Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-project-milestone-1/grade/msa224

IT114-004-S2024 - [IT114] Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 4/5/2024 1:17:41 AM

Instructions

COLLAPSE

1. Create a new branch called Milestone1
2. At the root of your repository create a folder called Project if one doesn't exist yet
    1. You will be updating this folder with new code as you do milestones
    2. You won't be creating separate folders for milestones; milestones are just branches
3. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
4. Copy in the latest Socket sample code from the most recent Socket Part example of the lessons
    1. Recommended Part 5 (clients should be having names at this point and not ids)
        1. https://github.com/MattToegel/IT114/tree/Module5/Module5
5. Fix the package references at the top of each file (these are the only edits you should do at this point)
6. Git add/commit the baseline and push it to github
7. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
8. Ensure the sample is working and fill in the below deliverables
    1. Note: The client commands likely are different in part 5 with the /name and /connect options instead of just "connect"
9. Generate the worksheet output file once done and add it to your local repository
10. Git add/commit/push all changes
11. Complete the pull request merge from step 7
12. Locally checkout main
13. 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

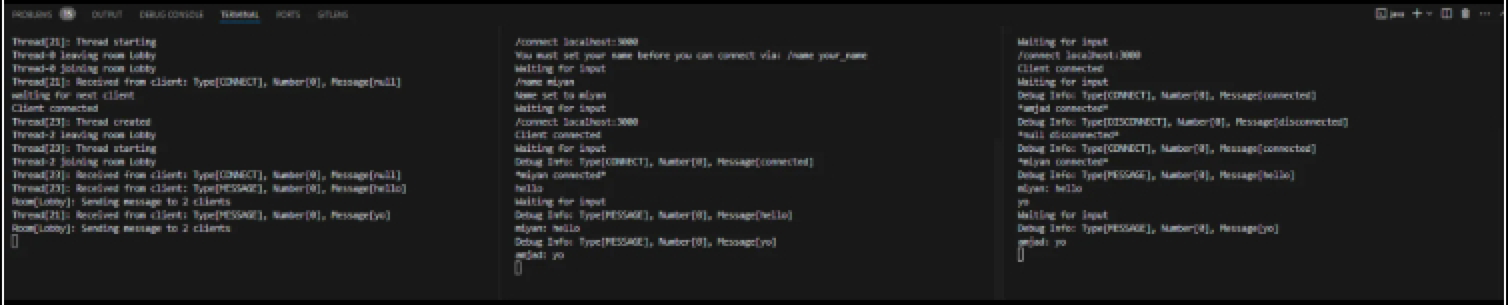
Checklist *The checkboxes are for your own tracking

#	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



```
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]
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[yo]
Room[Lobby]: Sending message to 2 clients
```

```
/connect localhost:3000
You must set your name before you can connect via: /name your_name
Waiting for input
/name miyan
Name set to miyan
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
miyan: hello
Debug Info: Type[MESSAGE], Number[0], Message[yo]
amjad: yo
```

```
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*amjad connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
Debug Info: Type[MESSAGE], Number[0], Message[hello]
miyan: hello
yo
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[yo]
amjad: yo
```

Shows all the checklist items that are being asked for. Clearly working as they should.

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)

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:

So to get the server up and running, first we compile the server by running javac Project/Server.java, then we run the actual server by running its counterpart, java Project.Server. This gets the server up and running and ready to receive clients on the serverport.

To get the clients up and running to joined to the server, we run the same codes to compile and start with the respective file names. Once that happens in each client terminal we first must enter a name for the client and then we are able to join the server port and send messages, etc.

Communication (3 pts.)

