Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-project-milestone-1/grade/ms75

IT114-006-S2024 - [IT114] Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 3/19/2024 10:58:11 PM

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

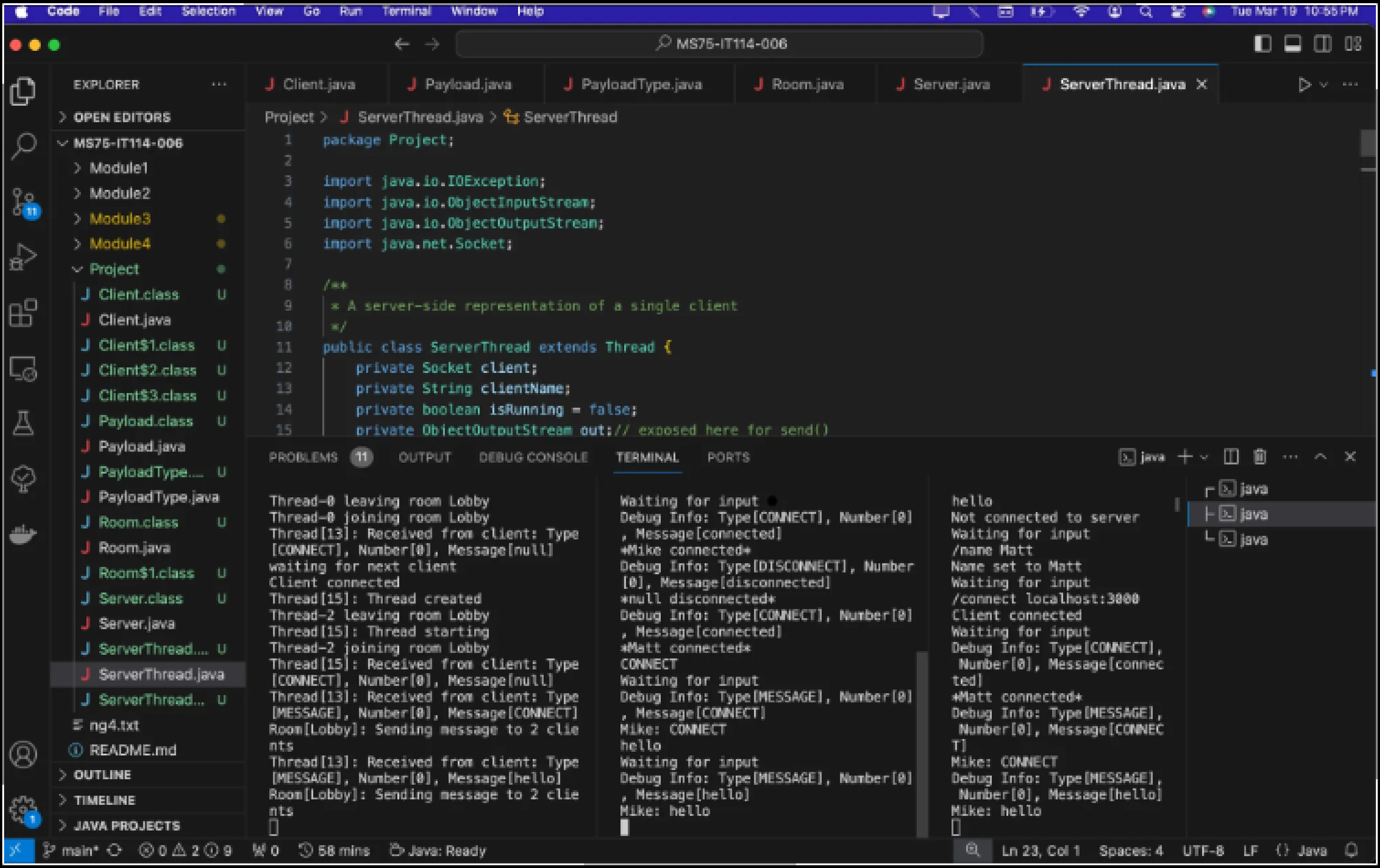
Checklist *The checkboxes are for your own tracking

