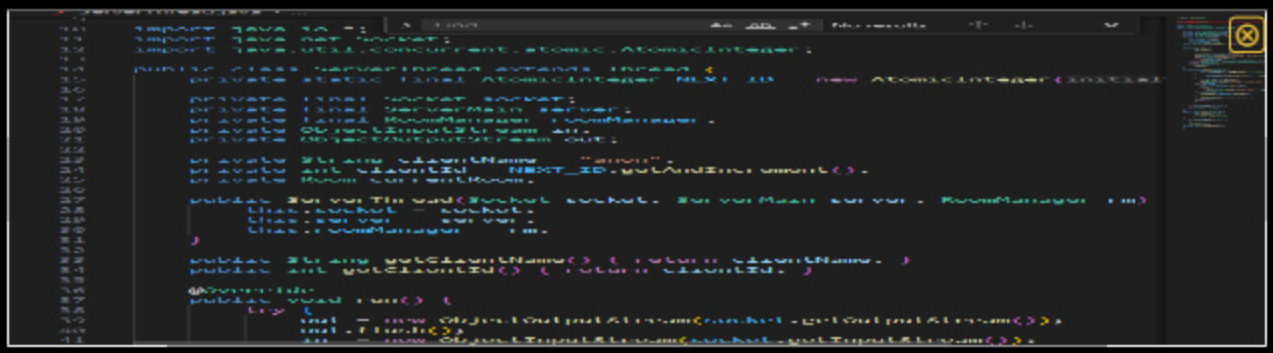
Three clients connected concurrently; server prints each connection.



Three clients connected concurrently; server prints each connection.



ServerMain & ServerThread handle each client on its own thread.



ServerMain & ServerThread handle each client on its own thread.

Saved: 7/24/2025 7:23:17 PM

Part 2:

Progress: 100%

Details:

- Briefly explain how the server-side handles multiple connected clients

Your Response:

Every client runs in its own ServerThread. The server stores threads in a concurrent set, so multiple clients can connect simultaneously without blocking each other. Each thread reads

payloads and sends responses independently.

Saved: 7/24/2025 7:23:17 PM

Section #3: (2 pts.) Feature: Server Will Implement The Concept Of Rooms (With The Default Being "l obby")

Progress: 100%

Task #1 (2 pts.) - Evidence

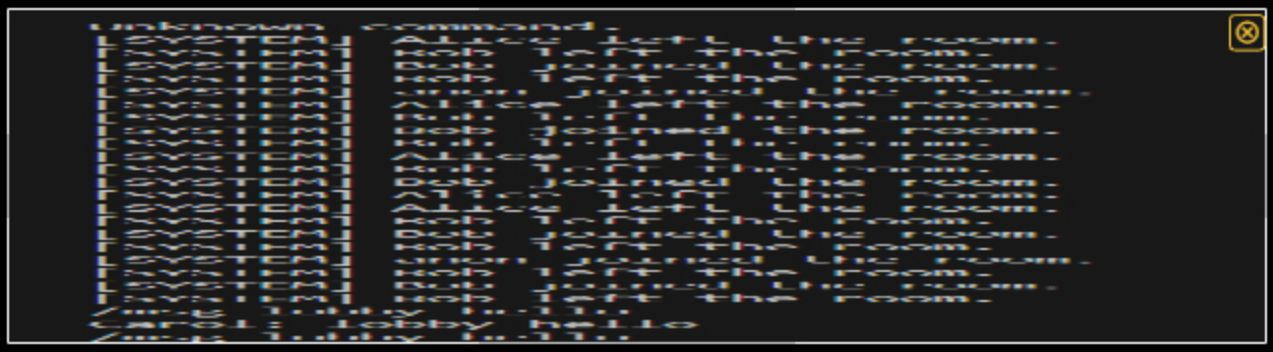
Progress: 100%

Part 1:

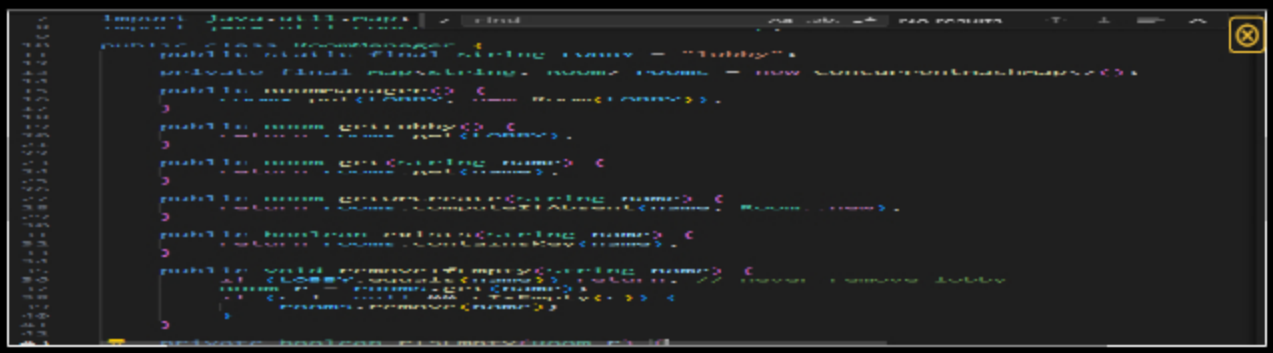
Progress: 100%

Details:

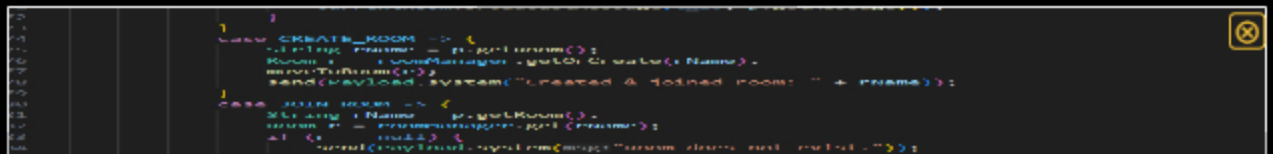
- Show the terminal output of rooms being created, joined, and removed (server-side)
- Show the relevant snippets of code that handle room management (create, join, leave, remove) (server-side)

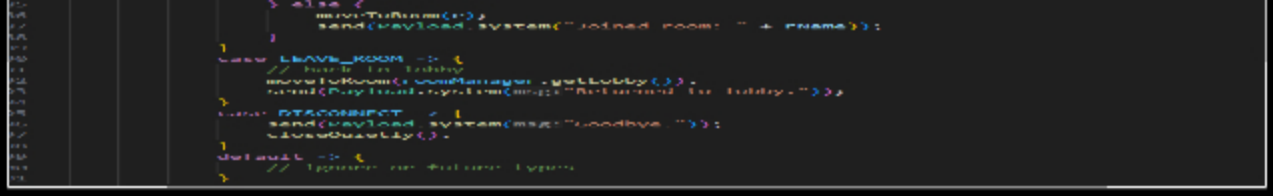


RoomManager + Room classes manage create/join/leave and broadcast."



RoomManager + Room classes manage create/join/leave and broadcast."





Server logs room creation, joins, leaves."

Saved: 7/24/2025 2:56:43 AM

Part 2:

Progress: 100%

Details:

- Briefly explain how the server-side handles room creation, joining/leaving, and removal

Your Response:

The server keeps rooms in RoomManager (a map). getOrCreate() returns an existing room or creates one. ServerThread.moveToRoom() removes the client from the current room, adds them to the new one, and calls broadcast() so everyone sees join/leave messages.

