Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-project-milestone-1/grade/rn364

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

Submissions:

Submission Selection

1 Submission [active] 3/19/2024 11:34:20 AM

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

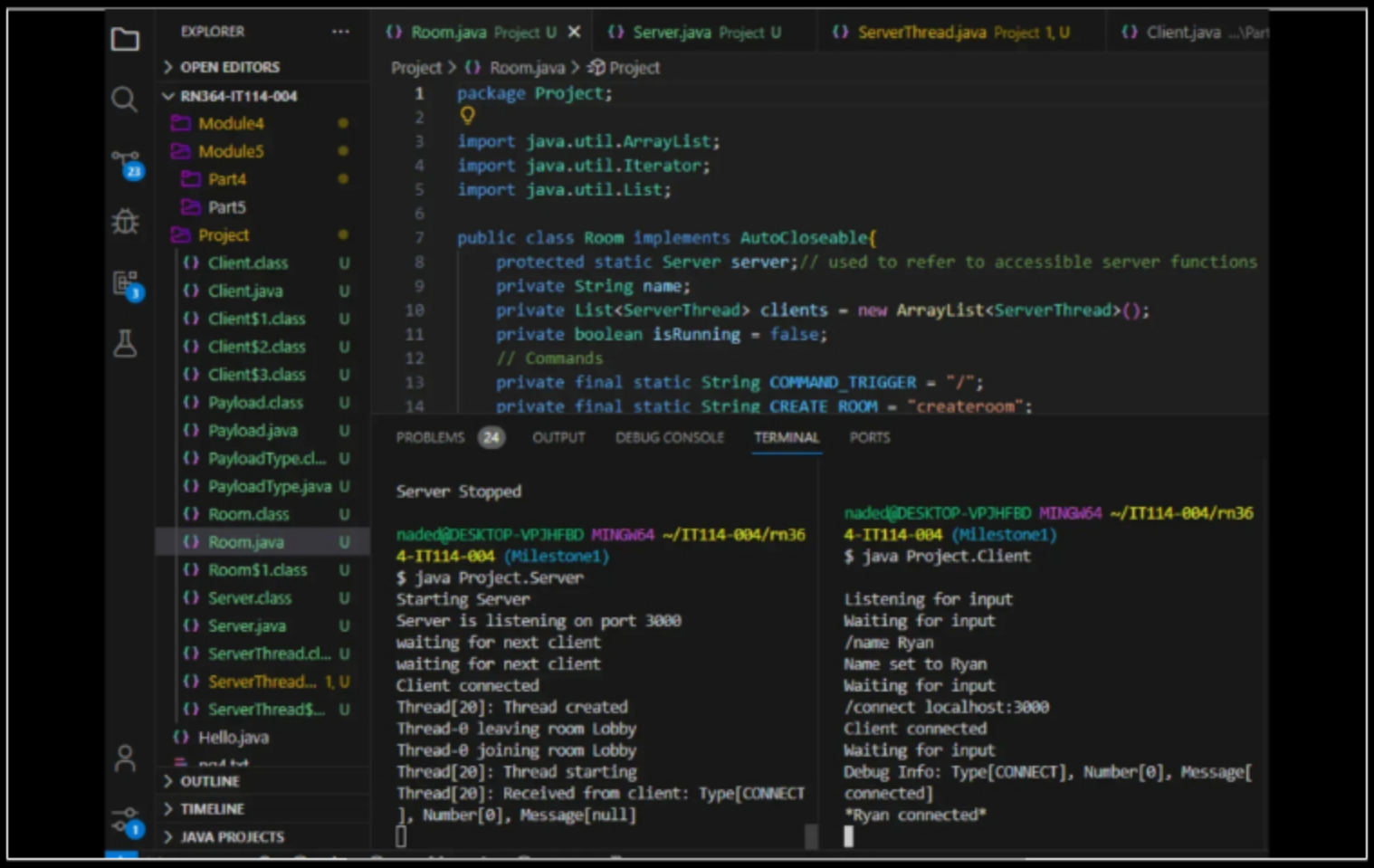
*The checkboxes are for your own tracking

#	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



Server started, Server waited for input and Client successfully connected to the 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:

The server gets set up to receive input from the client server and waits on port 3000 after compiling. Before the client can connect, it needs to pick a name. so on the client side i typed /name Ryan to create a name for the user. After choosing a name you will be able to connect to the server. I was able to connect by typing /connect localhost:3000 and Once connected, both the client side and server side are linked up.

