Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-project-milestone-1/grade/ns87

IT114-002-S2024 - [IT114] Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 3/15/2024 8:17:35 PM

Instructions

COLLAPSE

Create a new branch called Milestone1
At the root of your repository create a folder called Project if one doesn't exist yet
- You will be updating this folder with new code as you do milestones
- You won't be creating separate folders for milestones; milestones are just branches

Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
Copy in the latest Socket sample code from the most recent Socket Part example of the lessons
- Recommended Part 5 (clients should be having names at this point and not ids)
- https://github.com/MattToegel/IT114/tree/Module5/Module5

Fix the package references at the top of each file (these are the only edits you should do at this point)
Git add/commit the baseline and push it to github
Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
Ensure the sample is working and fill in the below deliverables
- Note: The client commands likely are different in part 5 with the /name and /connect options instead of just "connect"

Generate the worksheet output file once done and add it to your local repository
Git add/commit/push all changes
Complete the pull request merge from step 7
Locally checkout main
git pull origin main

Branch name: Milestone1

Tasks: 9 Points: 10.00

● Start Up (3 pts.)

COLLAPSE

^COLLAPSE^

Task #1 - Points: 1

Text: Server and Client Initialization

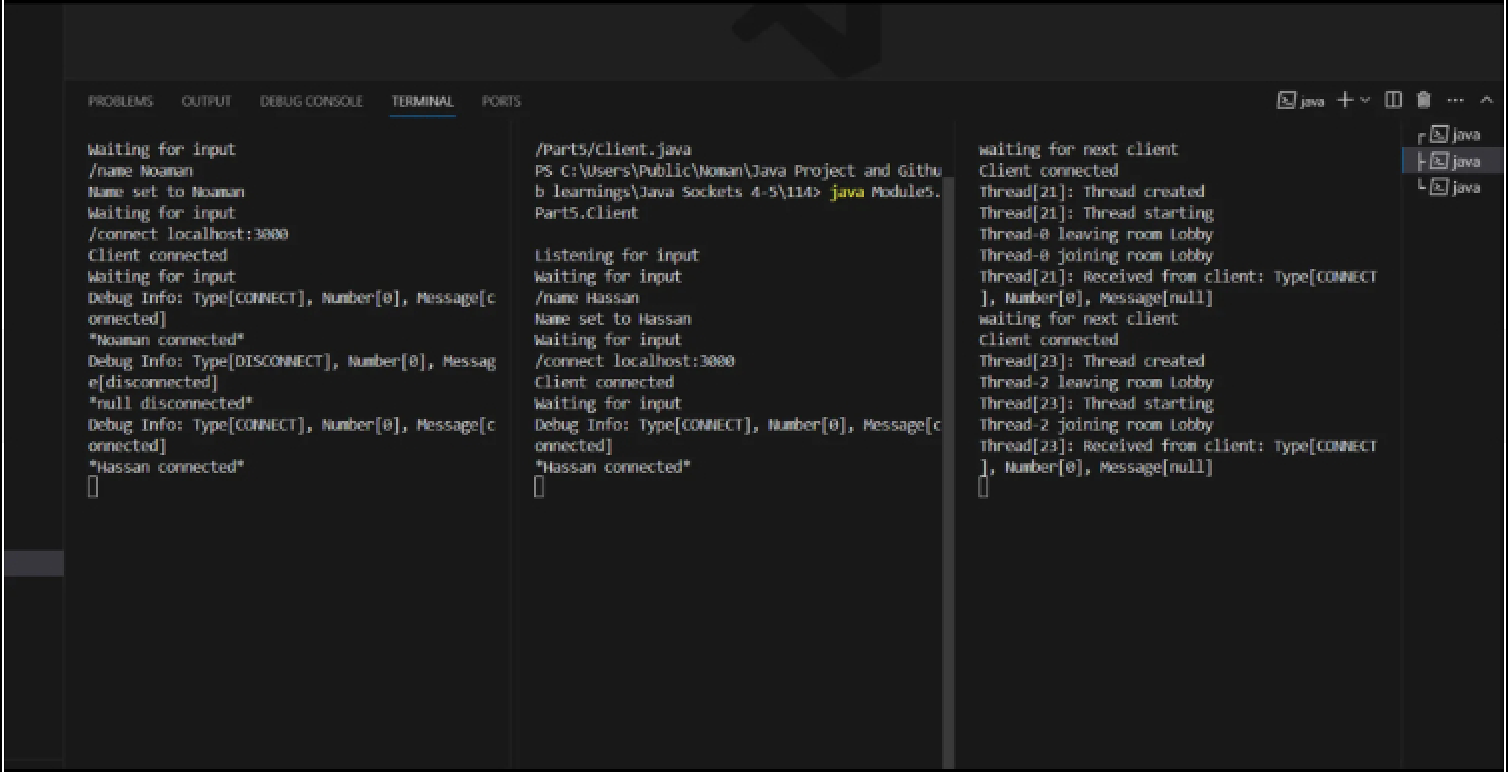
Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Server should properly be listening to its port from the command line (note the related message)
☐ #2	1	Clients should be successfully waiting for input
☐ #3	1	Clients should have a name and successfully connected to the server (note related messages)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
Waiting for input
/name Noaman
Name set to Noaman
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Noaman connected*
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
```