#	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



```java
package Project;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

/**
 * A server-side representation of a single client
 */
public class ServerThread extends Thread {
    private Socket client;
    private String clientName;
    private boolean isRunning = false;
    private ObjectOutputStream out;// exposed here for send()
```

```
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[13]: Received from client: Type
[CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[15]: Thread created
Thread-2 leaving room Lobby
Thread[15]: Thread starting
Thread-2 joining room Lobby
Thread[15]: Received from client: Type
[CONNECT], Number[0], Message[null]
Thread[13]: Received from client: Type
[MESSAGE], Number[0], Message[CONNECT]
Room[Lobby]: Sending message to 2 clie
nts
Thread[13]: Received from client: Type
[MESSAGE], Number[0], Message[hello]
Room[Lobby]: Sending message to 2 clie
nts
```

```
Waiting for input
Debug Info: Type[CONNECT], Number[0]
, Message[connected]
*Mike connected*
Debug Info: Type[DISCONNECT], Number
[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0]
, Message[connected]
*Matt connected*
CONNECT
Waiting for input
Debug Info: Type[MESSAGE], Number[0]
, Message[CONNECT]
Mike: CONNECT
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0]
, Message[hello]
Mike: hello
```

```
hello
Not connected to server
Waiting for input
/name Matt
Name set to Matt
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT],
 Number[0], Message[connec
ted]
*Matt connected*
Debug Info: Type[MESSAGE],
 Number[0], Message[CONNEC
T]
Mike: CONNECT
Debug Info: Type[MESSAGE],
 Number[0], Message[hello]
Mike: hello
```

Screenshot shows (from left to right) Server, Client 1(Mike) and Client 2 (Matt)

Checklist Items (0)

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 class is managing how the server is setup and initialized and how connections are made between clients. The server class has the createNewRoom function which creates a new instance of a "room" which is where the chat functionality works. The server first needs to be compiled and ran. It enters a loop to allow new client connections and creates server threads for each client connection. Each thread handles connection for one client and data such as commands or messages are broadcasted to the room.

The Client class manages how a client can join servers and communicate between other connected clients and processing commands. The connect method connects to the server of the specified port and address. The client class creates a socket connection with the server and allows input and output streams as well as a connection request to the server.

After a client class is compiled and ran it has to run the /name "NAME" command where the client enters their name in the quotation marks. Then a client can enter the /connect localhost:3000 command to connect to the server as the port number for Server is specified as 3000 in this case.