COLLAPSE

Communication (3 pts.)

COLLAPSE

Task #1 - Points: 1

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

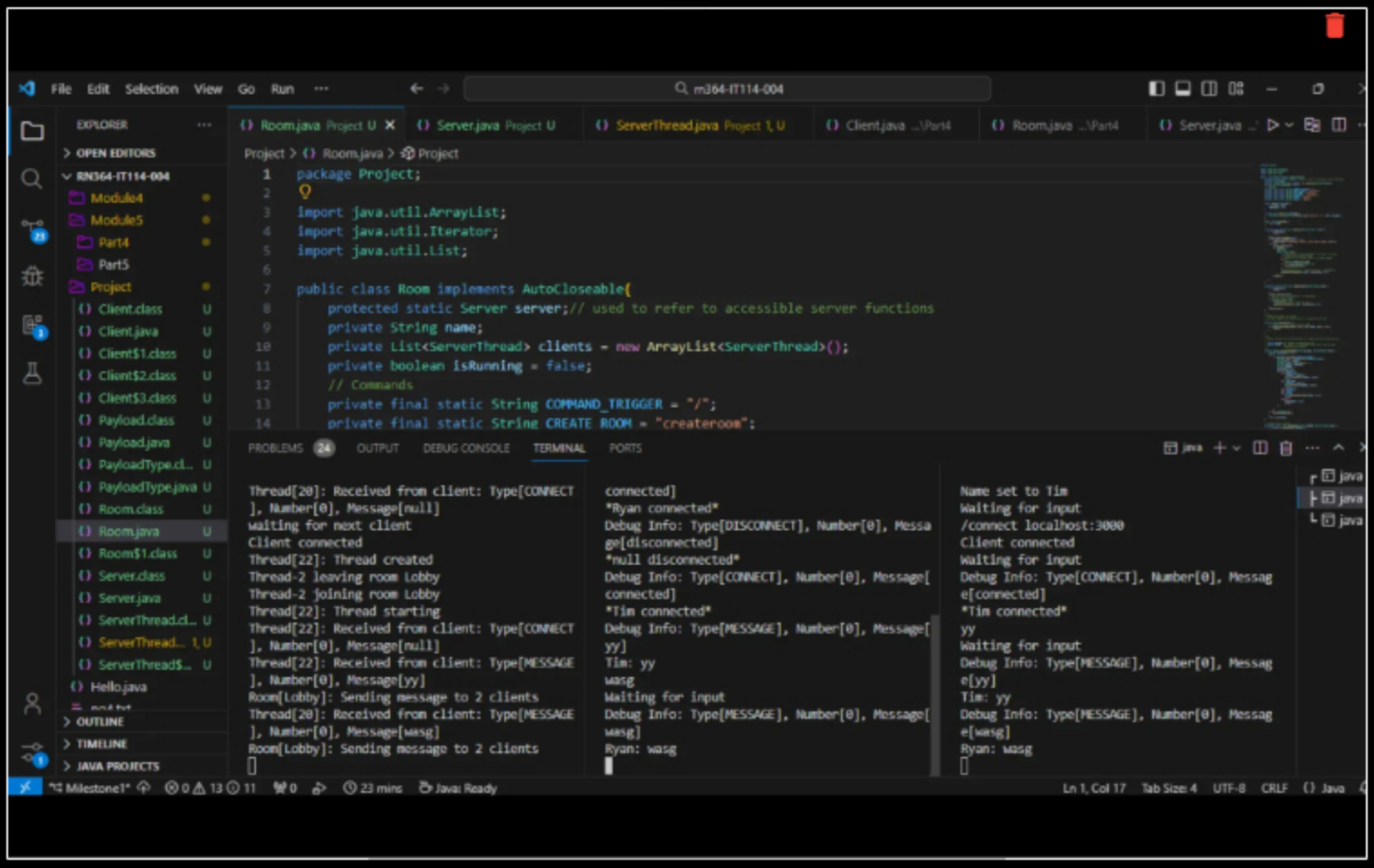
Checklist *The checkboxes are for your own tracking