```
/Part5/Client.java
PS C:\Users\Public\Noman\Java Project and Githu
b learnings\Java Sockets 4-5\114> java Module5.
Part5.Client

Listening for input
Waiting for input
/name Hassan
Name set to Hassan
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
```

```
waiting for next client
Client connected
Thread[21]: Thread created
Thread[21]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[21]: Received from client: Type[CONNECT
], Number[0], Message[null]
waiting for next client
Client connected
Thread[23]: Thread created
Thread-2 leaving room Lobby
Thread[23]: Thread starting
Thread-2 joining room Lobby
Thread[23]: Received from client: Type[CONNECT
], Number[0], Message[null]
```

A server listens to its port from the command line. Clients successfully wait for input. Clients have a name and are connected to the server.

Checklist Items (0)

^COLLAPSE^

Task #2 - Points: 1

Text: Explain the connection process

Details:

Note the various steps from the beginning to when the client is fully connected and able to communicate in the room.

Emphasize the code flow and the sockets usage.

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Mention how the server-side of the connection works
☐ #2	1	Mention how the client-side of the connection works
☐ #3	1	Describe the socket steps until the server is waiting for messages from the client

Response:

Server-Side Connection:

The server listens for incoming connections on a specified port using a ServerSocket.
Once a client connects, the server accepts the connection and creates a new ServerThread to handle that client's communication.
The ServerThread communicates with the client over the socket using ObjectInputStream and ObjectOutputStream.

Client-Side Connection:

The client initiates a connection to the server by creating a socket and providing the server's port.
After establishing the connection, the client creates an ObjectOutputStream and ObjectInputStream to send and receive objects over the socket

Socket Steps Until Server Waits for Messages:

The server creates a ServerSocket and starts listening on a specified port.
The server enters a loop to accept incoming client connections using serverSocket.accept().
When a client connects, the server creates a new ServerThread to handle that client.
Inside the ServerThread, the server sets up an ObjectOutputStream and ObjectInputStream to communicate with the client.
The server then enters a loop to read incoming messages from the client using in.readObject(), processing each message accordingly.

In summary, the server listens for connections using a ServerSocket, and when a client connects, it creates a separate thread (ServerThread) to handle that client's communication. On the client side, it establishes a connection to the server using a socket and sets up streams for communication. This allows both the server and client to send and receive messages over the network.

● Communication (3 pts.)

^COLLAPSE ^

●

^COLLAPSE ^

Task #1 - Points: 1

Text: Add screenshot(s) showing evidence related to the checklist

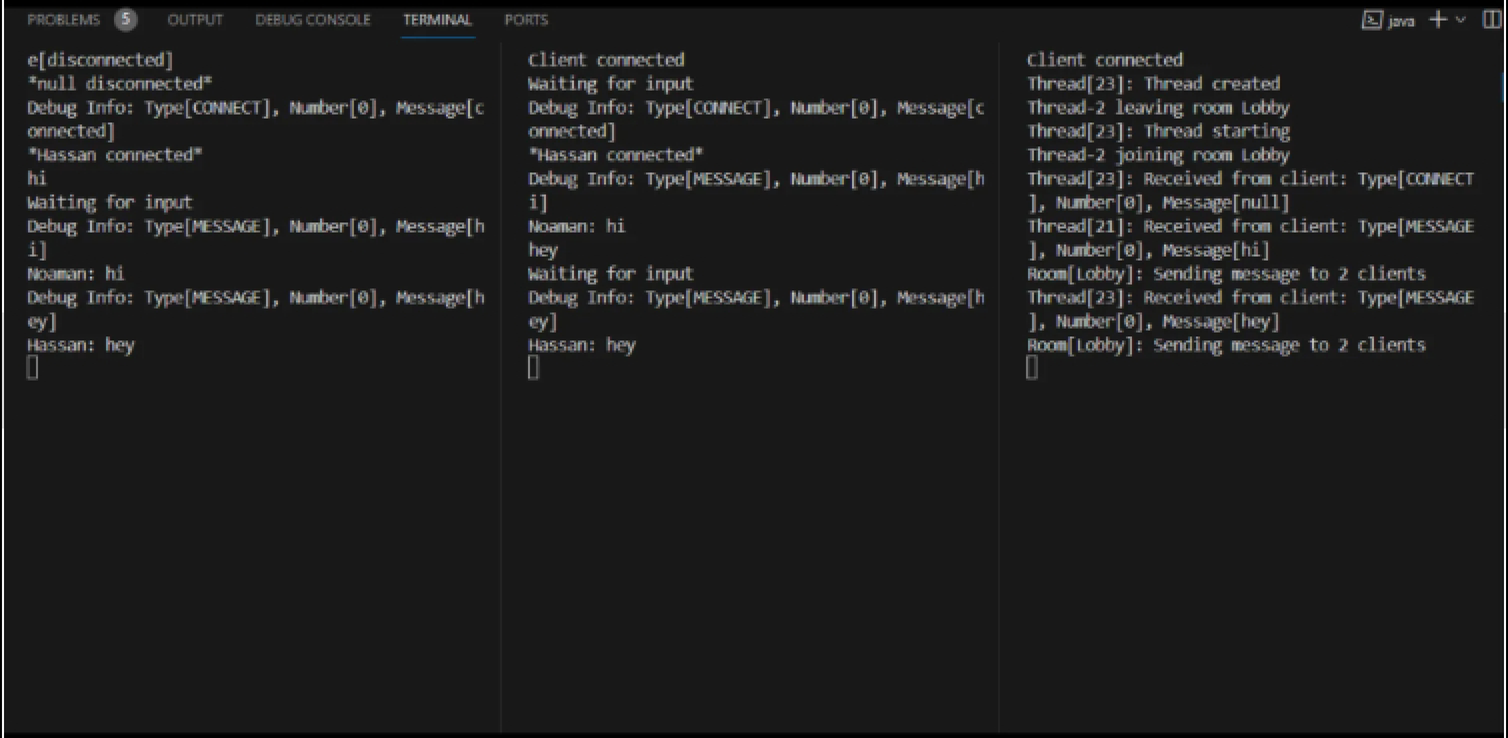
Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	At least two clients connected to the server
☐ #2	1	Client can send messages to the server
☐ #3	1	Server sends the message to all clients in the same room
☐ #4	1	Messages clearly show who the message is from (i.e., client name is clearly with the message)
☐ #5	2	Demonstrate clients in two different rooms can't send/receive messages to each other (clearly show the clients are in different rooms via the commands demonstrated in the lessons
☐ #6	1	Clearly caption each image regarding what is being shown

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
e[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
Debug Info: Type[MESSAGE], Number[0], Message[h
ey]
Hassan: hey
```

```
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
hey
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
ey]
Hassan: hey
```

```
Client connected
Thread[23]: Thread created
Thread-2 leaving room Lobby
Thread[23]: Thread starting
Thread-2 joining room Lobby
Thread[23]: Received from client: Type[CONNECT
], Number[0], Message[null]
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[hi]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[hey]
Room[Lobby]: Sending message to 2 clients
```