Task #1 - Points: 1

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

Checklist *The checkboxes are for your own tracking

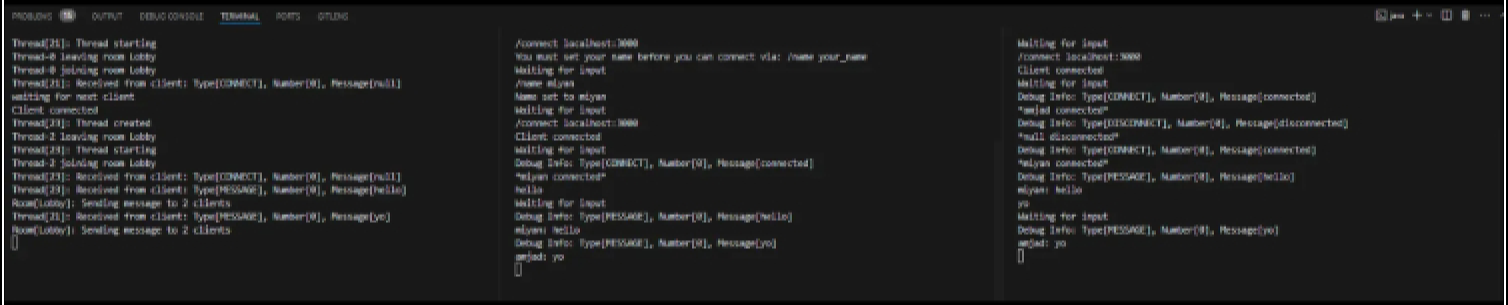
#	Points	Details
☐ #1	1	At least two clients connected to the server
☐ #2	1	Client can send messages to the server
☐ #3	1	Server sends the message to all clients in the same room
☐ #4	1	Messages clearly show who the message is from (i.e., client name is clearly with the message)

🟦 #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



```
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]
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[yo]
Room[Lobby]: Sending message to 2 clients
```

```
/connect localhost:3000
You must set your name before you can connect via: /name your_name
Waiting for input
/name miyan
Name set to miyan
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
miyan: hello
Debug Info: Type[MESSAGE], Number[0], Message[yo]
amjad: yo
```

```
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*amjad connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
Debug Info: Type[MESSAGE], Number[0], Message[hello]
miyan: hello
yo
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[yo]
amjad: yo
```

Shows everything being asked, clients are connected and sending messages that are being received by all other clients in the same room. Also shows who the messages are from.

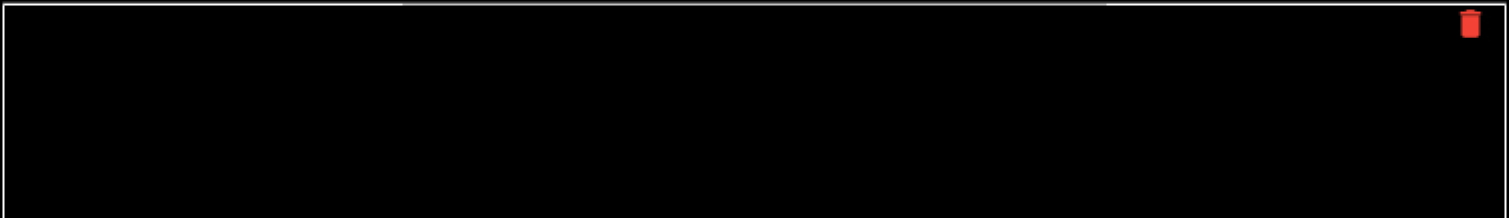
Checklist Items (4)

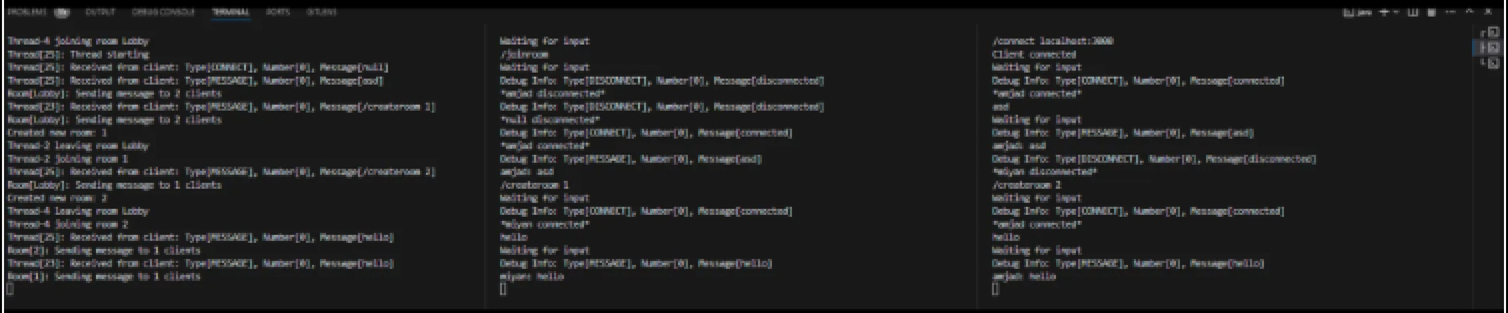
#1 At least two clients connected to the server

