Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-project-milestone-1/grade/oha2

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

Submissions:

Submission Selection

1 Submission [active] 3/17/2024 3:17:35 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

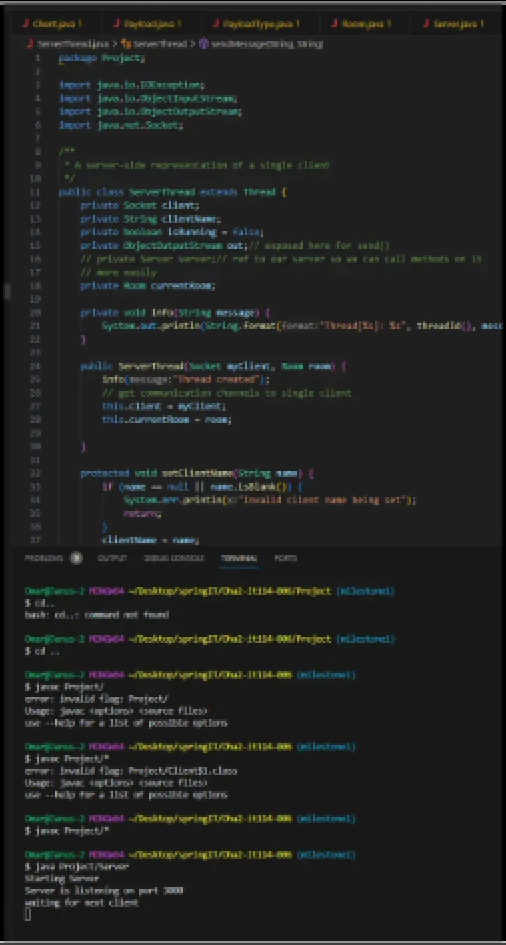
Checklist *The checkboxes are for your own tracking

#	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

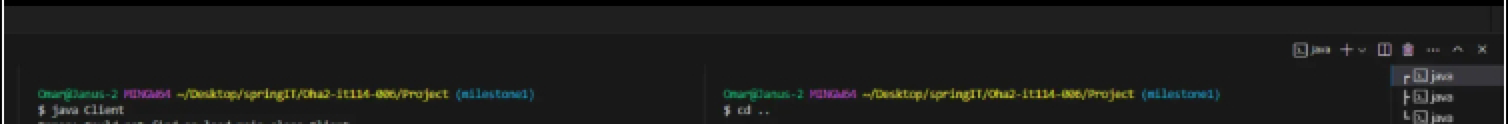


```
package Project;

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()
    // private Server server;// ref to our server so we can call methods on it
    // more easily
    private Room currentRoom;

    private void info(String message) {
        System.out.println(String.format(format:"Thread[%s]: %s", threadId(), mess
    }

    public ServerThread(Socket myClient, Room room) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
        this.currentRoom = room;

    }

    protected void setClientName(String name) {
        if (name == null || name.isBlank()) {
            System.err.println(x:"Invalid client name being set");
            return;
        }
        clientName = name;
```

```
Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ cd..
bash: cd..: command not found

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ cd ..

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ javac Project/
error: invalid flag: Project/
Usage: javac <options> <source files>
use --help for a list of possible options

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ javac Project/*
error: invalid flag: Project/Client$1.class
Usage: javac <options> <source files>
use --help for a list of possible options

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ javac Project/*

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Project/Server
Starting Server
Server is listening on port 3000
waiting for next client
```