Server sends the message to all clients in the same room

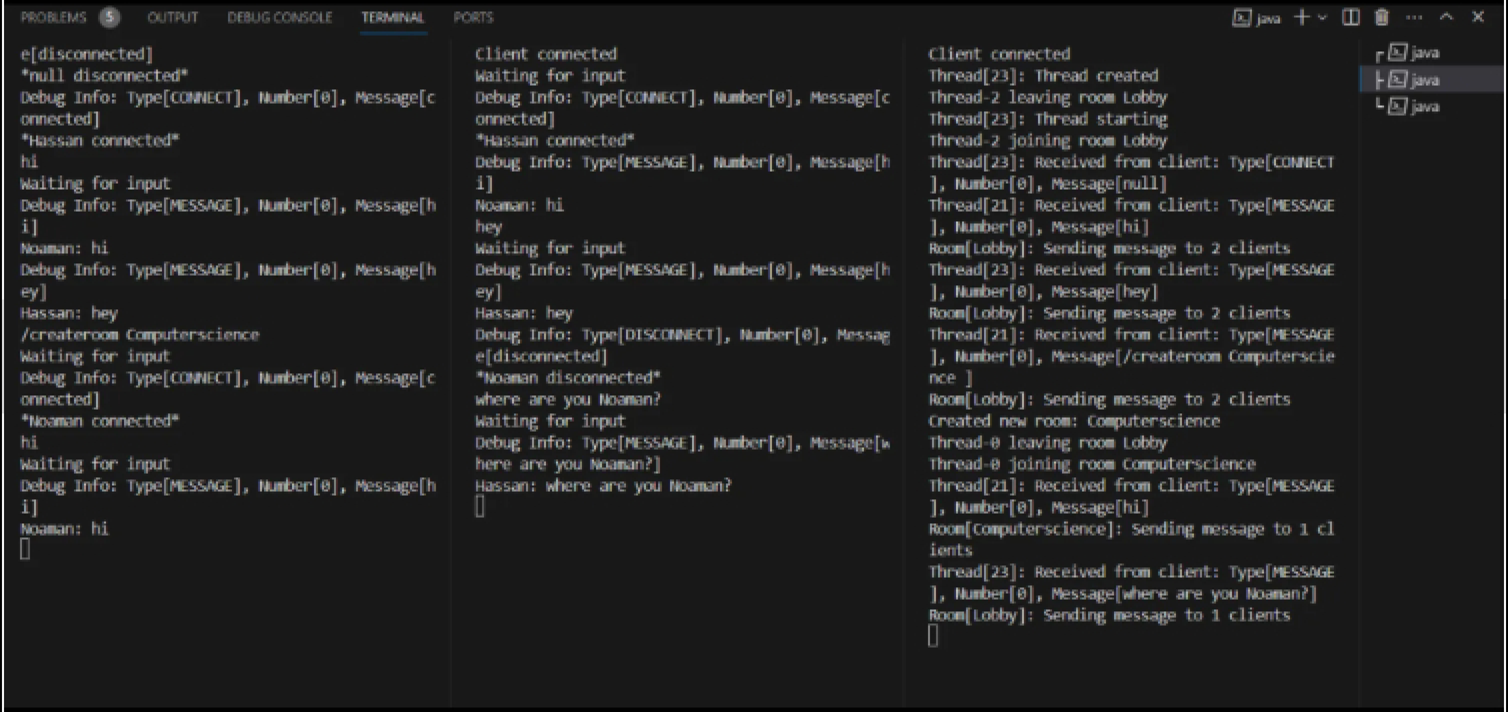
Checklist Items (4)