#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)





```
Thread-4 joining room Lobby
Thread[25]: Thread starting
Thread[25]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[25]: Received from client: Type[MESSAGE], Number[0], Message[asd]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/createroom 1]
Room[Lobby]: Sending message to 2 clients
Created new room: 1
Thread-2 leaving room Lobby
Thread-2 joining room 1
Thread[25]: Received from client: Type[MESSAGE], Number[0], Message[/createroom 2]
Room[Lobby]: Sending message to 1 clients
Created new room: 2
Thread-4 leaving room Lobby
Thread-4 joining room 2
Thread[25]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[2]: Sending message to 1 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[1]: Sending message to 1 clients
```

```
Waiting for input
/joinroom
Waiting for input
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*amjad disconnected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*amjad connected*
Debug Info: Type[MESSAGE], Number[0], Message[asd]
amjad: asd
/createroom 1
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
miyan: hello
```

```
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*amjad connected*
asd
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[asd]
amjad: asd
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*miyan disconnected*
/createroom 2
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*amjad connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
amjad: hello
```

Shows two clients who created two different rooms and shows thier messages only being sent to their respective rooms.

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):

Users type messages in the chat window or terminal.
These messages are then sent over to the server using the Client class.

ServerThread's Involvement:
The server's ServerThread receives messages from clients.
It decides what to do with each message, like sending it to the right chat room or clients.

Room's Perspective:
When the ServerThread gets a message, it sends it to the relevant chat room or clients.
The room then shares the message with everyone in that chat room using sendMessage().

Client-side (Receiving):
Messages arrive on clients' screen or terminal through their ServerThread connections.
Users see these messages pop up in their chat window, showing what others have said who are also in the same room.

Disconnecting/Termination (3 pts.)

^COLLAPSE ^

Task #1 - Points: 1

^COLLAPSE ^

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

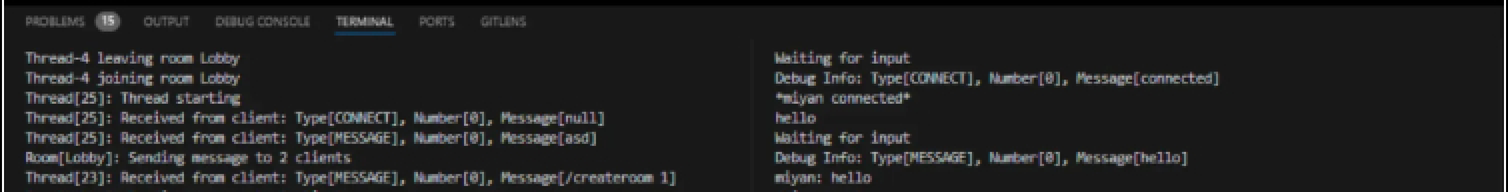
Checklist *The checkboxes are for your own tracking

#	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 a different person disconnected and should show the specific client disconnected)
#4	1	Clearly caption each image regarding what is being shown

Task Screenshots:

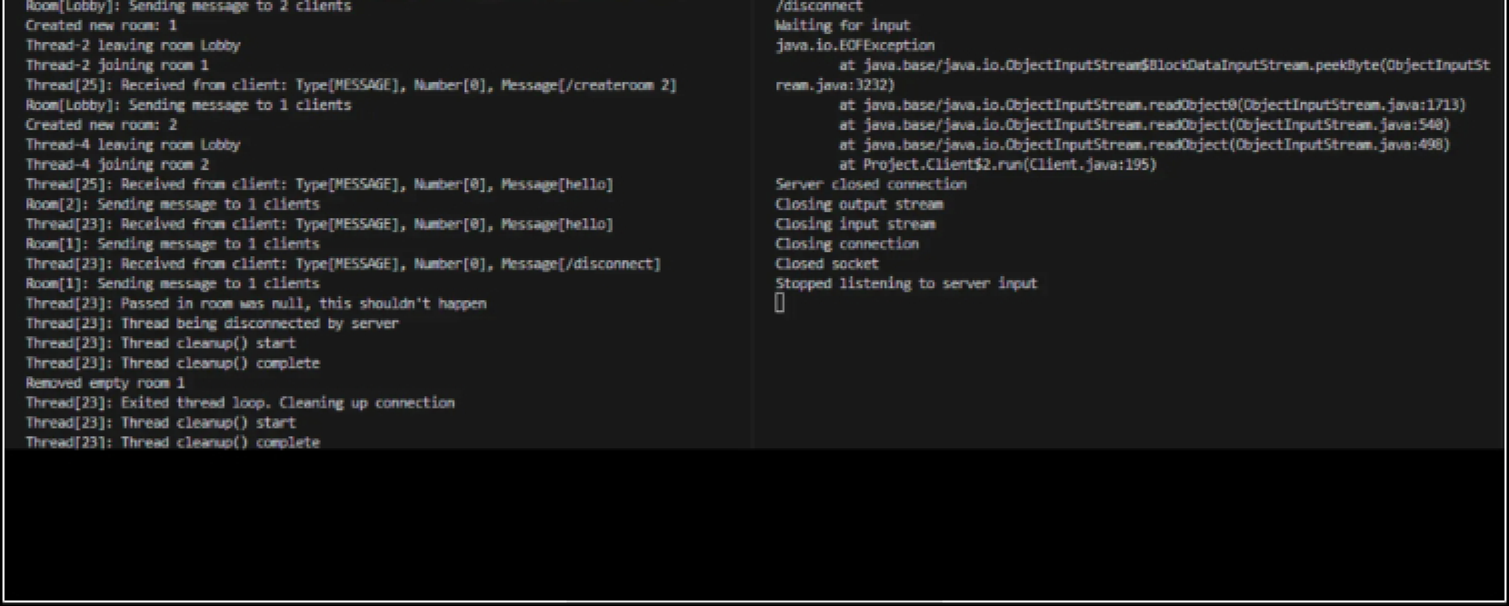
Gallery Style: Large View

Small Medium Large



```
Thread-4 leaving room Lobby
Thread-4 joining room Lobby
Thread[25]: Thread starting
Thread[25]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[25]: Received from client: Type[MESSAGE], Number[0], Message[asd]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/createroom 1]
```

```
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
miyan: hello
```



```
Room[Lobby]: Sending message to 2 clients
Created new room: 1
Thread-2 leaving room Lobby
Thread-2 joining room 1
Thread[25]: Received from client: Type[MESSAGE], Number[0], Message[/createroom 2]
Room[Lobby]: Sending message to 1 clients
Created new room: 2
Thread-4 leaving room Lobby
Thread-4 joining room 2
Thread[25]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[2]: Sending message to 1 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[1]: Sending message to 1 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[1]: Sending message to 1 clients
Thread[23]: Passed in room was null, this shouldn't happen
Thread[23]: Thread being disconnected by server
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
Removed empty room 1
Thread[23]: Exited thread loop. Cleaning up connection
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
```

```
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputSt
ream.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 Project.Client$2.run(Client.java:195)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
▯
```

