Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-project-milestone-1/grade/jeo29

IT114-002-S2024 - [IT114] Project Milestone 1

Submissions:

Submission Selection

1 Submission [active] 3/17/2024 2:14:16 AM

Instructions

COLLAPSE

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. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
4. Copy in the latest Socket sample code from the most recent Socket Part example of the lessons
   1. Recommended Part 5 (clients should be having names at this point and not ids)
   2. https://github.com/MattToegel/IT114/tree/Module5/Module5
5. Fix the package references at the top of each file (these are the only edits you should do at this point)
6. Git add/commit the baseline and push it to github
7. Create a pull request from Milestone1 to main (don't complete/merge it yet, just have it in open status)
8. Ensure the sample is working and fill in the below deliverables
   1. Note: The client commands likely are different in part 5 with the /name and /connect options instead of just "connect"
9. Generate the worksheet output file once done and add it to your local repository
10. Git add/commit/push all changes
11. Complete the pull request merge from step 7
12. Locally checkout main
13. 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

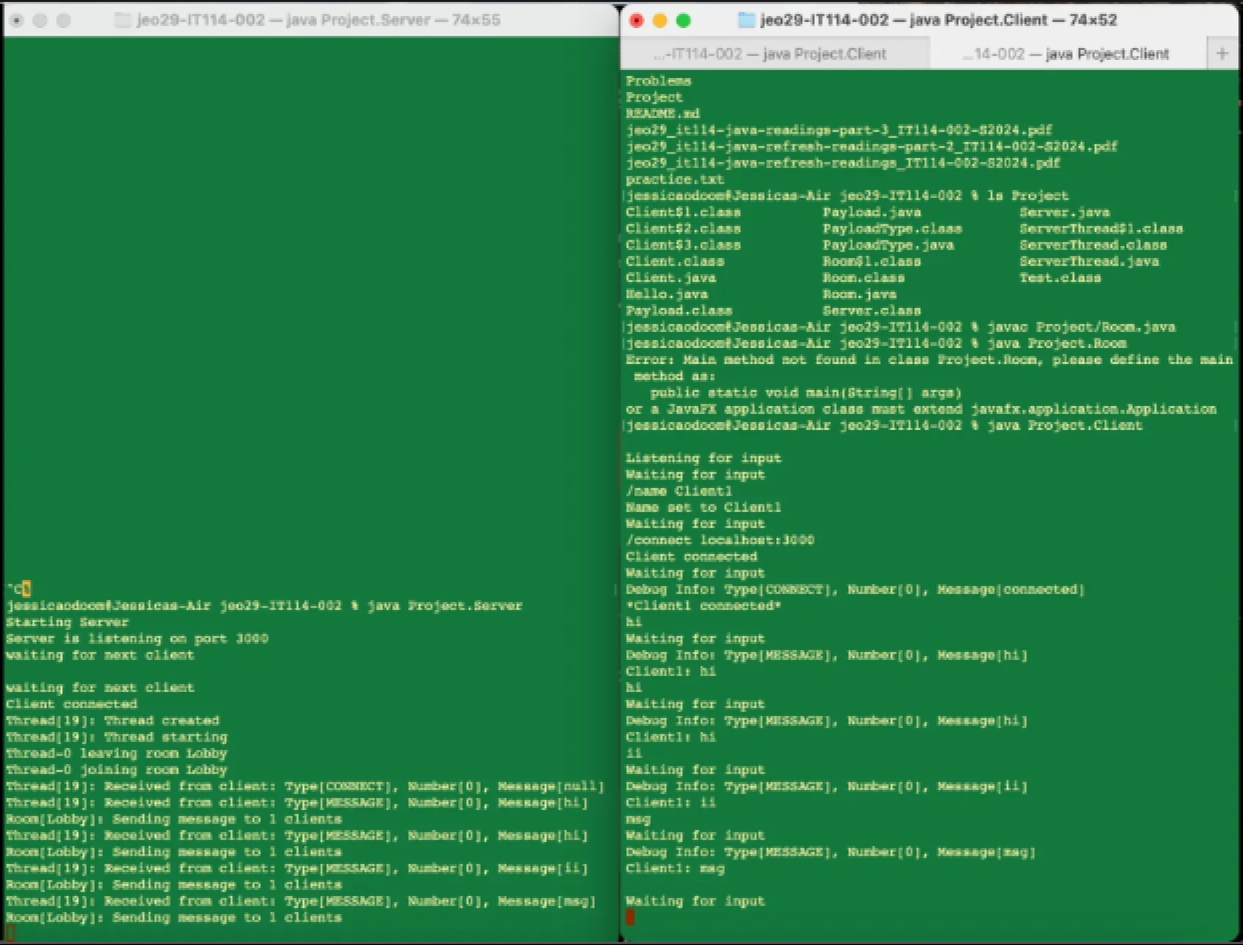
Checklist *The checkboxes are for your own tracking