#1 At least two clients connected to the server

#2 Client can send messages to the server

#3 Server sends the message to all clients in the same room

#4 Messages clearly show who the message is from (i.e., client name is clearly with the message)



```
e[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
Debug Info: Type[MESSAGE], Number[0], Message[h
ey]
Hassan: hey
/createroom Computerscience
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Noaman connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
```

```
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
hey
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
ey]
Hassan: hey
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*Noaman disconnected*
where are you Noaman?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[w
here are you Noaman?]
Hassan: where are you Noaman?
```

```
Client connected
Thread[23]: Thread created
Thread-2 leaving room Lobby
Thread[23]: Thread starting
Thread-2 joining room Lobby
Thread[23]: Received from client: Type[CONNECT
], Number[0], Message[null]
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[hi]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[hey]
Room[Lobby]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[/createroom Computerscie
nce ]
Room[Lobby]: Sending message to 2 clients
Created new room: Computerscience
Thread-0 leaving room Lobby
Thread-0 joining room Computerscience
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[hi]
Room[Computerscience]: Sending message to 1 cl
ients
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[where are you Noaman?]
Room[Lobby]: Sending message to 1 clients
```

Noaman is in Room Computerscience both the clients can not send/receive messages to each other. The server shows that Room [Computerscience] sends 1 message while client in lobby also send 1 message to the client.

Checklist Items (0)

^COLLAPSE ^

Task #2 - Points: 1

Text: Explain the communication process

Details:
How are messages entered from the client side and how do they propagate to other clients?

Note all the steps involved and use specific terminology from the code.
Don't just translate the code line-by-line to plain English, keep it concise.

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention the client-side (sending)
#2	1	Mention the ServerThread's involvement

#3	1	Mention the Room's perspective
#4	1	Mention the client-side (receiving)

Response:

Client-Side (Sending):

The client enters a message via the keyboard input stream.
The client's Client class processes the message and sends it as a Payload object to the server.
The Client class uses an ObjectOutputStream to send the Payload object to the server over the established socket connection.
ServerThread's Involvement:

On the server side, each connected client is managed by a separate ServerThread.
The ServerThread listens for incoming Payload objects from its respective client using an ObjectInputStream.
When a Payload object is received, the ServerThread processes it and takes appropriate actions based on the payload type (e.g., sending a message to the room, handling connect/disconnect events).
Room's Perspective:

The Room class manages a group of clients within the same chat room.
When a ServerThread receives a message from its client, it forwards that message to the respective Room object.
The Room object then broadcasts the message to all clients in the same room by using the sendMessage method of each connected ServerThread.

Client-Side (Receiving):

On the receiving client's side, its respective ServerThread receives the message from the server.
The ServerThread processes the incoming message and sends it to the client's Client class.
The Client class displays the received message to the user via the console.

In summary, messages entered by the client are sent to the server as Payload objects via an ObjectOutputStream. The server's ServerThread processes these objects and forwards messages to the appropriate Room, which then broadcasts the messages to all clients in that room. On the receiving end, clients' ServerThread instances handle incoming messages from the server and pass them to the client for display.

