

Submission Worksheet

Course: IT114-002-S2025

Assignment: IT114 Milestone 1

Student: Trinity C. (tdc28)

Status: Submitted | Worksheet Progress: 100.00%

Potential Grade: 10.00/10.00 (100.00%)

Received Grade: 0.00/10.00 (0.00%)

Grading Link: <https://learn.ethereallab.app/assignment/v3/IT114-002-S2025/it114-milestone-1/grading/tdc28>

Instructions

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
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
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

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: 100%

Objective: Evidence

≡, Image Prompt

Weight: 50%

Details:

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

tdc28, 04/09/2025 this is the terminal showing the server startd and is listening

```

    final String serverName = socket.getServerName();
    final String port = socket.getPort();
    final String connectionType = socket.getConnectionType();

    try (ServerSocket serverSocket = new ServerSocket(Integer.parseInt(port))) {
        System.out.println("Creating " + serverName + " on port " + port + " with connection type " + connectionType);
        serverSocket.bind(new InetSocketAddress(host, port)); // create the first room (lobby)
        while (!isRunning) {
            Info(message: "Waiting for next client");
            Socket incomingClient = serverSocket.accept(); // blocking call, waits for a client connection
            Info(message: "Client connected");
            new ClientThread(incomingClient, host, port).start(); // new thread, pass a callback to notify the Server when
            // they're initialized
            ServerThread serverThread = new ServerThread(incomingClient, this); // this is an ServerThreadInitHandler;
            // start the thread (typically an external entity manages the lifecycle and we
            // don't have the thread start itself)
        }
    } catch (IOException e) {
        Info(message: "An error occurred while trying to bind the socket: " + e.getMessage());
    }
}

```

tdc28.04/09/2025 - this is the code that waits for incoming connections.

Text Prompt

Weight: 50%

Details:

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

Your Response:

The server-side waits for connections by using ServerSocket to listen for new connections. Once the connection is accepted, a new thread is created for the client to use. The client then chooses a name and would have to type /connect localhost:3000 to join.



Saved: 4/9/2025 9:01:53 PM

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

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: 100%

Objective: Evidence

Image Prompt

Weight: 50%

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 client output for both the clients for multiple connections

- Show the relevant snippets of code that handle logic for multiple connections

```
04/09/2025 10:16:45 [Project.Server.Server] (TINFO):
> Server: *trinity#1 added to Lobby*
04/09/2025 10:18:55 [Project.Server.Server] (INFO):
> Server: Client connected
04/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
> Thread[-1]: ServerThread created
04/09/2025 10:18:55 [Project.Server.Server] (INFO):
> Server: Waiting for next client
04/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
```

tdc28, 04/09 - connection 1

```
Server: *matt#2 initialized*
4/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
> Thread[2]: Sending to client: Payload[ROOM_JOIN] Client Id [-1] Message: [null]
ClientName: [null]
4/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
> Thread[2]: Sending to client: Payload[SYNC_CLIENT] Client Id [1]
ssage: [null] ClientName: [trinity]
4/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[ROOM_JOIN] Client Id [2] Message: [null] ClientName: [matt]
4/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[MESSAGE] Client Id [0] Message: [Room[lobby] matt#2 joined the room]
4/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
> Thread[2]: Sending to client: Payload[ROOM_JOIN] Client Id [2] Message: [null] ClientName: [matt]
4/09/2025 10:18:55 [Project.Server.ServerThread] (INFO):
> Thread[2]: Sending to client: Payload[MESSAGE] Client Id [0] Message: [Room[lobby] You joined the room]
4/09/2025 10:18:55 [Project.Server.Server] (INFO):
```