#	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


jeo29-IT114-002 — java Project.Server — 74x55
jeo29-IT114-002 — java Project.Client — 74x52
...-IT114-002 — java Project.Client
...14-002 — java Project.Client

```
^C
jessicaodoom@Jessicas-Air jeo29-IT114-002 % java Project.Server
Starting Server
Server is listening on port 3000
waiting for next client

waiting for next client
Client connected
Thread[19]: Thread created
Thread[19]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[19]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[ii]
Room[Lobby]: Sending message to 1 clients
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[msg]
Room[Lobby]: Sending message to 1 clients
```

```
Problems
Project
README.md
jeo29_it114-java-readings-part-3_IT114-002-S2024.pdf
jeo29_it114-java-refresh-readings-part-2_IT114-002-S2024.pdf
jeo29_it114-java-refresh-readings_IT114-002-S2024.pdf
practice.txt
jessicaodoom@Jessicas-Air jeo29-IT114-002 % ls Project
Client$1.class          Payload.java            Server.java
Client$2.class          PayloadType.class       ServerThread$1.class
Client$3.class          PayloadType.java        ServerThread.class
Client.class            Room$1.class            ServerThread.java
Client.java             Room.class              Test.class
Hello.java              Room.java
Payload.class           Server.class
jessicaodoom@Jessicas-Air jeo29-IT114-002 % javac Project/Room.java
jessicaodoom@Jessicas-Air jeo29-IT114-002 % java Project.Room
Error: Main method not found in class Project.Room, please define the main
 method as:
   public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
jessicaodoom@Jessicas-Air jeo29-IT114-002 % java Project.Client

Listening for input
Waiting for input
/name Client1
Name set to Client1
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Client1 connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi]
Client1: hi
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi]
Client1: hi
ii
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[ii]
Client1: ii
msg
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[msg]
Client1: msg

Waiting for input
```

Server and Client Initialization

Checklist Items (1)

#1 Server should properly be listening to its port from the command line (note the related message)

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 side of the connection works with the creation of the socket binding to an IP address and port number. This determines where the server will listen for incoming connections and when the server receives a connection, it accepts.

For the client side, the client initiates a connection to the server with the IP address and port number. The server needs to accept and once that is accepted the client can communicate with the server.

The socket steps require a socket to be set up as a listening socket. This socket then hasd to accept the connection The listening socket allows for the server to handle multiple connections for each new client.

COLLAPSE

Communication (3 pts.)

