Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-project-milestone-1/grade/fj28

IT114-006-S2024 - [IT114] Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 3/16/2024 7:33:57 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

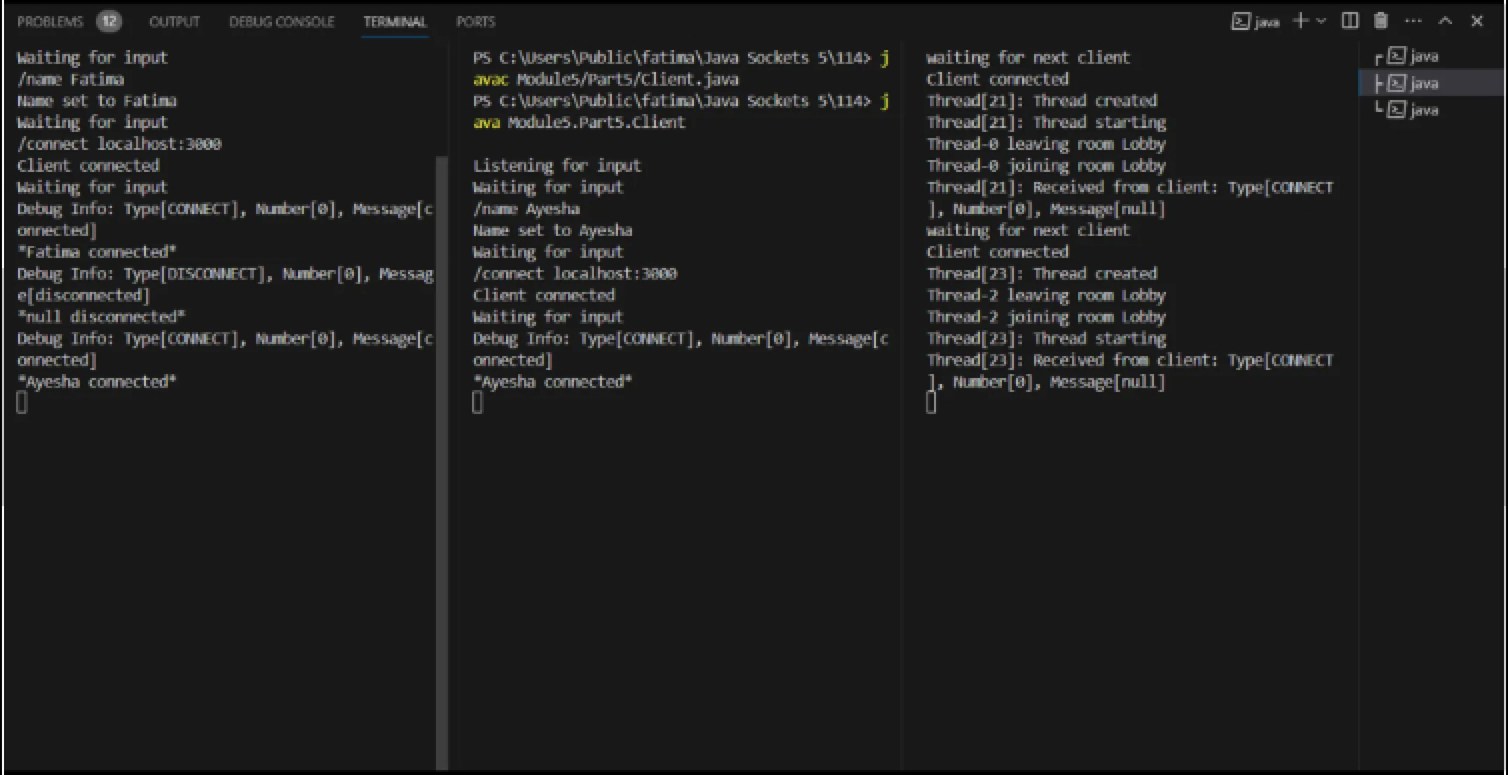
Checklist *The checkboxes are for your own tracking

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

```
PS C:\Users\Public\fatima\Java Sockets 5\114> j
avac Module5/Part5/Client.java
PS C:\Users\Public\fatima\Java Sockets 5\114> j
ava Module5.Part5.Client

Listening for input
Waiting for input
/name Ayesha
Name set to Ayesha
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha 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-2 joining room Lobby
Thread[23]: Thread starting
Thread[23]: Received from client: Type[CONNECT
], Number[0], Message[null]
```

Server listening to its port clients waiting for input clients have name and connect to server