tdc28, 04/09/2025 - connection 2

```
* Server: number#3 initialized
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[ROOM_JOIN] Client Id [-1] Message: [null]
ClientName: [null]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[SYNC_CLIENT] Client Id [1] Message: [null]
ClientName: [null]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[ROOM_JOIN] Client Id [3] Message: [null] ClientName: [emberil]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[MESSAGE] Client Id [0] Message: [Room[lobby] v1_emberil joined the room]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[ROOM_JOIN] Client Id [3] Message: [null] ClientName: [emberil]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Sending to client: Payload[MESSAGE] Client Id [0] Message: [Room[lobby] v1_You joined the room]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Received from my client: Payload[MESSAGE] Client id [3] Message: [Room[lobby] v1_You joined the room]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Received from my client: Payload[MESSAGE] Client id [3] Message: [Room[lobby] v1_You joined the room]
4/09/2025 10:45:38 [Project.Server.ServerThread] (INFO):
> Thread[1]: Received from my client: Payload[MESSAGE] Client id [3] Message: [Room[lobby] v1_You joined the room]
```

tdc28, 04/09/2025 - connection 3

```
Info(Message("Client connected"));
// we're back to our ServerThread, pass a callback to notify the Server when
// it's initialized
ServerThread serverThread = new ServerThread(incomingClient, this);
serverThread.initialize();
// Start the thread (typically an external entity manages the lifecycle and we
// don't have the thread start itself)
serverThread.start();
// Note we don't yet add the ServerThread reference to our connectedClients map
// yet so while it's running!
```

tdc28, 04/09/2025 - snippet 1

```

// add initialized client to the lobby
info(String.format("New initial client", serverThread.getDisplayName()));
try {
    joinRoom(Room.Lobby, serverThread);
    info(String.format("was added to Lobby", serverThread.getDisplayName()));
} catch (RoomNotFoundException e) {
    info(String.format("Error adding %s to Lobby", serverThread.getDisplayName()));
    e.printStackTrace();
}

```

tdc28, 04/09/2025 - snippet 2



```

/*
 * Relays the message from the sender to all rooms
 * Adding the synchronized keyword ensures that only one thread can execute
 * these methods at a time,
 * preventing concurrent modification issues and ensuring thread safety
 */
private synchronized void relayToAllRooms(ServerThread sender, String message) {
    // Note: any desired changes to the message must be done before this line
    String senderString = sender == null ? "Server" : sender.getDisplayName();
    // Note: formattedMessage must be final (or effectively final) since outside
    // scope can't change inside a callback function (see removeIt() below)
    final String formattedMessage = String.format("%s: %s", senderString, message);
    // end temp identifier
}

```

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

Task #1 (2 pts.) - Evidence

Combo Task:

Weight: 100%

Objective: Evidence

☞ Image Prompt

Weight: 50%

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)

tdc28, 04/09/2025 - snippet 1

tdc28, 04/09/2025 - snippet 2

```
0 Trinitychatmonymac tdc2A-1t114-002 % ./run.sh Project server  
04/09/2025 22:28:51 [Project.Server.Server] (INFO):  
~ Server starting  
04/09/2025 22:28:51 [Project.Server.Server] (INFO):  
~ Server: Listening on port 3000  
04/09/2025 22:28:51 [Project.Server.Room1] (INFO):  
~ Room lobby1 created  
04/09/2025 22:28:51 [Project.Server.Server] (INFO):  
~ Server: Created new Room lobby  
04/09/2025 22:28:51 [Project.Server.Server] (INFO):  
~ Server: Waiting for next client
```