COLLAPSE

Task #1 - Points: 1

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

Checklist *The checkboxes are for your own tracking

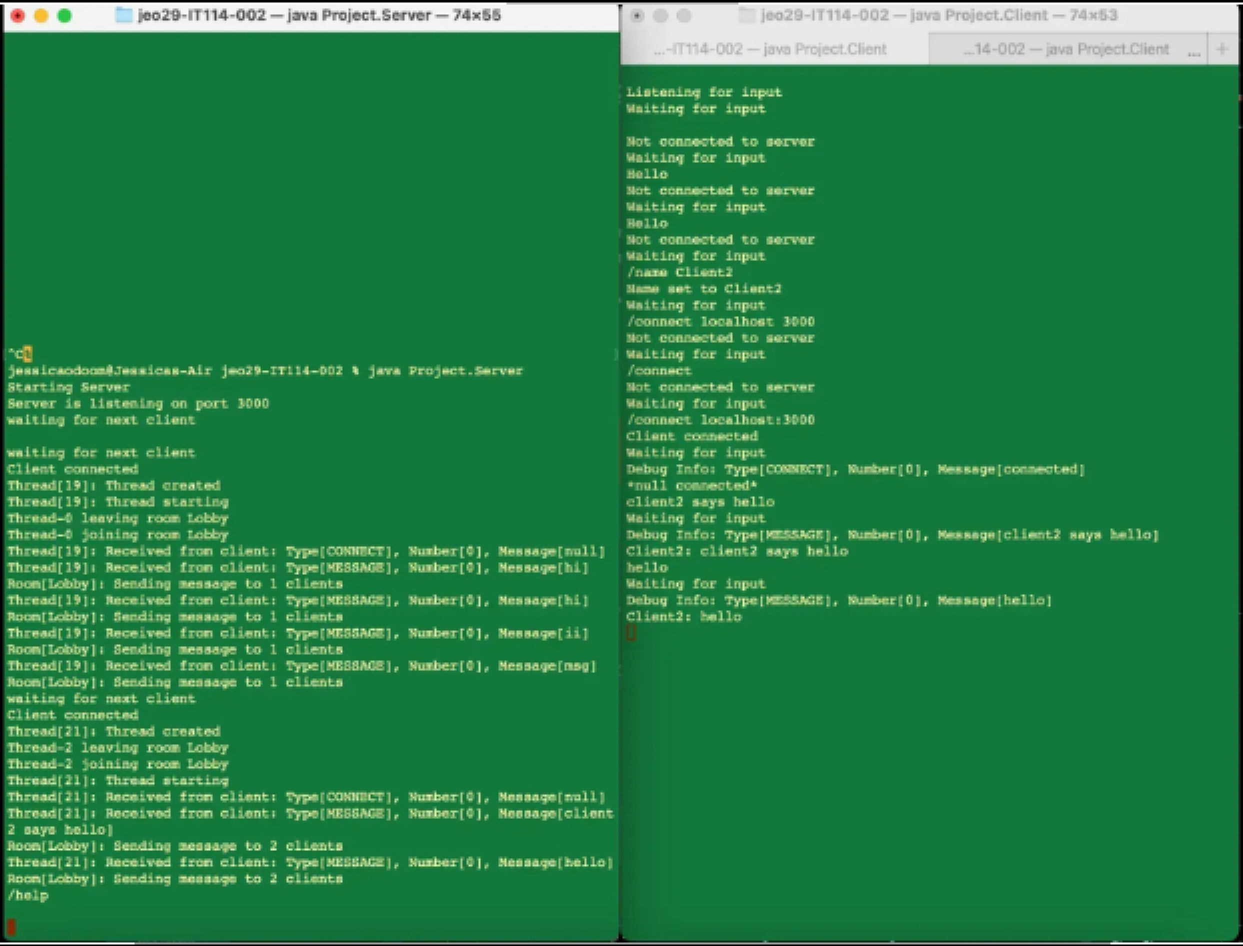
#	Points	Details
#1	1	At least two clients connected to the server
#2	1	Client can send messages to the server
#3	1	Server sends the message to all clients in the same room