Checklist Items (3)

#1 Server should properly be listening to its port from the command line (note the related message)

#2 Clients should be successfully waiting for input

#3 Clients should have a name and successfully connected to the server (note related messages)

^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 initializes a ServerSocket to listen for incoming connections on a specified port.
Upon receiving a connection request from a client, the server accepts the connection and creates a new ServerThread to manage communication with that client.
The ServerThread utilizes ObjectInputStream and ObjectOutputStream to exchange data with the client over the established socket connection.
Client-Side Connection:

The client initiates a connection to the server by creating a socket and specifying the server's port.
Once the connection is established, the client sets up an ObjectOutputStream and ObjectInputStream to send and receive objects via the socket.
Socket Steps Until Server Awaits Messages:

The server initializes a ServerSocket and begins listening for incoming connections on a designated port.
As clients attempt to connect, the serverSocket.accept() method is used to accept incoming connection requests, spawning a new ServerThread for each connected client.
Within each ServerThread, the server sets up ObjectInputStream and ObjectOutputStream to handle communication with the respective client.
The server enters a continuous loop to read incoming messages from the client using in.readObject(), processing each message accordingly.

Communication (3 pts.)

^COLLAPSE ^

^COLLAPSE ^

Task #1 - Points: 1

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

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

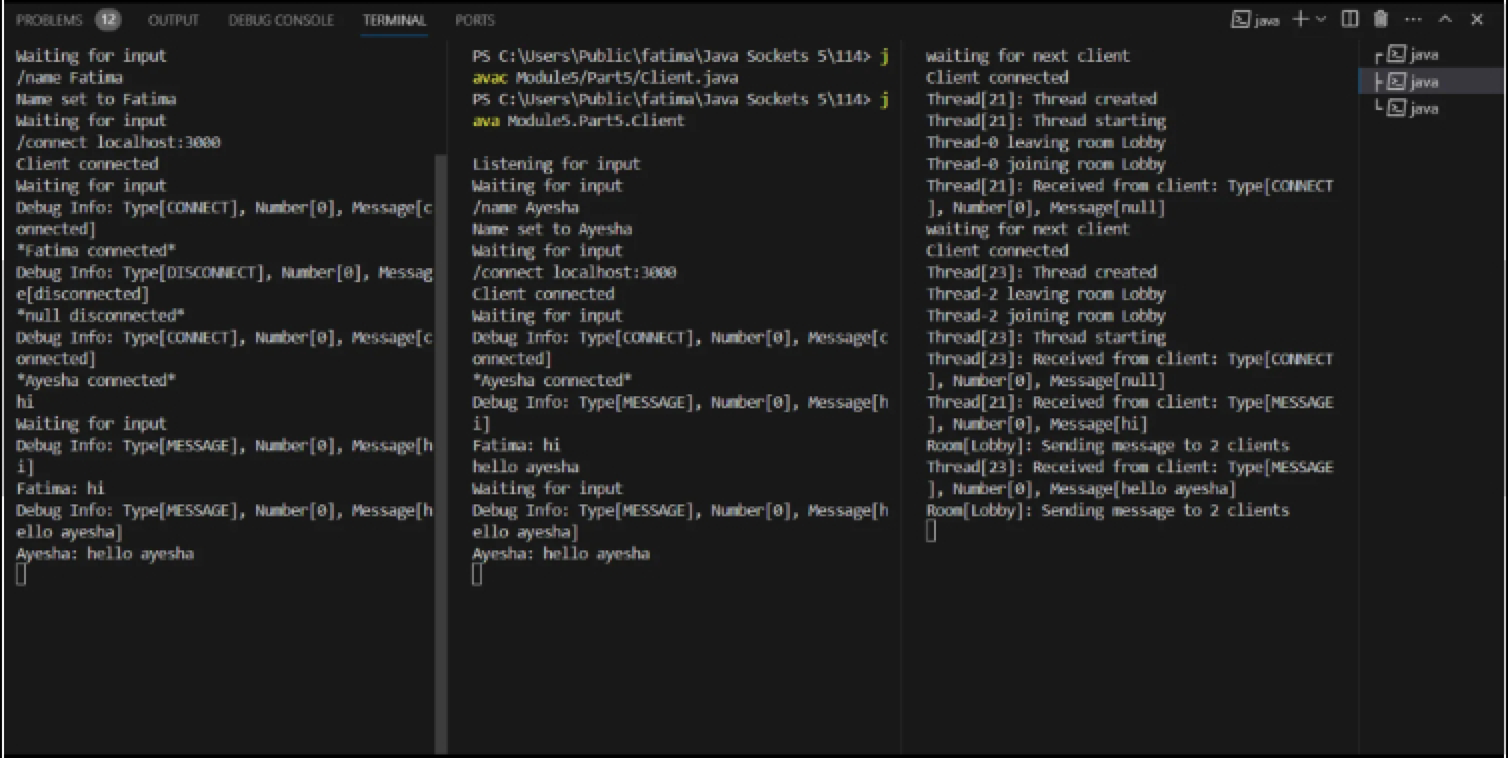
Checklist *The checkboxes are for your own tracking