#	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



2 clients connected and are able to send messages to each other

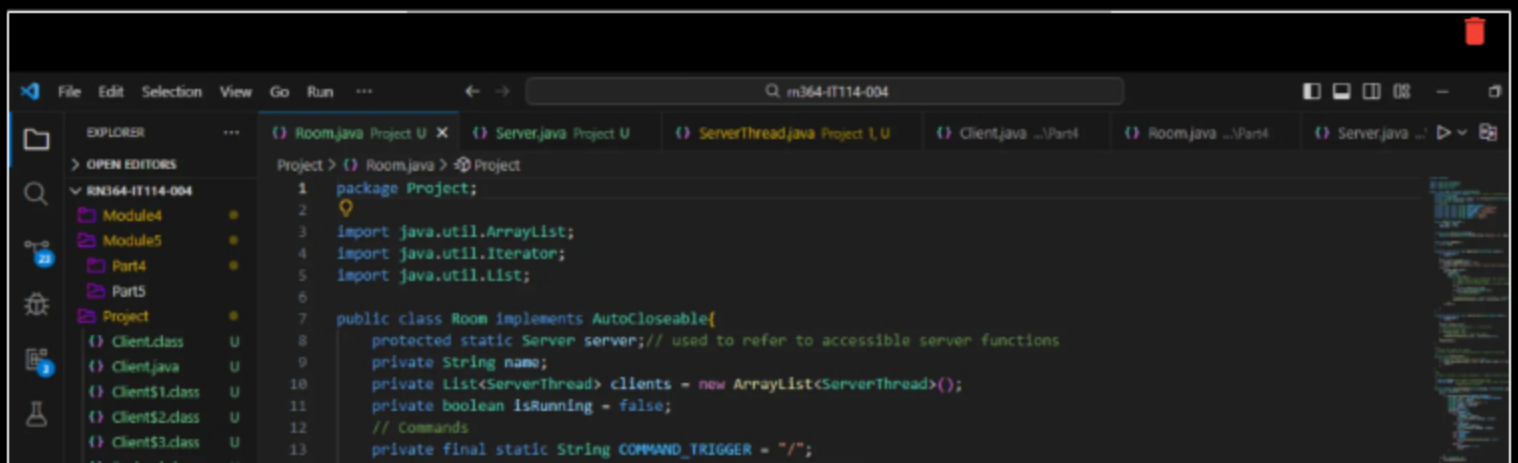
Checklist Items (4)

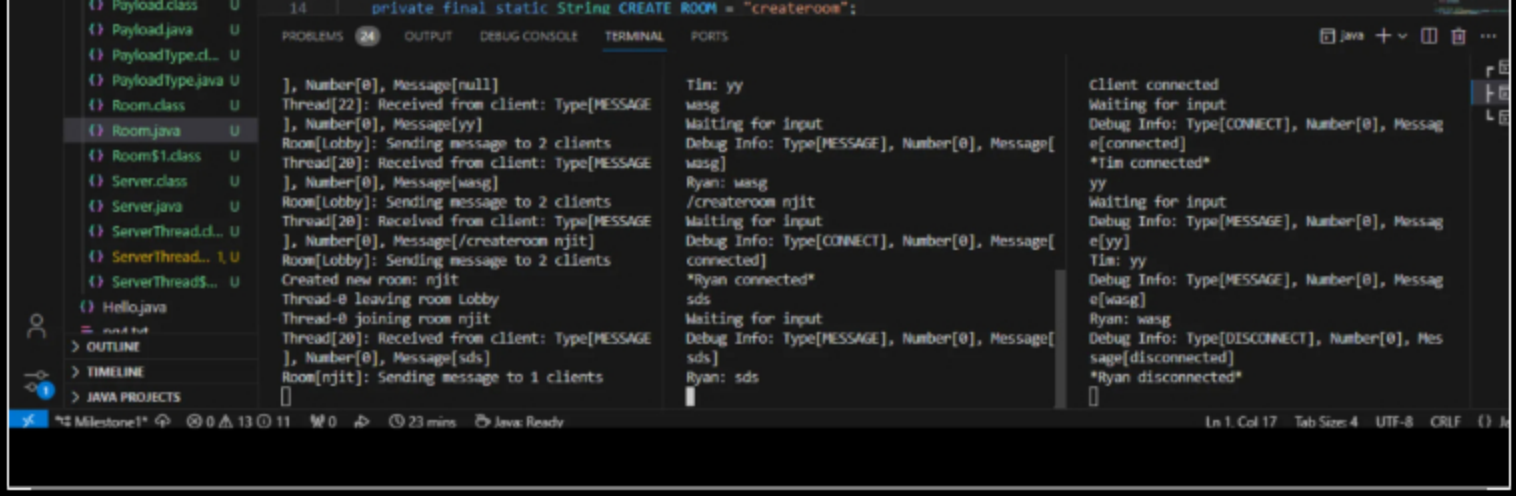
#1 At least two clients connected to the server

#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)





I made a new room called NJIT and joined it in the image, when I type a message it doesnt show for the other client.