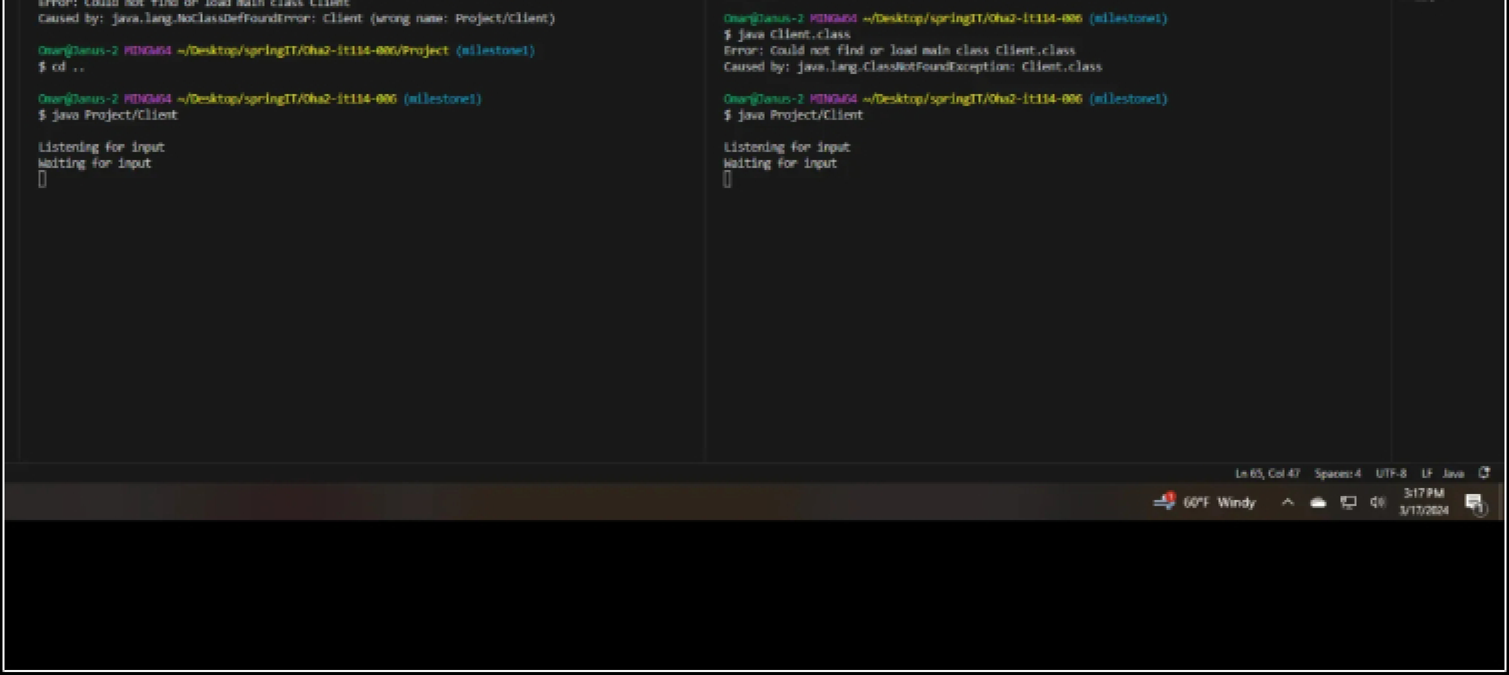
server waiting for client input

Checklist Items (0)



```
Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ java Client
```

```
Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ cd ..
```



```
Error: Could not find or load main class Client
Caused by: java.lang.NoClassDefFoundError: Client (wrong name: Project/Client)

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ cd ..

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Project/Client

Listening for input
Waiting for input
```

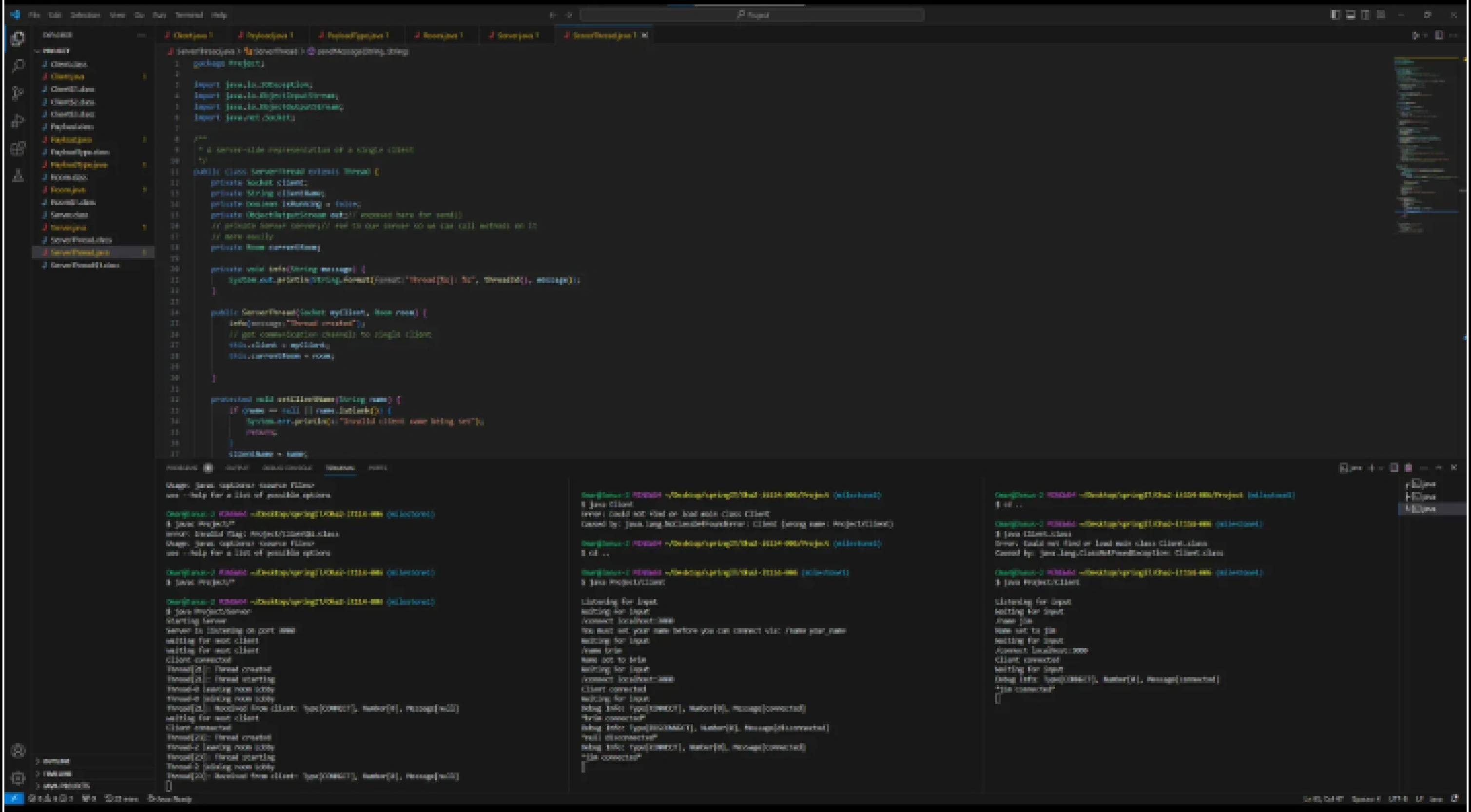
```
Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Client.class
Error: Could not find or load main class Client.class
Caused by: java.lang.ClassNotFoundException: Client.class

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Project/Client

Listening for input
Waiting for input
```