● Disconnecting/Termination (3 pts.)

^COLLAPSE ^

● ^COLLAPSE ^

Task #1 - Points: 1

Text: Add screenshot(s) showing evidence related to the checklist

Checklist *The checkboxes are for your own tracking

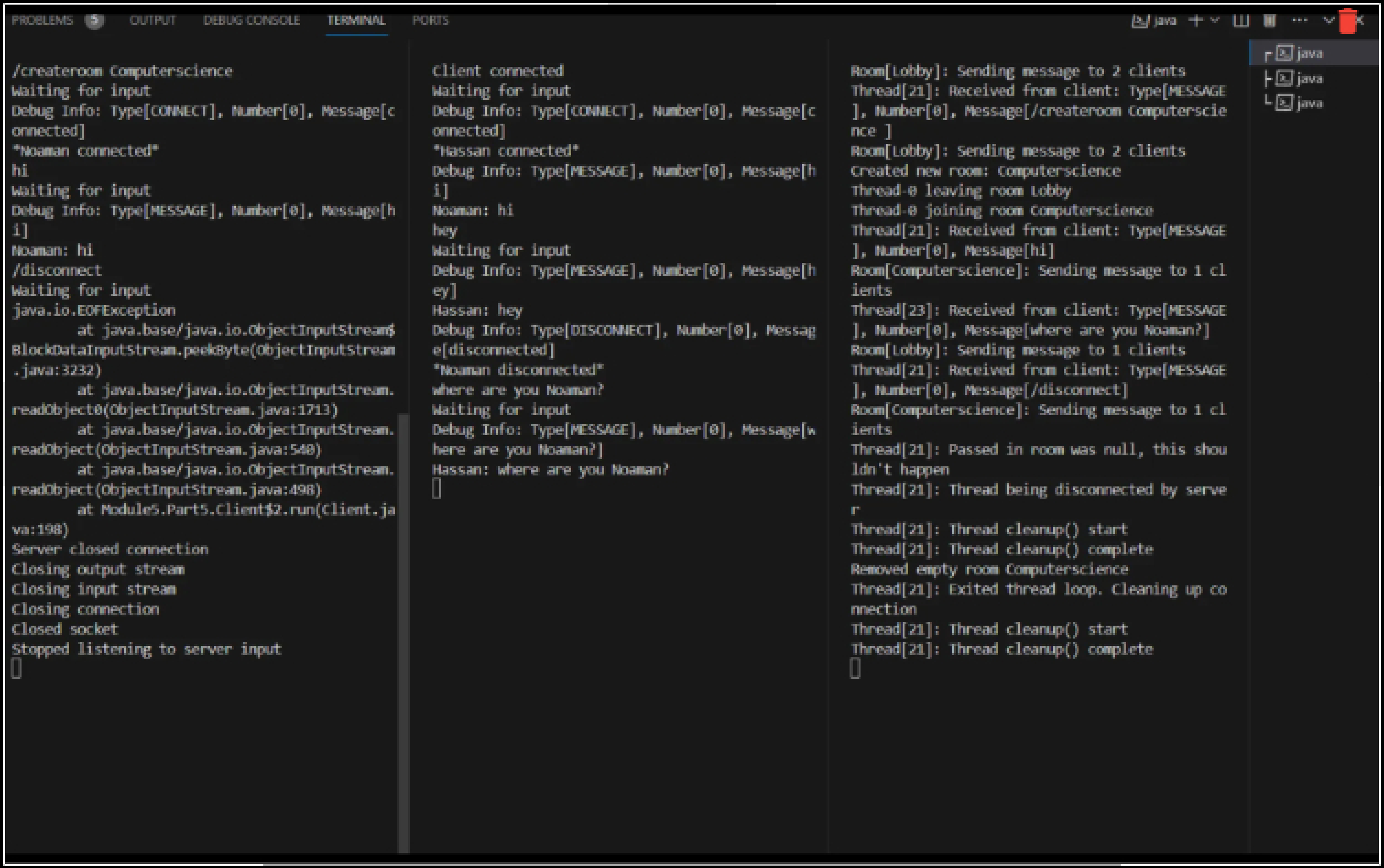
#	Points	Details
#1	1	Show a client disconnecting from the server; Server should still be running without issue (it's ok if an exception message shows as it's part of the lesson code, the server just shouldn't terminate)
		Show the server terminating; Clients should be disconnected but still running and

#2	1	able to reconnect when the server is back online (demonstrate this)
#3	1	For each scenario, disconnected messages should be shown to the clients (should show a different person disconnected and should show the specific client disconnected)
#4	1	Clearly caption each image regarding what is being shown