tdc28, 04/09/2025 - terminal ss 1

tdc28, 04/09/2025 - terminal ss 2

```
04/09/2023 23:10:02 [Project.Server.ServerThread] (INFO): Thread[1]: Received from my Client: Payload[DISCONNECT] Client Id [0] Message: [null]
04/09/2023 23:10:02 [Project.Server.Room] (INFO): Room[Lobby]: sending message to 0 recipients: Room[lobby]: Trinity#1 disconnected
04/09/2023 23:10:02 [Project.Server.ServerThread] (INFO): Thread[1]: Client disconnected
04/09/2023 23:10:02 [Project.Server.ServerThread] (INFO): Thread[1]: ServerThread cleanup start
04/09/2023 23:10:02 [Project.Server.ServerThread] (INFO): Thread[1]: ServerThread cleanup end
```

```
04/07/2025 23:18:02 [Project, Server, ServerThread] (INFO):  
~ ThreadId=11: ServerThread cleanup() end  
04/07/2025 23:18:02 [Project, Server, ServerThread] (INFO):  
~ ThreadId=11: ServerThread cleanup() starting up connection  
04/07/2025 23:18:02 [Project, Server, ServerThread] (INFO):  
~ ThreadId=11: ServerThread cleanup() start  
04/07/2025 23:18:02 [Project, Server, ServerThread] (INFO):  
~ ThreadId=11: ServerThread cleanup() end  
04/07/2025 23:18:02 [Project, Server, ServerThread] (INFO):  
~ ThreadId=11: ServerThread cleanup() end
```

tdc28,04/09/2025 - terminal ss 3



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

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: 100%

Objective: Evidence

☞ Image Prompt

Weight: 50%

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

```
> Server: Client connected
04/09/2025 23:21:47 [Project.Server.ServerThread] (INFO):
> Thread-1-111 Server Thread created
04/09/2025 23:21:47 [Project.Server.Server1] (INFO):
> New Client Waiting for next client
04/09/2025 23:21:47 [Project.Server.ServerThread] (INFO):
> Thread-1-111 Thread starting
04/09/2025 23:21:47 [Project.Server.ServerThread] (INFO):
> Thread-1-111 Received from my client: Payload(CLIENT_CONNECT) Client Id: 101 Messages: 10 ClientName: Trinity
04/09/2025 23:21:47 [Project.Server.Server1] (INFO):
> Thread-1-111 Sending to clients: Payload(CLIENT_ID) Client Id: 101 Messages: 10 ClientName: Trinity
04/09/2025 23:21:47 [Project.Server.ServerThread] (INFO):
> Thread-1-111 Sending to clients: Payload(ROOM_JOIN) Client Id: 101 Messages: 10 ClientName: Trinity
04/09/2025 23:21:47 [Project.Server.Server1] (INFO):
> Thread-1-111 Sending to clients: Payload(ROOM_JOIN) Client Id: 101 Messages: 10 ClientName: Trinity
04/09/2025 23:21:47 [Project.Server.ServerThread] (INFO):
> Thread-1-111 Received from my client: Payload(CLIENT_MESSAGE) Client Id: 101 Messages: (Room lobby) You joined the room
04/09/2025 23:21:47 [Project.Server.Server1] (INFO):
```

tdc28, 04/09/2025 - terminal ss 1

```
04/09/2025 23:21:37 [Project.Client.Client] (INFO):
> Client starting
04/09/2025 23:21:37 [Project.Client.Client] (INFO):
> Waiting for input
/name Trinity
04/09/2025 23:21:41 [Project.Client.Client] (INFO):
> Name set to Trinity
/connect localhost:3000
04/09/2025 23:21:47 [Project.Client.Client] (INFO):
> Client connected
04/09/2025 23:21:47 [Project.Client.Client] (INFO):
> Connected
04/09/2025 23:21:47 [Project.Client.Client] (INFO):
> Room[lobby] You joined the room
HELLOOOOO EARTHLINGS
04/09/2025 23:21:57 [Project.Client.Client] (INFO):
> Trinity#1: HELLOOOOO EARTHLINGS
```

tdc28, 04/09/2025 - terminal ss2

```
private void onServerThreadToClientEvent(ServerThread serverThread, ServerEvent event) {
    // Generate Server controlled ClientId
    nextClientId = Math.max(nextClientId, 0);
    serverThread.setClientId(nextClientId);
    serverThread.sendData("101"); // sync the data to the client
    addInitializedClientToTheLobby();
    info(String.format("was initialized", serverThread.getDisplayName()));
    try {
        info(Room.lobby, serverThread);
        info(String.format("was added to lobby", serverThread.getDisplayName()));
    } catch (IllegalStateException e) {
        info(String.format("already adding %s to lobby", serverThread.getDisplayName()));
        e.printStackTrace();
    }
}
--> private void onServerThreadInitialized(ServerThread serverThread)
```

