Submission Worksheet

Submission Data

Course: IT114-450-M2025
Assignment: IT114 Module 4 Sockets Part3 Challenge
Student: Anthony L. (agl8)
Status: In Progress | **Worksheet Progress:** 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Started: 8/11/2025 2:15:09 AM
Updated: 8/11/2025 2:14:30 PM
Grading Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/grading/agl8
View Link: https://learn.ethereallab.app/assignment/v3/IT114-450-M2025/it114-module-4-sockets-part3-challenge/view/agl8

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

```
//agl8, 8-9-25
//flip command
else if ("/flip".equalsIgnoreCase(text)){
        String [] commandData = {Constants.COMMAND_TRIGGER, "flip"};
        sendToServer(String.join(delimiter:",",commandData));
        wasCommand = true;
```

```
} <- #127-131 else if ("/flip".equalsIgnoreCase(text))
```

code part 1

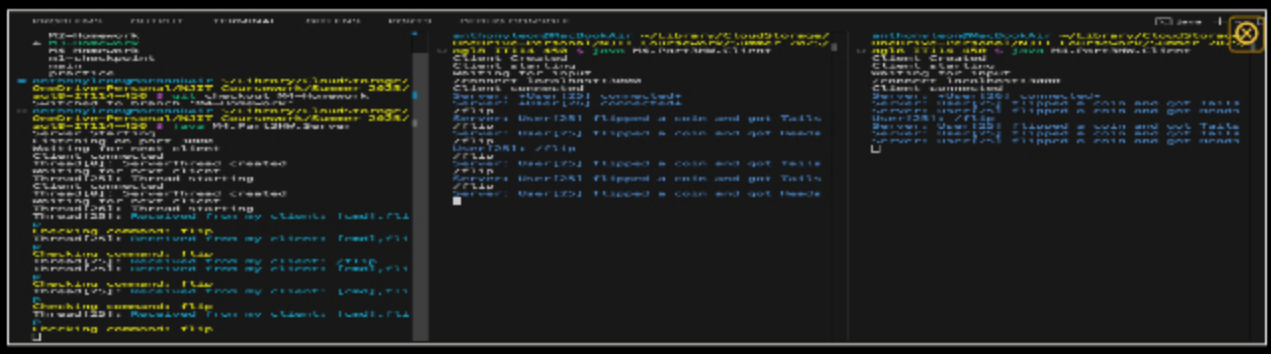
```java
//agl8- 8-9-25
public synchronized void handleFlip(ServerThread sender) {
    String result = (Math.random() < 0.5) ? "Heads" : "Tails";
    long clientId = sender.getClientId();
    String message = String.format(format:"User[%d] flipped a coin and got %s", clientId, result);
    relay(sender:null, message);
} <- #134-139 public synchronized void handleFlip(ServerThread sender)

protected synchronized void handleMessage(ServerThread sender, String text) {
    relay(sender, text);
}
```

code part 2

```java
//agl8, 8-9-25
//case flip
case "flip":
server.handleFlip(this);
wasCommand = true;
break;
// added more cases/breaks as needed for other commands
```

code part 3



output

Saved: 8/11/2025 1:59:52 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/agl8-2025/agl8-IT114-450/M4-Homework/M4/Part3HW/Server.java

URL
https://github.com/agl8-2025/agl

URL #2
https://github.com/agl8-2025/agl8-IT114-450/M4-Homework/M4/Part3HW/ServerThread.java

URL
https://github.com/agl8-2025/agl

URL #3
https://github.com/agl8-2025/agl8-IT114-450/M4-Homework/M4/Part3HW/Client.java

URL
https://github.com/agl8-2025/agl

Saved: 8/11/2025 1:59:52 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:

Client sends the flip command to the ServerThread. ServerThread then passes it to the handleFlip method on the Server. Server picks heads or tails and then relays it to everyone connected.