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
/createroom Computerscience
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Noaman connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$
BlockDataInputStream.peekByte(ObjectInputStream
.java:3232)
        at java.base/java.io.ObjectInputStream.
readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.
readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.
readObject(ObjectInputStream.java:498)
        at Module5.Part5.Client$2.run(Client.ja
va:198)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
```

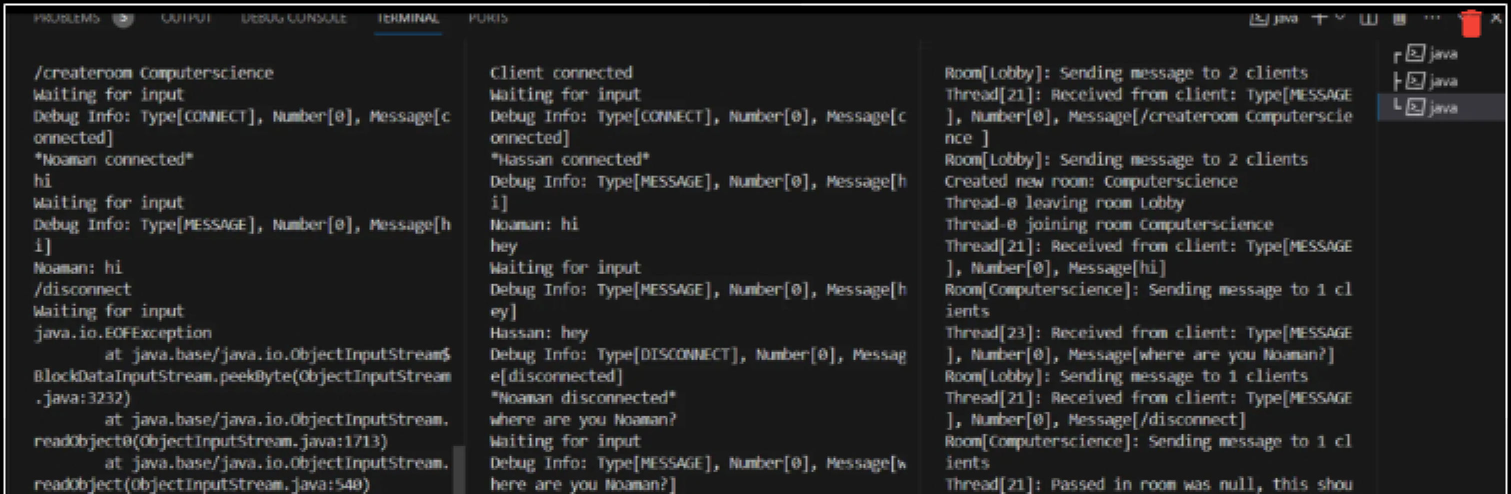
```
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
hey
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
ey]
Hassan: hey
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*Noaman disconnected*
where are you Noaman?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[w
here are you Noaman?]
Hassan: where are you Noaman?
```

```
Room[Lobby]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[/createroom Computerscie
nce ]
Room[Lobby]: Sending message to 2 clients
Created new room: Computerscience
Thread-0 leaving room Lobby
Thread-0 joining room Computerscience
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[hi]
Room[Computerscience]: Sending message to 1 cl
ients
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[where are you Noaman?]
Room[Lobby]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[/disconnect]
Room[Computerscience]: Sending message to 1 cl
ients
Thread[21]: Passed in room was null, this shou
ldn't happen
Thread[21]: Thread being disconnected by serve
r
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Removed empty room Computerscience
Thread[21]: Exited thread loop. Cleaning up co
nnection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
```

client disconnecting from the server.

Checklist Items (1)

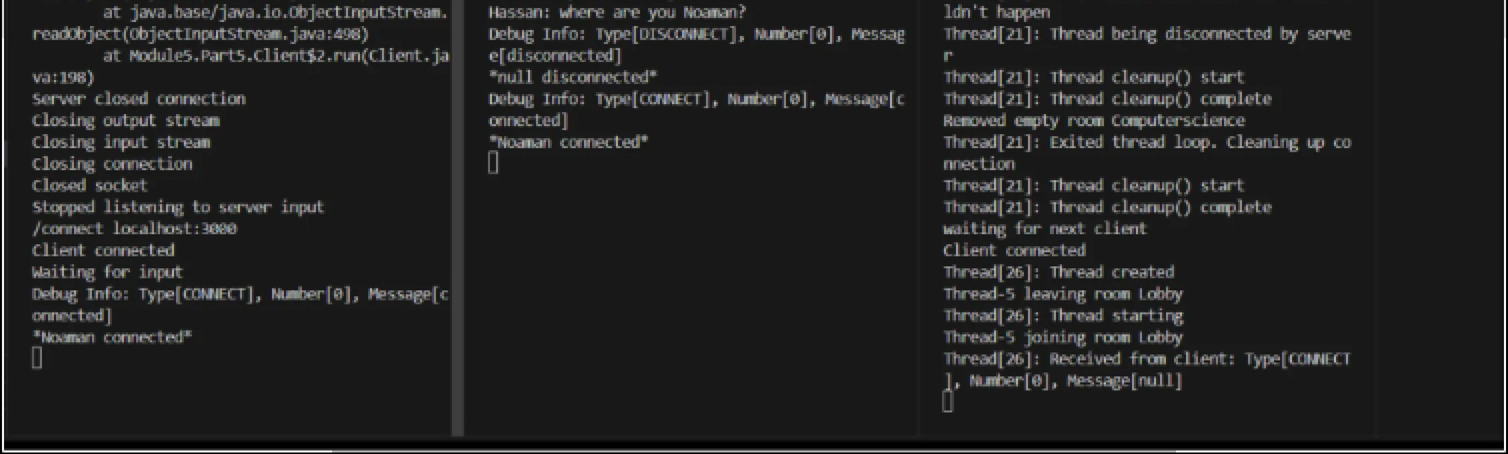
#1 Show a client disconnecting from the server; Server should still be running without issue (it's ok if an exception message shows as it's part of the lesson code, the server just shouldn't terminate)



