

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-project-milestone-1/grade/sb57>

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

Submissions:

Submission Selection

1 Submission [active] 4/30/2024 4:14:36 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 ^

Task #1 - Points: 1

Text: Server and Client Initialization

Checklist

*The checkboxes are for your own tracking

| # | Points | Details |
|-----------------------------|--------|--|
| <input type="checkbox"/> #1 | 1 | Server should properly be listening to its port from the command line (note the related message) |
| <input type="checkbox"/> #2 | 1 | Clients should be successfully waiting for input |
| <input type="checkbox"/> #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

```
PS C:\Users\Shreya\Desktop\5.55.Server>
, are only accepted if annotation proc
server
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[15]: Thread created
Thread[15]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[15]: Received from client: Type
[CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[17]: Thread created
Thread-2 leaving room Lobby
Thread[17]: Thread starting
Thread-2 joining room Lobby
Thread[17]: Received from client: Type
[CONNECT], Number[0], Message[null]
[]
```

```
PS C:\Users\Shreya\Desktop\IT114\sb57-
IT114-006\Project> java Module5.Part5.
Client

Listening for input
Not connected to server
Waiting for input
/connect localhost:3000
You must set your name before you can
connect via: /name your_name
Waiting for input
/name Shreya
Name set to Shreya
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0],
Message[connected]
*Shreya connected*
Debug Info: Type[DISCONNECT], Number[0
], Message[disconnected]
>null disconnected*
Debug Info: Type[CONNECT], Number[0],
Message[connected]
*Bud connected*
[]
```

```
nt.java
PS C:\Users\Shreya\Desktop\IT114\sb57
-IT114-006\Project> java Module5.Part
5.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 Bud
Name set to Bud
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0],
Message[connected]
*Bud connected*
[]
```

This screenshot shows the Server listening to Port 3000. It also shows two Clients connected successfully to the server with names to each, and is successfully waiting for input.

Checklist Items (3)

#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

#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 |
|-----------------------------|--------|--|
| <input type="checkbox"/> #1 | 1 | Mention how the server-side of the connection works |
| <input type="checkbox"/> #2 | 1 | Mention how the client-side of the connection works |
| <input type="checkbox"/> #3 | 1 | Describe the socket steps until the server is waiting for messages from the client |

Response:

Both the server and the client connect to a port, the server listens to it and the client needs to connect to the port the server is listening to. The client connects to the server by socket which is the pathway for the connection to the client/server relationship.



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 |
|-----------------------------|--------|--|
| <input type="checkbox"/> #1 | 1 | At least two clients connected to the server |
| <input type="checkbox"/> #2 | 1 | Client can send messages to the server |
| <input type="checkbox"/> #3 | 1 | Server sends the message to all clients in the same room |
| <input type="checkbox"/> | | |

| | | |
|----|---|---|
| #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

| | | |
|--|---|--|
| <pre>PS C:\Users\Shreya\Desktop> 5.S5.Server , are only accepted if annotation proc server Starting Server Server is listening on port 3000 waiting for next client waiting for next client Client connected Thread[15]: Thread created Thread[15]: Thread starting Thread-0 leaving room Lobby Thread-0 joining room Lobby Thread[15]: Received from client: Type [CONNECT], Number[0], Message[null] waiting for next client Client connected Thread[17]: Thread created Thread-2 leaving room Lobby Thread[17]: Thread starting Thread-2 joining room Lobby Thread[17]: Received from client: Type [CONNECT], Number[0], Message[null] Thread[15]: Received from client: Type [MESSAGE], Number[0], Message[hi] Room[Lobby]: Sending message to 2 clie nts Thread[17]: Received from client: Type [MESSAGE], Number[0], Message[hello] Room[Lobby]: Sending message to 2 clie nts []</pre> | <pre>Not connected to server Waiting for input /connect localhost:3000 You must set your name before you can connect via: /name your_name Waiting for input /name Shreya Name set to Shreya Waiting for input /connect localhost:3000 Client connected Waiting for input Debug Info: Type[CONNECT], Number[0], Message[connected] *Shreya connected* Debug Info: Type[DISCONNECT], Number[0], Message[disconnected] >null disconnected* Debug Info: Type[CONNECT], Number[0], Message[connected] *Bud connected* hi Waiting for input Debug Info: Type[MESSAGE], Number[0], Message[hi] Shreya: hi Debug Info: Type[MESSAGE], Number[0], Message[hello] Bud: hello []</pre> | <pre>nt.java PS C:\Users\Shreya\Desktop> IT114\sb57 -IT114-006\Project> java Module5.Part 5.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 Bud Name set to Bud Waiting for input /connect localhost:3000 Client connected Waiting for input Debug Info: Type[CONNECT], Number[0], Message[connected] *Bud connected* Debug Info: Type[MESSAGE], Number[0], Message[hi] Shreya: hi hello Waiting for input Debug Info: Type[MESSAGE], Number[0], Message[hello] Bud: hello []</pre> |
|--|---|--|

This screenshot shows two clients successfully connected to the server on port 3000, Shreya and Bud. Both clients successfully send messages to the server and as a response, the server sends messages to all the clients since they are in the same room. The messages clearly outline who sent the message.

