Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-module-5-project-milestone-1/grade/jns

Course: IT114-003-F2024
Assigment: [IT114] Module 5 Project Milestone 1
Student: Jimmy S. (jns)

Submissions:

Submission Selection

1 Submission [submitted] 12/11/2024 4:17:02 PM

Instructions

COLLAPSE

Overview Video: https://youtu.be/A2yDMS9TS1o

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. Copy in the code from Sockets Part 5 into the Project folder (just the files)
    2. https://github.com/MattToegel/IT114/tree/M24-Sockets-Part5
4. Fix the package references at the top of each file (these are the only edits you should do at this point)
5. Git add/commit the baseline and push it to github
6. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
7. Ensure the sample is working and fill in the below deliverables 1. Note: Don't forget the client commands are /name and /connect
8. Generate the output file once done and add it to your local repository
9. Git add/commit/push all changes
10. Complete the pull request merge from the step in the beginning
11. Locally checkout main
12. git pull origin main

Branch name: Milestone1

Group

100%

Group: Start Up
Tasks: 2
Points: 3

COLLAPSE

Task

100%

Group: Start Up
Task #1: Start Up
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

100%

Group: Start Up
Task #1: Start Up
Sub Task #1: Show the Server starting via command line and listening for connections

Task Screenshots

Gallery Style: 2 Columns

4 2 1

server properly running after compiling all dependencies in the folder

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Start Up
Task #1: Start Up
Sub Task #2: Show the Server Code that listens for connections

Task Screenshots

Gallery Style: 2 Columns

4 2 1

server listening

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code related to starting up and waiting for connections

Response:

this code is first run at the start/initialization of a server enum, in which after it trys to create a serversocket, create the first room, lobby, and while it is running, it will wait for incoming clients, in which the serversocket will try to accept. If accepted, the incomingCLient is made a serverThread.

Sub-Task

100%

Group: Start Up

Task #1: Start Up

Sub Task #3: Show the Client starting via command line

Task Screenshots

Gallery Style: 2 Columns

4 2 1

clients connected

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Start Up

Task #1: Start Up

Sub Task #4: Show the Client Code that prepares the client and waits for user input

Task Screenshots

Gallery Style: 2 Columns

4 2 1

client is initialized in start

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow leading up to and including waiting for user input

Response:

header

what happens is that the sys will print that the client is starting, in which the completablefuture object creates a thread that will listen to when the user sends input.

End of Task 1

Task

100%

Group: Start Up
Task #2: Connecting
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:

Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

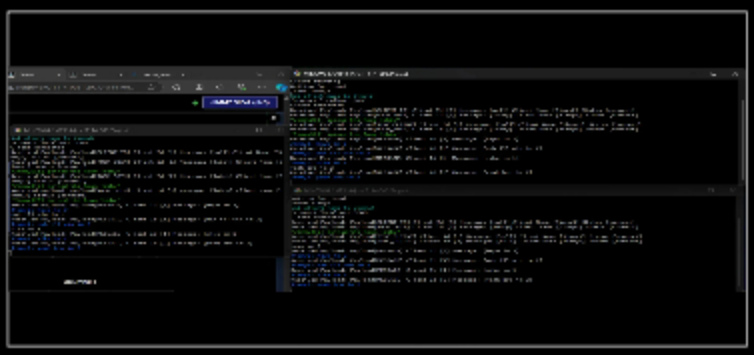
Sub-Task 100%

Group: Start Up
Task #2: Connecting
Sub Task #1: Show 3 Clients connecting to the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1



3 clients connected and communicating

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task 100%

Group: Start Up
Task #2: Connecting
Sub Task #2: Show the code related to Clients connecting to the Server (including the two needed commands)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

proccess client command, deals with logic for user when inputed /name /connect and /user

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow

Response:

Process client commands method, deals with logic for user when inputed /name /connect and /user. the /connect

speficially calls the isConnection method, which checks for the correct port, and if the user is connected via localhost or speficic ip address.