Ln 65, Col 47 Spaces: 4 UTF-8 LF Java

60°F Windy 3:17 PM 3/17/2024

2 clients waiting

Checklist Items (0)



```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()
    // private Server server;// ref to our server so we can call methods on it
    // more easily
    private Room currentRoom;

    private void info(String message) {
        System.out.println(String.format(format:"Thread[%s]: %s", threadId(), message));
    }

    public ServerThread(Socket myClient, Room room) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
        this.currentRoom = room;

    }

    protected void setClientName(String name) {
        if (name == null || name.isBlank()) {
            System.err.println(x:"Invalid client name being set");
            return;
        }
        clientName = name;
```

```
Usage: javac <options> <source files>
use --help for a list of possible options

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ javac Project/*
error: Invalid flag: Project/Client$1.class
Usage: javac <options> <source files>
use --help for a list of possible options

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ javac Project/*

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Project/Server
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[21]: Thread created
Thread[21]: Thread starting
Thread-0 leaving room lobby
Thread-0 joining room lobby
Thread[21]: Received from client: Type[CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[23]: Thread created
Thread-2 leaving room lobby
Thread[23]: Thread starting
Thread-2 joining room lobby
Thread[23]: Received from client: Type[CONNECT], Number[0], Message[null]
```

```
Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ java Client
Error: Could not find or load main class Client
Caused by: java.lang.NoClassDefFoundError: Client (wrong name: Project/Client)

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ cd ..

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Project/Client

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 brie
Name set to brie
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*brie connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
```

```
Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006/Project (milestone1)
$ cd ..

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Client.class
Error: Could not find or load main class Client.class
Caused by: java.lang.ClassNotFoundException: Client.class

Omar@Janus-2 MINGW64 ~/Desktop/springIT/Oha2-it114-006 (milestone1)
$ java Project/Client

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

both clients connected to server with different names

Checklist Items (0)

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:

A server starts up and it starts running the main method in the server class, then it waits for clients to connect by using the port that it specified and it opens sockets. The clients who want to connect have connect to by first making a name for themselves so that the server accepts them and then make a network socket connection with the server's address. The server sees an incoming client and accepts them while making a server thread for them to communicate with. Now the socket they connected to is waiting for something from the server thread like a message from the client.

Communication (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

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

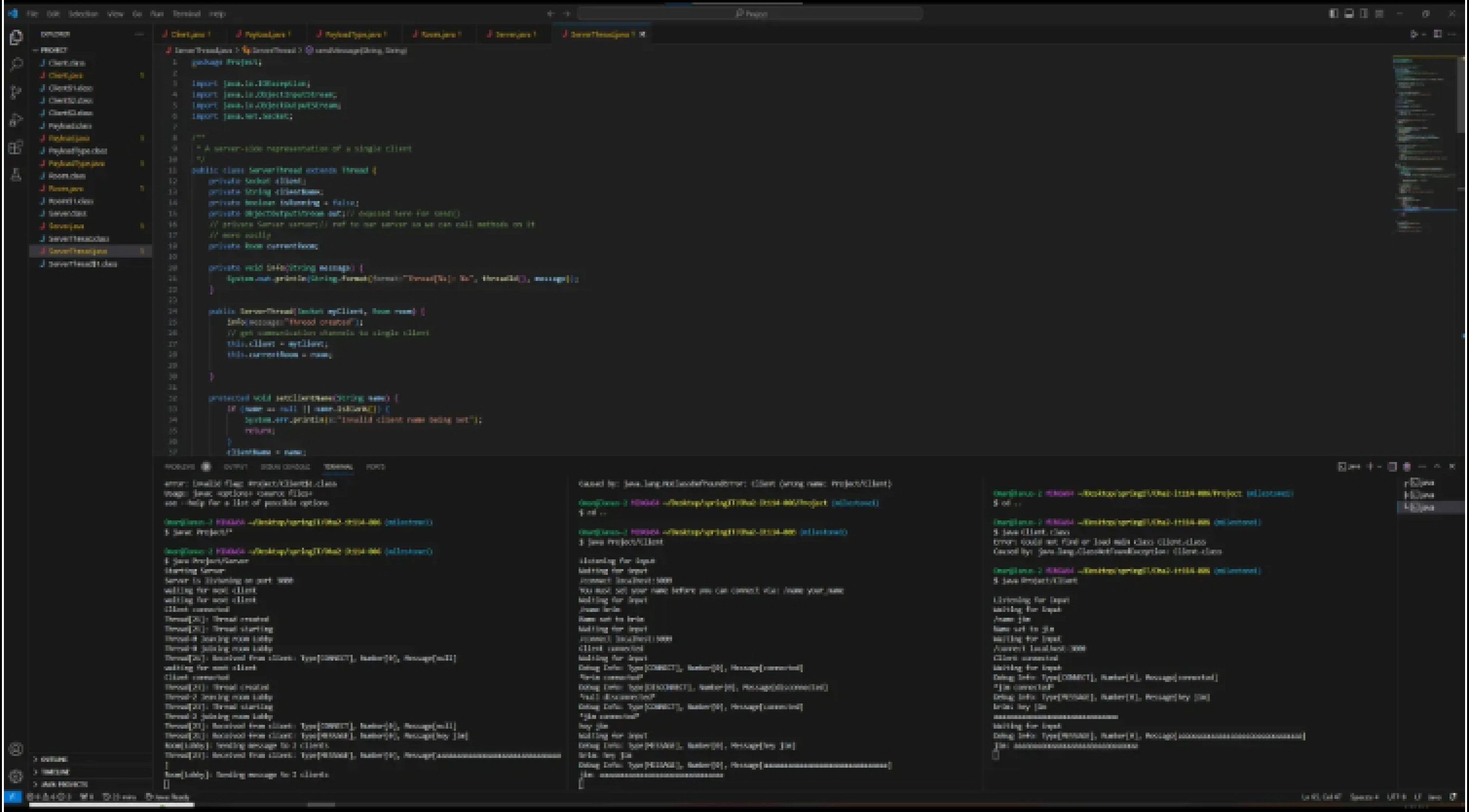
Checklist *The checkboxes are for your own tracking

#	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)
☐ #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



```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()
    // private Server server;// ref to our server so we can call methods on it
    // more easily
    private Room currentRoom;

    private void info(String message) {
        System.out.println(String.format(format:"Thread[%s]: %s", threadId(), message));
    }

    public ServerThread(Socket myClient, Room room) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
        this.currentRoom = room;