Saved: 7/24/2025 2:56:43 AM

Section #4: (1 pt.) Feature: Client Can Be Started Via The Command Line

Progress: 100%

Task #1 (1 pt.) - Evidence

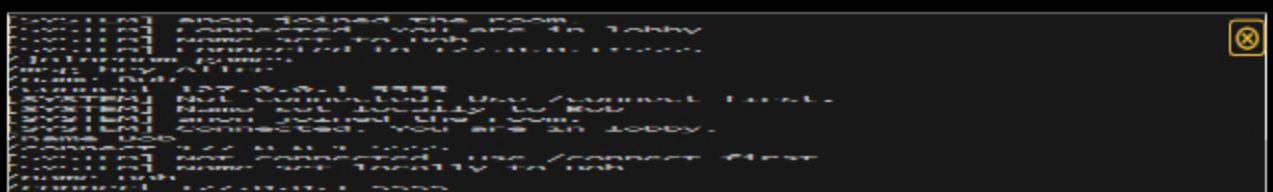
Progress: 100%

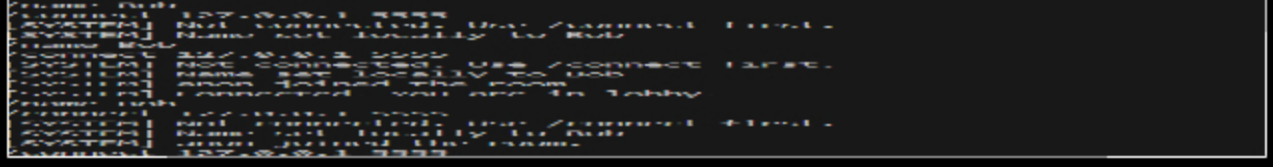
Part 1:

Progress: 100%

Details:

- Show the terminal output of the /name and /connect commands for each of 3 clients (best to use the split terminal feature)
- Output should show evidence of a successful connection
- Show the relevant snippets of code that handle the processes for /name, /connect, and the confirmation of being fully setup/connected





Client CLI accepts /name and /connect, then confirms connection.

```
        if (!connected) {
            if (line.startsWith(prefix:"/name")) {
                String name = line.replaceFirst(regex:"/name\\s*", replacement:"").trim();
                if (name.isEmpty()) {
                    System.out.println(x:"[SYSTEM] Usage: /name <displayName>");
                    return;
                }
                localName = name;
                System.out.println("[SYSTEM] Name set locally to " + localName);
            } else {
                System.out.println(x:"[SYSTEM] Not connected. Use /connect first.");
            }
            return;
        }
```

ClientMain: parses commands, opens socket, sends NAME payload

Saved: 7/24/2025 3:00:30 AM

Part 2:

Progress: 100%

Details:

- Briefly explain how the /name and /connect commands work and the code flow that leads to a successful connection for the client

Your Response:

/name sets a local variable and sends a Payload of type NAME to the server. /connect opens the Socket, sets up ObjectInputStream/ObjectOutputStream, and starts ClientReader. After that, the server confirms the connection and room placement (“You are in lobby”).

Saved: 7/24/2025 3:00:30 AM

Section #5: (2 pts.) Feature: Client Can Create/j oin Rooms

Progress: 100%

Task #1 (2 pts.) - Evidence

Progress: 100%

Part 1:

Progress: 100%

Details:

- Show the terminal output of the /createroom and /joinroom
- Output should show evidence of a successful creation/join in both scenarios
- Show the relevant snippets of code that handle the client-side processes for room creation and joining

```
/msg hi from games
[SYSTEM] Alice joined the room.
[SYSTEM] Created & joined room: games
Alice: hi from games

Unknown command.
[SYSTEM] Bob joined the room.
Bob: hey Alice
[SYSTEM] Bob left the room.
[SYSTEM] Disconnected from server.
```

Alice created 'games'; Bob joined; both got success messages

```
1   // UCID: oka
2   // Date: 2025 07 24
3   // Summary: Enum of all message/command types exchanged between client and
4
5   package Common;
6
7   public enum PayloadType {
8       NAME,               // client sets or updates its display name
9       MESSAGE,            // normal room chat message
10      CREATE_ROOM,        // request to create a room
11      JOIN_ROOM,          // request to join an existing room
12      LEAVE_ROOM,         // leave current room (back to lobby)
13      DISCONNECT,         // client is quitting
14      SYSTEM              // server to client informational/system messages
15  }
16
```

Client sends CREATE_ROOM/JOIN_ROOM payloads to server

Saved: 7/24/2025 7:03:44 PM

Part 2:

Progress: 100%

Details:

- Briefly explain how the /createroom and /join room commands work and the related code flow for each

Your Response:

On /createroom, the client wraps the room name in a Payload with type CREATE_ROOM and sends it. The server creates the room and moves the client; client prints confirmation. /joinroom uses JOIN_ROOM type, and the server checks if room exists, then moves the client accordingly