Communication (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

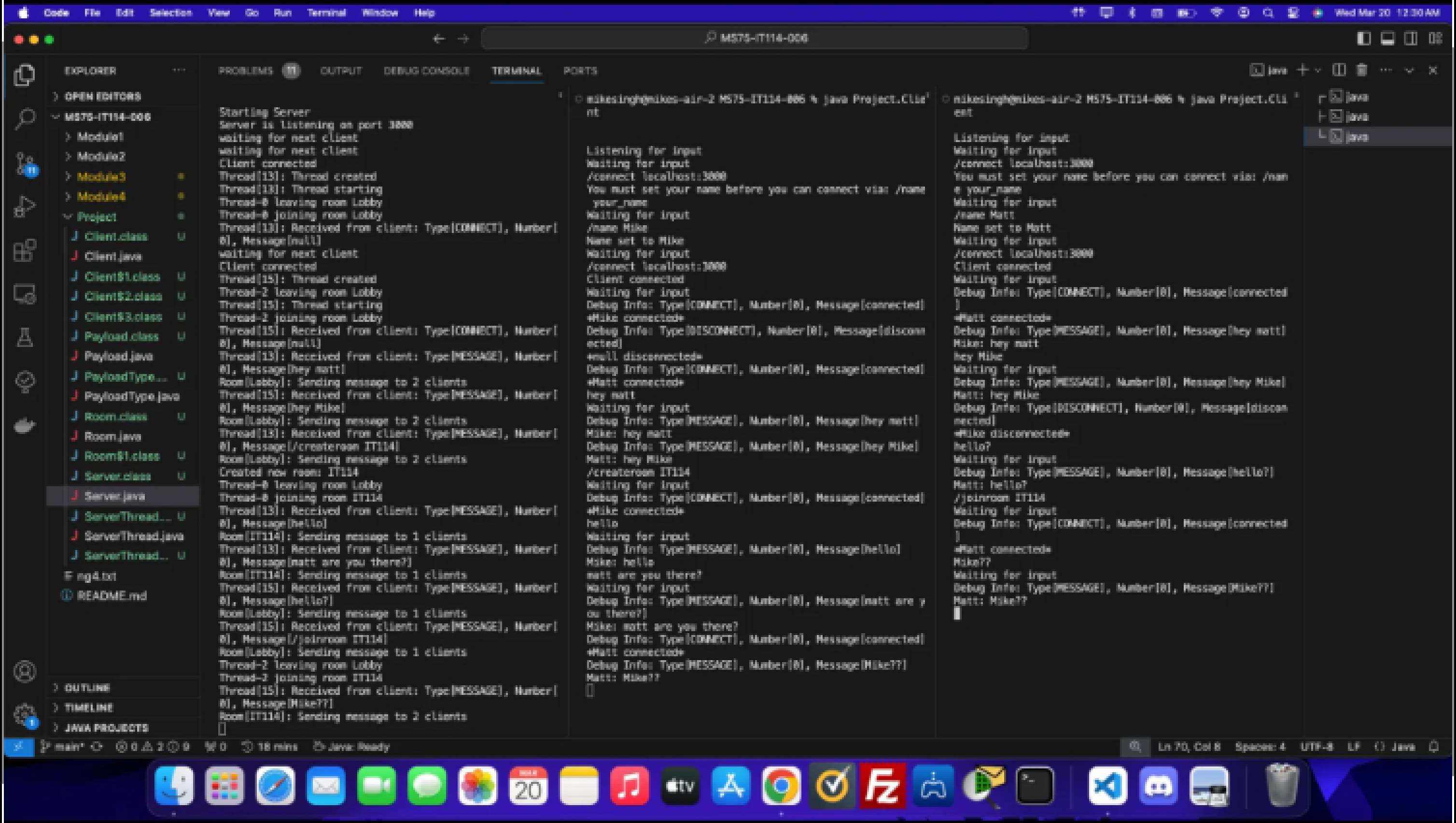
#	Points	Details
#1	1	At least two clients connected to the server
#2	1	Client can send messages to the server
#3	1	Server sends the message to all clients in the same room
#4	1	Messages clearly show who the message is from (i.e., client name is clearly with the message)
		Demonstrate clients in two different rooms can't send/receive messages to each other

#5	2	(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



```
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[13]: Thread created
Thread[13]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[13]: Received from client: Type[CONNECT], Number[
0], Message[null]
waiting for next client
Client connected
Thread[15]: Thread created
Thread-2 leaving room Lobby
Thread[15]: Thread starting
Thread-2 joining room Lobby
Thread[15]: Received from client: Type[CONNECT], Number[
0], Message[null]
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[hey matt]
Room[Lobby]: Sending message to 2 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[hey Mike]
Room[Lobby]: Sending message to 2 clients
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[/createroom IT114]
Room[Lobby]: Sending message to 2 clients
Created new room: IT114
Thread-0 leaving room Lobby
Thread-0 joining room IT114
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[hello]
Room[IT114]: Sending message to 1 clients
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[matt are you there?]
Room[IT114]: Sending message to 1 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[hello?]
Room[Lobby]: Sending message to 1 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[/joinroom IT114]
Room[Lobby]: Sending message to 1 clients
Thread-2 leaving room Lobby
Thread-2 joining room IT114
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[Mike??]
Room[IT114]: Sending message to 2 clients
```

```
mikesingh@mikes-air-2 MS75-IT114-006 % java Project.Clie
nt


Listening for input
Waiting for input
/connect localhost:3000
You must set your name before you can connect via: /name
 your_name
Waiting for input
/name Mike
Name set to Mike
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Mike connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconn
ected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Matt connected*
hey matt
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hey matt]
Mike: hey matt
Debug Info: Type[MESSAGE], Number[0], Message[hey Mike]
Matt: hey Mike
/createroom IT114
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Mike connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
Mike: hello
matt are you there?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[matt are y
ou there?]
Mike: matt are you there?
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Matt connected*
Debug Info: Type[MESSAGE], Number[0], Message[Mike??]
Matt: Mike??
```

```
mikesingh@mikes-air-2 MS75-IT114-006 % java Project.Cli
ent

Listening for input
Waiting for input
/connect localhost:3000
You must set your name before you can connect via: /nam
e your_name
Waiting for input
/name Matt
Name set to Matt
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected
]
*Matt connected*
Debug Info: Type[MESSAGE], Number[0], Message[hey matt]
Mike: hey matt
hey Mike
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hey Mike]
Matt: hey Mike
Debug Info: Type[DISCONNECT], Number[0], Message[discon
nected]
*Mike disconnected*
hello?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello?]
Matt: hello?
/joinroom IT114
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected
]
*Matt connected*
Mike??
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[Mike??]
Matt: Mike??
```