tdc28, 04/09/2025 - snippet of code 1

```
info("Waiting for next client");
Socket incomingClient = serverSocket.accept(); // blocking action, waits for a client connection
info("Client connected");
// wrap socket in a ServerThread, pass a callback to notify the Server when
// the Client has connected
ServerThread serverThread = new ServerThread(incomingClient, this.onServerThreadInitialized);
// start the thread (possibly an external utility manages the lifecycle and we
// don't have the thread start itself)
serverThread.start();
// add the serverThread reference to our connectedClients map
--> --> while (isRunning)
```

tdc28, 04/09/2025 - snippet of code 2



Saved: 4/9/2025 11:55:36 PM

=, Text Prompt

Weight: 50%

Details:

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

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

Task #1 (2 pts.) - Evidence

Combo Task:

Weight: 100%

Objective: Evidence

☞ Image Prompt

Weight: 50%

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

```
Connected
04/09/2025 23:56:06 [Project.Client.Client] (INFO):
=> Room[lobby] You joined the room
/createroom leo
04/09/2025 23:56:12 [Project.Client.Client] (INFO):
=> Room[lobby] You left the room
04/09/2025 23:56:12 [Project.Client.Client] (INFO):
=> Room[leo] You joined the room
/createroom sagittarius
04/09/2025 23:56:37 [Project.Client.Client] (INFO):
=> Room[leo] You left the room
04/09/2025 23:56:37 [Project.Client.Client] (INFO):
=> Room[sagittarius] You joined the room
/joinroom leo
04/09/2025 23:57:35 [Project.Client.Client] (INFO):
=> Room leo doesn't exist
yes it does
04/09/2025 23:57:55 [Project.Client.Client] (INFO):
=> Trinity#2: yes it does
lol
04/09/2025 23:57:57 [Project.Client.Client] (INFO):
=> Trinity#2: lol
```

tdc28, 04/09/2025 - terminal /joinroom & /createroom

```
protected void createBsd(String name) throws BsdCreateException {
    final String bsdName = name.substring(0, 1).toUpperCase() + name.substring(1);
    if (!BsdKey.isBsdKey(bsdName)) {
        Util.create(new DataInputStream(new ByteArrayInputStream("".getBytes())));
        Bsd bsd = new Bsd(name);
        bsd.setBsdName(bsdName);
        DATA.setString(FormatFormat.format("created new Bsd %s", name));
    } else {
        Util.create(new DataInputStream(new ByteArrayInputStream("".getBytes())));
        DATA.setString(FormatFormat.format("Bsd %s already exists", name));
    }
}
```

tdc28, 04/09/2025 - /createroom code

```

protected void joinRoom(String name, ServerThread client) throws RoomNotfoundException {
    String roomName = name.substring(0, name.indexOf('@'));
    if (!rooms.containsKey(roomName)) {
        Room room = new Room();
        room.setRoomName(roomName);
        rooms.put(roomName, room);
    }
    Room currentRoom = client.getCurrentRoom();
    if (currentRoom != null) {
        info("Removing client from previous Room " + currentRoom.getName());
        currentRoom.removeClient(client);
    }
    Room newRoom = rooms.get(name.substring(0, name.indexOf('@')));
    next.addClient(client);
}

```

tdc28.04/09/2025 - /joinroom code

tdc28,04/09/2025 - /joinroom, /createroom terminal pt.2



Saved: 4/10/2025 12:45:06 AM

Text Prompt

Weight: 50%

Details:

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

Messages

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: 100%

Objective: Evidence

≡, Image Prompt

Weight: 50%

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

```
04/10/2025 00:47:13 [Project.Client.Client] (INFO): ~ Name set to jannette
/connected localhost:3000
04/10/2025 00:47:19 [Project.Client.Client] (INFO): ~ Client connected
04/10/2025 00:47:19 [Project.Client.Client] (INFO): ~ Connected
04/10/2025 00:47:19 [Project.Client.Client] (INFO): ~ Room|lobby| You joined the room
04/10/2025 00:47:23 [Project.Client.Client] (INFO): ~ jcunelli#441 Jvc arrived
04/10/2025 00:47:40 [Project.Client.Client] (INFO): ~ Room|lobby| Trinity#3 joined the room
04/10/2025 00:47:40 [Project.Client.Client] (INFO): ~ Trinity#3: /you've arrived? finally
04/10/2025 00:47:52 [Project.Client.Client] (INFO): ~ Trinity#3: where the other guy
04/10/2025 00:48:45 [Project.Client.Client] (INFO): ~ Room|lobby| leonardo_da_vinci#5 joined the room
04/10/2025 00:48:53 [Project.Client.Client] (INFO): ~ leonardo_da_vinci#5: ouui ouui i have arrived
04/10/2025 00:49:15 [Project.Client.Client] (INFO): ~ leonardo_da_vinci#5: my name is leonardo, i am a renaissance man
```

tdc28.04/09/2025 - terminal msgs

tdc28,04/09/2025 - clients grouped into other rooms pt.1

```

>VVVVVVVVV
04/10/2025 00:55:15 [Project.Client.Client] (INFO):
> Room[lobby] You joined the room
/createroom Trinity
04/10/2025 00:55:15 [Project.Client.Client] (INFO):
> Room[lobby] You left the room
04/10/2025 00:55:24 [Project.Client.Client] (INFO):
> Room[aries] You joined the room
hi!!!!
04/10/2025 00:55:26 [Project.Client.Client] (INFO):
= Trinity#6: hi!!!!

```

tdc28,04/09/2025 - example of client in other rooms pt.2

```

04/10/2025 00:52:38 [Project.Client.Client] (INFO):
> Room[lobby] You left the room
04/10/2025 00:52:38 [Project.Client.Client] (INFO):
> Room[leo] You joined the room
hi
04/10/2025 00:52:43 [Project.Client.Client] (INFO):
= jeanette#4: hi
□

```

tdc28,04/09/2025 - example of client in other rooms pt.3

```

04/10/2025 00:48:45 [Project.Client.Client] (INFO):
> Room[lobby] You joined the room
out out I have arrived
04/10/2025 00:48:53 [Project.Client.Client] (INFO):
> Leonardo da Vinci#5: out but I have arrived
my name is Leonardo, I am a renaissance man
04/10/2025 00:49:15 [Project.Client.Client] (INFO):
= Leonardo da Vinci#5: my name is Leonardo, I am a renaissance
man
04/10/2025 00:52:30 [Project.Client.Client] (INFO):
> Room[lobby] jeanette#4 left the room
04/10/2025 00:52:59 [Project.Client.Client] (INFO):
> Room[lobby] Trinity#6 left the room
/createroom picasso
04/10/2025 00:53:28 [Project.Client.Client] (INFO):
> Room[lobby] You left the room
04/10/2025 00:53:28 [Project.Client.Client] (INFO):
> Room[picasso] You joined the room
boom boom boom boom
04/10/2025 00:59:42 [Project.Client.Client] (INFO):
= Leonardo da Vinci#5: boom boom boom boom
■

```

tdc28,04/09/2025 - example of client in different rooms pt.4

```

/*
private synchronized void relayToAllRooms(ServerThread sender, String message) {
    // Note: we can't add a scope to the message since it must be the same for all rooms
    String senderString = sender.getName() + " (" + sender.getDisplayName() + ")";
    // Note: formattedMessage must be final (or effectively final) since outside
    // scope can't change inside a callback function (see removeIf() below)
    final String formattedMessage = String.format(format:"%s: %s", senderString, message);
    // end temp identifier

    // Loop over Room and send out the message
    // Note: this uses a lambda expression for each item in the values() collection
    rooms.values().forEach(room -> {
        room.relay(sender, formattedMessage);
    });
}
*/
private synchronized void relayToAllRooms(ServerThread sender...

