Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-450-M2024/it114-module-5-project-milestone-1/grade/arc73

IT114-450-M2024 - [IT114] Module 5 Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 6/22/2024 7:50:05 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

Tasks: 8 Points: 10.00

Start Up (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Start Up

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. The goal is to show you understand what segments are related to the prompts.

#1) Show the Server starting via

Caption (required) ✓

Describe/highlight what's being shown

Server listening for connections

#2) Show the Server Code that listens

Caption (required) ✓

Describe/highlight what's being shown (ucid/date must be present)

Server code demonstrating how it listens for connections

Explanation (required)

✓

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

PREVIEW RESPONSE

The "start(int port)" method is the foundation for the server to listen on a specific port by creating a "ServerSocket" which

#3) Show the Client starting via

Caption (required) ✓

Describe/highlight what's being shown

Client starting via command line

#4) Show the Client Code that prepares

Caption (required) ✓

Describe/highlight what's being shown (ucid/date must be present)

Client code demonstrating how the client is prepared and waits for user input

Explanation (required)

✓

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

PREVIEW RESPONSE

The "start()" method initializes the client and prepares it. Once it is creating, a thread to listen for input is created

explicitly states that it is listening and waiting for connections. The int port was declared as 3000 based on the previous section of the code. "serverSocket.accept()" waits for a client and adds the connection, for each added connection, a ServerThread is started to handle the client. However, if an error occurs, the server disconnects/shuts down.

using the "listenToInput()" method. The while loop allows for continuous processing of user input if connected to the server. Otherwise, if there is no connection to the server, the error will be caught in the catch case which defaults out of the loop and no longer processes client input.

COLLAPSE

Task #2 - Points: 1

Text: Connecting

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. The goal is to show you understand what segments are related to the prompts.

#1) Show 3 Clients connecting

Caption (required) ✓

Describe/highlight what's being shown

3 clients connecting to the server

#2) Show the code related to Clients

Caption (required) ✓

Describe/highlight what's being shown (ucid/date must be present)

Code relating to clients connecting to the server

Explanation (required)

Briefly explain the code/logic/flow

PREVIEW RESPONSE

The "connect()" method allows the client to connect to the server. The method takes an IP address and port as parameters and sets-up a socket for communication between the server and client. From the "processClientCommand" method, validity of input from the user is checked by "isConnection". if valid, it grabs the IP and port and allows for the "connect" method to execute and establishes a connection from the client to the server.

Communication (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Communication

Details:

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

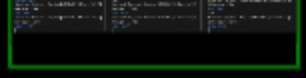
Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade. The goal is to show you understand what segments are related to the prompts.

#1) Show each Client sending and

#2) Show the code related to the Client-

#3) Show the code related to the Server-

#4) Show the code related to the Client



Caption (required) ✓
Describe/highlight what's being shown
Clients sending and receiving messages

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Code related to the client side of getting a user message and sending it over the socket

Explanation (required)
✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
The "listenToInput" method uses a while loop to continuously wait for and process user input and messaging. If connected to the server, and the input is not a command, it executes the "sendMessage" method which packages the input/message into a "payload" and send it over the socket using the "send" method.

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Code relating to the server side receiving the message and relaying it to each connected client

Explanation (required)
✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
The "ServerThread" class listens for communication from clients through the "run" method which utilizes a while loop that continuously takes in "payload" objects from client-side. Once a message payload is received, it executes the case "Message", which calls "currentRoom.sendMessag payload.getMessage());" which broadcasts the message to the room that the clients are currently in.

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Code related to the client receiving messages from the server side and presenting them

Explanation (required)
✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
The "Client" class creates a socket connection to the server and uses "ObjectInputStream" to recieve "payload" messgae objects from the server. The "listenToServer()" method continously waits for and processes messages from the client. Each recieved "payload" calls the "processPayload()" method which formats/displays messgaes to the client side.

COLLAPSE

Task #2 - Points: 1
Text: Rooms

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

Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade.
The goal is to show you understand what segments are related to the prompts.

#1) Show Clients can Create

Caption (required) ✓
Describe/highlight what's being shown
Client created room "test"

#2) Show Clients can Join Rooms

Caption (required) ✓
Describe/highlight what's being shown
Other clients joined room "test"

#3) Show the Client code related to the

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Client code relating to the create/join rooms commands

Explanation (required)
✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
The "sendCreateRoom" and "sendJoinRoom" method work in similar ways in how they send messages to the server to process requests in creating or joining rooms. They package the room name into a "payload" with the appropriate type (ROOM_CREATE/JOIN) and send it over the socket to the server using the "send" method.

#4) Show the ServerThread/Rc code

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Code handling the create/join process

Explanation (required)
✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
The "handleCreateRoom" method checks if a room with the given name already exists using "Server.INSTANCE.createRoom(r if-else statement. If not, then the room is created and the user who attempted to create the room is joined to it. However, if it already exists, the client is informed that the room exists. The "handleJoinRoom" method tries to add the client to the room using "Server.INSTANCE.joinRoom(roo sender)". However, if the room does not exist, the client is informed it does

not exist.