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:

From the client's side, their input is taken and passed through the process command function. Afterwards, the relevant data is sent out. In the server thread, the process message function handles three scenarios: connecting, disconnecting, and messaging. It receives input from the client and navigates through these different cases. Within the room class, the process command function manages input from both the client and the server thread. It employs a case structure to handle actions like creating, joining, and disconnecting from rooms. The resulting message is then sent back to the client. If clients are in the same room, they receive the same message, but not if they're in separate rooms.

Disconnecting/Termination (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

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

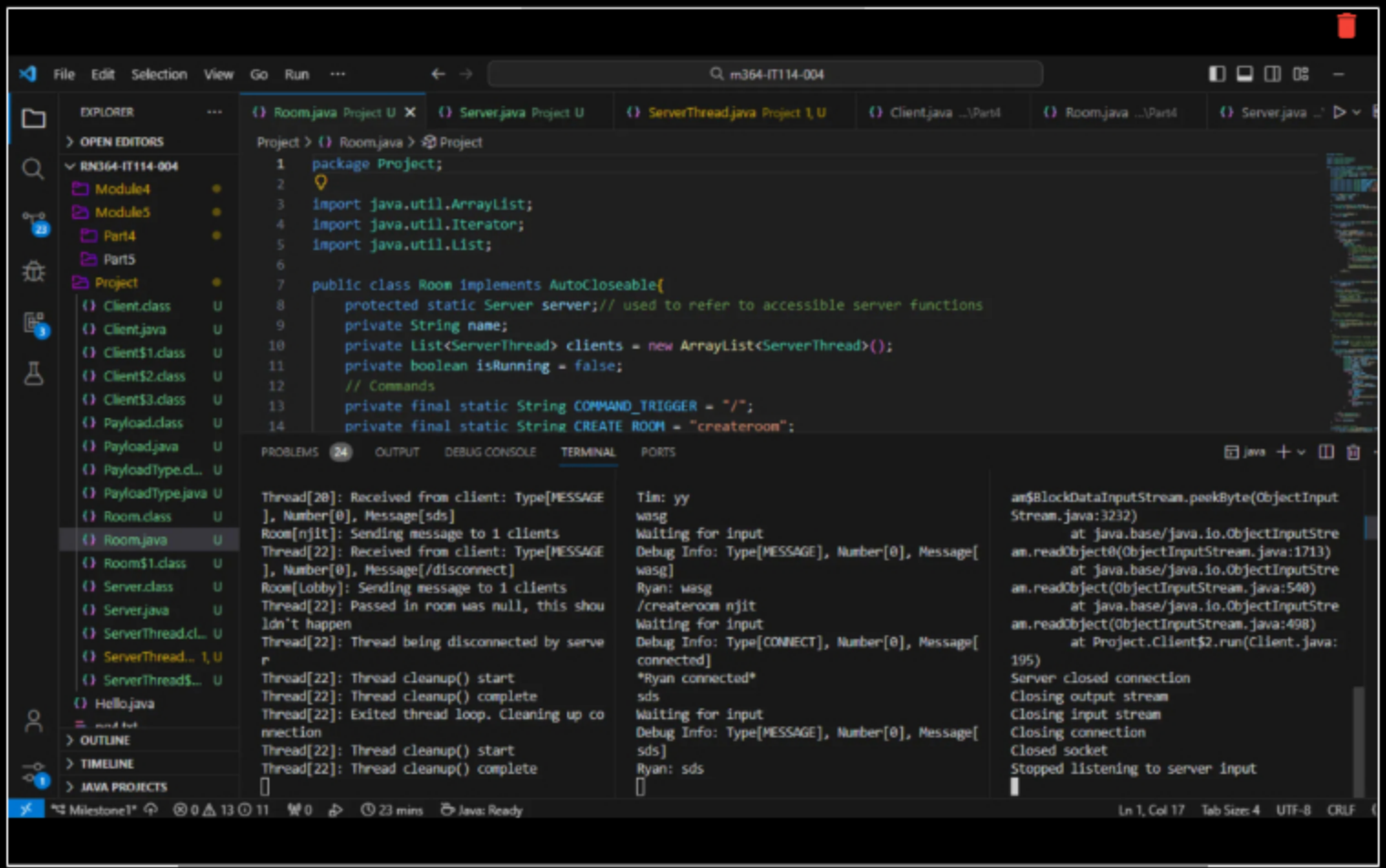
Checklist *The checkboxes are for your own tracking