Checklist Items (5)

- #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)
- #6 Clearly caption each image regarding what is being shown


```
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[15]: Thread created
Thread[15]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[15]: Received from client: Type
[CONNECT], Number[0], Message[null]
waiting for next client
client connected
Thread[17]: Thread created
Thread-2 leaving room Lobby
Thread[17]: Thread starting
Thread-2 joining room Lobby
Thread[17]: Received from client: Type
[CONNECT], Number[0], Message[null]
Thread[15]: Received from client: Type
[MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 2 clients
Thread[17]: Received from client: Type
[MESSAGE], Number[0], Message[hello]
Room[Lobby]: Sending message to 2 clients
Thread[15]: Received from client: Type
[MESSAGE], Number[0], Message[/createroom test1]
Room[Lobby]: Sending message to 2 clients
Created new room: test1
Thread-0 leaving room Lobby
Thread-0 joining room test1
Thread[17]: Received from client: Type
[MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
Thread[15]: Received from client: Type
[MESSAGE], Number[0], Message[hello]
Room[test1]: Sending message to 1 clients
[]



Waiting for input
/connect localhost:3000
You must set your name before you can
connect via: /name your_name
Waiting for input
/name Shreya
Name set to Shreya
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0],
Message[connected]
*Shreya connected*
Debug Info: Type[DISCONNECT], Number[0],
Message[disconnected]
>null disconnected*
Debug Info: Type[CONNECT], Number[0],
Message[connected]
*Bud connected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0],
Message[hi]
Shreya: hi
Debug Info: Type[MESSAGE], Number[0],
Message[hello]
Bud: hello
/createroom test1
Waiting for input
Debug Info: Type[CONNECT], Number[0],
Message[connected]
*Shreya connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0],
Message[hello]
Shreya: hello
[]

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 Bud
Name set to Bud
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0],
Message[connected]
*Bud connected*
Debug Info: Type[MESSAGE], Number[0],
Message[hi]
Shreya: hi
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0],
Message[hello]
Bud: hello
Debug Info: Type[DISCONNECT], Number[0],
Message[disconnected]
*Shreya disconnected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0],
Message[hi]
Bud: hi
[]
```

This screenshot shows client Shreya create and join a new room called test1. From here it communicates with the server, and client Bud does not receive the message since it is in the room Lobby.


Checklist Items (2)

- #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)
- #6 Clearly caption each image regarding what is being shown

 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) | |
|  | 1 | Mention the ServerThread's involvement | |

| | | |
|----|---|---|
| #2 | 1 | Mention the Server Thread's involvement |
| #3 | 1 | Mention the Room's perspective |
| #4 | 1 | Mention the client-side (receiving) |

Response:

For the server-side of the connection, the server needs to be compiled and then run. For the server-side of the connection, the server needs to be compiled and then run on a different terminal. You would then use the /name command to assign a name to the client and then the /connect command to connect to port 3000 (or whichever port the server is listening to). The client sends requests to the ServerThread and the server replies, creating individual threads for each client. When the /createroom and /joinroom commands are used, only the clients that are in the same room will receive messages from the room by the server.