    }

    protected void setClientName(String name) {
        if (name == null || name.isBlank()) {
            System.err.println(x:"Invalid client name being set");
            return;
        }
        clientName = name;
```

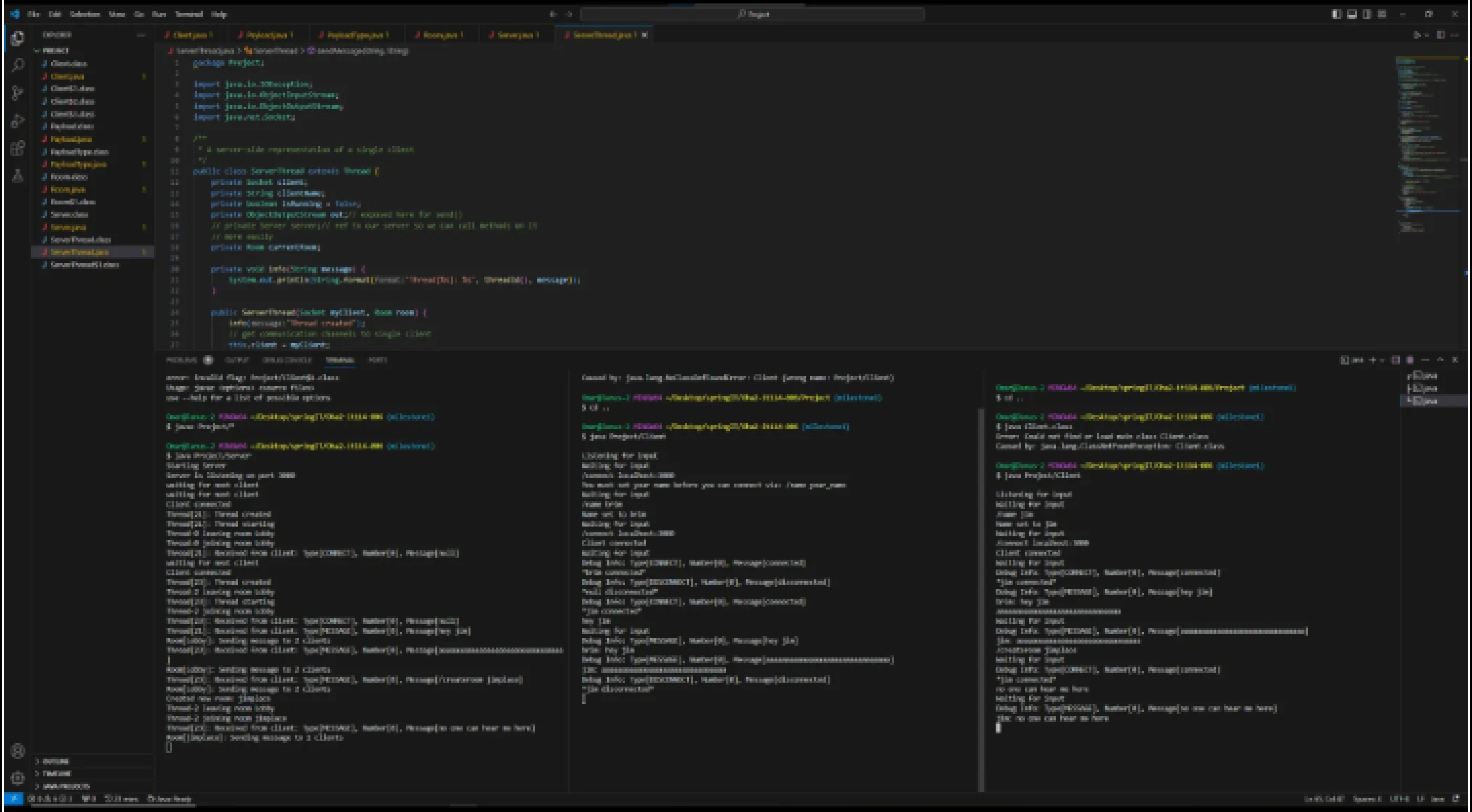
```
error: invalid flag: Project/Client$1.class
Usage: javac <options> <source files>
use --help for a list of possible options

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ javac Project/*

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Project/Server
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], Number[0], Message[null]
waiting for next client
Client connected
Thread[21]: Thread created
Thread-2 leaving room Lobby
Thread[21]: Thread starting
Thread-2 joining room Lobby
Thread[21]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[hey jim]
Room[Lobby]: Sending message to 2 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
]
Room[Lobby]: Sending message to 2 clients
```

```
Caused by: java.lang.NoClassDefFoundError: Client (wrong name: Project/Client)

[illegible] ~/Desktop/spring23/[illegible]/Project ([illegible])
$ cd ..

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Project/Client

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 brie
Name set to brie
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*brie connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
hey jim
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hey jim]
brie: hey jim
Debug Info: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
jim: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

```
[illegible] ~/Desktop/spring23/[illegible]/Project ([illegible])
$ cd ..

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Client.class
Error: Could not find or load main class Client.class
Caused by: java.lang.ClassNotFoundException: Client.class

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Project/Client

Listening for input
Waiting for input
/name jim
Name set to jim
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
Debug Info: Type[MESSAGE], Number[0], Message[hey jim]
brie: hey jim
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
jim: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

clients sending messages to each other and receiving the server messages

Checklist Items (0)



```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()
    // private Server server;// ref to our server so we can call methods on it
    // more easily
    private Room currentRoom;

    private void info(String message) {
        System.out.println(String.format(format:"Thread[%s]: %s", threadId(), message));
    }

    public ServerThread(Socket myClient, Room room) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
```

```
error: invalid flag: Project/Client$1.class
Usage: javac <options> <source files>
use --help for a list of possible options

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ javac Project/*

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Project/Server
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[21]: Thread created
Thread[21]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[21]: Received from client: Type[CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[23]: Thread created
Thread-2 leaving room Lobby
Thread[23]: Thread starting
Thread-2 joining room Lobby
Thread[23]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[hey jim]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/createroom jimplace]
Room[Lobby]: Sending message to 2 clients
Created new room: jimplace
Thread-2 leaving room Lobby
Thread-2 joining room jimplace
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[no one can hear me here]
Room[jimplace]: Sending message to 1 clients
```

```
Caused by: java.lang.NoClassDefFoundError: Client (wrong name: Project/Client)

[illegible] ~/Desktop/spring23/[illegible]/Project ([illegible])
$ cd ..