Saved: 7/24/2025 7:03:44 PM

Section #6: (1 pt.) Feature: Client Can Send Messages

Messages

Progress: 100%

Task #1 (1 pt.) - Evidence

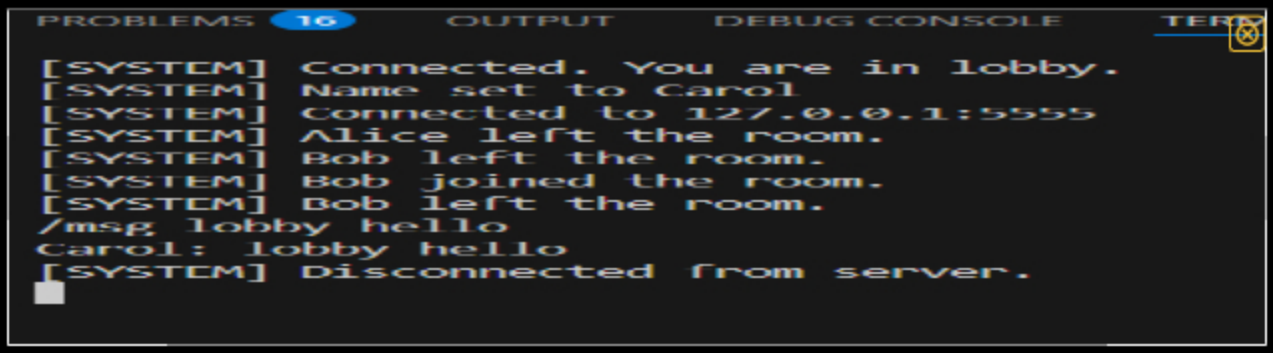
Progress: 100%

Part 1:

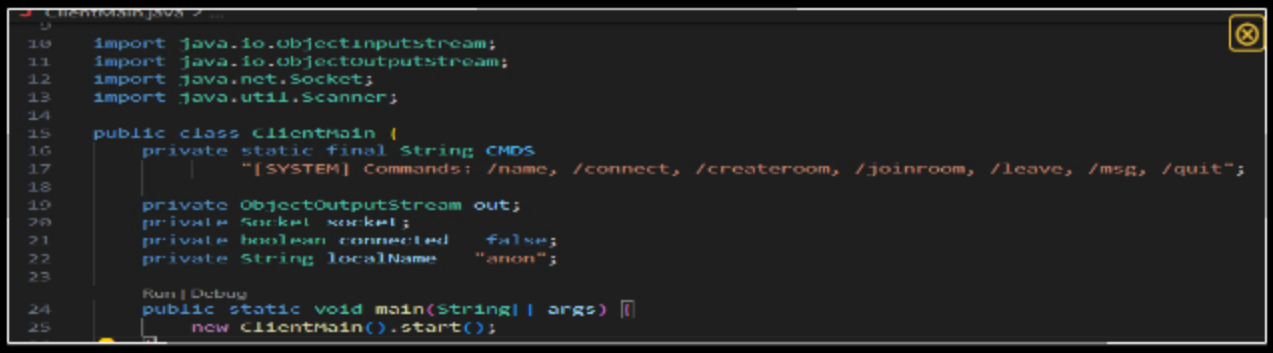
Progress: 100%

Details:

- Show the terminal output of a few messages from each of 3 clients
- Include examples of clients grouped into other rooms
- Show the relevant snippets of code that handle the message process from client to server-side and back



Room-specific chat: games msgs stay in games, lobby sees only lobby msgs.



```java
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.Scanner;

public class ClientMain {
    private static final String CMDS =
            "[SYSTEM] Commands: /name, /connect, /createroom, /joinroom, /leave, /msg, /quit";

    private ObjectOutputStream out;
    private Socket socket;
    private boolean connected = false;
    private String localName = "anon";

    Run | Debug
    public static void main(String[] args) {
        new ClientMain().start();
```

Server broadcasts only to members of the current Room.

Saved: 7/24/2025 7:06:45 PM

Part 2:

Progress: 100%

Details:

- Briefly explain how the message code flow works

Your Response:

/msg becomes a MESSAGE payload. The server handles it and calls currentRoom.broadcast(), which loops over members and sends them the text. Because broadcast is per-room, clients in

other rooms never see those messages.