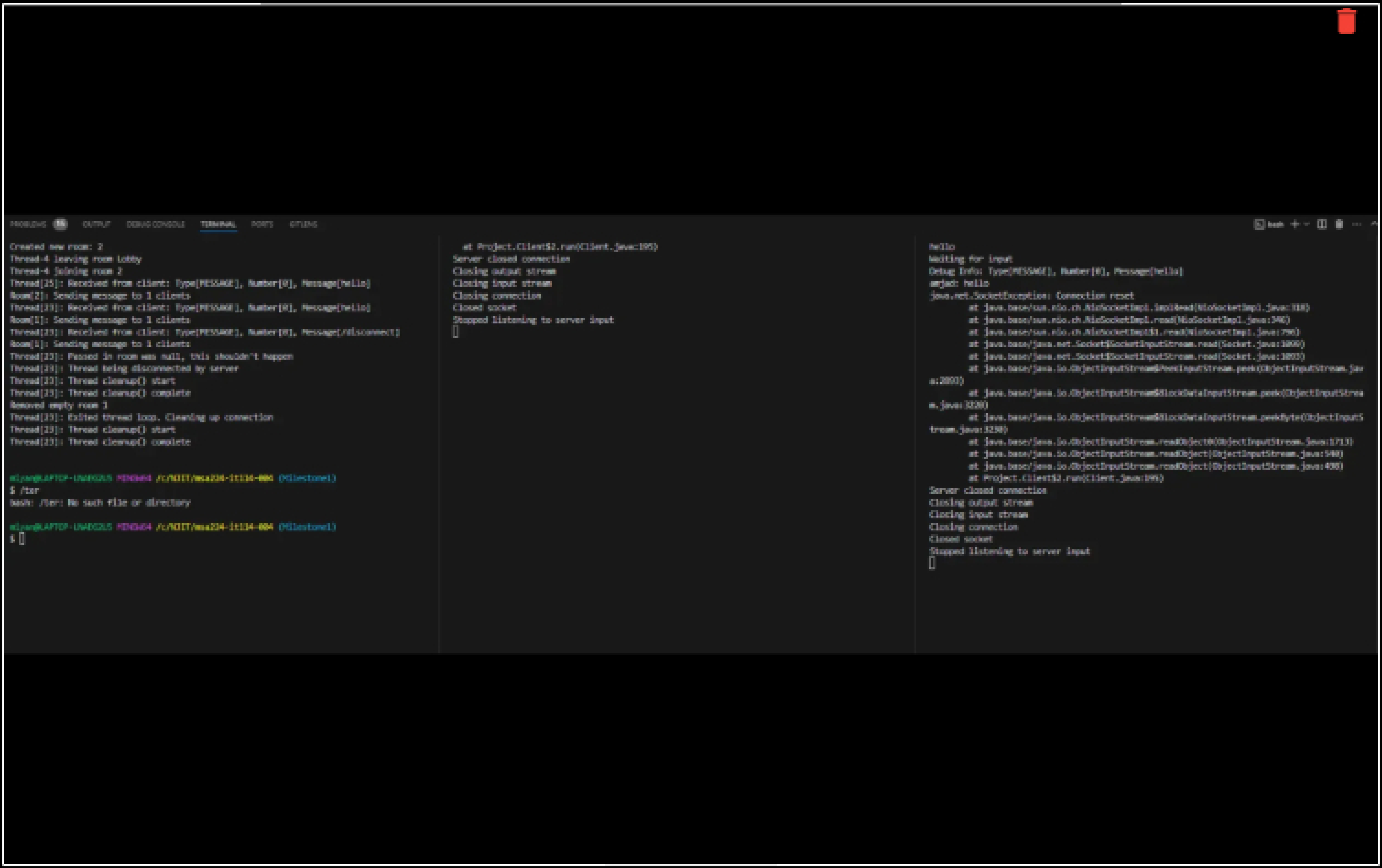
shows a client disconnecting from the server but the server still running.

Checklist Items (3)

#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)

#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



```
Created new room: 2
Thread-4 leaving room Lobby
Thread-4 joining room 2
Thread[25]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[2]: Sending message to 1 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[1]: Sending message to 1 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[1]: Sending message to 1 clients
Thread[23]: Passed in room was null, this shouldn't happen
Thread[23]: Thread being disconnected by server
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
Removed empty room 1
Thread[23]: Exited thread loop. Cleaning up connection
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete


alyan@LAPTOP-LNA0G2US MINGW64 /c/NJIT/msa224-it114-004 (Milestone1)
$ /ter
bash: /ter: No such file or directory

alyan@LAPTOP-LNA0G2US MINGW64 /c/NJIT/msa224-it114-004 (Milestone1)
$ ▯
```

```
  at Project.Client$2.run(Client.java:195)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
▯
```

```
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
amjad: hello
java.net.SocketException: Connection reset
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:318)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:346)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:796)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:1099)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:1093)
        at java.base/java.io.ObjectInputStream$PeekInputStream.peek(ObjectInputStream.jav
a:2893)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peek(ObjectInputStrea
m.java:3220)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputS
tream.java:3230)
        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 Project.Client$2.run(Client.java:195)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
▯
```