#4	1	Messages clearly show who the message is from (i.e., client name is clearly with the message)
#5	2	Demonstrate clients in two different rooms can't send/receive messages to each other (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



jeo29-IT114-002 — java Project.Server — 74×55

```
^C
jessicaodoom@Jessicas-Air jeo29-IT114-002 % java Project.Server
Starting Server
Server is listening on port 3000
waiting for next client

waiting for next client
Client connected
Thread[19]: Thread created
Thread[19]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[19]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[ii]
Room[Lobby]: Sending message to 1 clients
Thread[19]: Received from client: Type[MESSAGE], Number[0], Message[msg]
Room[Lobby]: Sending message to 1 clients
waiting for next client
Client connected
Thread[21]: Thread created
Thread-2 leaving room Lobby
Thread-2 joining room Lobby
Thread[21]: Thread starting
Thread[21]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[client
2 says hello]
Room[Lobby]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
/help
```

jeo29-IT114-002 — java Project.Client — 74×53

...-IT114-002 — java Project.Client | ...14-002 — java Project.Client

```
Listening for input
Waiting for input

Not connected to server
Waiting for input
Hello
Not connected to server
Waiting for input
Hello
Not connected to server
Waiting for input
/name Client2
Name set to Client2
Waiting for input
/connect localhost 3000
Not connected to server
Waiting for input
/connect
Not connected to server
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*null connected*
client2 says hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[client2 says hello]
Client2: client2 says hello
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
Client2: hello
```

Evidence related to checklist

Checklist Items (1)

#2 Client can send messages to the server

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.

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:

Client Side: The client composes a message and sends it to the server.

ServerThread: Once the ServerThread receives the message the client processes the incoming message.
Rooms Perspective: The ServerThread forwards the message to the appropriate Room. The Room then sends the message to its appropriate clients.
ClientSideReceiving: The message is received. The client then processes the incoming message so that all clients in the same room receive the message, allowing for real-time communication.

Disconnecting/Termination (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

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

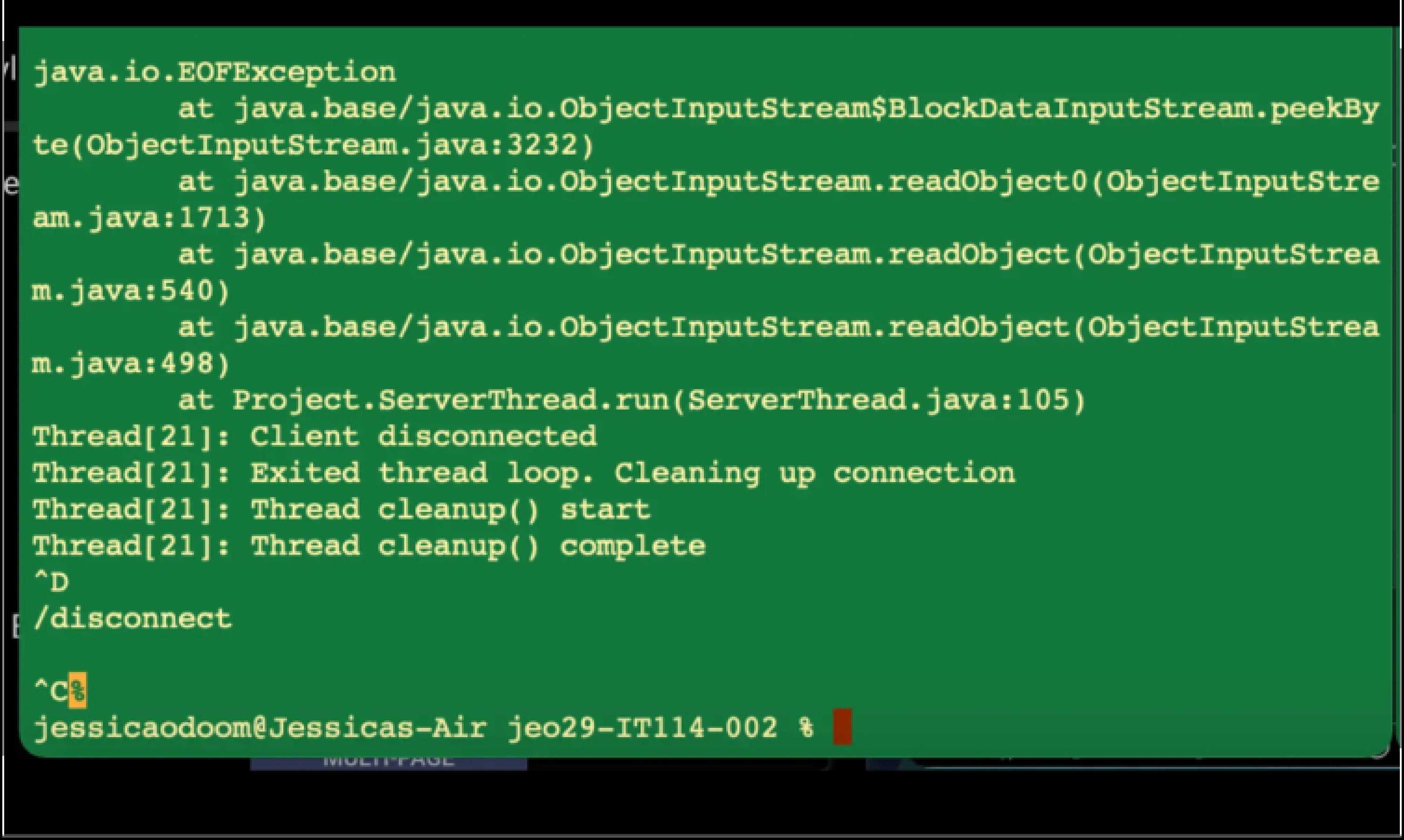
Checklist *The checkboxes are for your own tracking