End of Task 2

End of Group: Start Up
Task Status: 2/2

Group

100%

Group: Communication
Tasks: 2
Points: 3

COLLAPSE

Task

100%

Group: Communication
Task #1: Communication
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

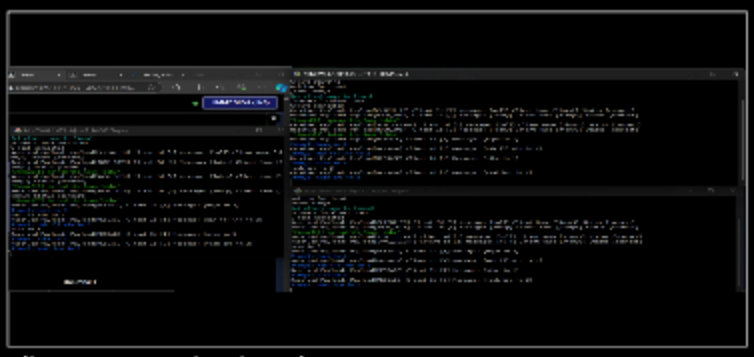
100%

Group: Communication
Task #1: Communication
Sub Task #1: Show each Client sending and receiving messages

Task Screenshots

Gallery Style: 2 Columns

4 2 1



client communication shown

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task 100%

Group: Communication
Task #1: Communication
Sub Task #2: Show the code related to the Client-side of getting a user message and sending it over the socket

Task Screenshots

Gallery Style: 2 Columns

4 2 1

proccessPayload, deals with proccess payload info sent through server thread

deals specifically for when the payload is enum type MESSAGE

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

proccessPayload, deals with proccess payload info sent through server thread deals specifically for when the payload is enum type MESSAGE

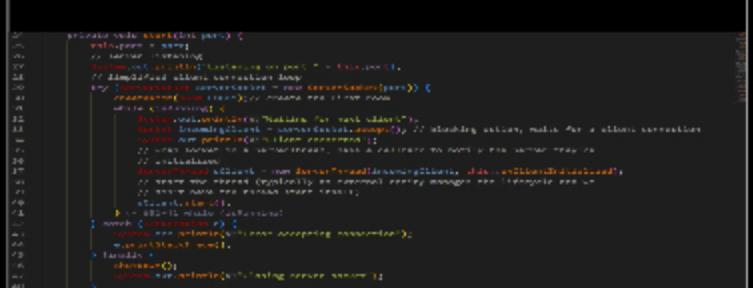
Sub-Task 100%

Group: Communication
Task #1: Communication
Sub Task #3: Show the code related to the Server-side receiving the message and relaying it to each connected Client

Task Screenshots

Gallery Style: 2 Columns

4 2 1



start method in server thread

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

this is ran in the server method, in which it will wait to accept any incoming server threads, in which it will accept the socket, and have it initilized server side.

Subtask Feedback 12/5/2024 9:18:48 PM:

wrong screenshot

Sub-Task

100%

Group: Communication
Task #1: Communication
Sub Task #4: Show the code related to the Client receiving messages from the Server-side and presenting them

Task Screenshots

Gallery Style: 2 Columns

4 2 1

process payload function

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

this command process when payloads come back from the serverthread, and is dealt with accordingly, either as a command or message.