#	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



```
Waiting for input
/name Fatima
Name set to Fatima
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Fatima connected*
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Fatima: hi
Debug Info: Type[MESSAGE], Number[0], Message[h
ello ayesha]
Ayesha: hello ayesha
```

```
PS C:\Users\Public\fatima\Java Sockets 5\114> j
avac Module5/Part5/Client.java
PS C:\Users\Public\fatima\Java Sockets 5\114> j
ava Module5.Part5.Client

Listening for input
Waiting for input
/name Ayesha
Name set to Ayesha
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha connected*
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Fatima: hi
hello ayesha
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
ello ayesha]
Ayesha: hello ayesha
```

```
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-2 joining room Lobby
Thread[23]: Thread starting
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[hello ayesha]
Room[Lobby]: Sending message to 2 clients
```

2 clients connected to server and can send messages to each other displaying clients name.

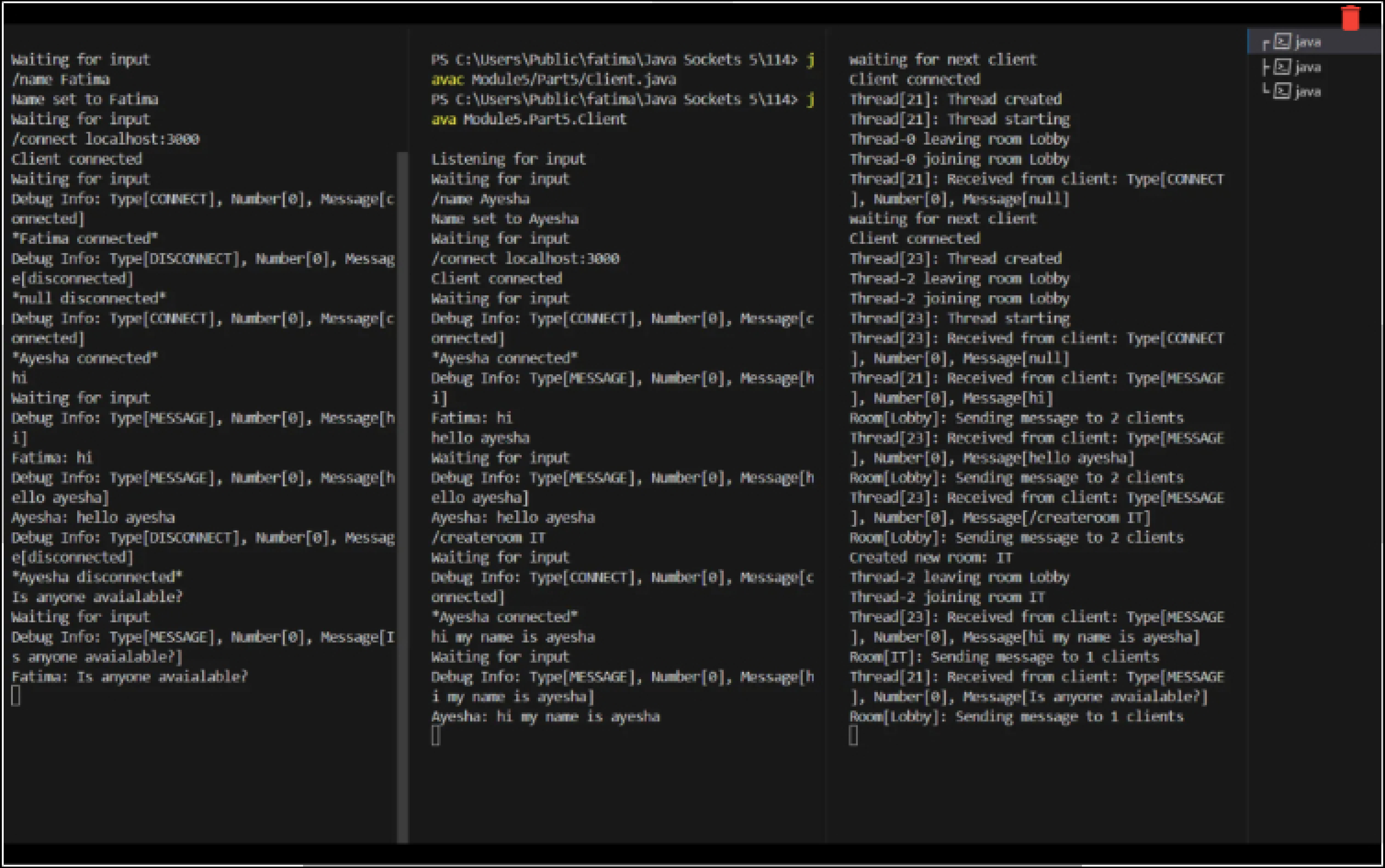
Checklist Items (4)