#	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 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

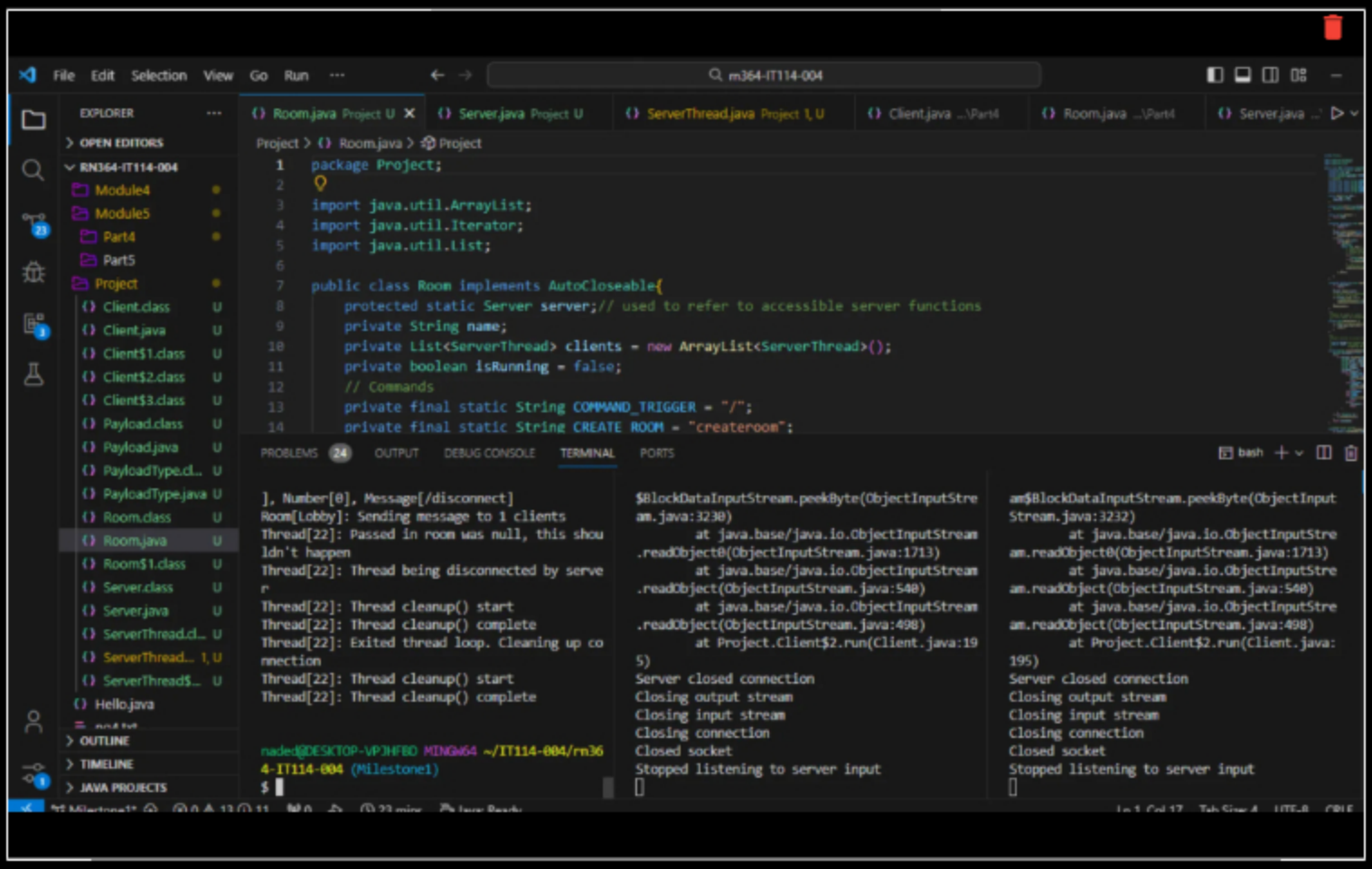


Client Disconnects from server, server is still running

Checklist Items (2)

#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)



Server is terminated

Checklist Items (1)

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

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
		Mention how/why the client program doesn't crash when the server

#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:

A client gets disconnected when the running status is false because there's no server thread running. To handle this, the client program uses a try-catch block to prevent crashing if the server isn't connected. Instead, it keeps running and informs the client that the server is disconnected. The server does something similar to check if a client is connected, also preventing the code from stopping if there's an issue.