```

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

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: 100%

Objective: Evidence

≡, Image Prompt

Weight: 50%

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 occurring
 - 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

2024/10/20 10:11:53 [Project.Server.Server] (INFO): Server: JVM is shutting down. Please cleanup tasks.

tdc28,04/09/2025 - server shutting down ss1

```
84/10/2025 10:11:53 [Project.Client.Client] (WARNING):
  Connection_dropped.
java.io.EOFException
at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1570)
at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:430)
at Project.C1.readObject(C1.java:10)
at java.util.concurrent.CompletableFuture$AsynchronousFuture$1.run(CompletableFuture.java:1848)
at java.base/java.util.concurrent.CompletableFuture$AsynchronousFuture$1.run(CompletableFuture.java:1846)
at java.base/java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:597)
at java.base/java.util.concurrent.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1460)
at java.base/java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:2030)
at java.base/java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:109)
84/10/2025 10:11:53 [Project.Client.Client] (INFO):
> Closing output Stream
84/10/2025 10:11:53 [Project.Client.Client] (INFO):
Closing input Stream
84/10/2025 10:11:53 [Project.Client.Client] (INFO):
> Closing connection
84/10/2025 10:11:53 [Project.Client.Client] (INFO):
84/10/2025 10:11:53 [Project.Client.Client] (INFO):
> listeningServer thread stopped
```

tdc28.04/09/2025 - server shutting down ss 2

```
infinitychatmanager: tde28 41114 860 N ./run.sh Project see  
ver:  
04/10/2025 10:14:40 [Project.Server.Server] (INFO):  
+ Server Starting up...  
04/10/2025 10:14:40 [Project.Server.Server] (INFO):  
+ SERVER: Listening on port 3000  
04/10/2025 10:14:40 [Project.Server.Bnnm] (TINFO):  
+ Manager lobby created  
04/10/2025 10:14:40 [Project.Server.Server] (INFO):  
+ Server Created new Room lobby  
04/10/2025 10:14:40 [Project.Server.Server] (INFO):  
+ SERVER: lobby created  
04/10/2025 10:15:01 [Project.Server.Server] (TINFO):  
+ Server Client connected  
04/10/2025 10:15:01 [Project.Server.ServerThread] (INFO):  
+ Thread-11 ServerThread created  
04/10/2025 10:15:01 [Project.Server.Server] (INFO):  
+ SERVER: waiting for client  
04/10/2025 10:15:01 [Project.Server.ServerThread] (TINFO):  
+ Thread-11 Thread starting  
04/10/2025 10:15:01 [Project.Server.ServerThread] (INFO):  
+ Thread-11 Client-11 Received CLIENT_CONNECT_CLIENT_TO_IP Multiplayer Thread-11 Client(Numerus) ITs In
```

tdc28,04/09/2025 - client reconnecting ss 1

tdc28,04/09/2025 - client reconnecting pt.2

```
/* Sends a message to the server */
private void sendDisconnect() throws IOException {
    Payload payload = new Payload();
    payload.setPayloadType(PayloadType.DISCONNECT);
    sendRawData(payload);
}

private void sendDisconnect() throws IOException {
    /* Sends a message to the server */
}
```

tdc28, 04/09/2025 - client disconnect ss1

```
private void processBlockHeader(Header header) {
    if(header.getSectionID() == Header.getSectionID()) {
        currentClient.setClientID(header.getClientID());
        suggestedSectionID = header.getSectionID();
        suggestedSectionName = header.getSectionName();
        > write(JF_CurrentClient.setClientHeader(header, sectionID));
        if(header.isLastSection() == null || header.isLastSection() != true)
            & (blockHeader.isLastSection() == null || blockHeader.isLastSection() != true);
        suggestedSectionName = header.getSectionName();
        & (blockHeader.getSectionName() == null || blockHeader.getSectionName() != header.getSectionName());
    }
    & (header.getSectionID() == Header.getSectionID());
    & (header.getClientID() == currentClient.getClientID());
}
```