#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)



```
Waiting for input
/name Fatima
Name set to Fatima
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Fatima connected*
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Fatima: hi
Debug Info: Type[MESSAGE], Number[0], Message[h
ello ayesha]
Ayesha: hello ayesha
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*Ayesha disconnected*
Is anyone avaialable?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[I
s anyone avaialable?]
Fatima: Is anyone avaialable?
```

```
PS C:\Users\Public\fatima\Java Sockets 5\114> j
avac Module5/Part5/Client.java
PS C:\Users\Public\fatima\Java Sockets 5\114> j
ava Module5.Part5.Client

Listening for input
Waiting for input
/name Ayesha
Name set to Ayesha
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha connected*
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Fatima: hi
hello ayesha
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
ello ayesha]
Ayesha: hello ayesha
/createroom IT
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha connected*
hi my name is ayesha
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i my name is ayesha]
Ayesha: hi my name is ayesha
```

```
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-2 joining room Lobby
Thread[23]: Thread starting
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[hello ayesha]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[/createroom IT]
Room[Lobby]: Sending message to 2 clients
Created new room: IT
Thread-2 leaving room Lobby
Thread-2 joining room IT
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[hi my name is ayesha]
Room[IT]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[Is anyone avaialable?]
Room[Lobby]: Sending message to 1 clients
```

Both clients are in different rooms and cant send/receive message to each other.

Checklist Items (2)

#5 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 Clearly caption each image regarding what is being shown

^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 inputs a message through the keyboard input stream.
The Client class processes the message and packages it into a Payload object.
Using an ObjectOutputStream, the Client class sends the Payload object to the server via the established socket connection.
ServerThread's Role:

Each connected client is managed by a distinct ServerThread on the server side.
The ServerThread listens for incoming Payload objects from its respective client using an ObjectInputStream.
Upon receiving a Payload object, the ServerThread interprets its content and takes appropriate actions based on the payload type (e.g., handling messages, connect/disconnect events).
Room's Functionality:

The Room class oversees a collection of clients within the same chat room.
When a ServerThread receives a message from its client, it forwards the message to the corresponding Room object.
The Room object distributes the message to all clients in the same room by utilizing the sendMessage method of each connected ServerThread.
Client-Side (Receiving):

The respective ServerThread of the receiving client obtains the message from the server.
The ServerThread processes the incoming message and forwards it to the client's Client class.
The Client class then displays the received message to the user via the console or user interface.