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/ryann2n/rn364-IT114-004/pull/8

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:

It was confusing at first because I was used to just running the files with the play button on vscode, when I tried it in the terminal I was getting errors but thats just because I forgot to compile first.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

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

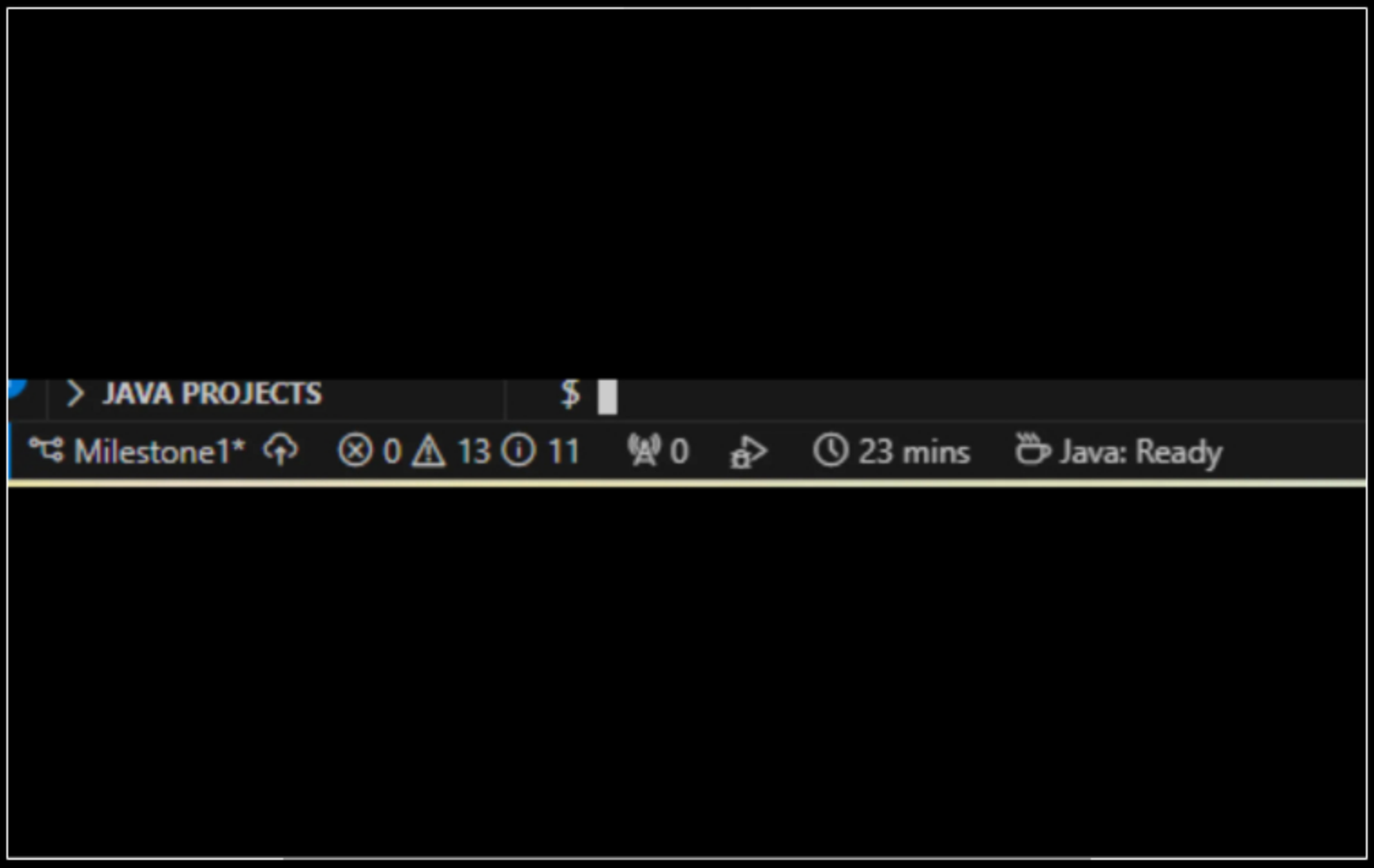
The duration isn't considered for grading, but there should be some time involved.

Task Screenshots:

Gallery Style: Large View

Gallery Style: Large View

Small Medium Large



Waka Time

End of Assignment