The screenshot from left to right shows the server, Client 1(Mike) and Client 2 (Matt). The server is just connected and keeping track of all client activity. When the first Client was connected the server created a Thread[13]. In client 1 you can see that first I set client's name as Mike and then connected to the server. I then went to client 2 and set the name as Matt and connected to the server. I typed "hey matt" from Mike's client and Matt recieved the message and replied saying "hey Mike". Both clients were in the lobby room and were able to share messages. I then created a room called IT114 from Mike and tried sending messages but Matt did not receive them. I then connected Matt to the same IT114 room and sent a message saying"Mike??" and the clients were able to communicate to with other again.

Checklist Items (0)

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:

Messages are entered from the client side once they set their name and connect to the server. A new server thread, individual connection to the server, is started and the client is placed into the lobby room. The lobby room is the default room set where all new clients join. Once clients are in the lobby anything they type will be broadcasted to the server and back to all clients in the lobby showing the sending clients name and what their message was.

Disconnecting/Termination (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

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

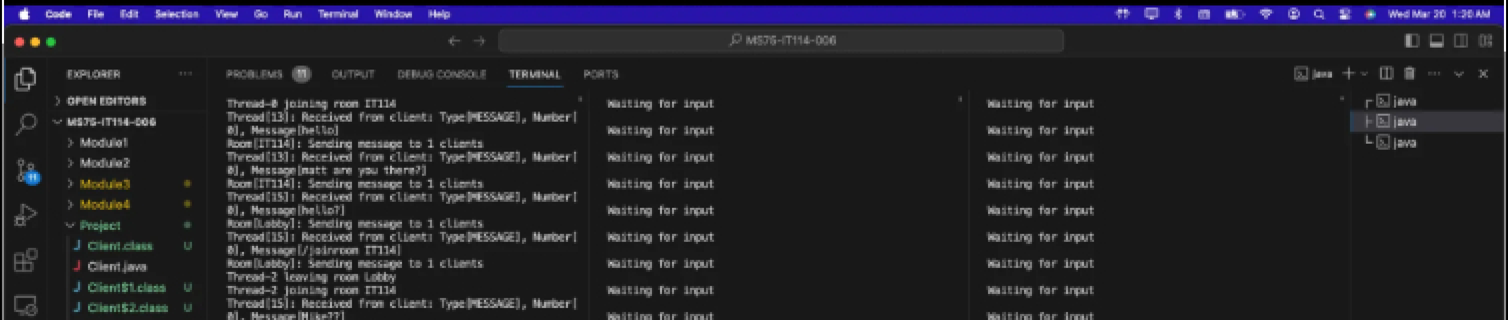
Checklist *The checkboxes are for your own tracking