End of Task 1

Task

100%

Group: Communication
Task #2: Rooms
Weight: ~50%
Points: ~1.50

COLLAPSE

Details:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

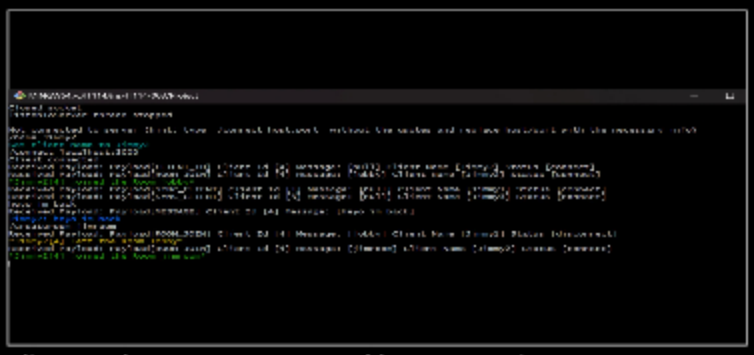
100%

Group: Communication
Task #2: Rooms
Sub Task #1: Show Clients can Create Rooms

Task Screenshots

Gallery Style: 2 Columns

4 2 1



client makes a new room and leaves previous one

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

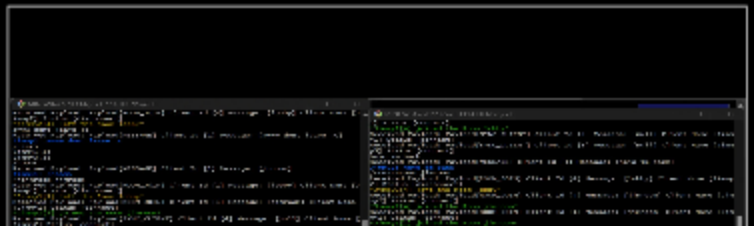
100%

Group: Communication
Task #2: Rooms
Sub Task #2: Show Clients can Join Rooms (leave/join messages should be visible)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



user joins existing room created in last question

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Communication

Task #2: Rooms

Sub Task #3: Show the Client code related to the create/join room commands

Task Screenshots

Gallery Style: 2 Columns

4 2 1

process client command, checks for join and create room cases

methods for send create and join room message to payload

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

in client process command, it checks if the string contains the command char "/", and then if the case = joinroom or createroom, and sends it with the according methods, send join/create method

Sub-Task

100%

Group: Communication

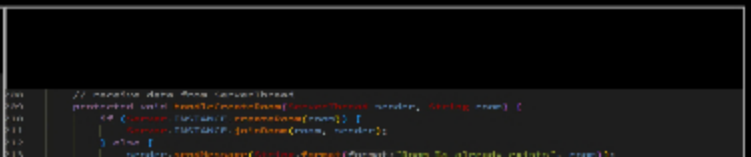
Task #2: Rooms

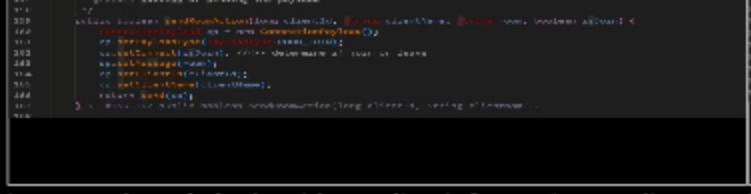
Sub Task #4: Show the ServerThread/Room code handling the create/join process

Task Screenshots

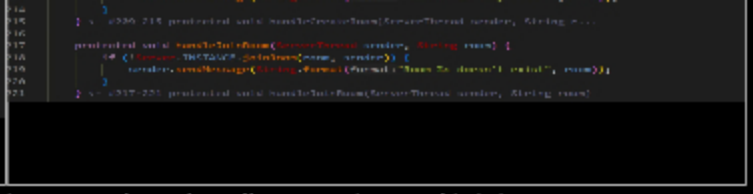
Gallery Style: 2 Columns

4 2 1





in serverthread, deals with sending information to client when there is changes to a users room.



in room class, handles creating and joining room commands

in room class, method adds client when command processes successfully

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

photo1: in serverthread, deals with sending information to client when there is changes to a users room. photo2: in room class, handles creating and joining room commands photo3: in room class, method adds client when command processes successfully