● 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

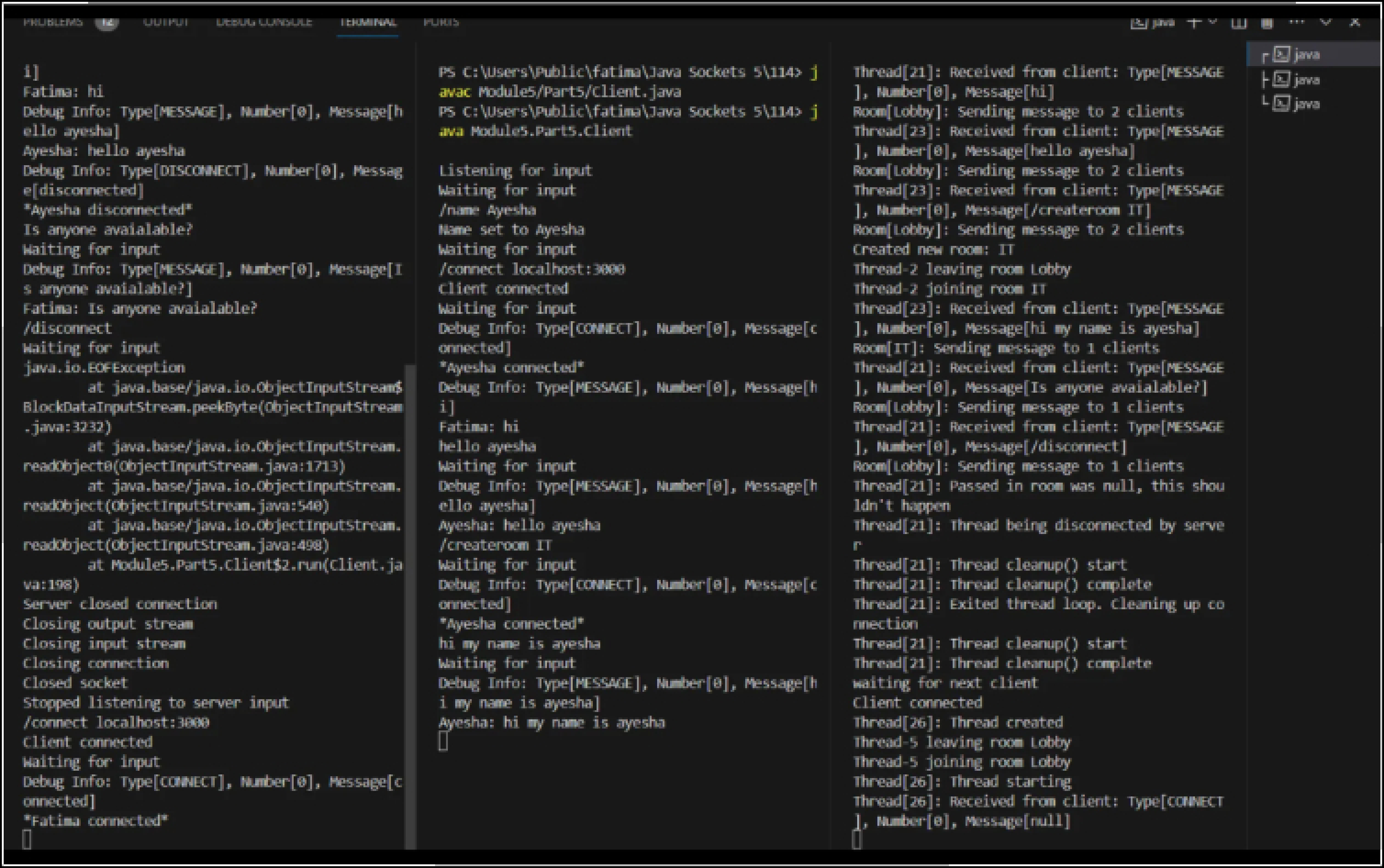
#	Points	Details
#1	1	Show a client disconnecting from the server; Server should still be running without issue (it's ok if an exception message shows as it's part of the lesson code, the server just shouldn't terminate)
#2	1	Show the server terminating; Clients should be disconnected but still running and 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

#3	1	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



```
i]
Fatima: hi
Debug Info: Type[MESSAGE], Number[0], Message[h
ello ayesha]
Ayesha: hello ayesha
Debug Info: Type[DISCONNECT], Number[0], Messag
e[disconnected]
*Ayesha disconnected*
Is anyone avaialable?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[I
s anyone avaialable?]
Fatima: Is anyone avaialable?
/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
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Fatima connected*
```

```
PS C:\Users\Public\fatima\Java Sockets 5\114> j
avac Module5/Part5/Client.java
PS C:\Users\Public\fatima\Java Sockets 5\114> j
ava Module5.Part5.Client

Listening for input
Waiting for input
/name Ayesha
Name set to Ayesha
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha connected*
Debug Info: Type[MESSAGE], Number[0], Message[h
i]
Fatima: hi
hello ayesha
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
ello ayesha]
Ayesha: hello ayesha
/createroom IT
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[c
onnected]
*Ayesha connected*
hi my name is ayesha
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[h
i my name is ayesha]
Ayesha: hi my name is ayesha
```

```
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[hello ayesha]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[/createroom IT]
Room[Lobby]: Sending message to 2 clients
Created new room: IT
Thread-2 leaving room Lobby
Thread-2 joining room IT
Thread[23]: Received from client: Type[MESSAGE
], Number[0], Message[hi my name is ayesha]
Room[IT]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[Is anyone avaialable?]
Room[Lobby]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE
], Number[0], Message[/disconnect]
Room[Lobby]: Sending message to 1 clients
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
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-5 joining room Lobby
Thread[26]: Thread starting
Thread[26]: Received from client: Type[CONNECT
], Number[0], Message[null]
```

The client disconnected and can reconnect.