#	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

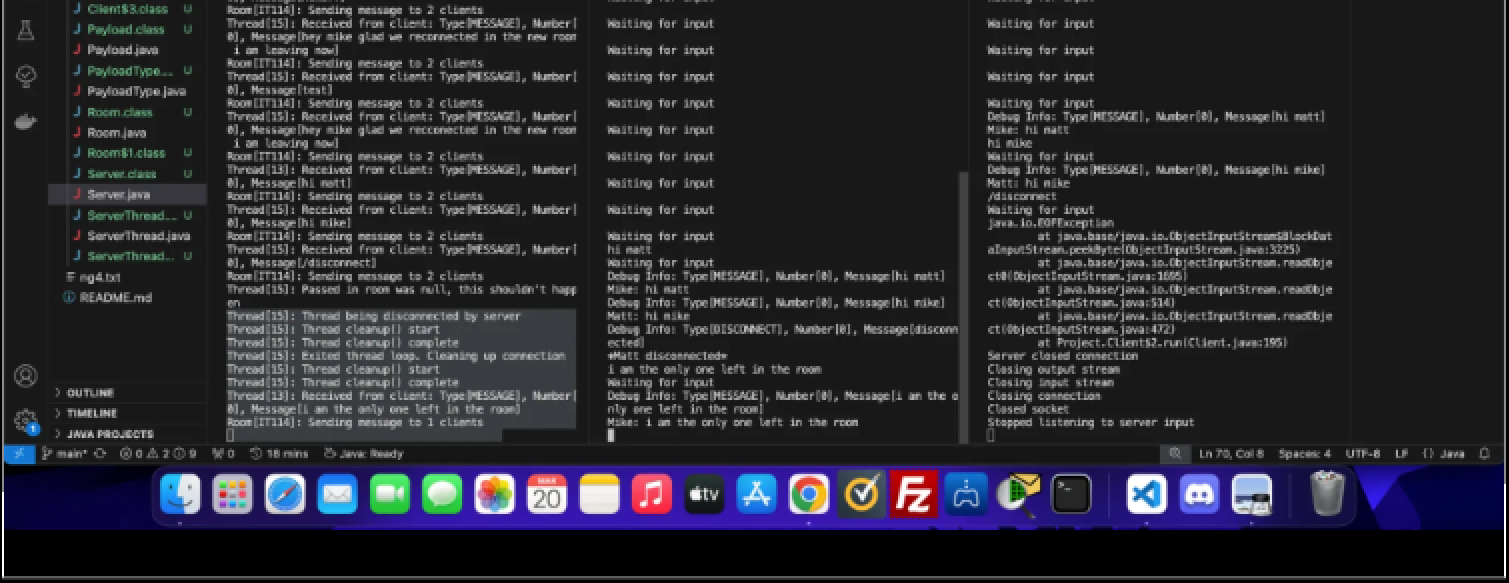
Small Medium Large



```
Thread-0 joining room IT114
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[hello]
Room[IT114]: Sending message to 1 clients
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[matt are you there?]
Room[IT114]: Sending message to 1 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[hello?]
Room[Lobby]: Sending message to 1 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[/joinroom IT114]
Room[Lobby]: Sending message to 1 clients
Thread-2 leaving room Lobby
Thread-2 joining room IT114
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[Mike??]
```

```
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
```

```
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
Waiting for input
```



```
Room[IT114]: Sending message to 2 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[hey mike glad we reconnected in the new room
 i am leaving now]
Room[IT114]: Sending message to 2 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[test]
Room[IT114]: Sending message to 2 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[hey mike glad we recconected in the new room
 i am leaving now]
Room[IT114]: Sending message to 2 clients
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[hi matt]
Room[IT114]: Sending message to 2 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[hi mike]
Room[IT114]: Sending message to 2 clients
Thread[15]: Received from client: Type[MESSAGE], Number[
0], Message[/disconnect]
Room[IT114]: Sending message to 2 clients
Thread[15]: Passed in room was null, this shouldn't happ
en
Thread[15]: Thread being disconnected by server
Thread[15]: Thread cleanup() start
Thread[15]: Thread cleanup() complete
Thread[15]: Exited thread loop. Cleaning up connection
Thread[15]: Thread cleanup() start
Thread[15]: Thread cleanup() complete
Thread[13]: Received from client: Type[MESSAGE], Number[
0], Message[i am the only one left in the room]
Room[IT114]: Sending message to 1 clients
```

```
Waiting for input

Waiting for input

Waiting for input

Waiting for input

Waiting for input

Waiting for input

Waiting for input

Waiting for input

Waiting for input
hi matt
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi matt]
Mike: hi matt
Debug Info: Type[MESSAGE], Number[0], Message[hi mike]
Matt: hi mike
Debug Info: Type[DISCONNECT], Number[0], Message[disconn
ected]
*Matt disconnected*
i am the only one left in the room
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[i am the o
nly one left in the room]
Mike: i am the only one left in the room
```

```
Waiting for input

Waiting for input

Waiting for input

Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi matt]
Mike: hi matt
hi mike
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi mike]
Matt: hi mike
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDat
aInputStream.peekByte(ObjectInputStream.java:3225)
        at java.base/java.io.ObjectInputStream.readObje
ct0(ObjectInputStream.java:1695)
        at java.base/java.io.ObjectInputStream.readObje
ct(ObjectInputStream.java:514)
        at java.base/java.io.ObjectInputStream.readObje
ct(ObjectInputStream.java:472)
        at Project.Client$2.run(Client.java:195)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
```

