Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-450-M2024/it114-module-5-project-milestone-1/grade/jah89

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

Submissions:

Submission Selection

1 Submission [active] 6/17/2024 2:23:15 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 ^

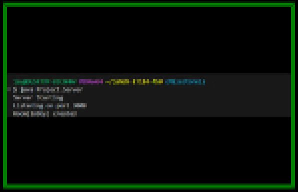
Task #1 - Points: 1

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
showing server starting

#2) Show the Server Code that listens

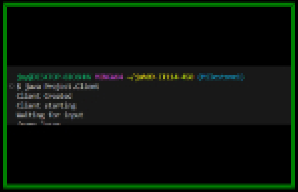
Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Showing section of code that nadles

Explanation (required) ✓
Briefly explain the code related to starting up and waiting for connections

PREVIEW RESPONSE

The first thing the code does is initializes a port by using this.port=port. It then creeates a server socket by using try(ServerSocker serversocket).... This is

#3) Show the Client starting via



Caption (required) ✓
Describe/highlight what's being shown
Showing client starting in cmd

#4) Show the Client Code that

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Showing the client code that prepares client

Explanation (required) ✓
Briefly explain the code/logic/flow leading up to and including waiting for user input

PREVIEW RESPONSE

The first step in the code is to start the client. In the code above the start method is called and prints a message "Client Starting". It then uses "CompletableFuture.runAsync....

created to listen for the incoming connections on the specified port. Once this occurs the code creates a room using CreateRoom(Room.lobby). This is where the clients will be placed once they connect to the server. Next there is a while loop (isRunning), while this is true the server waits for an incoming client connection using the serversocket.accept(). And finally at the end of the code they are exceptions to catch if an error was to occur and a finally statement to shutdown the server once

which is used to run the "listentoinput" method in a separate thread so it doesn't block the main thread. Then the code calls the listenToInput method to listen for keyboard input from the user. We track the user input using the scanner object "si". It then prints a message out to show its waiting for the user input. Then there is a while loop "isRunning" that reads the input from the user. The processClientCommand(line) method is called in the code to check if the input entered was a command or not, if it is a command it will return true. If it is not a command it will just print the message sent as a normal chat message using "sendMessage(line).

COLLAPSE

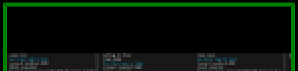
Task #2 - Points: 1

Text: Connecting

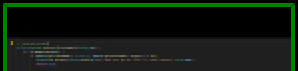
Details:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

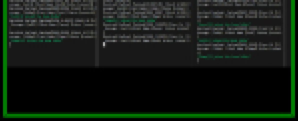
Capturing all possible code (i.e., including a lot of irrelevant code) can lead to a reduced grade. The goal is to show you understand what segments are related to the prompts.

#1) Show 3 Clients connecting

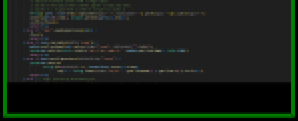


#2) Show the code related to Clients





Caption (required) ✓
Describe/highlight what's being shown
showing 3 clients connecting to the server



Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Showing the snppet of code related to clients connecting to server

Explanation (required) ✓
Briefly explain the code/logic/flow

PREVIEW RESPONSE

The first step in the code is checking that the input the user gives matches the correct format and isText. It then checks to make sure the user has their name set to something and if not they get alerted with a message. It then trims the text and turns multipled spaces into just one space. This occurs to get the host and port, and calls the "connect" method to establish the connection. There then is else ifs for when user types /quit which will close the client connection. Also if the input from the client starts with /name it will take the text and send it to the "myData.setClientName" which will complete the connection process.