Sub-Task

100%

Group: Communication

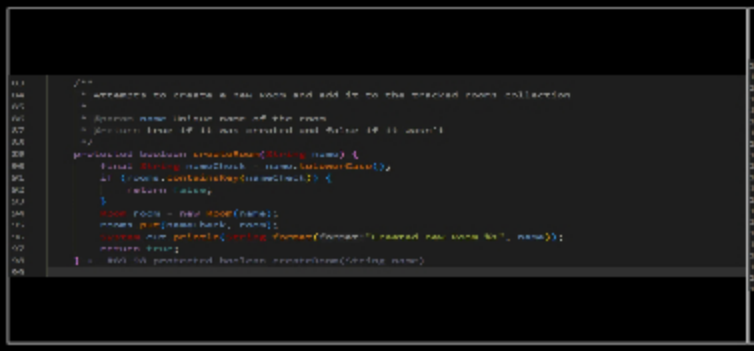
Task #2: Rooms

Sub Task #5: Show the Server code for handling the create/join process

Task Screenshots

Gallery Style: 2 Columns

4 2 1



create room method in server

handles join room in server

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

server deals with actual creation and approval of join/switching rooms, and these two methods handle it

Sub-Task

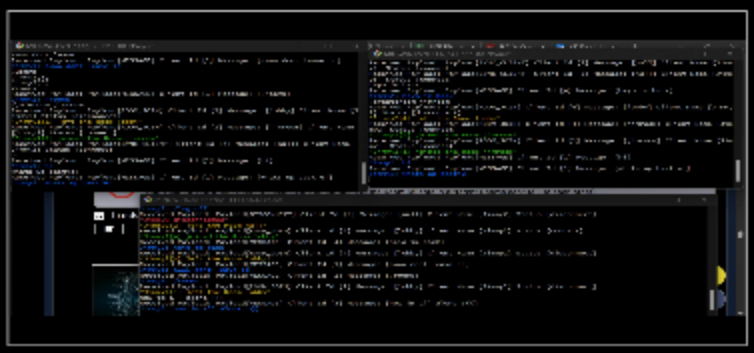
100%

Group: Communication
Task #2: Rooms
Sub Task #6: Show that Client messages are constrained to the Room (clients in different Rooms can't talk to each other)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



top left and right are in a different room then the bottom, so message in the bottom terminal is not seen by jimmy1/ jimmy2

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain why/how it works this way

Response:

messages are echoed depending on what room the client is in, so clients in different rooms dont get message echoed in different rooms

End of Task 2

End of Group: Communication
Task Status: 2/2

Group

100%

Group: Disconnecting/Termination
Tasks: 1
Points: 3

COLLAPSE

Task

100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Weight: ~100%
Points: ~3.00

COLLAPSE

Details:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.

Columns: 1

Sub-Task

100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Sub Task #1: Show Clients gracefully disconnecting (should not crash Server or other Clients)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

jimmy2 gracefully disconnects, can be seen in jimmy1 terminal bc they are in the same room

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

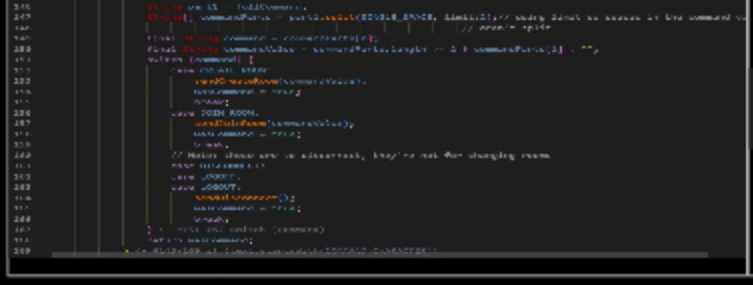
100%

Group: Disconnecting/Termination
Task #1: Disconnecting
Sub Task #2: Show the code related to Clients disconnecting