```
/createroom Computerscience
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Noaman connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$
BlockDataInputStream.peekByte(ObjectInputStream
.java:3232)
        at java.base/java.io.ObjectInputStream.
readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.
readObject(ObjectInputStream.java:540)
```

```
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Hassan connected*
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Noaman: hi
hey
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
ey]
Hassan: hey
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*Noaman disconnected*
where are you Noaman?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[w
here are you Noaman?]
```

```
Room[Lobby]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[/createroom Computerscie
nce ]
Room[Lobby]: Sending message to 2 clients
Created new room: Computerscience
Thread-0 leaving room Lobby
Thread-0 joining room Computerscience
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[hi]
Room[Computerscience]: Sending message to 1 cl
ients
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[where are you Noaman?]
Room[Lobby]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[/disconnect]
Room[Computerscience]: Sending message to 1 cl
ients
Thread[21]: Passed in room was null, this shou
```



```
        at java.base/java.io.ObjectInputStream.
readObject(ObjectInputStream.java:498)
        at Module5.Part5.Client$2.run(Client.ja
va:198)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Noaman connected*
```

```
Hassan: where are you Noaman?
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Noaman connected*
```

```
ldn't happen
Thread[21]: Thread being disconnected by serve
r
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Removed empty room Computerscience
Thread[21]: Exited thread loop. Cleaning up co
nnection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
waiting for next client
Client connected
Thread[26]: Thread created
Thread-5 leaving room Lobby
Thread[26]: Thread starting
Thread-5 joining room Lobby
Thread[26]: Received from client: Type[CONNECT
], Number[0], Message[null]
```

Client reconnected

Checklist Items (3)

#2 Show the server terminating; Clients should be disconnected but still running and able to reconnect when the server is back online (demonstrate this)

#3 For each scenario, disconnected messages should be shown to the clients (should show a different person disconnected and should show the specific client disconnected)

#4 Clearly caption each image regarding what is being shown

Task #2 - Points: 1

Text: Explain the various Disconnect/termination scenarios

Details:
Include the various scenarios of how a disconnect can occur. There should be around 3 or so.