● Communication (3 pts.)
^COLLAPSE ^

● ^COLLAPSE ^

Task #1 - Points: 1

Text: Communication

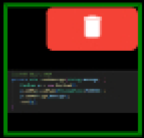
ⓘDetails:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

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

Caption (required) ✓
Describe/highlight what's being shown
showing the clients sending and recieving messages

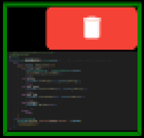
#2) Show the code related to the Client-



Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Showing client side of user getting message over socket

Explanation (required) ✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
In this code the listentoInput method is always listening for user input from the console. The method si.NextLine checks the input from the user and checks to

#3) Show the code related to the



Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Showing code for server side recieving the msg

Explanation (required) ✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
First in this code is the processPayload method where it handles different types of payloads recieved from the client. In this method it checks the type of payload and then

#4) Show the code related to the Client

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Showing code related to client recieving msgs from server side.

Explanation (required) ✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
The listentoServer method is set to run in a separate thread and always listen for messages from the server while the client is running and connected. There's a while loop for isRunning and

the user and checks to see if it is a command with the the "processClientCommand(lin Then there is a sendMessage method that creates a payload object to input to the output stream and flushes to make sure it is sent right away.

payload and then processes it depending on the type. In the room class there is a sendMessage method that relays the message from the sender to all the other clients in the room. Finally in the serverthread class there is a sendMessage method that creates a payload object with the message from the user and sends it to the client.

isConnected to ensure that the client always is listening for messages as long as it is running and connected to server. The payload fromServer=(payload).... is a blocking operation that waits for a message from the server. It reads an object from the input and casts a payload type. If the payload is recieved the method calls "processPayload(fromServer)" to handle the receieved messages. It processes ther payloads based on its type and then performs the actions based off the type, such as displaying the message to the client. Then near the end of the code there is a try catch block to handle exceptions such as connection dropped.

^COLLAPSE^

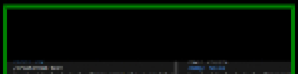
Task #2 - Points: 1

Text: Rooms

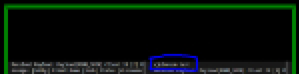
Details:
Important: Code screenshots should be fairly concise (try to show only the sections of code relevant to the question)

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

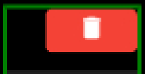
#1) Show Clients can Create



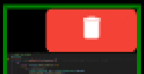
#2) Show Clients can Join Rooms

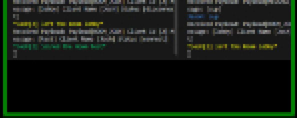


#3) Show the Client code related to the



#4) Show the ServerThread/Ro code

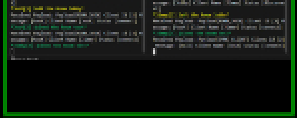




Caption (required) ✓

Describe/highlight what's being shown

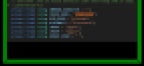
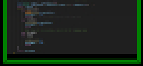
Showing creating A room



Caption (required) ✓

Describe/highlight what's being shown

Showing clients an join and leave

Caption (required) ✓

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

Showing client code related to create and joins commands

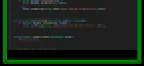
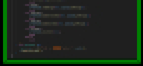
Explanation (required)

✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

In the first screenshot the code shows the constants that are assigned. These are used to define the different commands that the client can issue. Such as joinroom and createroom. In the next code snippet the processClientCommand method is able to process different client commands. It parses the command and its value and uses a switch statement to determine the command to process. For the createroom command it calls the "sendCreateRoom" etc. There is then a sendcreateroom method that creates a payload for ROOM_CREATE command and sends it to the server. Finally the sendJoinRoom method

Caption (required) ✓

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

Showing serverthread/room code handling join/create process

Explanation (required)

✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The first method of handleCreateRoom is used when "ROOM_CREATE" payload is recieved. This code checks if the room can be created by using "Server.instance.createRoom(roo If the room is created it lets the sender to the created room using "Server.instance.joinroom(room, sender)". If a room already exists it sends a message to the user showing the room already exists. The handleJoinRoom method is used when "ROOM_JOIN" payload is recieved. In the serverthread class the method processpayloads processes different types of payloads recieved from the client. Some of these include client connect, disconnect, room join, room create, etc.

creates a payload for ROOM_JOIN that sends it to the server.