Checklist Items (4)

#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)

#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

●

COLLAPSE

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-Side Disconnect:

1. The client sends "/disconnect" command to the server.
2. The server's ServerThread closes the socket connection, disconnecting the client.

Server Disconnects Due to Inactivity:

1. Server's ServerThread detects client inactivity.
2. The server closes the socket connection after a timeout period.

Client Crashes or Loses Connection:

1. The client's socket connection closes due to program crash or network issue.
2. Server's ServerThread manages the disconnection and cleans up resources.

Socket Perspective:

The client disconnects by closing the socket.
The program handles disconnections gracefully without crashing.

Program Stability:

The client program handles server disconnects without crashing.
The server manages unexpected client disconnects without affecting other clients or crashing.

● Misc (1 pt.)
COLLAPSE

● COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1
https://github.com/fj29/-Milestone-Milestone-1/pull/1

●

Task #2 - Points: 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:

Challenges Faced:

Managing Concurrent Connections: Learned efficient thread management and resource allocation for handling multiple clients concurrently.

Insights Gained:

Socket Programming Understanding: Deepened knowledge of sockets, streams, and input/output handling.

^COLLAPSE ^

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

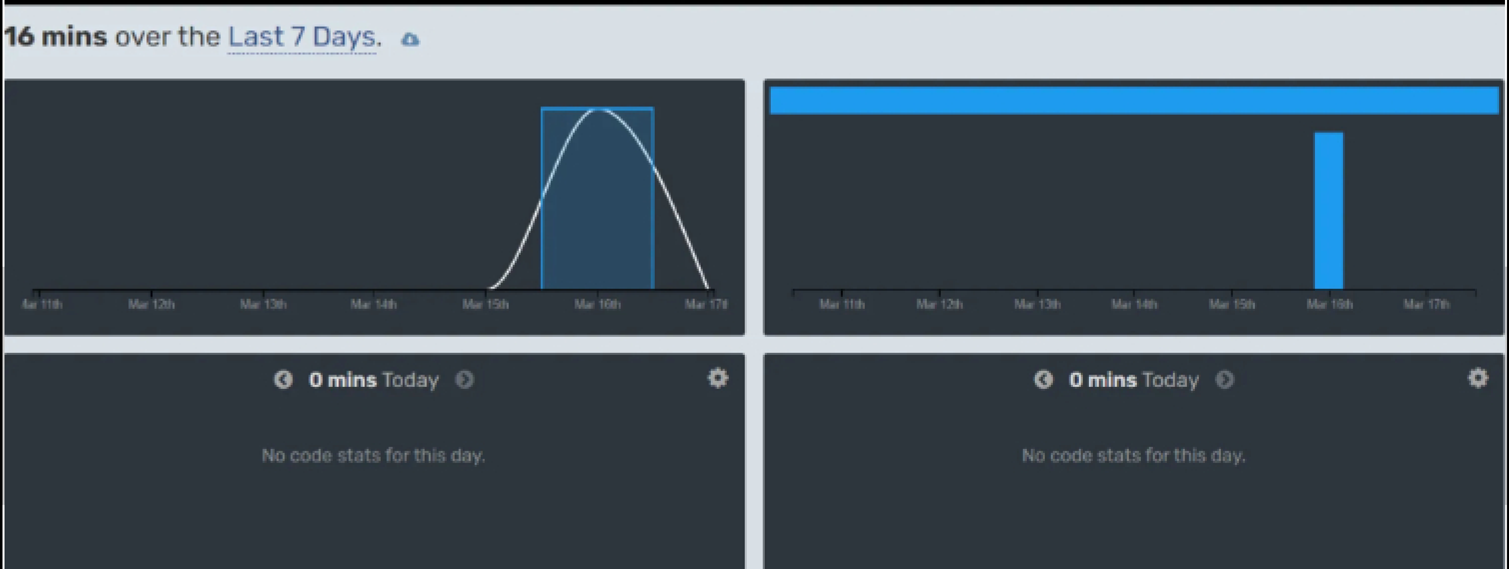
Grab a snippet showing the approximate time involved that clearly shows your repository.

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



16 mins over the Last 7 Days.

0 mins Today

No code stats for this day.

0 mins Today

No code stats for this day.

Wakatime

End of Assignment