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Project/Client

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 brie
Name set to brie
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*brie connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
hey jim
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hey jim]
brie: hey jim
Debug Info: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
jim: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*jim disconnected*
```

```
[illegible] ~/Desktop/spring23/[illegible]/Project ([illegible])
$ cd ..

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Client.class
Error: Could not find or load main class Client.class
Caused by: java.lang.ClassNotFoundException: Client.class

[illegible] ~/Desktop/spring23/[illegible] ([illegible])
$ java Project/Client

Listening for input
Waiting for input
/name jim
Name set to jim
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
Debug Info: Type[MESSAGE], Number[0], Message[hey jim]
brie: hey jim
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
jim: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
/createroom jimplace
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
no one can hear me here
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[no one can hear me here]
jim: no one can hear me here
```

Client jim in their own room where brim can not see their message

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:

When a client connects they are placed into the room with their server thread. The room is what interprets everything from a server thread. The server thread interprets everything that a client inputs and send it to the room where it displays the input, a client sends a message it calls the send Message method and the thread interprets that and sends the message which displays it in the room. This displays it in the room for all connected clients including the one that sent it effectively having them receive the message.

●

^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

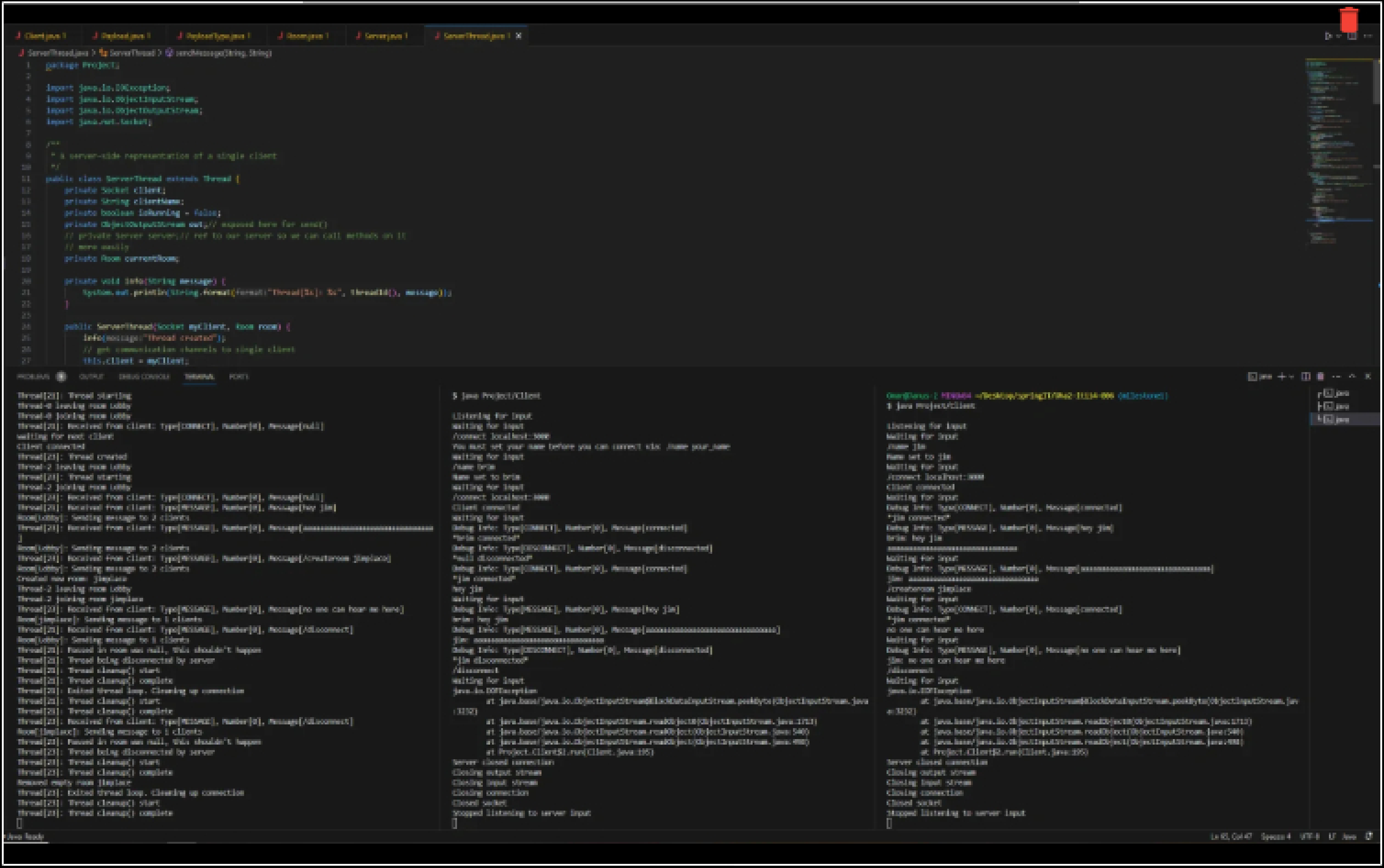
#	Points	Details
#1	1	Show a client disconnecting from the server; Server should still be running without issue (it's ok if an exception message shows as it's part of the lesson code, the server just shouldn't terminate)
#2	1	Show the server terminating; Clients should be disconnected but still running 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