#5) Show the Server code for handling

Caption (required) ✓
Describe/highlight what's being shown (ucid/date must be present)
Showing server code to handle create/join

Explanation (required) ✓
Briefly explain the code/logic/flow involved
PREVIEW RESPONSE
For the createRoom method the first thing it does is convert the room name to lowercase to make sure everything is consistent. The code then uses "rooms.containsKey(nameC to check if the room already exists or not. If the room doesnt exist it creates a new room instance with the name provided by the user. The code then logs the creation of the room and returns the value true if the room was created or false if it already existed. The joinroom method does basically the same thing except

#6) Show that Client messages

Caption (required) ✓
Describe/highlight what's being shown
Showing client not in room their msg doesnt go the others

Explanation (required) ✓
Briefly explain why/how it works this way
PREVIEW RESPONSE
This works that clients can only talk to people in the same room because each room manages its own list of clients. Messages that are sent by the client are only broadcasted to clients within the same room. The serverthread class handles the communication between the server and single client so when a client sends a message the serverthread passes the message to the room the client is in and that room only. When a client sends a message it is put into a payload object with the type "MESSAGE". The

the same thing except for joining a room. The code checks if the room exists when trying to join. If the client is already in a different room if they join a different room it will remove them from the current room using current.removedClient(clien Then the code uses "next.addClient(client)" to add the client to the current room.

"MESSAGE": The serverthread then processes the payload and calls the sendMessage method. This sendmessage method in the room class iterates over all the clients in the the clients in the "ClientsInRoom" map. So since each room has its own "ClientsInRoom" map messages can only be sent to the client in the same room.

● 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

Showing discnnecting

#2) Show the code related to Clients

Caption (required) ✓

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

Showing code related to clients disconnecting

#3) Show the Server terminating

Caption (required) ✓

Describe/highlight what's being shown

showing server terminated

#4) Show the Server code related to

Caption (required) ✓

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

Showing server code related to termination

Explanation (required)

✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

In the disconnect method the code uses synchronized keyword because it makes sure that only one thread can execute it at a time for a given instance of a room. This method first checks ifthe room is running by using isRunning. If the room isn't running it will return to prevent any further action. The code also gets the clientid by using long id = client.getCLientID(). Once the user disconnects a notifcation is sent out to notify other clients in the room. The disconnect method gets called client.disconnect() to disconnect the client and then is removed from the clientRooms map using clientsInRoom.Remove(client.getClientID()).

Explanation (required)

✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The first code snippet shows a constructor that sets up a shutdown hook that the JVM calls when it is shutting down. This executes shutdown() method to perform important tasks before the server terminates. Then in the shutdown method it sets the isRunning to false to show that the server is no longer running. This iterates to every room and calls "disconnectAll() on each room and disconnects the clients.

● Misc (1 pt.)

COLLAPSE

● COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/jah89/jah89-IT114-450/pull/14

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:

The only issue I had with this assignment was finding the snippets of code and making sure to provide all code related to that part of the problem. For some problems I added 3 screenshots just to ensure I was covering every part of the topic necessary. I'm hoping this doesn't end up being wrong for adding snippets of code to make sure I was really showing every part of the code that corresponds to what it was asking for .