In this screenshot Client 2 (Matt) used the "/disconnect" command to leave the IT114 room. Client 1(Mike) got the message that Matt Disconnected and in the server terminal it is seen that Thread[15],which was Matt's connection to the IT 114 room, is closed along with the server connection and input output streams. To confirm this I sent a message from Client 1 to the room and Matt did not receive the message and the server shows that the message was only broadcasted to 1 client.

Checklist Items (0)

●

^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
☐ #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:

1. A client can be disconnected from a socket in a few ways whether if the server crashes or disconnects, or the client disconnects or crashes. When the client is disconnected from a socket there is no endpoint for client data to be sent. A client can be trying to send messages across a room but if there is no socket connection then an error will be triggered since there is no address or sending destination for the clients message to reach. A socket works as a route to the end destination.
2. The client program does not crash when a server terminates. The client socket, thread and input output streams are closed but the client application is still active. The client is not able to interact with the server or other clients as well as not be able to send or receive messages but it does not crash.
3.The server does not crash when a client(s) disconnect because a server does not rely on a client to stay up it is

independent. A client however relies on an instance of the server (a server thread) in order to stay up so if the client disconnects then the server thread is deleted and closed.

● Misc (1 pt.)

COLLAPSE

●

COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/mikes1302/MS75-IT114-006/pull/11

●

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:

I had an issue understanding the concept of sockets and how it was different from a server thread. I also accidentally merged my pull request before finishing and pushing all my local changes so when I went to push the changes I was getting an error of my local Milestone1 branch being behind my remote repository. After trying for a while I just fixed my project folder by making and pushing the remaining changes in the Main branch instead of Milestone1.

●

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

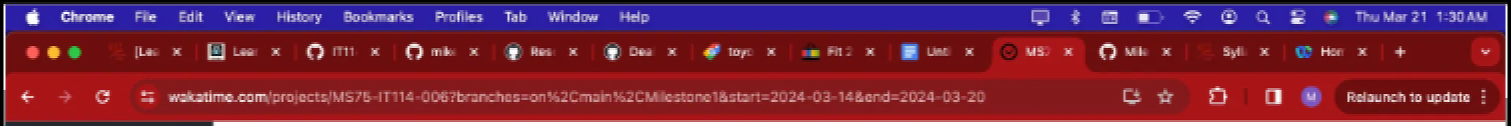
Grab a snippet showing the approximate time involved that clearly shows your repository.