```
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()
    // private Server server;// ref to our server so we can call methods on it
    // more easily
    private Room currentRoom;

    private void info(String message) {
        System.out.println(String.format(format:"Thread[%s]: %s", threadId(), message));
    }

    public ServerThread(Socket myClient, Room room) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
```

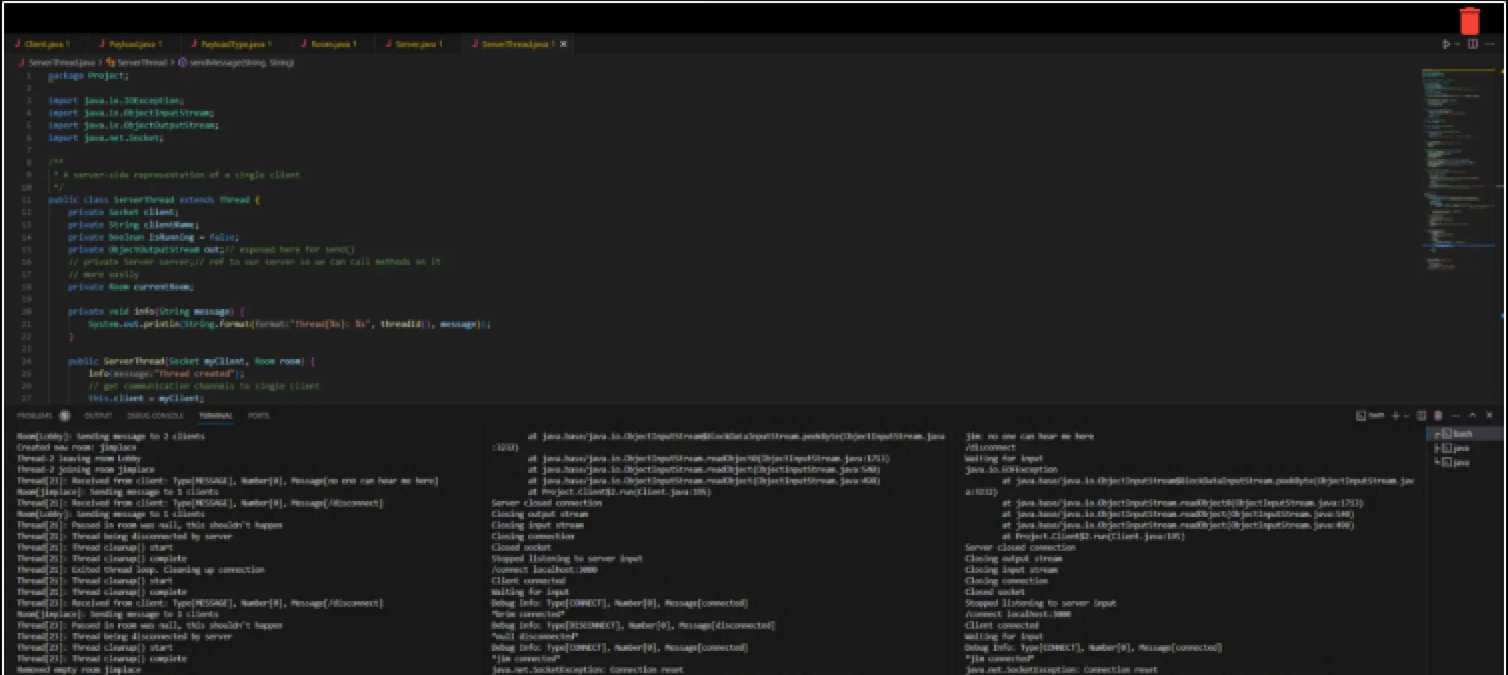
```
Thread[21]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[21]: Received from client: Type[CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[23]: Thread created
Thread-2 leaving room Lobby
Thread[23]: Thread starting
Thread-2 joining room Lobby
Thread[23]: Received from client: Type[CONNECT], Number[0], Message[null]
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[hey jim]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
]
Room[Lobby]: Sending message to 2 clients
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/createroom jimplace]
Room[Lobby]: Sending message to 2 clients
Created new room: jimplace
Thread-2 leaving room Lobby
Thread-2 joining room jimplace
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[no one can hear me here]
Room[jimplace]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[Lobby]: Sending message to 1 clients
Thread[21]: Passed in room was null, this shouldn't happen
Thread[21]: Thread being disconnected by server
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Thread[21]: Exited thread loop. Cleaning up connection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[jimplace]: Sending message to 1 clients
Thread[23]: Passed in room was null, this shouldn't happen
Thread[23]: Thread being disconnected by server
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
Removed empty room jimplace
Thread[23]: Exited thread loop. Cleaning up connection
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
```

```
$ java Project/Client

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 brim
Name set to brim
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*brim connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
hey jim
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hey jim]
brim: hey jim
Debug Info: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
jim: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*jim disconnected*
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.java
:3232)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.readObject(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
```

```
$ java Project/Client

Listening for input
Waiting for input
/name jim
Name set to jim
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
Debug Info: Type[MESSAGE], Number[0], Message[hey jim]
brim: hey jim
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa]
jim: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
/createroom jimplace
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
no one can hear me here
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[no one can hear me here]
jim: no one can hear me here
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.jav
a:3232)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.readObject(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
```

clients leaving but server still running

Checklist Items (0)

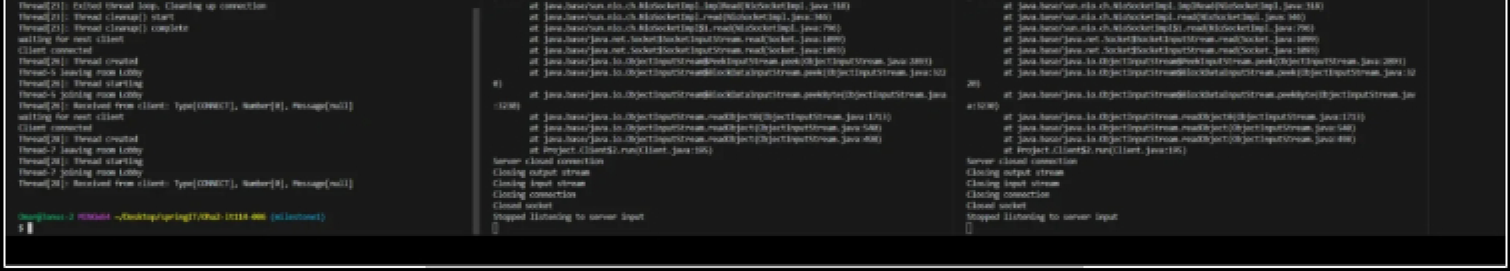