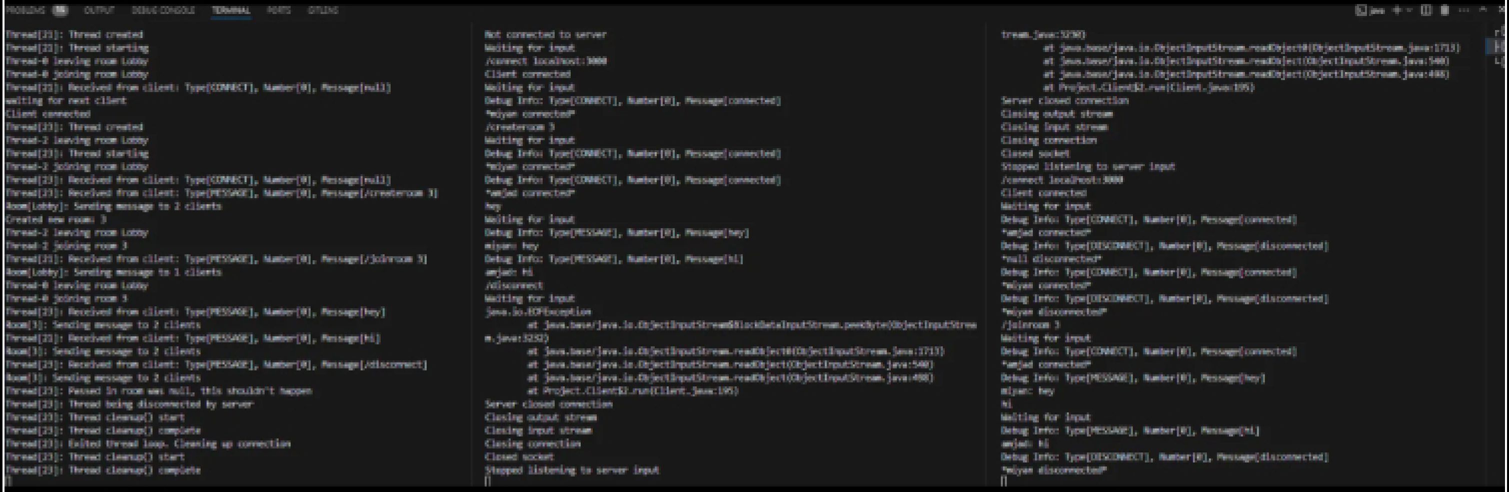
Shows the server terminated but the client still running ready for the server to be up and running again.

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



```
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]
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/createroom 3]
Room[Lobby]: Sending message to 2 clients
Created new room: 3
Thread-2 leaving room Lobby
Thread-2 joining room 3
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[/joinroom 3]
Room[Lobby]: Sending message to 1 clients
Thread-0 leaving room Lobby
Thread-0 joining room 3
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[hey]
Room[3]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[hi]
Room[3]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[3]: Sending message to 2 clients
Thread[23]: Passed in room was null, this shouldn't happen
Thread[23]: Thread being disconnected by server
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
Thread[23]: Exited thread loop. Cleaning up connection
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
```

```
Not connected to server
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
/createroom 3
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*amjad connected*
hey
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hey]
miyan: hey
Debug Info: Type[MESSAGE], Number[0], Message[hi]
amjad: hi
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStrea
m.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 Project.Client$2.run(Client.java:195)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
```

```
tream.java:3230)
        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 Project.Client$2.run(Client.java:195)
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[connected]
*amjad connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*miyan connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*miyan disconnected*
/joinroom 3
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*amjad connected*
Debug Info: Type[MESSAGE], Number[0], Message[hey]
miyan: hey
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi]
amjad: hi
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*miyan disconnected*
```

shows the client being able to reconnect when the server is back online as well as the disconnection messages being shown when a specific client disconnects.

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

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:

3 ways they can disconnect is the client disconnects themselves, there is a network interruption which causes them to lose contact, and/or there is a crash in the program itself.

