Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-450-M2024/it114-module-5-project-milestone-1/grade/yh68

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

Submissions:

Submission Selection

1 Submission [active] 6/22/2024 7:03:56 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 the Server starting via command line and listening for connections.

#2) Show the Server Code that listens

Caption (required) ✓

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

Showing the server code that listens for connections

Explanation (required) ✓

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

PREVIEW RESPONSE

A new socket is assigned and bound to a specific port.

#3) Show the Client starting via

Caption (required) ✓

Describe/highlight what's being shown

Showing the 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)

Showing the client code that prepares the client and waits for the user input.

Explanation (required) ✓

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

PREVIEW RESPONSE

A scanner object is initialized to accept user input. It then checks if the message is a client command, if it isn't then

it is sent as a message.

COLLAPSE

Task #2 - Points: 1

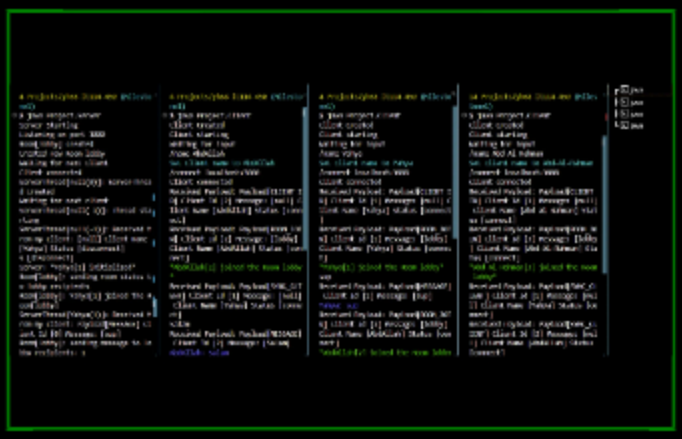
Text: Connecting

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 to the Server



Caption (required) ✓

Describe/highlight what's being shown

Showing 3 Clients connecting to the Server

#2) Show the code related to Clients connecting to the Server (including the two needed commands)

```
// yh68 6/22/2024
private boolean connect(String address, int port) {
    try {
        server = new Socket(address, port);
        // channel to send to server
        out = new ObjectOutputStream(server.getOutputStream());
        // channel to listen to server
        in = new ObjectInputStream(server.getInputStream());
        System.out.println("Client connected");
        // Use CompletableFuture to run listenToServer() in a separate thread
        CompletableFuture.runAsync(this::listenToServer);
    } catch (UnknownHostException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return isConnected();
}
```

Caption (required) ✓

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

Showing the code related to Clients connecting to the Server (the two needed commands dont fit in the screenshot)

Explanation (required) ✓

Briefly explain the code/logic/flow

PREVIEW RESPONSE

A socket instance is initialized named "server" and connects to the "address" and "port" which are parameter. The objectouputstream allows the client to send objects to the server. The objectinputstream allows the client to receive objects from the server. The 2 needed commands are /connect and /disconnect.

Communication (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

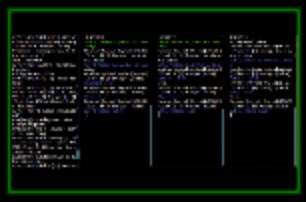
Text: Communication

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 each client sending and receiving messages.

#2) Show the code related to the Client-

Caption (required) ✓

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

Showing the 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 "payload" which contains the message is sent using the out.writeobject() method to send it over the socket. The out.flush()

#3) Show the code related to the Server-

Caption (required) ✓

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

Showing the code related to the serverside receiving the message and relaying it to each connected client

Explanation (required) ✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The processpayload() method manages different types of client requests using a switch statement to decide what happens based on

#4) Show the code related to the Client

Caption (required) ✓

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

Showing the 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 listentoserver() method constantly listens for incoming messages from the server. It uses a while loop to keep checking

method makes sure that the data is actually sent out to the server.

the payload type, client_connect, message, room_create, room_join, disconnect. The sendclientsync() method creates a connectionpayload with its parameters and then sends this payload to the client using the send() method to keep the client updated in real time.

for new payloads from the servers objectinputstream. When a new payload is received it gets processed it by calling processpayload().

^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.