Saved: 8/11/2025 1:59:52 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:

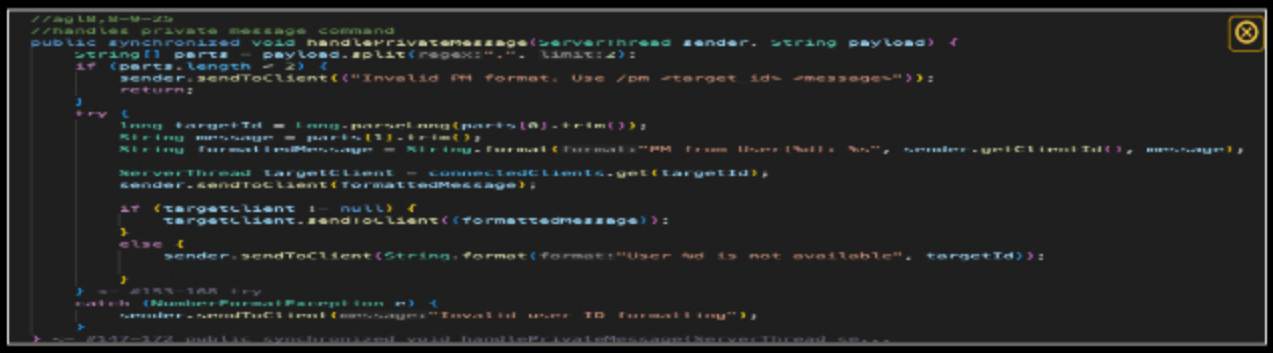
Progress: 100%

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 `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 send the result message to just the sender and receiver
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)

```
//agl8, 8-9-25
//checks for pm and sends to server
else if (text.startsWith(prefix:"/pm ")) {
    String[] pmData = text.substring("/pm ".length()).split(regex:" ", limit:2);
    if (pmData.length == 2) {
        String[] commandData = { Constants.COMMAND_TRIGGER, "pm", pmData[0], pmData[1] };
        sendToServer(String.join(delimiter:",", commandData));
    } else {
        System.out.println(x:"Invalid format. Use /pm <target_id> <message>");
    }
    wasCommand = true;
} <- #134-143 else if (text.startsWith("/pm "))
```

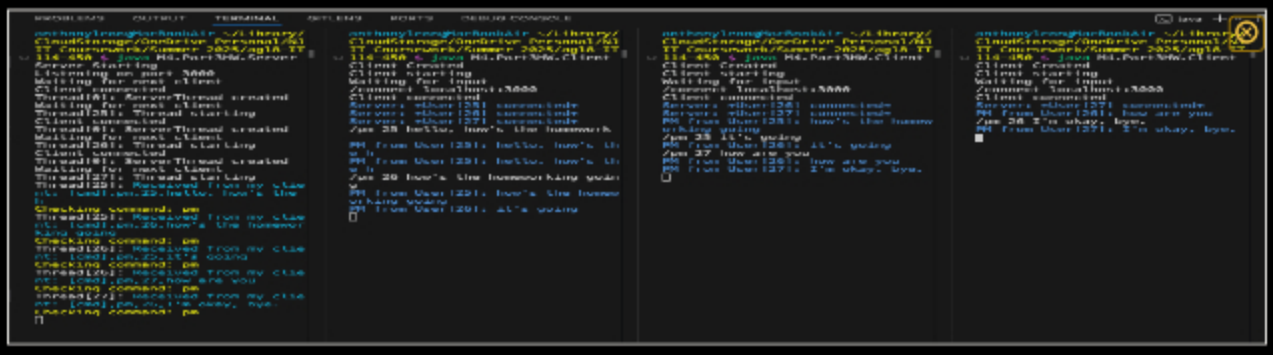
code part 1



code part 2

```
//agl8, 8-9-25
//catches pm
case "pm":
String pmDetails = String.join(delimiter:",", Arrays.copyOfRange(commandData, from:2, commandData.length));
server.handlePrivateMessage(this, pmDetails);
wasCommand = true;
break;
```

code part 3



output

Saved: 8/11/2025 2:02:32 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/agl8-2025/agl8-IT114-[redacted]-Homework/M4/Part3HW/Server.java