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention how a client gets disconnected from a Socket perspective
#2	1	Mention how/why the client program doesn't crash when the server disconnects/terminates.
#3	1	Mention how the server doesn't crash from the client(s) disconnecting

Response:

Client Initiates Disconnect:

The client program sends a disconnect command (e.g., "/disconnect") to the server.
The server's ServerThread receives the disconnect request and initiates the disconnection process.
The ServerThread closes the socket connection with the client, triggering a disconnection event.

Server Disconnects Client Due to Inactivity:

If a client remains inactive for a specified period, the server's ServerThread may disconnect the client to manage

resources efficiently.
The server detects the inactivity by monitoring the time since the last message received from the client.
When the timeout threshold is reached, the ServerThread closes the socket connection with the inactive client.

Client Crashes or Loses Connection:

If a client's program crashes or the network connection is lost abruptly, the client's socket connection to the server is terminated.
The server's ServerThread detects the socket closure or connection loss event.
The ServerThread handles the unexpected disconnection by cleaning up resources associated with the disconnected client.

● Misc (1 pt.)

^COLLAPSE^

● ^COLLAPSE^

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1
https://github.com/Noaman4/-Milestone-Milestone-1/pull/1

● ^COLLAPSE^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Details:
Few related sentences about the Project/sockets topics

Response:

During this assignment, I encountered several challenges and gained valuable insights related to the project and socket programming:

Handling Concurrent Connections: One of the significant challenges was implementing a server capable of handling multiple simultaneous client connections efficiently. I learned about thread management, synchronization, and resource allocation to ensure smooth communication among clients.

Learning Socket Programming: This assignment deepened my understanding of socket programming concepts such as sockets, streams, input/output handling.

● ^COLLAPSE^

Task #3 - Points: 1

Text: WakaTime Screenshot

ⓘ Details:

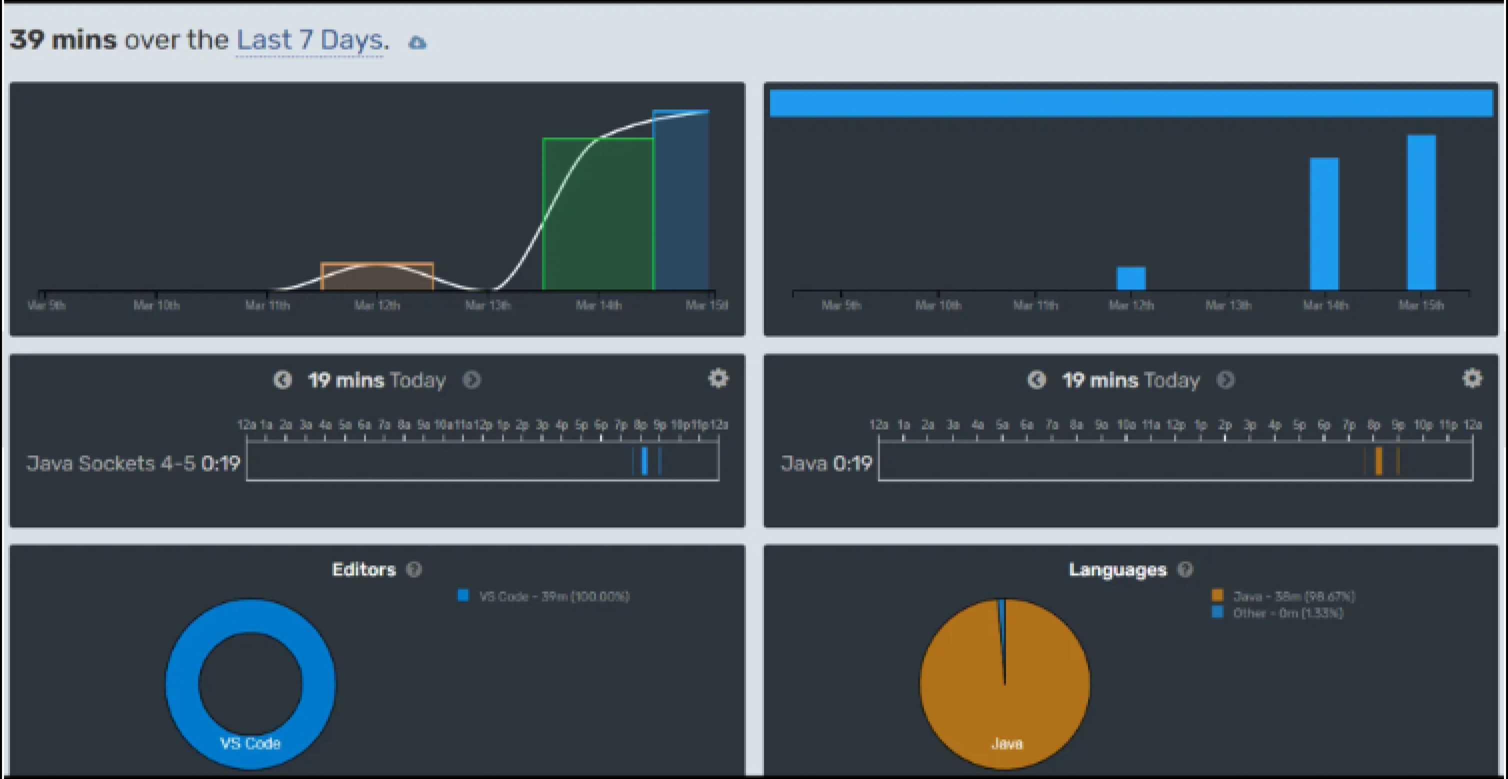
Grab a snippet showing the approximate time involved that clearly shows your repository.

The duration isn't considered for grading, but there should be some time involved.

Task Screenshots:

Gallery Style: Large View

Small Medium Large



39 mins over the Last 7 Days.

19 mins Today

Java Sockets 4-5 **0:19**

19 mins Today

Java **0:19**

Editors

🟦 VS Code - 39m (100.00%)

Languages

🟧 Java - 38m (98.67%)
🟦 Other - 0m (1.33%)

wakatime

End of Assignment