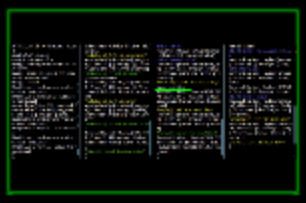
#1) Show Clients can Create

Caption (required) ✓

Describe/highlight what's being shown

Showing clients creating rooms

#2) Show Clients can Join Rooms



Caption (required) ✓

Describe/highlight what's being shown

Showing clients joining rooms

#3) Show the Client code related to the

Caption (required) ✓

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

Showing the client code related to the create/join room commands

Explanation (required)

✓

Briefly explain the

#4) Show the ServerThread/Rc code

Caption (required) ✓

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

Showing the serverthread/room code handling the create/join process

Explanation (required)

✓

Briefly explain the

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The sendcreateroom() method sends a request to create a room on the server. It creates a new payload object and sets the payload type to ROOM_CREATE and then the method calls send() to send the payload to the server.

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The serverthread class receives payloads from clients such as create room or join room. When ROOM_CREATE or ROOM_JOIN is received the method in the room class is triggered. The room class allows clients to create rooms or join them with handlecreateroom() and handlejoinroom() methods.

#5) Show the Server code for handling

Caption (required) ✓

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

Showing the server code for handling the create/join process

Explanation (required)

✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The createroom() method creates a new chat room with a given name and converts it to lowercase and checks if the room already exists. If the room doesn't

#6) Show that Client messages

Caption (required) ✓

Describe/highlight what's being shown

Showing that client messages are constrained to the room

Explanation (required)

✓

Briefly explain why/how it works this way

PREVIEW RESPONSE

When a client sends a message the processpayload() method in the serverthread class, the message is sent only to the current room by calling currentRoom.sendMessage().

already exist it creates a new room object and adds it to the list of rooms and returns true. If the room exists it returns false. The joinroom() method allows a client to join a chat room. It also converts the room name to lowercase and verifies if the room exists and if it does it removes the client from the current room they are in adds them to the new room and returns true. If the room does not exist it returns false.

This method makes sure that the client only sends messages to clients 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

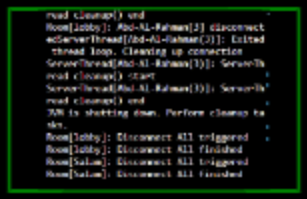
#2) Show the code related to Clients

```
// ykkk 1/23/2025
private void sendDisconnect() {
    Payload p = new Payload();
    p.setPayloadType(PayloadType.DISCONNECT);
    send(p);
}
```

Caption (required) ✓

Describe/highlight what's being shown

#3) Show the Server terminating



```
read cleanup() end
Room[lobby]: Abd-Al-Rahman[3] disconnect
eClientThread[Abd-Al-Rahman(3)]: Exited
 thread loop. Cleaning up connection
ServerThread[Abd-Al-Rahman(3)]: ServerTh
read cleanup() start
ServerThread[Abd-Al-Rahman(3)]: ServerTh
read cleanup() end
JEB is shutting down. Perform cleanup ta
sks.
Room[lobby]: Disconnect All triggered
Room[lobby]: Disconnect All finished
Room[kalam]: Disconnect All triggered
Room[kalam]: Disconnect All finished
```

Caption (required) ✓

Describe/highlight what's being shown

#4) Show the Server code related to

Caption (required) ✓

Describe/highlight what's being shown

Showing clients gracefully disconnecting

(ucid/date must be present)
Showing the code related to clients disconnecting

Explanation (required)

✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

When a client types in the ~~disconnect~~ the senddisconnect() method in the client code is responsible for notifying the server that the client wants to disconnect. It creates a payload object and sets its type to disconnect. This payload is then sent to the server using the send() method. By sending this payload the client is sending a disconnection request, allowing the server to remove the client from any active rooms.

Showing the server terminating

(ucid/date must be present)
Showing the server code related to handling termination

Explanation (required)

✓

Briefly explain the code/logic/flow involved

PREVIEW RESPONSE

The shutdown() method disconnects all clients and cleaning up resources when the server is shutting down. For each room removed, it calls disconnectAll() on the room object, which iterates through all connected clients and disconnects them. Overall, shutdown makes sure that the server gracefully terminates while disconnecting clients.