#	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



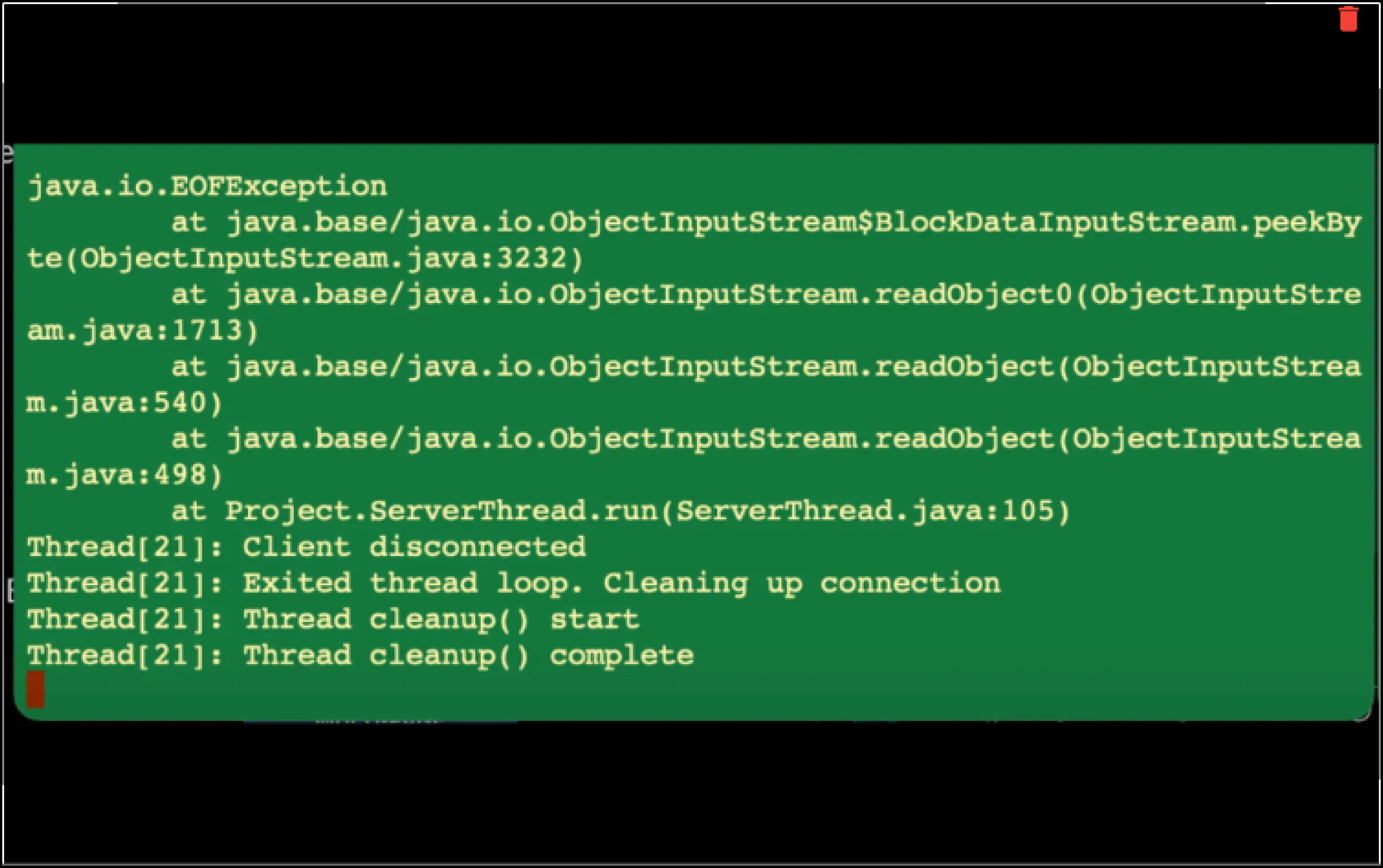
```
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekBy
te(ObjectInputStream.java:3232)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStre
am.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStrea
m.java:540)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStrea
m.java:498)
        at Project.ServerThread.run(ServerThread.java:105)
Thread[21]: Client disconnected
Thread[21]: Exited thread loop. Cleaning up connection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
^D
/disconnect

^C%
jessicaodoom@Jessicas-Air jeo29-IT114-002 %
```

Disconnections2

Checklist Items (1)

#3 For each scenario, disconnected messages should be shown to the clients (should show a different person disconnected and should show the specific client disconnected)



```
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekBy
te(ObjectInputStream.java:3232)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStre
am.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStrea
m.java:540)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStrea
m.java:498)
        at Project.ServerThread.run(ServerThread.java:105)
Thread[21]: Client disconnected
Thread[21]: Exited thread loop. Cleaning up connection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
```

Disconnections

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:

Disconnecting a client from a socket includes clients dropping due to network issues or intentional exits, which servers detect by receiving zero by causing them to close the socket and update their client management structures.

Client applications, to avoid crashing when servers disconnect or terminate, implement error handling around networking calls, allowing them to catch disconnections and offer users reconnection options.

Servers are designed to handle client disconnects, and socket checks to close clients and remove their data, making sure the server is stable and uninterrupted service to remaining clients.

● COLLAPSE Misc (1 pt.)

● COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1
https://github.com/jessicaodoom/jeo29-IT114-002/pull/15

● 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 a lot of issues for starters changed the package name but when I tried to compile it, it failed. Not using javac correctly. The package is not being fixed properly. As a result, it took me about 3 days to do this assignment because I was going back and forth with the professor to get my questions answered.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

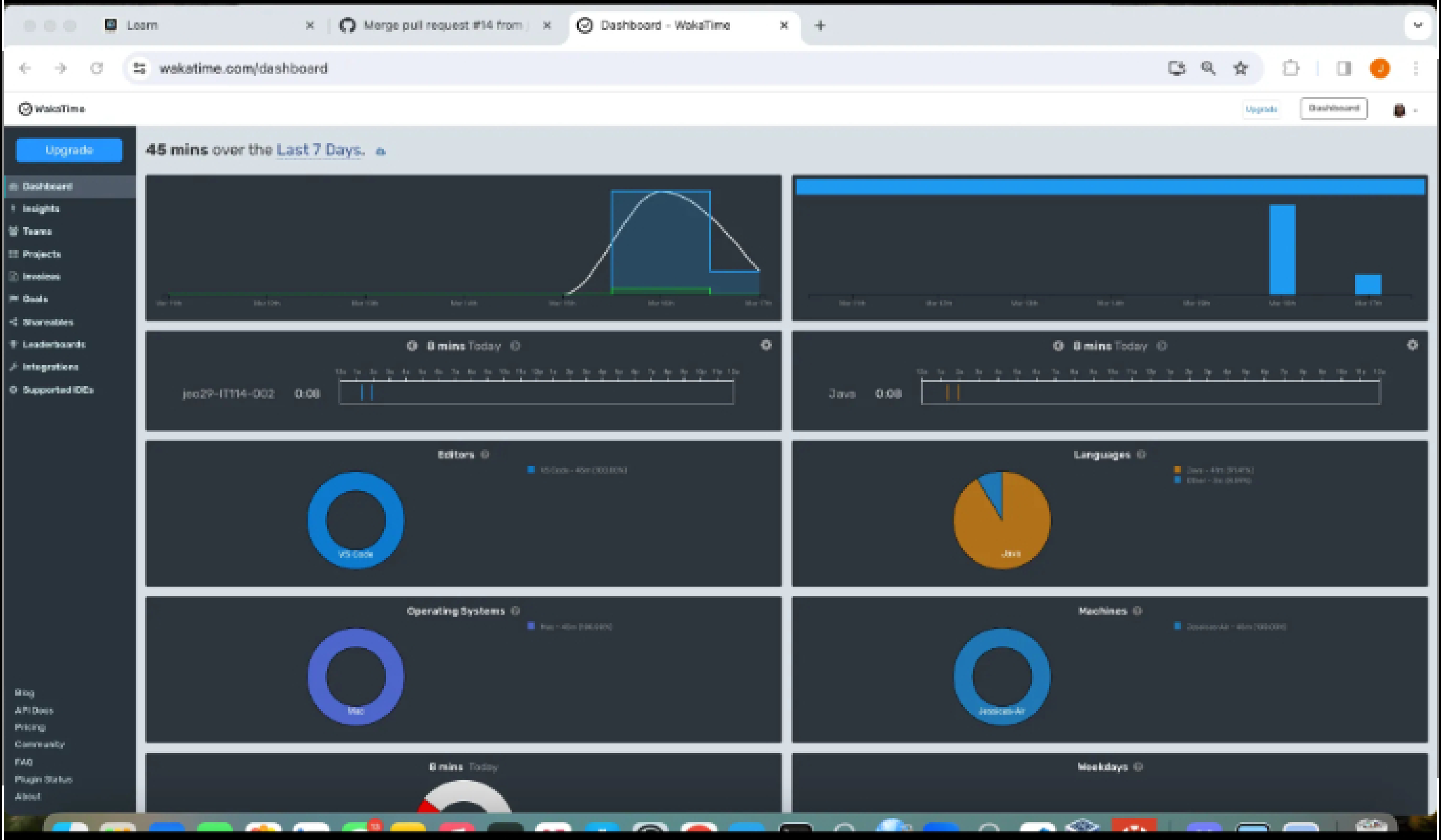
Grab a snippet showing the approximate time involved that clearly shows your repository.

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



45 mins over the Last 7 Days.

0 mins Today

jeo29-IT114-002 0:08

0 mins Today

Java 0:08

0 mins Today

Wakatime

End of Assignment