Section #8: (1 pt.) Misc

Task #1 (0.25 pts.) - Show the proper workspace structure with the

Image Prompt

Weight: 25%

Objective: Show the proper workspace structure with the new Client, Common, Server, and Exceptions packages



tdc28, 04/09/2025 - ss1



Task #2 (0.25 pts.) - Github Details

Combo Task:

Weight: 25%

Objective: Github Details

Image Prompt

Weight: 60%

Details:

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

```
git log --oneline --graph --all --date=iso --format='* %Cred%h %ad %s' --abbrev-commit
commit 61f114d8623e0f1179005a20mc2010 (HEAD --> MILESTONEone, origin/MILESTONEone, main)
Author: Tariq Alabdullah <tariq.alabdullah@tariqabdullah.com>
Date:   Thu Apr 19 12:00:38 2025 +0000

Revert "I can't get this to work"
This reverts commit 2105f7932ef0eb4592c3dc3c6fe0c3b59ebc6ed.

commit 2105f7932ef0eb4592c3dc3c6fe0c3b59ebc6ed
Author: Tariq Alabdullah <tariq.alabdullah@tariqabdullah.com>
Date:   Thu Apr 19 12:01:43 2025 +0000

I can't get this to work
commit da0ec5fffd0065000000003340656ab6a437c1d0e5
Author: Tariq Alabdullah <tariq.alabdullah@tariqabdullah.com>
Date:   Thu Apr 19 12:00:38 2025 +0000
```

tdc28, 04/10/2025 - commits



Saved: 4/10/2025 1:52:43 PM

🔗 Url Prompt

Weight: 40%

Details:

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

URL #1

<https://github.com/trinitydessyna/tdc28-it114-p0025>



URL

<https://github.com/trinitydessyna/tdc28-it114-p0025>



Saved: 4/10/2025 1:52:43 PM

Task #3 (0.25 pts.) - WakaTime - Activity

🖼 Image Prompt

Weight: 25%

Objective: *WakaTime - Activity*

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	Lines
Project/Server/Server.java	110
Project/Common/Client.java	23
Project/Common/Client.java	20
Project/Common/Client.java	20
Project/Server/ServerThread.java	19
Project/Client.java	18
run.sh	14
Project/Server/Room.java	14
Project/Common/ServerThread.java	13
server.Glog	6
build.sh	2
Project/Common/RoomAction.java	2
Project/User.java	1
Project/Common/Room.java	1
Project/Client/Client.java	1
Project/Common/TestFix.java	1
Project/Common/CommonDataPayload.java	1
./Promotion/CustomT14Promotion.java	1
Usage.java	0

Task #4 (0.25 pts.) - Reflection

Weight: 25%

Objective: *Reflection*

Sub-Tasks:

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

Text Prompt

Weight: 33.33%

Objective: *What did you learn?*

Details:

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

Your Response:

I learned how to identify what different pieces of code do. I also learned how to essentially create a chatroom (granted, I still have to learn how to code each file it takes to do so) and how to run the entire "Project".



Saved: 4/10/2025 8:47:53 PM

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

≡, Text Prompt

Weight: 33.33%

Objective: *What was the easiest part of the assignment?*

Details:

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

Your Response:

The easiest part of the assignment was honestly screenshotting the snippets of code and the terminal, adding comments to the screenshots, and then publishing them here.



Saved: 4/10/2025 8:46:40 PM

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

≡, Text Prompt

Weight: 33.33%

Objective: *What was the hardest part of the assignment?*

Details:

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

Your Response:

The hardest part was identifying which parts of the code needed to be screenshotted. As someone who is still kind of new to learning Java, everything looked somewhat the same to me, and it took me reading the code over and over again and even searching up which part meant what to be able to identify which part was which and what it did.



Saved: 4/10/2025 8:49:41 PM