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/FreePalestine7/yh68-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:

No issues during this assignment. Just a lot of code analyzing, thankfully I didn't have to debug the code. It was a lot of code to read through. I feel like I am getting a better grasp on these new methods and learning how servers function in java which is really cool for me. I learned a little about sockets and payloads and feel like once I get used to them I will be able to elaborate on them some more.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

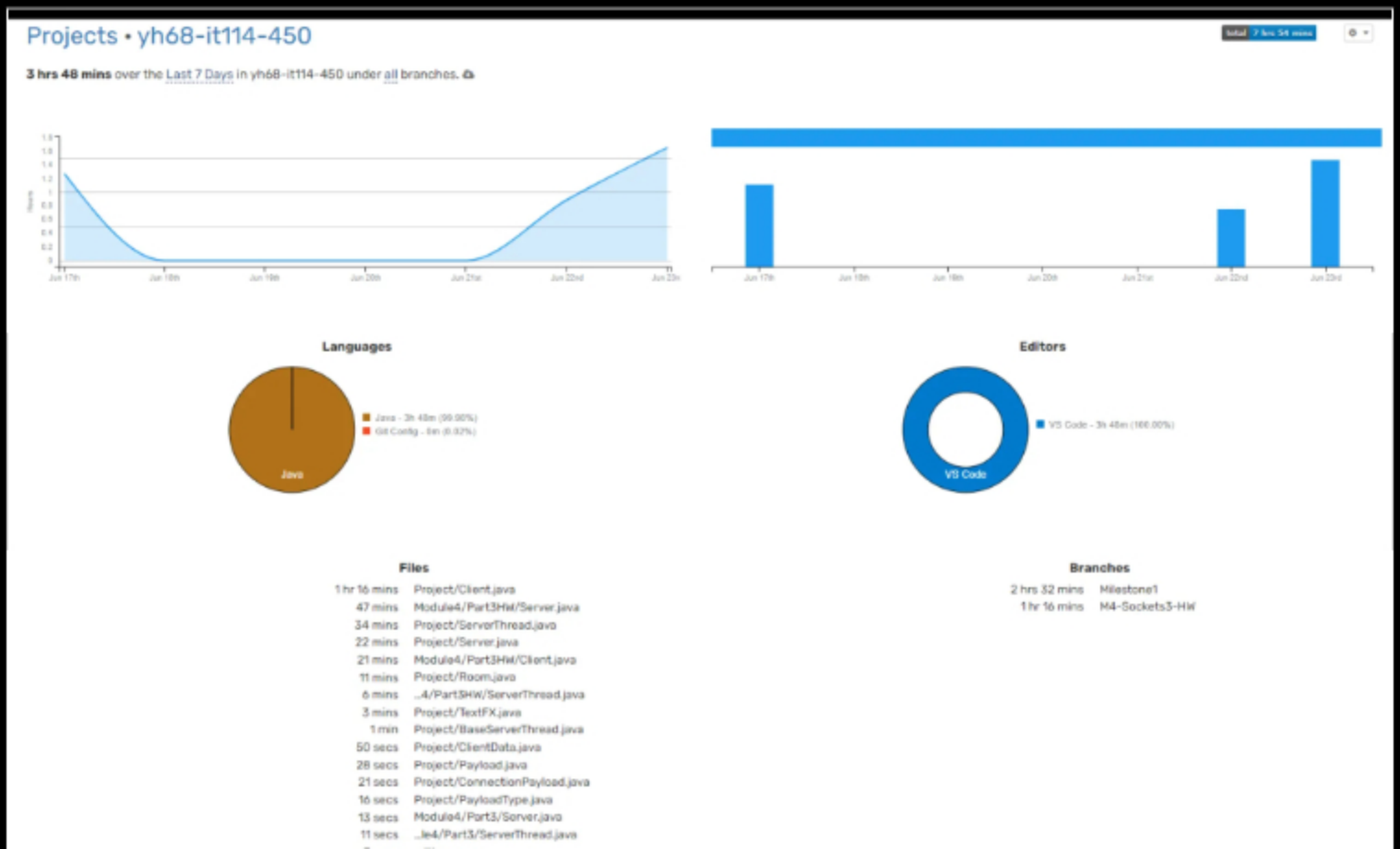
Grab a snippet showing the approximate time involved that clearly shows your repository.

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



3 secs .gitignore

Showing wakatime overview

End of Assignment