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

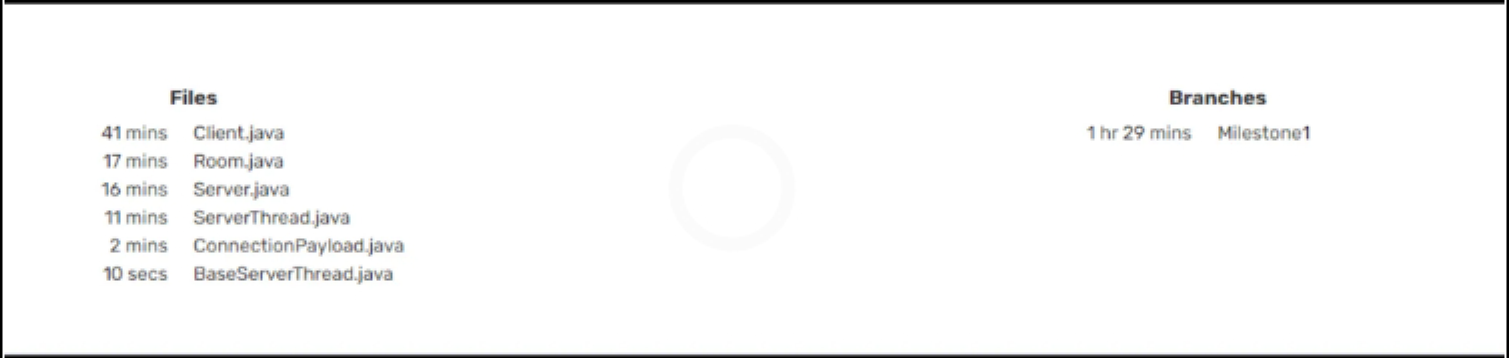
Grab a snippet showing the approximate time involved that clearly shows your repository.

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

Task Screenshots:

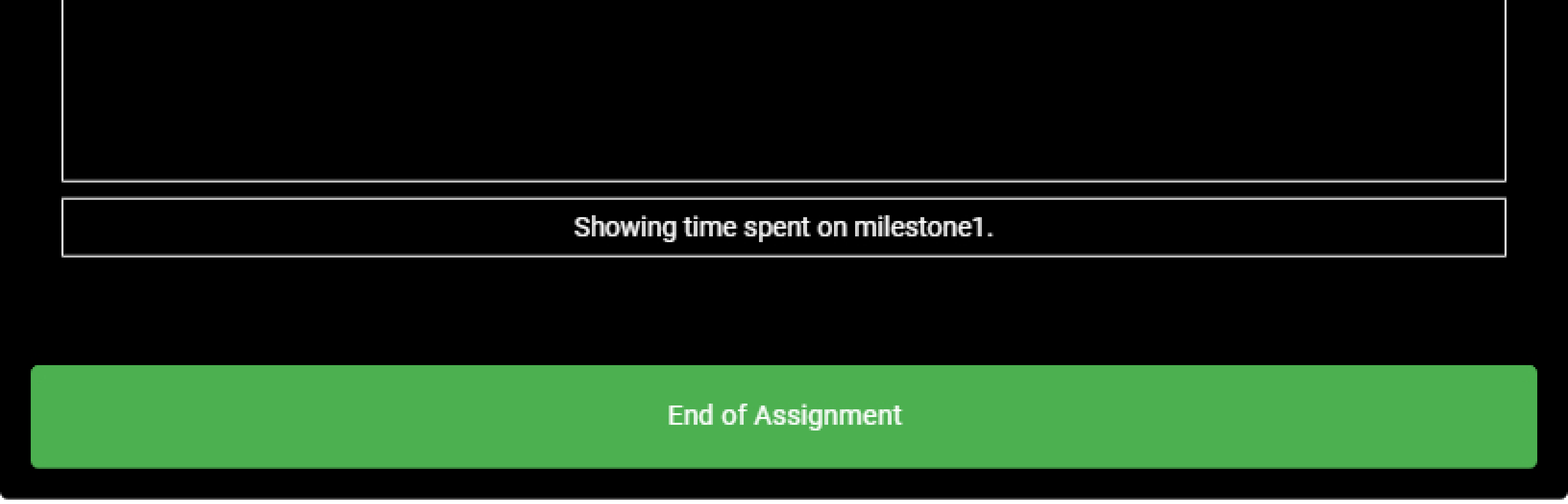
Gallery Style: Large View

Small Medium Large



Files	
41 mins	Client.java
17 mins	Room.java
16 mins	Server.java
11 mins	ServerThread.java
2 mins	ConnectionPayload.java
10 secs	BaseServerThread.java

Branches	
1 hr 29 mins	Milestone1



Showing time spent on milestone1.

End of Assignment