Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-project-milestone-1/grade/ob75

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

Submissions:

Submission Selection

1 Submission [active] 3/17/2024 7:37:04 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

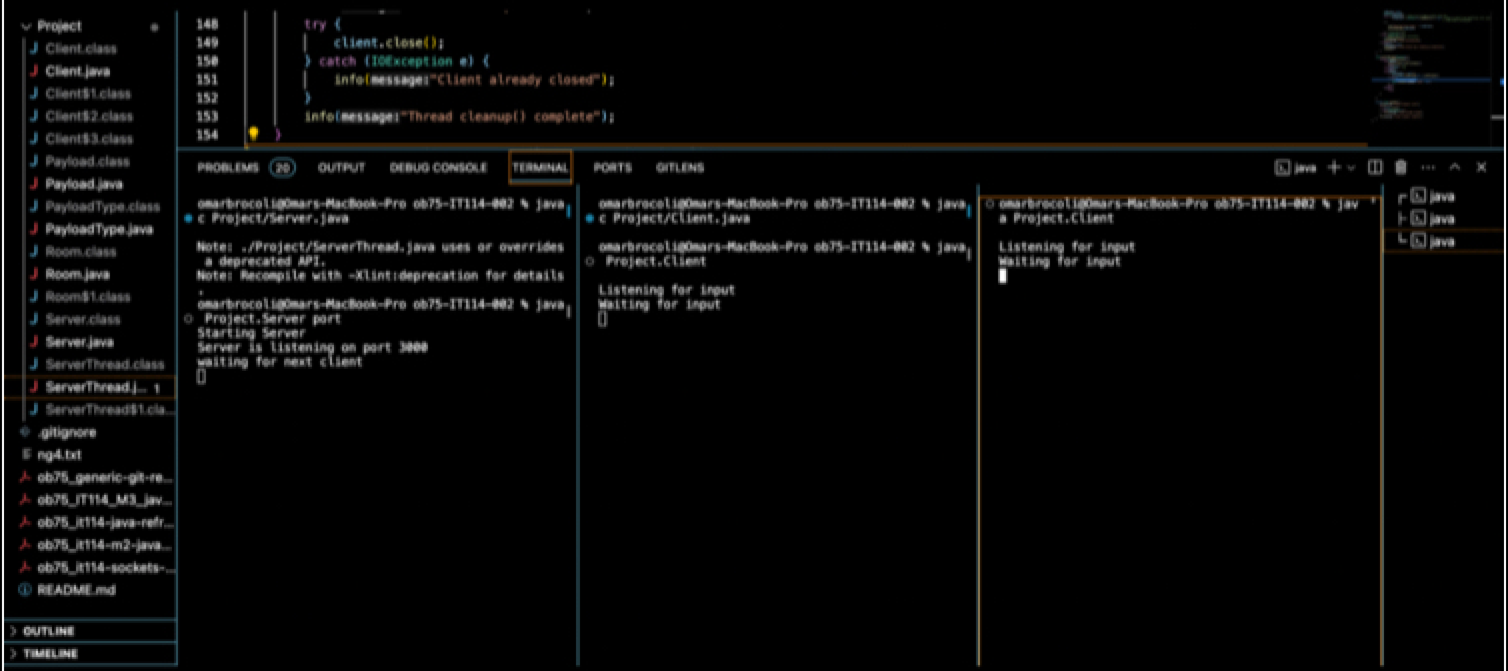
Checklist *The checkboxes are for your own tracking

#	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
            try {
                client.close();
            } catch (IOException e) {
                info(message:"Client already closed");
            }
            info(message:"Thread cleanup() complete");
        }
```

```
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % javac Project/Server.java
Note: ./Project/ServerThread.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java Project.Server port
Starting Server
Server is listening on port 3000
waiting for next client
```

```
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % javac Project/Client.java
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java Project.Client

Listening for input
Waiting for input
```

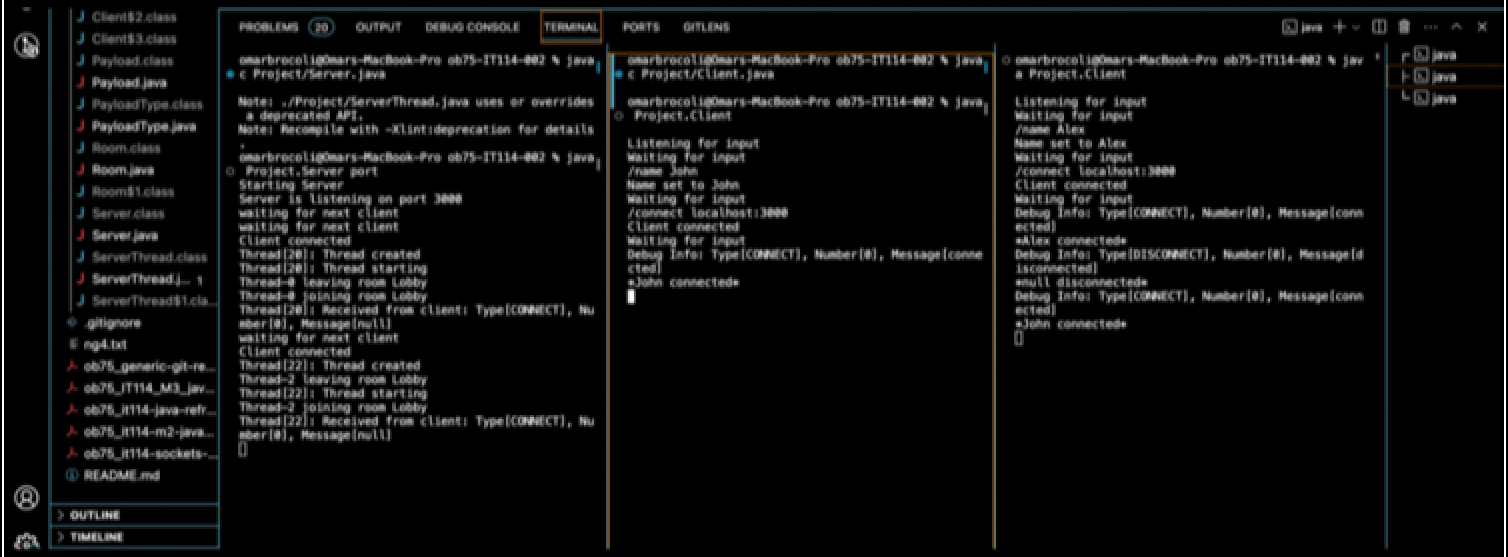
```
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java Project.Client