Saved: 7/24/2025 7:06:45 PM

Section #7: (1 pt.) Feature: Disconnection

Progress: 100%

Task #1 (1 pt.) - Evidence

Progress: 100%

Part 1:

Progress: 100%

Details:

- Show examples of clients disconnecting (server should still be active)
- Show examples of server disconnecting (clients should be active but disconnected)
- Show examples of clients reconnecting when a server is brought back online
- Examples should include relevant messages of the actions occuring
- Show the relevant snippets of code that handle the client-side disconnection process
- Show the relevant snippets of code that handle the server-side termination process

```
/msg hi from games
[SYSTEM] Alice joined the room.
[SYSTEM] Created & joined room: games
Alice: hi from games

Unknown command.
[SYSTEM] Bob joined the room.
Bob: hey Alice
[SYSTEM] Bob left the room.
[SYSTEM] Disconnected from server.
```

Bob quits; server remains running.

```
10  import java.io.ObjectInputStream;
11  import java.io.ObjectOutputStream;
12  import java.net.Socket;
13  import java.util.Scanner;
14
15  public class ClientMain {
16      private static final String CMDS =
17              "[SYSTEM] Commands: /name, /connect, /createroom, /joinroom, /leave, /msg, /quit";
18
19      private ObjectOutputStream out;
20      private Socket socket;
21      private boolean connected = false;
22      private String localName = "Anon";
23
24      Run | Debug
25      public static void main(String[] args) {
26          new ClientMain().start();
27      }
```

Server stopped, clients auto-disconnected; after restart, clients reconnected

Saved: 7/24/2025 7:10:11 PM

Part 2:

Progress: 100%

Details:

- Briefly explain how both client and server gracefully handle their disconnect/termination logic

Your Response:

The client sends a DISCONNECT payload (or just closes), and the server catches EOF/exception, calls cleanup() to remove the user from the room and client list. If the server dies, the client's ClientReader thread catches an exception and prints a disconnected message; the user can reconnect when the server restarts.

Saved: 7/24/2025 7:10:11 PM

Section #8: (1 pt.) Misc

Progress: 100%

Task #1 (0.25 pts.) - Show the proper workspace structure with the new Client, Common, Server, and Exceptions packages

Progress: 100%

```
 Folder PATH listing for volume OS
 Volume serial number is 8ADE-1F58
 C:\USERS\17323\DESKTOP\IT114\PROJECT
 |   ClientMain.java
 |   ClientReader.java
 |   compile_errors.txt
 |   ExampleException.java
 |   Payload.java
 |   PayloadType.java
 |   Room.java
 |   RoomManager.java
 |   ServerMain.java
 |   ServerThread.java
 |
 \---bin
PS C:\Users\17323\Desktop\It114\Project> ||
```

Project/src split into Client, Common, Server, Exceptions per Milestone 1 requirements.

Saved: 7/24/2025 7:17:14 PM

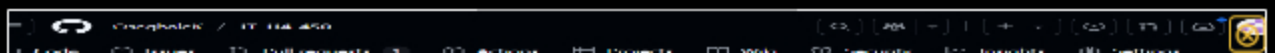
Task #2 (0.25 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

Details:
From the Commits tab of the Pull Request screenshot the commit history



Commits tab of Milestone1 PR showing PDF export and code restructure commits.

Saved: 7/24/2025 7:21:24 PM

Part 2:

Progress: 100%

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

URL #1
https://github.com/OseghaleK/IT-114-450/pull/1/Milestone1?search=1

URL
https://github.com/OseghaleK/IT-

Saved: 7/24/2025 7:21:24 PM

Task #3 (0.25 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

WakaTime project summary with total time and per-file breakdown for IT-114-450.

Saved: 7/24/2025 7:21:08 PM

Task #4 (0.25 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 build a socket-based client/server app in Java. Setting up ServerSocket.accept() and spawning a thread per client finally clicked. Using Payload objects kept commands organized, and managing rooms on the server taught me how to separate concerns cleanly. I also picked up a lot of practical Git/PowerShell skills (ports, classpaths, branches)

Saved: 7/24/2025 7:21:41 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:

Following the lesson template to start the server and client was straightforward. Once the baseline worked, adding commands like /name, /connect, /createroom, and /msg was mostly just packaging data in a payload and printing the server's response

Saved: 7/24/2025 7:22:16 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 environment/setup issues: wrong folders, wrong classpaths, and the port already being in use. I kept typing slash commands in PowerShell instead of inside the client. After I standardized the workflow (one terminal per role, always cd to root, use taskkill for ports) it got much easier

Saved: 7/24/2025 7:22:31 PM