URL
https://github.com/agl8-2025/agl

URL #2
https://github.com/agl8-2025/agl8-IT114-[redacted]-Homework/M4/Part3HW/ServerThread.java

URL
https://github.com/agl8-2025/agl

URL #3
https://github.com/agl8-2025/agl8-IT114-[redacted]-Homework/M4/Part3HW/Client.java

URL
https://github.com/agl8-2025/agl

Saved: 8/11/2025 2:02:32 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:

Client sends the pm command with an ID and message to the ServerThread. ServerThread then passes it to a handlePrivateMessage method on the Server. Server finds the correct sender and receiver and sends the message only to them.

Saved: 8/11/2025 2:02:32 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

```
//agl8, 8-10-25
//checks for shuffle and send to server
else if (text.startsWith(prefix:"/shuffle ")) {
    String message = text.substring("/shuffle ".length());
    String[] commandData = {Constants.COMMAND_TRIGGER, "shuffle", message};
    sendToServer(String.join(delimiter:",", commandData));
    wasCommand = true;
} <- #146-151 else if (text.startsWith("/shuffle "))
return wasCommand;
<- #101-153 private boolean processClientCommand(String text) throws IOEx...
```

code part 1

```java
    //agl8,8-9-25
    //handles shuffle        You, 3 hours ago • solved ShuffleCommand
public synchronized void handleShuffle(ServerThread sender, String text) {
    List<Character> characters = new ArrayList<>();
    for (char c : text.toCharArray()) {
        characters.add(c);
    }
    Collections.shuffle(characters);

    StringBuilder shuffledText = new StringBuilder();
    for (char c : characters) {
        shuffledText.append (c);
    }
    String finalMessage = String.format(format:"Shuffled from User[%d]: %s", sender.getClientId(), shuffledText.toString());
    relay(sender:null, finalMessage);
} <- #176-189 public synchronized void handleShuffle(ServerThread sender, S...
// end handle actions
```

code part 2

```java
//agl8, 8-10-25
//handles shuffle
case "shuffle":
String textToShuffle = String.join(delimiter:",", Arrays.copyOfRange(commandData, from:2, commandData.length));
server.handleShuffle(this, textToShuffle);
wasCommand = true;
break;

default:
    break;
<- #190-230 switch (command)
```

code part 3

output

Saved: 8/11/2025 2:04:42 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/agl8-2025/agl8-IT114-450/M4-Homework/M4/Part3HW/Server.java

URL
https://github.com/agl8-2025/agl

URL #2

URL
https://github.com/agl8-2025/agl

IT11[redacted]M4-Homework/M4/Part3HW/ServerThread.java

URL #3
https://github.com/agl8-2025/agl8-IT11[redacted]M4-Homework/M4/Part3HW/Client.java

URL
https://github.com/agl8-2025/agl

Saved: 8/11/2025 2:04:42 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:

Client sends the shuffle command and a message to the ServerThread. This gets passed to a handleShuffle method on the Server. Server then shuffles the message and sends the result to all clients.

Saved: 8/11/2025 2:04:42 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

commits

Saved: 8/11/2025 2:39:56 AM

Part 2:

Progress: 100%

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

URL #1
https://github.com/agl8-2025/agl8-IT114-450/

URL
https://github.com/agl8-2025/agl

Saved: 8/11/2025 2:39:56 AM

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

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

waka top

waka bottom

Saved: 8/11/2025 2:40:42 AM

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 how a client server chat program works. This included using sockets to connect them and having the server manage multiple client chats at once.

Saved: 8/11/2025 2:08:11 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:

The easiest part was the coin flip command. The logic was simple because it didn't need any extra arguments from the user, and it just had to send the result to everyone.

Saved: 8/11/2025 2:13:47 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:

The hardest part was the private message command. Figuring out how to get the target user's ID and send the message to only them and the sender was confusing.

Saved: 8/11/2025 2:14:30 PM