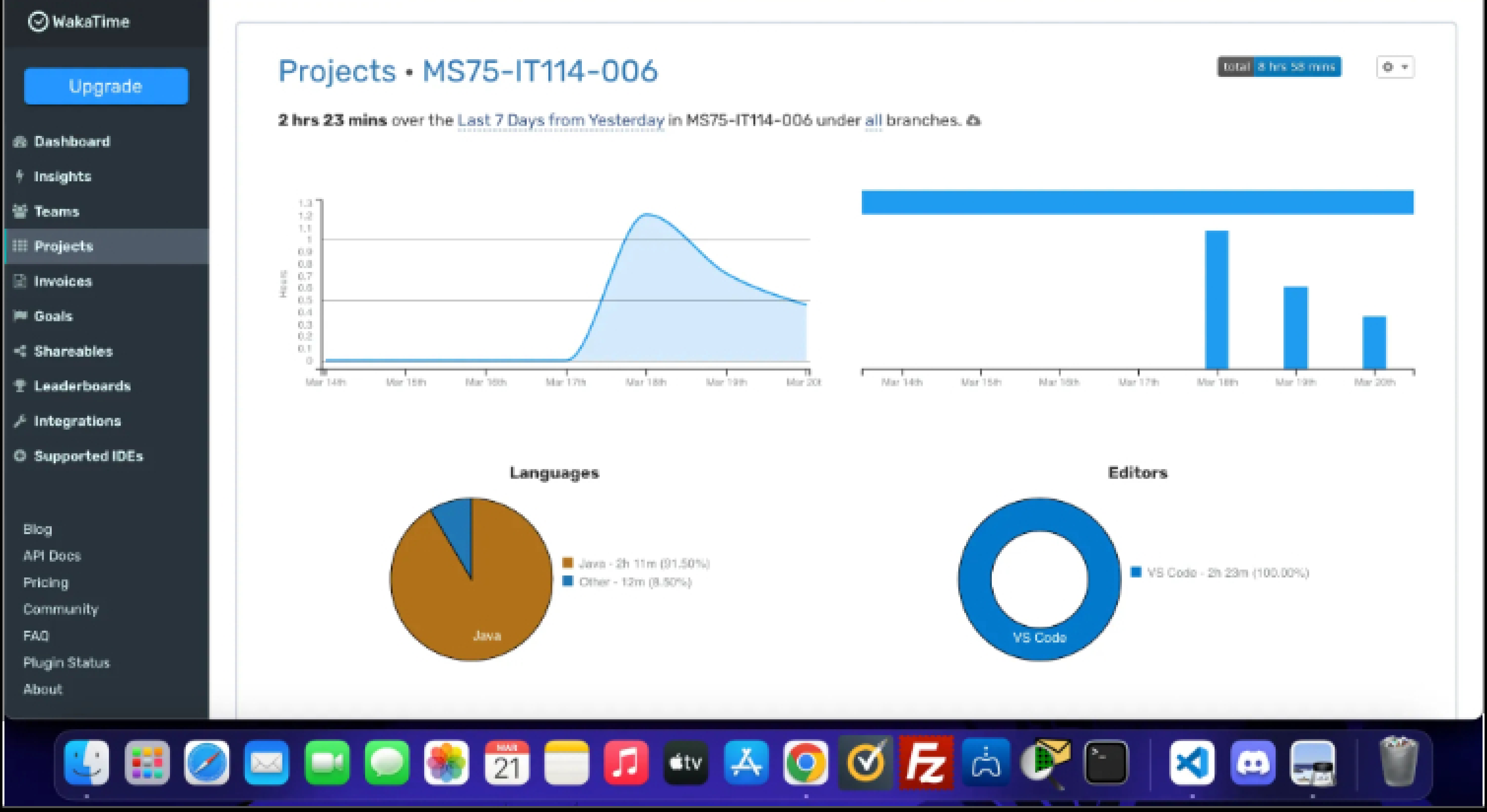
The duration isn't considered for grading, but there should be some time involved.

Task Screenshots:

Gallery Style: Large View

Small Medium Large


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Thu Mar 21 1:30 AM
wakatime.com/projects/MS75-IT114-006?branches=on%2Cmain%2CMilestone1&start=2024-03-14&end=2024-03-20
Relaunch to update



Projects • MS75-IT114-006

2 hrs 23 mins over the Last 7 Days from Yesterday in MS75-IT114-006 under all branches.

Java - 2h 11m (91.50%)
Other - 12m (8.50%)

VS Code - 2h 23m (100.00%)

WakaTime screenshot shows last 7 days (3/14/24 - 3/21/24) of time spent on my local repository.

End of Assignment