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

```
Room[Lobby]: Sending message to 2 clients
Thread[15]: Received from client: Type [MESSAGE], Number[0], Message[/createroom test1]
Room[Lobby]: Sending message to 2 clients
Created new room: test1
Thread-0 leaving room Lobby
Thread-0 joining room test1
Thread[17]: Received from client: Type [MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
Thread[15]: Received from client: Type [MESSAGE], Number[0], Message[hello]
Room[test1]: Sending message to 1 clients
Thread[15]: Received from client: Type [MESSAGE], Number[0], Message[/joinroom Lobby]
```

```
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi]
Shreya: hi
Debug Info: Type[MESSAGE], Number[0], Message[hello]
Bud: hello
/createroom test1
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Shreya connected*
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
Shreya: hello
/joinroom Lobby
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
```

```
-IT114-006\Project> java Module5.Part5.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 Bud
Name set to Bud
Waiting for input
/connect localhost:3000
Client connected
Waiting for input
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Bud connected*
Debug Info: Type[MESSAGE], Number[0], Message[hi]
Shreya: hi
```

```

Room[Lobby]: Sending message to 1 clients
Room[test1]: Sending message to 1 clients
Thread-0 leaving room test1
Removed empty room test1
Thread-0 joining room lobby
Thread[15]: Received from client: Type [MESSAGE], Number[0], Message[/disconnect]
Room[Lobby]: Sending message to 2 clients
Thread[15]: Passed in room was null, this shouldn't happen
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[17]: Received from client: Type [MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
[ ]

Message[connected]
*Shreya connected*
/d disconnect
Waiting for input
java.io.EOFException
    at java.base/java.io.ObjectInputStream$BlockDataInputStream.peekByte(ObjectInputStream.java:3214)
    at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1684)
    at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:509)
    at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:467)
    at Module5.Part5.Client$2.run(Client.java:195)
Server closed connection
Closing output stream
Closing input stream
Closing connection
Closed socket
Stopped listening to server input
[ ]

Shreya: hi
hello
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hello]
Bud: hello
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*Shreya disconnected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi]
Bud: hi
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Shreya connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*Shreya disconnected*
hi
Waiting for input
Debug Info: Type[MESSAGE], Number[0], Message[hi]
Bud: hi

```

This screenshot shows client Shreya disconnecting from the server and the server still continuing to run successfully. The disconnected message was shown to client Bud who is still connected successfully to the server.

Checklist Items (3)

- #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)
- #4 Clearly caption each image regarding what is being shown

```

Thread[15]: Thread cleanup() start
Thread[15]: Thread cleanup() complete
Thread[17]: Received from client: Type [MESSAGE], Number[0], Message[hi]
Room[Lobby]: Sending message to 1 clients
PS C:\Users\Shreya\Desktop\IT114\sb57-IT114-006\Project> java Module5.Part5.Server
Starting Server
Server is listening on port 3000
waiting for next client
waiting for next client
Client connected
Thread[15]: Thread created
Thread[15]: Thread starting
Thread-0 leaving room Lobby
Thread-0 joining room Lobby
Thread[15]: Received from client: Type [CONNECT], Number[0], Message[null]
waiting for next client
Client connected
Thread[17]: Thread created
Thread-2 leaving room Lobby
Thread[17]: Thread starting
Thread-2 joining room Lobby
Thread[17]: Received from client: Type [CONNECT], Number[0], Message[null]
[ ]

utStream.readObject0(ObjectInputStream.java:1684)
    at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:509)
    at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:467)
    at Module5.Part5.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]
*Shreya connected*
Debug Info: Type[DISCONNECT], Number[0], Message[disconnected]
*null disconnected*
Debug Info: Type[CONNECT], Number[0], Message[connected]
*Bud connected*
[ ]

putStream$BlockDataInputStream.peek(ObjectInputStream.java:3202)
    at java.base/java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1684)
    at java.base/java.io.ObjectInputStream.readObject(ObjectInputStream.java:467)
    at Module5.Part5.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]
*Bud connected*

```

This screenshot shows the server terminating and then coming back online. Both clients received the message that the sever closed the connection and once the server was back online, both clients were able to successfully reconnect to the server.

Checklist Items (3)

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

#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)

#4 Clearly caption each image regarding what is being shown



^ 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 |
|-----------------------------|--------|--|
| <input type="checkbox"/> #1 | 1 | Mention how a client gets disconnected from a Socket perspective |
| <input type="checkbox"/> #2 | 1 | Mention how/why the client program doesn't crash when the server disconnects/terminates. |
| <input type="checkbox"/> #3 | 1 | Mention how the server doesn't crash from the client(s) disconnecting |

Response:

The client gets disconnected from the Socket using the command /disconnect and the server gets terminated using the Ctrl-C command. The client program doesn't crash when the server terminates and the server doesn't crash when the client(s) disconnects because the server and the client(s) are run independently.



Misc (1 pt.)

^ COLLAPSE ^



^ COLLAPSE ^

Task #1 - Points: 1

Text: Add the pull request link for this branch

URL #1

<https://github.com/sb57-shreya/sb57-IT114-006/pull/9>



^ COLLAPSE ^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

i Details:

Few related sentences about the Project/sockets topics

Response:

This assignment gave me a much clearer understanding of the client/server relationship and connection. I am able to navigate my projects easier as well.



^ COLLAPSE ^

Task #3 - Points: 1

Text: WakaTime Screenshot

i 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

Screenshot of WakaTime running.

End of Assignment