Listening for input
Waiting for input
```

The terminal shows here the server compiled and ran successfully and is listening to port 3000. The Clients side is also compiled and ran and shows them listening for input and waiting for input.

Checklist Items (2)

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

#2 Clients should be successfully waiting for input



```
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java
c Project/Server.java

Note: ./Project/ServerThread.java uses or overrides
 a deprecated API.
Note: Recompile with -Xlint:deprecation for details
.
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java
 Project.Server port
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[20]: Thread created
Thread[20]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[20]: Received from client: Type[CONNECT], Nu
mber[0], Message[null]
waiting for next client
Client connected
Thread[22]: Thread created
Thread-2 leaving room Lobby
Thread[22]: Thread starting
Thread-2 joining room Lobby
Thread[22]: Received from client: Type[CONNECT], Nu
mber[0], Message[null]
```

```
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java
c Project/Client.java

omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java
 Project.Client

Listening for input
Waiting for input
/name John
Name set to John
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[conne
cted]
*John connected*
```

```
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % jav
a Project.Client

Listening for input
Waiting for input
/name Alex
Name set to Alex
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[conn
ected]
*Alex connected*
Debug Info: Type[DISCONNECT], Number[0], Message[d
isconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[conn
ected]
*John connected*
```

Terminal shows Clients successfully connect with Names created as well before hand. The *Alex connected* or *John connected* shows a successful connect from client to server as the server also displays information about the client connecting and creating the lobby.

Checklist Items (1)

#3 Clients should have a name and successfully connected to the server (note related messages)

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 listens for a port that matches the it and connect them if true or connects the client if they choose to connect with an IP address with the same port as before. Every time client joins the server, they are placed into a room named "lobby" where all clients can connect to first. From the client side, commands to connect and disconnect are done here and the commands are differentiated from the messages by using a "/" for the commands and that's what done here to connect to the server (/connect localhost:3000 is done here to connect to the port on the server and join the lobby). In the sendConnect() method of the client side, this creates a connect payload and allows, if the port matches server port, to connect to server. In steps for the socket before the server waits for messages from the client, the first is that the client side will process the connect command to the server so that the server reads and listen to the port the client types.The client must have also created a name for themselves using /name and then their name before they connect to the server. The server-thread will look for payload which is CONNECT and will connect the client if the port successfully connects from client to server (If there is no room in server-thread, then it will try to join the room "lobby"). The server reads off the output of the client side to check if they are connecting to the correct port/server. Once client connects to server, it will show to the whole server the clients name and saying they connected and the server will create a lobby for the clients to join first. Messages are then able to be sent and completed once clients are connected.

●

COLLAPSE

Communication (3 pts.)

●

COLLAPSE

Task #1 - Points: 1

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

Checklist

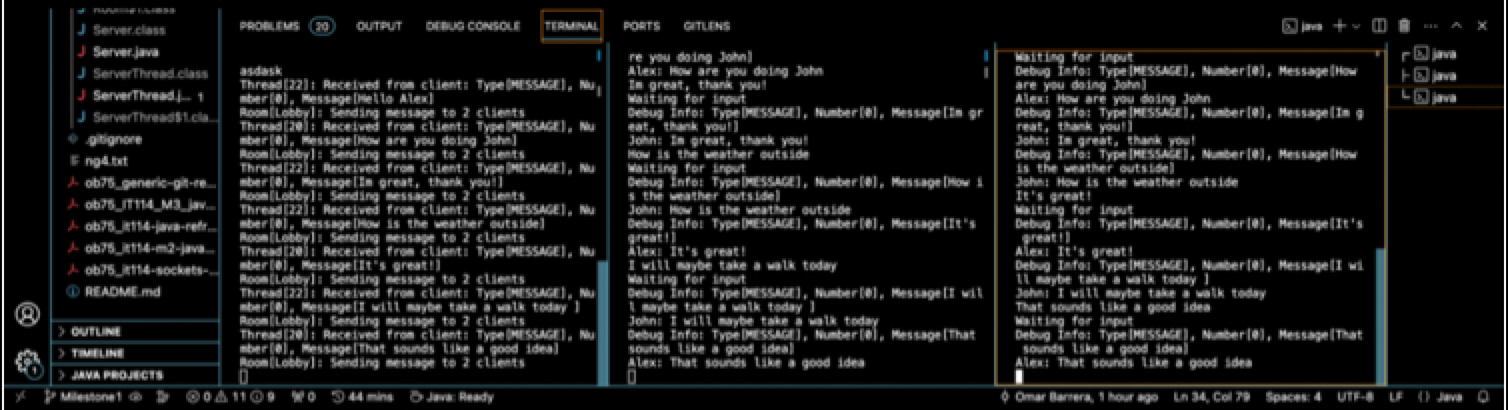
*The checkboxes are for your own tracking

#	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



```
asdask
Thread[22]: Received from client: Type[MESSAGE], Nu
mber[0], Message[Hello Alex]
Room[Lobby]: Sending message to 2 clients
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[How are you doing John]
Room[Lobby]: Sending message to 2 clients
Thread[22]: Received from client: Type[MESSAGE], Nu
mber[0], Message[Im great, thank you!]
Room[Lobby]: Sending message to 2 clients
Thread[22]: Received from client: Type[MESSAGE], Nu
mber[0], Message[How is the weather outside]
Room[Lobby]: Sending message to 2 clients
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[It's great!]
Room[Lobby]: Sending message to 2 clients
Thread[22]: Received from client: Type[MESSAGE], Nu
mber[0], Message[I will maybe take a walk today ]
Room[Lobby]: Sending message to 2 clients
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[That sounds like a good idea]
Room[Lobby]: Sending message to 2 clients
```

```
re you doing John]
Alex: How are you doing John
Im great, thank you!
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[Im gr
eat, thank you!]
John: Im great, thank you!
How is the weather outside
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[How i
s the weather outside]
John: How is the weather outside
Debug Info: Type[MESSAGE], Number[0], Message[It's
great!]
Alex: It's great!
I will maybe take a walk today
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[I wil
l maybe take a walk today ]
John: I will maybe take a walk today
Debug Info: Type[MESSAGE], Number[0], Message[That
sounds like a good idea]
Alex: That sounds like a good idea
```

```
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[How
are you doing John]
Alex: How are you doing John
Debug Info: Type[MESSAGE], Number[0], Message[Im g
reat, thank you!]
John: Im great, thank you!
Debug Info: Type[MESSAGE], Number[0], Message[How
is the weather outside]
John: How is the weather outside
It's great!
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[It's
 great!]
Alex: It's great!
Debug Info: Type[MESSAGE], Number[0], Message[I wi
ll maybe take a walk today ]
John: I will maybe take a walk today
That sounds like a good idea
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[That
 sounds like a good idea]
Alex: That sounds like a good idea
```

Two clients which are the two terminal windows on the right, are connected to the server in the terminal and are able to send message to the server. The server also shows in the terminal that it is sending the message to the 2 clients in the same room. The client side in the terminal shows who is name of the person sending the message and also shows the message they wrote.

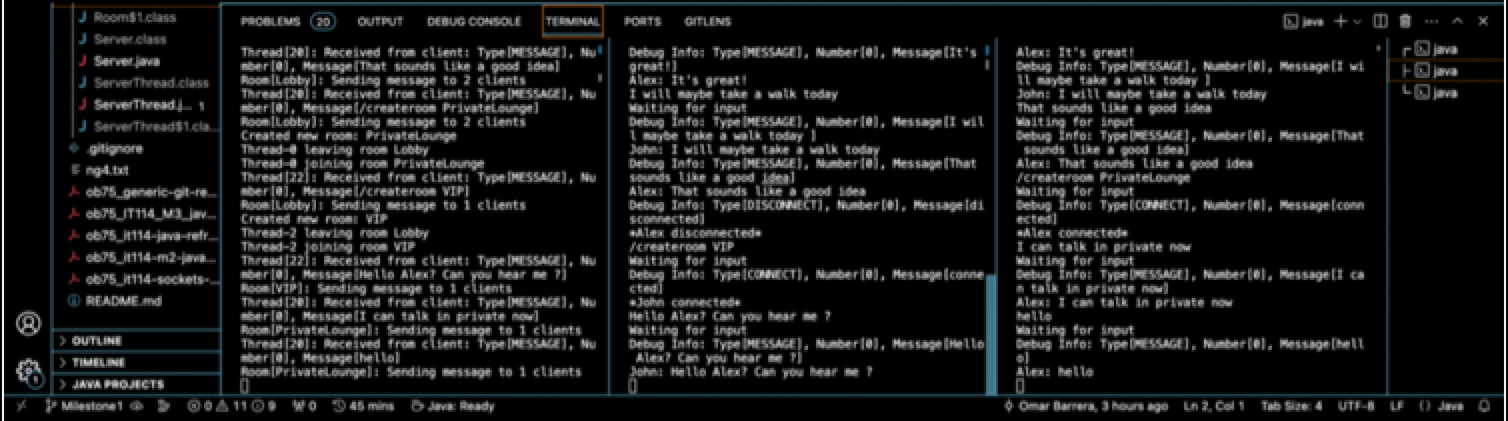
Checklist Items (4)

#1 At least two clients connected to the server

#2 Client can send messages to the server

#3 Server sends the message to all clients in the same room

#4 Messages clearly show who the message is from (i.e., client name is clearly with the message)



```
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[That sounds like a good idea]
Room[Lobby]: Sending message to 2 clients
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[/createroom PrivateLounge]
Room[Lobby]: Sending message to 2 clients
Created new room: PrivateLounge
Thread-0 leaving room Lobby
Thread-0 joining room PrivateLounge
Thread[22]: Received from client: Type[MESSAGE], Nu
mber[0], Message[/createroom VIP]
Room[Lobby]: Sending message to 1 clients
Created new room: VIP
Thread-2 leaving room Lobby
Thread-2 joining room VIP
Thread[22]: Received from client: Type[MESSAGE], Nu
mber[0], Message[Hello Alex? Can you hear me ?]
Room[VIP]: Sending message to 1 clients
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[I can talk in private now]
Room[PrivateLounge]: Sending message to 1 clients
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[hello]
Room[PrivateLounge]: Sending message to 1 clients
```

```
Debug Info: Type[MESSAGE], Number[0], Message[It's
great!]
Alex: It's great!
I will maybe take a walk today
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[I wil
l maybe take a walk today ]
John: I will maybe take a walk today
Debug Info: Type[MESSAGE], Number[0], Message[That
sounds like a good idea]
Alex: That sounds like a good idea
Debug Info: Type[DISCONNECT], Number[0], Message[di
sconnected]
*Alex disconnected*
/createroom VIP
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[conne
cted]
*John connected*
Hello Alex? Can you hear me ?
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[Hello
 Alex? Can you hear me ?]
John: Hello Alex? Can you hear me ?
```

```
Alex: It's great!
Debug Info: Type[MESSAGE], Number[0], Message[I wi
ll maybe take a walk today ]
John: I will maybe take a walk today
That sounds like a good idea
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[That
 sounds like a good idea]
Alex: That sounds like a good idea
/createroom PrivateLounge
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[conn
ected]
*Alex connected*
I can talk in private now
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[I ca
n talk in private now]
Alex: I can talk in private now
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hell
o]
Alex: hello
```

Clients both created and joined separate rooms. The last terminal window, with Alex as client, created room named "PrivateLounge" and middle terminal window with, John as client, created room named "VIP". Both clients send a message but neither can see each other's message and can only see their own since they are only ones in their respective rooms. Server also shows created rooms and the message being sent to one client which is themselves and where the message is sent(room).

Checklist Items (1)

#5 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

●
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 first entered to the client-side and send data to the ServerThread via payloads and does this in the sendMessage() method where the payload is created and sets the type which is payload.MESSAGE and writes it to the output stream. The ServerThread here is then getting the payload.MESSAGE data in through the sendMessage() method and sends the data out from the last code in the method, return send(p). There is also send() method which is a boolean to make sure message was sent successfully. The Room's perspective shows a sendMessage() method which takes a sender and message and would broadcast that message to every client in the room while the room is running. Lastly in the client-side for receiving, the processCommands() method allows for the case of a MESSAGE to print out to the room the client's name and the message typed.

●
COLLAPSE

Disconnecting/Termination (3 pts.)

COLLAPSE

Task #1 - Points: 1

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

Checklist

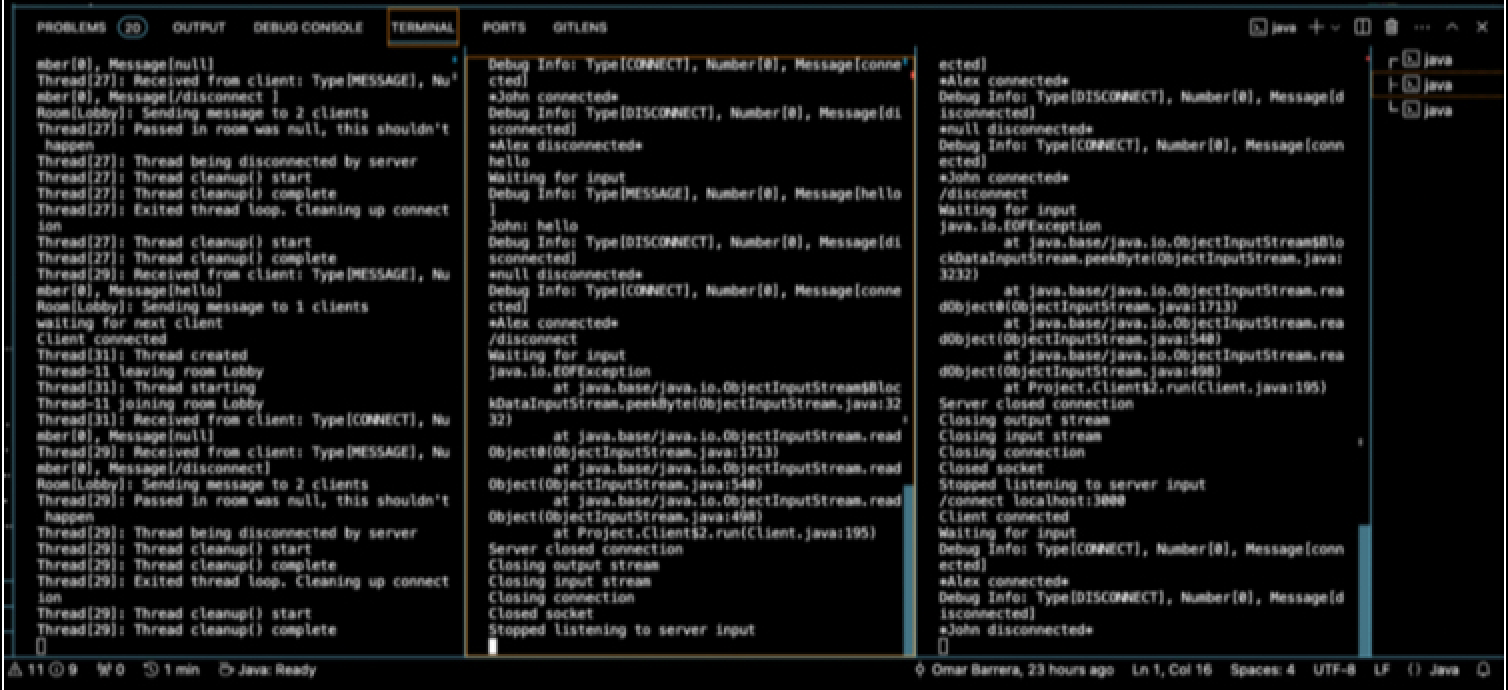
*The checkboxes are for your own tracking

#	Points	Details
#1	1	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



```
mber[0], Message[null]
Thread[27]: Received from client: Type[MESSAGE], Nu
mber[0], Message[/disconnect ]
Room[Lobby]: Sending message to 2 clients
Thread[27]: Passed in room was null, this shouldn't
 happen
Thread[27]: Thread being disconnected by server
Thread[27]: Thread cleanup() start
Thread[27]: Thread cleanup() complete
Thread[27]: Exited thread loop. Cleaning up connect
ion
Thread[27]: Thread cleanup() start
Thread[27]: Thread cleanup() complete
Thread[29]: Received from client: Type[MESSAGE], Nu
mber[0], Message[hello]
Room[Lobby]: Sending message to 1 clients
waiting for next client
Client connected
Thread[31]: Thread created
Thread-11 leaving room Lobby
Thread[31]: Thread starting
Thread-11 joining room Lobby
Thread[31]: Received from client: Type[CONNECT], Nu
mber[0], Message[null]
Thread[29]: Received from client: Type[MESSAGE], Nu
mber[0], Message[/disconnect]
Room[Lobby]: Sending message to 2 clients
Thread[29]: Passed in room was null, this shouldn't
 happen
Thread[29]: Thread being disconnected by server
Thread[29]: Thread cleanup() start
Thread[29]: Thread cleanup() complete
Thread[29]: Exited thread loop. Cleaning up connect
ion
Thread[29]: Thread cleanup() start
Thread[29]: Thread cleanup() complete
```

```
Debug Info: Type[CONNECT], Number[0], Message[conne
cted]
*John connected*
Debug Info: Type[DISCONNECT], Number[0], Message[di
sconnected]
*Alex disconnected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello
]
John: hello
Debug Info: Type[DISCONNECT], Number[0], Message[di
sconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[conne
cted]
*Alex connected*
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$Bloc
kDataInputStream.peekByte(ObjectInputStream.java:32
32)
        at java.base/java.io.ObjectInputStream.read
Object0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.read
Object(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.read
Object(ObjectInputStream.java:498)
        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
```

```
ected]
*Alex connected*
Debug Info: Type[DISCONNECT], Number[0], Message[d
isconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[conn
ected]
*John connected*
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$Blo
ckDataInputStream.peekByte(ObjectInputStream.java:
3232)
        at java.base/java.io.ObjectInputStream.rea
dObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.rea
dObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.rea
dObject(ObjectInputStream.java:498)
        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
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[conn
ected]
*Alex connected*
Debug Info: Type[DISCONNECT], Number[0], Message[d
isconnected]
*John disconnected*
```

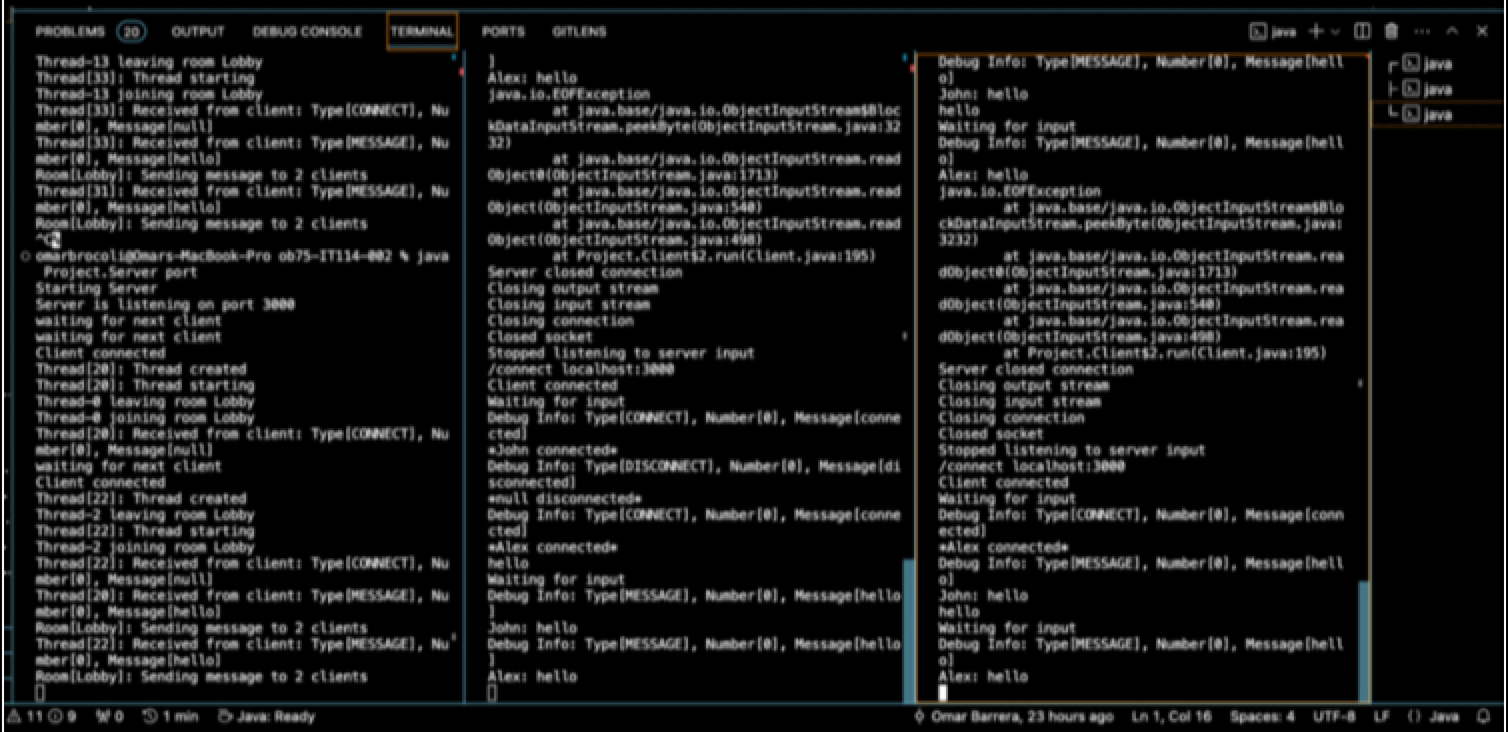
```
⚠ 11 ⓘ 9   ⚐ 0   1 min   Java: Ready        Omar Barrera, 23 hours ago   Ln 1, Col 16   Spaces: 4   UTF-8   LF   {} Java
```

Terminal shows clients disconnecting but the server is still running and sends out the message to all clients that the client disconnecting and shows specifically who. In this case, the 3rd window with client *Alex disconnected* and the other middle window is still able to chat and server still works. Same is done to middle window and shows to 3rd window that *John disconnected*

Checklist Items (2)

#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)

#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)


PROBLEMS 20
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
GITLENS
java
java
java
java
⚠ 11 ⓘ 9 W 0 1 min Java: Ready
Omar Barrera, 23 hours ago Ln 1, Col 16 Spaces: 4 UTF-8 LF {} Java

```
Thread-13 leaving room Lobby
Thread[33]: Thread starting
Thread-13 joining room Lobby
Thread[33]: Received from client: Type[CONNECT], Nu
mber[0], Message[null]
Thread[33]: Received from client: Type[MESSAGE], Nu
mber[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
Thread[31]: Received from client: Type[MESSAGE], Nu
mber[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
^C
omarbrocoli@Omars-MacBook-Pro ob75-IT114-002 % java
 Project.Server port
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[20]: Thread created
Thread[20]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[20]: Received from client: Type[CONNECT], Nu
mber[0], Message[null]
waiting for next client
Client connected
Thread[22]: Thread created
Thread-2 leaving room Lobby
Thread[22]: Thread starting
Thread-2 joining room Lobby
Thread[22]: Received from client: Type[CONNECT], Nu
mber[0], Message[null]
Thread[20]: Received from client: Type[MESSAGE], Nu
mber[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
Thread[22]: Received from client: Type[MESSAGE], Nu
mber[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
```

```
]
Alex: hello
java.io.EOFException
        at java.base/java.io.ObjectInputStream$Bloc
kDataInputStream.peekByte(ObjectInputStream.java:32
32)
        at java.base/java.io.ObjectInputStream.read
Object0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.read
Object(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.read
Object(ObjectInputStream.java:498)
        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
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[conne
cted]
*John connected*
Debug Info: Type[DISCONNECT], Number[0], Message[di
sconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[conne
cted]
*Alex connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello
]
John: hello
Debug Info: Type[MESSAGE], Number[0], Message[hello
]
Alex: hello
```

```
Debug Info: Type[MESSAGE], Number[0], Message[hell
o]
John: hello
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hell
o]
Alex: hello
java.io.EOFException
        at java.base/java.io.ObjectInputStream$Blo
ckDataInputStream.peekByte(ObjectInputStream.java:
3232)
        at java.base/java.io.ObjectInputStream.rea
dObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.rea
dObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.rea
dObject(ObjectInputStream.java:498)
        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
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[conn
ected]
*Alex connected*
Debug Info: Type[MESSAGE], Number[0], Message[hell
o]
John: hello
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hell
o]
Alex: hello
```

On the terminal server window (1st window) ctrl^c is done to terminate server and the client windows on the right are shown to be disconnected but still running. It closed connection to server but did not shut down the client. They are able to reconnect after the server come back online as shown of them doing /connect localhost:3000 and successfully connect to the server again.

Checklist Items (1)

#2 Show the server terminating; Clients should be disconnected but still running and able to reconnect when the server is back online (demonstrate this)

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:

Three ways a client disconnects is first they choose to disconnect by doing the command (/disconnect), second is if the client window is terminated which shuts down the connection for them, and third is if the server is terminated and then all clients become disconnected but are still running. The ServerThread handles connection status and checks for a client and who, and also processes whether they connected or disconnected to the server from the PayloadType. The client doesn't crash if server disconnects or terminates because if the server is not running, then It safely disconnects all clients from the server by using try catch block that catch exceptions in the client side and give the message that they simply disconnected and connections closed. The server doesn't crash from the clients disconnecting because ServerThread handles in the run() method if a client disconnects, it will catch the exception that happens when they disconnect and begins a thread cleanup after they disconnected and allows the server to keep running.

● **Misc** (1 pt.)

COLLAPSE

● COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/OmarBarrera1/ob75-IT114-002/pull/8

● 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:

There was almost no problems or issues with the assignment except for remembering some commands such as creating a room or joining the lobby again. In terms of learnings, going through all the code individually helped give me a better understanding of what each part of the code does and how data is handled from first being sent and in the end being broadcasted to everyone in the room. Also payload and payload types were all new at first but looking at them and understanding what the payload does, it helps understand how the data being sent from client generates a payload and the type gives the program information on how to process that payload.

● COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

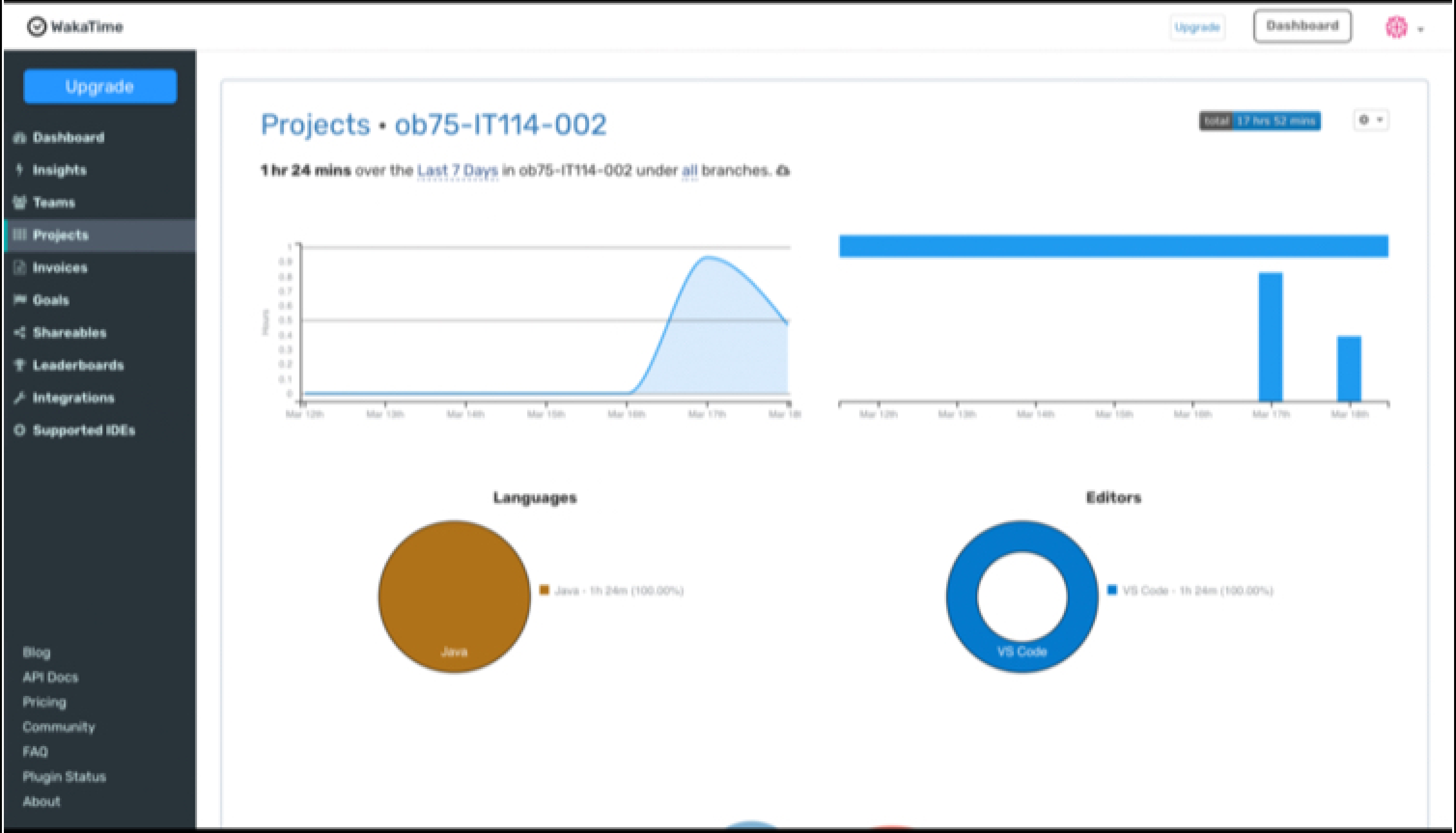
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository.

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Projects • ob75-IT114-002

1 hr 24 mins over the Last 7 Days in ob75-IT114-002 under all branches.

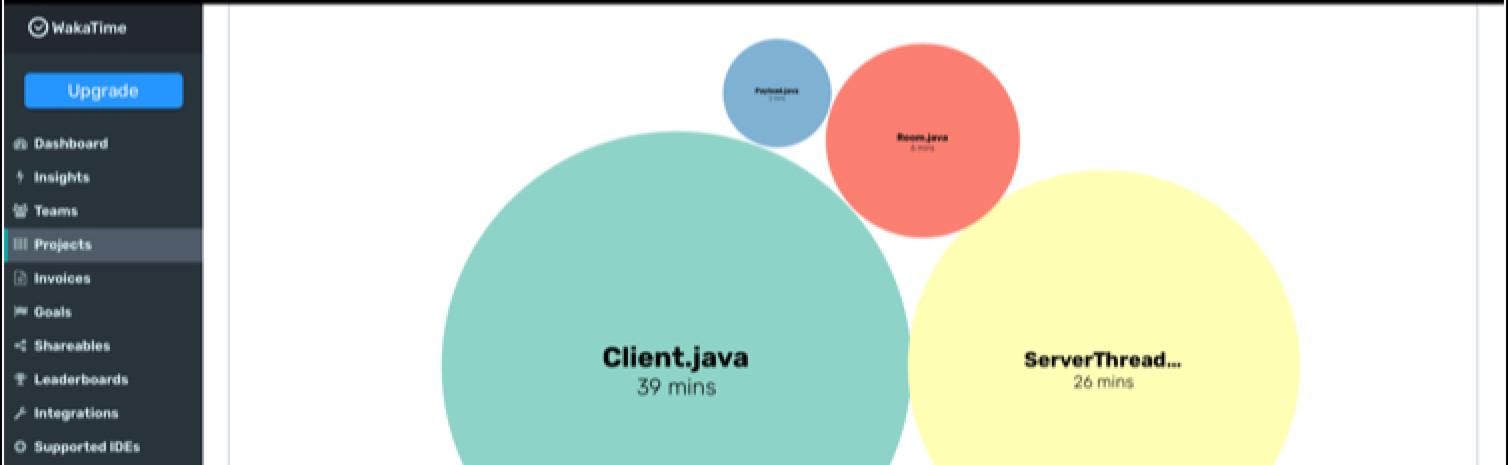
Languages

🟫 Java - 1h 24m (100.00%)

Editors

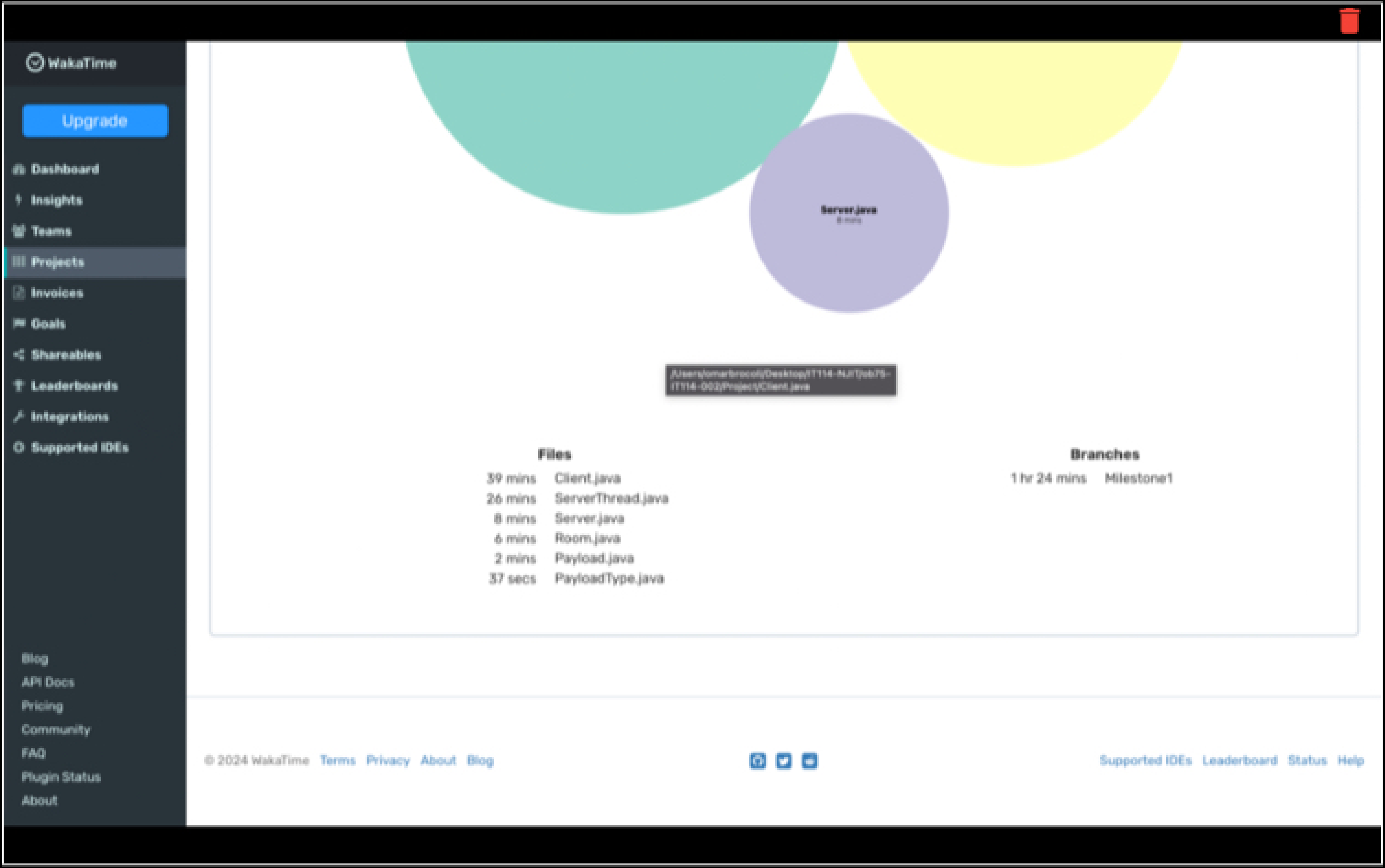
🟦 VS Code - 1h 24m (100.00%)

Waka time for Milestone1 top of page showing total time this week.



Server.java
8 mins
/Users/omarbrocoli/Desktop/IT114-NJIT/ob75-IT114-002/Project/ServerThread.java
Blog
API Docs
Pricing
Community
FAQ
Plugin Status
About

Waka time for Milestone 1, middle of page showing time in each file.



Files

39 mins	Client.java
26 mins	ServerThread.java
8 mins	Server.java
6 mins	Room.java
2 mins	Payload.java
37 secs	PayloadType.java

Branches

1 hr 24 mins	Milestone1

Waka time for Milestone 1, bottom of page showing time in each file and branches.

End of Assignment