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
    private void sendDisconnect() {
        Payload p = new Payload();
        p.setPayloadType(PayloadType.DISCONNECT);
        send(p);
    } <- #203-207 private void sendDisconnect()
```

client, process disconnect command as either disconnect, logoff, or logout

in client, sends required info to serverthread for disconnecting

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

proccess command check for the disconnect, logoff, or logout key words, in which it calls the senddisconnect method, which tells server thread there is a payload for disconnecting

Sub-Task

100%

Group: Disconnecting/Termination

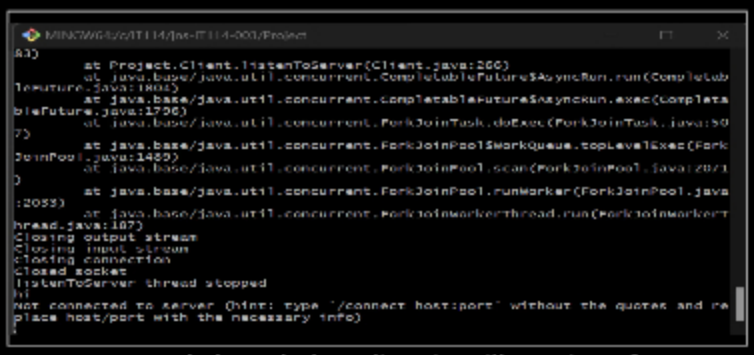
Task #1: Disconnecting

Sub Task #3: Show the Server terminating (Clients should be disconnected but still running)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
A3)
        at Project.Client.ListenToServer(Client.java:266)
        at java.base/java.util.concurrent.CompletableFuture$AsyncRun.run(CompletableFuture.java:1804)
        at java.base/java.util.concurrent.CompletableFuture$AsyncRun.exec(CompletableFuture.java:1796)
        at java.base/java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:507)
        at java.base/java.util.concurrent.ForkJoinPool$WorkQueue.topLevelExec(ForkJoinPool.java:1489)
        at java.base/java.util.concurrent.ForkJoinPool.scan(ForkJoinPool.java:2071)
        at java.base/java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:2035)
        at java.base/java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:187)
Closing output stream
Closing input stream
closing connection
Closed socket
ListenToServer thread stopped
hi
Not connected to server (hint: type `/connect host:port` without the quotes and replace host/port with the necessary info)
```

server stopped abruptly, but client is still running after

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Disconnecting/Termination

Task #1: Disconnecting

Sub Task #4: Show the Server code related to handling termination

Task Screenshots

Gallery Style: 2 Columns

4 2 1

code deals with exceptions, and if reaches bottom of method, something happened = server stopped

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown (ucid/date must be present)*

Task Response Prompt

Briefly explain the code/logic/flow involved

Response:

code deals with exceptions, and if reaches bottom of method, something happened = server stopped

End of Task 1

End of Group: Disconnecting/Termination
Task Status: 1/1

Group

100%

Group: Misc
Tasks: 3
Points: 1

COLLAPSE

Task

100%

Group: Misc
Task #1: Add the pull request link for this branch
Weight: ~33%
Points: ~0.33

COLLAPSE

Task URLs

URL #1
https://github.com/jnsnjit/jns-IT114-003/pull/13

URL
https://github.com/jnsnjit/jns-IT114-003/pull/13

End of Task 1

Task

100%

Group: Misc
Task #2: Talk about any issues or learnings during this assignment
Weight: ~33%
Points: ~0.33

COLLAPSE

Details:
Few related sentences about the Project/sockets topics

Task Response Prompt

Response:

well it took a while to go through each class and understand them, i think once milestone2 adds the extra folders inside the main one to split the class in client server and payload stuff, itll make understand the segments of the code easier. plus the screenshotting took forever...

End of Task 2

Task

100%

Group: Misc
Task #3: WakaTime Screenshot
Weight: ~33%
Points: ~0.33

COLLAPSE

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: 2 Columns

4 2 1

waka time src, about two hours in this repo for this

assignment

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment