Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 Module 4 Sockets Part3 Challenge
Student: Graham B. (gb373)
Status: Submitted | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 6/16/2025 3:38:48 PM
Updated: 6/16/2025 5:11:52 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/grading/gb373
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/view/gb373

Instructions

- Overview Link: https://youtu.be/_029E_aBTFo

1. Ensure you read all instructions and objectives before starting.
2. Create a new branch from `main` called `M4-Homework`
    1. `git checkout main` (ensure proper starting branch)
    2. `git pull origin main` (ensure history is up to date)
    3. `git checkout -b M4-Homework` (create and switch to branch)
3. Copy the template code from here: GitHub Repository - M4 Homework
    - It includes Sockets `Part1`, `Part2`, and `Part3`. Put all into an `M4` folder or similar if you don't have them yet (adjust `package` reference at the top if you chose a different folder name).
    - Make a copy of `Part3` and call it `Part3HW`
        - Fix the `package` and `import` references at the top of each file in this new folder (Note: you'll only be editing files in `Part3HW`)
    - Immediately record to history
        - `git add .`
        - `git commit -m "adding M4 HW baseline files"`
        - `git push origin M4-Homework`
        - Create a Pull Request from `M4-Homework` to `main` and keep it open
4. Fill out the below worksheet
    - Each Problem requires the following as you work
        - Ensure there's a comment with your UCID, date, and brief summary of how the problem was solved
        - Code solution (add/commit periodically as needed)
        - Hint: Note how `/reverse` is handled
5. Once finished, click "Submit and Export"
6. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
    1. `git add .`
    2. `git commit -m "adding PDF"`

3. `git push origin M4-Homework`
4. On Github merge the pull request from `M4-Homework` to `main`

7. Upload the same PDF to Canvas
8. Sync Local
    1. `git checkout main`
    2. `git pull origin main`

Section #1: (3 pts.) Challenge 1 - Coin Flip

Progress: 100%

Task #1 (3 pts.) - Implement a Coin Flip Command

Progress: 100%

Details:

- `Client` must capture the user entry and generate a valid command per the lesson details
    - Command format must be `/flip`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic and send the result to everyone
    - The message must be in the format of `<who> flipped a coin and got <result>` and be from the Server
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
    - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
    - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
    - Should only need to create a new method and pass the result message to `relay()`
4. Show 5 examples of the command being seen across all terminals (2+ Clients and 1 Server)
    1. This can be captured in one screenshot if you split the terminals side by side

Client side for Part 1 edited function

Serverthread edited for Part 1

Server side edited for part 1

Outputs for Part 1

Saved: 6/16/2025 5:11:52 PM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/Client.java

URL
https://github.com/GrahamBlack1

URL #2
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/Server.java

URL
https://github.com/GrahamBlack1

URL #3
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-

URL
https://github.com/GrahamBlack1

Homework/M4/Part3HW/ServerThread.java

Saved: 6/16/2025 5:11:52 PM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenge (note: this isn't the same as `what` the code does)

Your Response:

How the code solves the problem is on the Client side. I made a flip text where, when the client puts in /flip, it will flip a coin. The server thread then handles the flip command with a case. Finally, on the server side, the function handleFlip does the action of flipping a coin and giving an output.

Saved: 6/16/2025 5:11:52 PM

Section #2: (3 pts.) Challenge 2 - Private Message

Progress: 100%

Task #1 (3 pts.) - Implement a Private Message Command

Progress: 100%

Details:

- `Client` must capture the user entry and generate a valid command per the lesson details
  - Command format must be `/pm <target id> <message>`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic
  - The message must be in the format of `PM from <who>: <message>` and be from the Server
  - The result must only be sent to the original sender and to the receiver/target
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
   - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from

2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
    - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
    - Should only need to create a new method and pass the result message to `relay()`
4. Show 3 examples of the command being seen across all terminals (3+ Clients and 1 Server)
    1. This can be captured in one screenshot if you split the terminals side by side
    2. Note: Only the sender and the receiver should see the private message (show variations across different users)

Code for client side of Part 2

code for server thread side of part 2

code for server side of Part 2

Outputs for Part 2

Saved: 6/16/2025 5:11:47 PM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/Client.java
URL
https://github.com/GrahamBlack1

URL #2
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/Server.java
URL
https://github.com/GrahamBlack1

URL #3
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/ServerThread.java
URL
https://github.com/GrahamBlack1

Saved: 6/16/2025 5:11:47 PM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenges (note: this isn't the same as `what` the code does)

Your Response:

How the code solves the challenge is that the client side finds the input of what the client puts in, and it must be in a certain format. Then, for the server thread, the code uses a case to handle the message command with an if for the length. Finally, the server side does the message action while making sure that the other user who isn't getting the message is getting it, and it also gives what the user sent and what the user receives.

Saved: 6/16/2025 5:11:47 PM

Section #3: (3 pts.) Challenge 3 - Shuffle Message

Progress: 100%

Task #1 (3 pts.) - Implement a Shuffle Message Command

Progress: 100%

Details:

- `Client` must capture the user entry and generate a valid command per the lesson details
  - Command format must be `/shuffle <message>`
- `ServerThread` must receive the data and call the correct method on `Server`
- `Server` must expose a method for the logic and send the result to everyone
  - The message must be in the format of `Shuffled from <who>: <shuffled_message>` and be from the Server
- Add code to solve the problem (add/commit as needed)

Part 1:

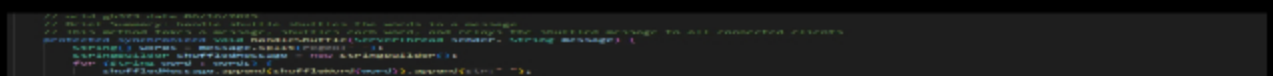
Progress: 100%

Details:
Multiple screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment) from `Client`
   - Should only need to edit `processClientCommands()`
2. Snippet of relevant code showing solution (with ucid/date comment) from `ServerThread`
   - Should only need to edit `processCommand()`
3. Snippet of relevant code showing solution (with ucid/date comment) from `Server`
   - Should only need to create a new method and do similar logic to `relay()`
4. Show 3 examples of the command being seen across all terminals (2+ Clients and 1 Server)
   1. This can be captured in one screenshot if you split the terminals side by side

Client side code for Part 3

Server Thread code for part 3



Server code for Part 3

Output for Part 3

Saved: 6/16/2025 5:11:41 PM

Part 2:

Progress: 100%

Details:
Direct link to the file in the homework related branch from Github (should end in `.java`)

URL #1
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/Client.java
URL
https://github.com/GrahamBlack1

URL #2
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/Server.java
URL
https://github.com/GrahamBlack1

URL #3
https://github.com/GrahamBlack10/gb373-IT114-450/blob/M4-Homework/M4/Part3HW/ServerThread.java
URL
https://github.com/GrahamBlack1

Saved: 6/16/2025 5:11:41 PM

Part 3:

Progress: 100%

Details:
Briefly explain `how` the code solves the challenges (note: this isn't the same as `what` the code does)

Your Response:

How the code solves the challenge is that the client-side will accept the shuffle command. Then

the server thread side will make a case that will handle the shuffle command. Finally, the server-side code does the action of the shuffle, where it takes the message, splits it into words, shuffles the words, then puts them back together, and it sends the new message. Also, the server side has code called shuffle word that is used for one-word responses, and it's able to shuffle that one word.

Saved: 6/16/2025 5:11:41 PM

Section #4: (1 pt.) Misc

Progress: 100%

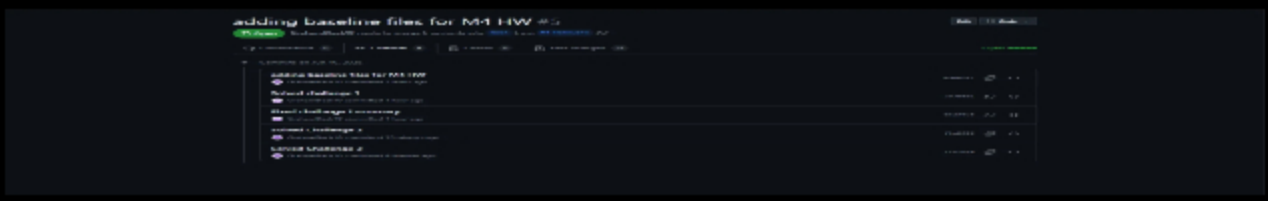
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

Details:
From the Commits tab of the Pull Request screenshot the commit history Following minimum should be present



Commit History

Saved: 6/16/2025 5:00:00 PM

Part 2:

Progress: 100%

Details:
Include the link to the Pull Request (should end in `/pull/#`)

URL #1
https://github.com/GrahamBlack10/gb373-IT114-450/pull/5

URL
https://github.com/GrahamBlack1

Saved: 6/16/2025 5:00:00 PM

Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

Progress: 100%

Details:

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary

Overall Time

Individual Time

Saved: 6/16/2025 5:00:49 PM

Task #3 (0.33 pts.) - Reflection

Progress: 100%

Task #1 (0.33 pts.) - What did you learn?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

I learned a lot about the client and server. I knew a lot about it from IT490, and this is a simpler version of that. I found it interesting about the ways that it needs a server thread to connect to each other. I also learned about different commands that a client can use to get a response and how the server will hold the action code for what that certain command does.

Saved: 6/16/2025 5:03:51 PM

Task #2 (0.33 pts.) - What was the easiest part of the assignment?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

Once I understood how the connection worked and where to put the code, most of the code was easy. I found it easy to create the cases in the server thread, and also the coding for the client side was pretty easy. For each part, parts 1 and 3 worked well, and I was able to finish those with pretty much no issues.

Saved: 6/16/2025 5:05:36 PM

Task #3 (0.33 pts.) - What was the hardest part of the assignment?

Progress: 100%

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

I found the hard part was getting started. I ran into issues of where to start and how to run the code, but after looking at the videos on Canvas, I was able to understand this part of my issue. I ran into an issue with part 2. In part 2, I ran into an issue where the code didn't run at the start and didn't give an output. Once I found the solution, the next issue I ran into was getting it so that the private message to only send to the users that were involved, which was the sender and receiver. The solution was defining the target user and sender, which made it output correctly.

Saved: 6/16/2025 5:10:04 PM