```
package Project;

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()
    // private Server server;// ref to our server so we can call methods on it
    // more easily
    private Room currentRoom;

    private void info(String message) {
        System.out.println(String.format(format:"Thread[%s]: %s", threadId(), message));
    }

    public ServerThread(Socket myClient, Room room) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
```

```
Room[Lobby]: Sending message to 2 clients
Created new room: jimplace
Thread-2 leaving room Lobby
Thread-2 joining room jimplace
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[no one can hear me here]
Room[jimplace]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[Lobby]: Sending message to 1 clients
Thread[21]: Passed in room was null, this shouldn't happen
Thread[21]: Thread being disconnected by server
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Thread[21]: Exited thread loop. Cleaning up connection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[jimplace]: Sending message to 1 clients
Thread[23]: Passed in room was null, this shouldn't happen
Thread[23]: Thread being disconnected by server
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
Removed empty room jimplace
```

```
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.java
:3232)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.readObject(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[connected]
*brim connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*jim connected*
java.net.SocketException: Connection reset
```

```
jim: no one can hear me here
/disconnect
Waiting for input
java.io.EOFException
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.jav
a:3232)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.readObject(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[connected]
*jim connected*
java.net.SocketException: Connection reset
```



```
Thread[21]: Exited thread loop. Cleaning up connection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
waiting for next client
Client connected
Thread[26]: Thread created
Thread-5 leaving room Lobby
Thread[26]: Thread starting
Thread-5 joining room Lobby
Thread[26]: Received from client: Type[CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[28]: Thread created
Thread-7 leaving room Lobby
Thread[28]: Thread starting
Thread-7 joining room Lobby
Thread[28]: Received from client: Type[CONNECT], Number[0], Message[null]

[illegible] ~/Desktop/springIT/Dha2-IT114-006 (milestone1)
$
```

```
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:318)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:346)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:796)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:1099)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:1093)
        at java.base/java.io.ObjectInputStream$PeekInputStream.peek(ObjectInputStream.java:2891)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peek(ObjectInputStream.java:322
8)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.java
:3238)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.readObject(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
```