When a client stops talking to the server, the server doesn't freak out or crash. It's because the server knows how to handle these situations . The part of the server that talks to clients, ServerThread, is smart enough to handle when clients leave without causing any trouble. Also, the server code has special things to deal with unexpected things, like when a client suddenly leaves. So, even if one or more clients leave suddenly, the server keeps on running smoothly.

Likewise, when the server goes quiet, the client doesn't shut down. It's because the client's code is smart enough to deal with these situations. If the server suddenly disappears, the client's code can handle it. So, even if the server vanishes unexpectedly, the client carries on without any issues.

● Misc (1 pt.)

^COLLAPSE ^

● ^COLLAPSE ^

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/msa224/msa224-it114-004/pull/10

● ^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:

Learning about sockets and how servers talk to clients has been really interesting. It's like discovering how the internet actually works behind the scenes. Seeing how servers handle lots of people connecting at once and how we can send messages back and forth in realtime is pretty cool. Overall, getting into socket stuff has been eye opening and made me realize how much goes on behind the scenes when we use apps and websites.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

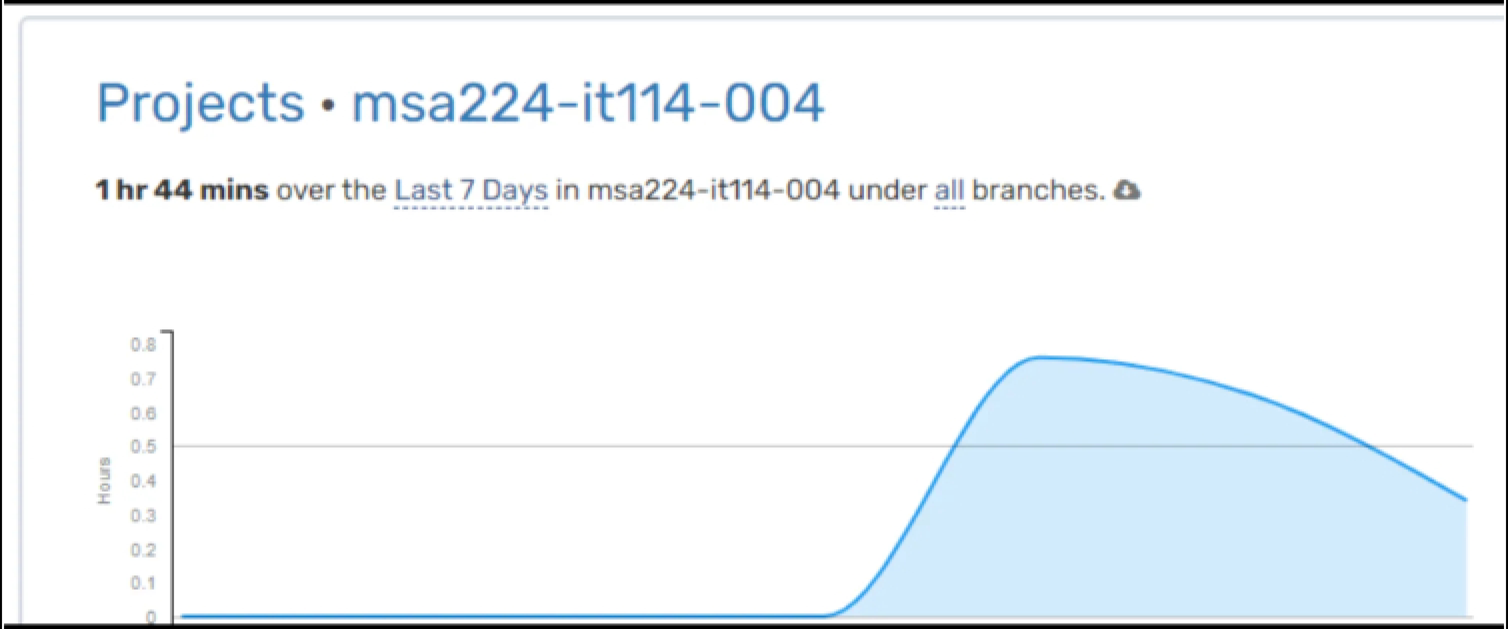
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository.

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Projects • msa224-it114-004

1 hr 44 mins over the Last 7 Days in msa224-it114-004 under all branches.

shows my wakatime in my repository

End of Assignment