#5) Show the Server code for handling

Caption (required) ✓

Describe/highlight what's being shown (ucid/date must be present)

Server code for handling the create/join process

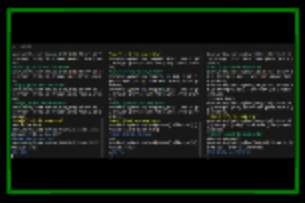
Explanation (required) ✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The "createRoom()" method checks if a room with the given name in the parameter already exists. If not, a new "Room" object is created and gets added to the "room" collection. Once added, a message is output to the console stating the room was succesfully created. Otherwise, if a room with the given name already exists, it will return false. The "joinRoom()" method checks if a room with the given name exists in the "room" collection. If so, it takes out the client from the current room and adds them to the specified room.

#6) Show that Client messages



Caption (required) ✓

Describe/highlight what's being shown

Message from John in lobby is not shown in room "test"

Explanation (required) ✓

Briefly explain why/how it works this way

PREVIEW RESPONSE

When a client joins a specific room, the client is assigned an attribute that relates to the room they are in using "client.setCurrentRoom(this);". This allows for proper management of tracking which room the client is currently in. This allows clients who have the same attribute to be the only recipients of the message. Otherwise, clients who do not have this attribute do not recieve the message on their end.

Otherwise, returns false.

Disconnecting/Termination (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Disconnecting

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. The goal is to show you understand what segments are related to the prompts.

#1) Show Clients gracefully

Caption (required) ✓

Describe/highlight what's being shown

Joe disconnected from room

#2) Show the code related to Clients

Caption (required) ✓

Describe/highlight what's being shown (ucid/date must be present)

Code relating to clients disconnecting

Explanation (required) ✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The "closeServerConnection" method removes the resources associated with the client's disconnection such as client data, known clients, and closing

#3) Show the Server terminating

Caption (required) ✓

Describe/highlight what's being shown

CTRL + C from server terminal terminates server, all 3 clients disconnected

#4) Show the Server code related to

Caption (required) ✓

Describe/highlight what's being shown (ucid/date must be present)

Server code relating to handling termination

Explanation (required) ✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The "shutdown()" method handles the disconnection of clients and cleanup tasks upon the server shutting down. The code iterates through all rooms stored

streams which the client communicated to the server socket. The "processDisconnect" method outputs a message about the client disconnecting.

in the "rooms" class and disconnects all clients from each room using the "removeIf()" method which executes when the condition of a client being in a room is true.

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/AhmedCho/arc73-IT114-450/pull/9

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:

Some issues that I've experienced while doing this assignment is transferring over the code from the Part 5 folder from the sample repository to my local files. I found that copy and pasting the code would result in numerous syntax issues. Instead, I downloaded the raw files from the sample repository and uploaded them to my folder which resolved the issue. Another issue I came across is being able to identify the specific parts of the code the questions were asking for. Any time I had a hard time understanding what I am looking for, I would start from the beginning of the file, and work my way through the logic of the code until I find the section of code that I am looking for regarding the server/client.

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.

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large

[redacted]

Projects • arc73-IT114-450

3 hrs 45 mins over the Last 7 Days in arc73-IT114-450 under all branches.

[redacted]

Wakatime Screenshot #1

Files

1 hr 49 mins	Module4/Part3HW/Server.java
27 mins	...4/Part3HW/ServerThread.java
22 mins	Project/BaseServerThread.java
9 mins	Client.java
7 mins	Project/ServerThread.java
7 mins	Module4/Part3HW/Client.java
5 mins	Part1/Server.java
3 mins	Server.java
3 mins	Project/Client.java
3 mins	Project/TextFX.java
3 mins	...le4/Part3/ServerThread.java
2 mins	Module4/Part2/Server.java
1 min	Project/.gitignore
1 min	Project/Server.java
1 min	Module4/Part3HW/.gitignore
1 min	Project/Room.java
1 min	Module4/Part1/Server.java
1 min	Project/ConnectionPayload.java
1 min	Module4/Client.java
1 min	Project/PayloadType.java
1 min	.gitignore
59 secs	Module4/Part3/Server.java
56 secs	Project/Payload.java
49 secs	Module4/Part1/Client.java
47 secs	Project/ClientData.java
44 secs	Module4/Part3/Client.java
30 secs	Part1/Client.java
29 secs	Module4/Part2/Client.java
16 secs	Module1/.gitignore
6 secs	Module4/Part2

Branches

2 hrs 33 mins	main
49 mins	Milestone1
22 mins	M4-Sockets3-Homework

5 secs	Module4/Part1
5 secs	Project/TestFX.java
3 secs	Module4/Part3
2 secs	..-part-3_IT114-450-M2024.pdf
0 secs	..art-1-3_IT114-450-M2024.pdf
0 secs	..-part-3_IT114-450-M2024.pdf

Wakatime for Milestone1

End of Assignment