```
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:318)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:346)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:796)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:1099)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:1093)
        at java.base/java.io.ObjectInputStream$PeekInputStream.peek(ObjectInputStream.java:2891)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peek(ObjectInputStream.java:32
28)
        at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.ja
va:3238)
        at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.readObject(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
```

server closing but clients still running

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:

Disconnects can happen from, a server willingly closing itself, a client willingly disconnected, or some kind of break in the connects between the server and client.
A client disconnects: a client disconnects or gets disconnected making removeclient get called in the room and the message that they disconnected is broadcast to everyone in the room they were in only. If they were the only person in that room it gets cleaned up after their disconnect unless it is the lobby.
When a client disconnects instead of crashing the client starts running the prompt to insert an address to connect to.
The server doesn't crash because only a single server thread is gone now and the socket the thread was in is closed with client.Disconnect so the client is removed from their and server list for clients and such.
A server disconnects: The serverthread and socket connects no longer exists to the client and they are reverted to their state of being prompted to insert an address to a server to connect to so they don't crash.

Misc (1 pt.)

^COLLAPSE ^

^COLLAPSE ^

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

https://github.com/Jomarrrrr/Oha2-it114-006/pull/6

●
^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 issues initially because I had to compile everything together instead of separately for the bash terminal to let me compile without a bunch of errors. I also didn't read the first step initially and had the folder they were in called milestone1 instead of project so I had to change the name of the folder and then the package info a second time. Other than that it went smoothly.

●
^COLLAPSE^

Task #3 - Points: 1

Text: WakaTime Screenshot

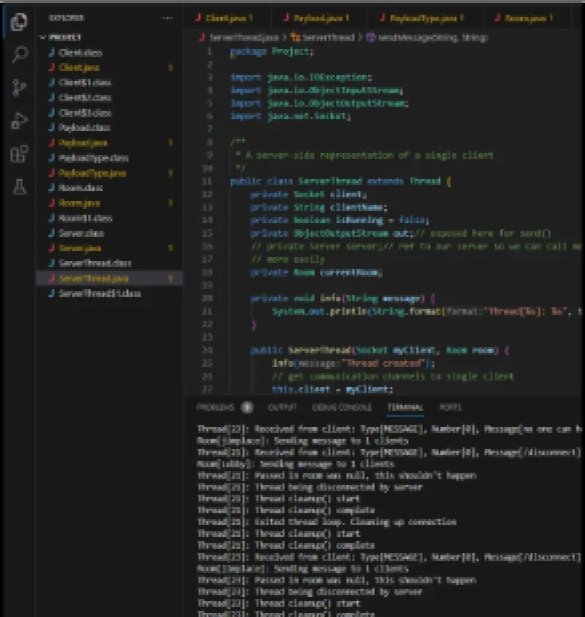
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository.

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

Task Screenshots:

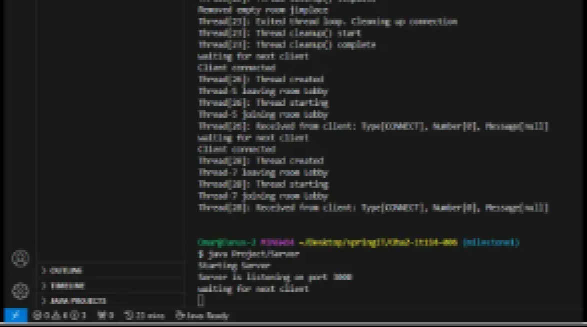
Gallery Style: Large View

Small Medium Large



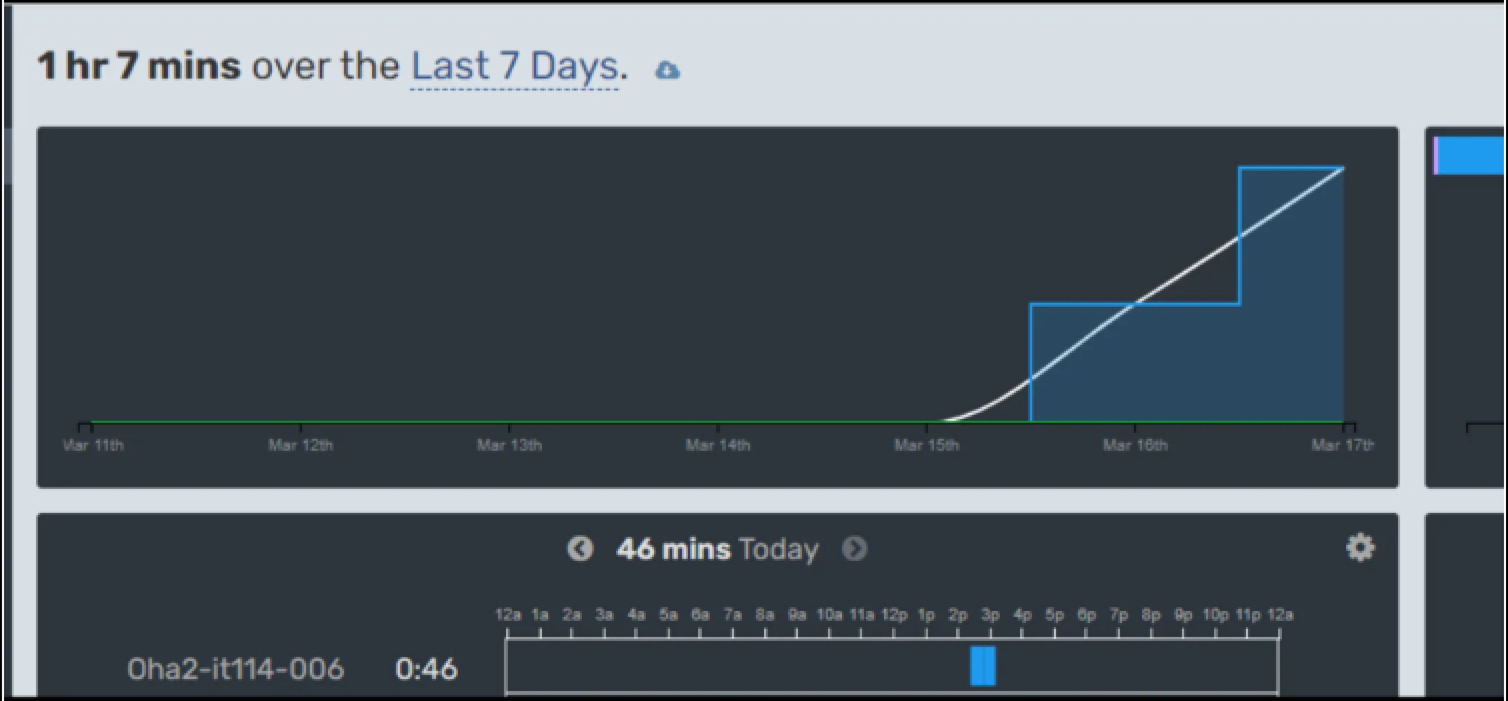
```java
package Project;

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()
    // private Server server;// ref to our server so we can call m
    // more easily
    private Room currentRoom;

    private void info(String message) {
        System.out.println(String.format(format:"Thread[%s]: %s", t
    }

    public ServerThread(Socket myClient, Room room) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
```

```
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[no one can h
Room[lobby]: Sending message to 1 clients
Thread[21]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[lobby]: Sending message to 1 clients
Thread[21]: Passed in room was null, this shouldn't happen
Thread[21]: Thread being disconnected by server
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Thread[21]: Exited thread loop. Cleaning up connection
Thread[21]: Thread cleanup() start
Thread[21]: Thread cleanup() complete
Thread[23]: Received from client: Type[MESSAGE], Number[0], Message[/disconnect]
Room[lobby]: Sending message to 1 clients
Thread[24]: Passed in room was null, this shouldn't happen
Thread[23]: Thread being disconnected by server
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
```



```
Removed empty room jimploc
Thread[23]: Exited thread loop. Cleaning up connection
Thread[23]: Thread cleanup() start
Thread[23]: Thread cleanup() complete
waiting for next client
Client connected
Thread[26]: Thread created
Thread-5 leaving room Lobby
Thread[26]: Thread starting
Thread-5 joining room Lobby
Thread[26]: Received from client: Type[CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[28]: Thread created
Thread-7 leaving room Lobby
Thread[28]: Thread starting
Thread-7 joining room Lobby
Thread[28]: Received from client: Type[CONNECT], Number[0], Message[null]


[illegible] ~/Desktop/spring23/Oha2-it114-006 (Milestone1)
$ java Project/Server
Starting Server
Server is listening on port 3000
waiting for next client
```

waka time symbol tracking on bottom of screen



1 hr 7 mins over the Last 7 Days.

Mar 11th
Mar 12th
Mar 13th
Mar 14th
Mar 15th
Mar 16th
Mar 17th

46 mins Today

12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p 12a

Oha2-it114-006 0:46

waketime website with my repository name

End of Assignment