Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-450-M2024/it114-module-4-sockets-part-1-3/grade/yk686

IT114-450-M2024 - [IT114] Module 4 Sockets Part 1-3

Submissions:

Submission Selection

1 Submission [active] 6/19/2024 8:35:23 PM

Instructions

^ COLLAPSE ^

Overview Video: https://youtu.be/5a5HL0n6jek

- Create a new branch for this assignment
- If you haven't, go through the socket lessons and get each part implemented (parts 1-3)
 - You'll probably want to put them into their own separate folders/packages (i.e., Part1, Part2, Part3) These are for your reference
- Part 3, below, is what's necessary for this HW
 - 3. https://github.com/MattToegel/IT114/tree/M24-Sockets-Part3
- Create a new folder called Part3HW (copy of Part3)
- Make sure you have all the necessary files from Part3 copied here and fix the package references at the top of each file
 - Add/commit/push the branch
 - 2. Create a pull request to main and keep it open
- Implement two of the following server-side activities for all connected clients (majority of the logic should be processed server-side and broadcasted/sent to all clients if/when applicable)
 - 1. Simple number guesser where all clients can attempt to guess while the game is active
 - Have a /start command that activates the game allowing guesses to be interpreted
 - Have a /stop command that deactivates the game, guesses will be treated as regular messages (i.e., guess messages are ignored)
 - 3. Have a /guess command that include a value that is processed to see if it matches the hidden number (i.e., /guess 5)
 - Guess should only be considered when the game is active
 - The response should include who guessed, what they guessed, and whether or not it was correct (i.e., Bob guessed 5 but it was not correct)
 - No need to implement complexities like strikes
 - 2. Coin toss command (random heads or tails)

- Command should be something logical like /flip or /toss or /coin or similar
- 2. The result should mention who did what and got what result (i.e., Bob Flipped a coin and got heads)
- 3. Dice roller given a command and text format of "/roll #d#" (i.e., /roll 2d6)
 - Command should be in the format of /roll #d# (i.e., /roll 1d10)
 - 2. The result should mention who did what and got what result (i.e., Bob rolled 1d10 and got 7)
- Math game (server outputs a basic equation, first person to guess it correctly gets congratulated and a new equation is given)
 - 1. Have a /start command that activates the game allowing equaiton to be answered
 - Have a /stop command that deactivates the game, answers will be treated as regular messages (i.e., any game related commands when stopped will be ignored)
 - Have an answer command that include a value that is processed to see if it matches the hidden number (i.e., /answer 15)
 - The response should include who answered, what they answered, and whether or not it was correct (i.e., Bob answered 5 but it was not correct)
- Private message (a client can send a message targetting another client where only the two can see the messages)
 - Command can be /pm, /dm followed by the user's name or an @ preceding the users name (clearly note which)
 - The server should properly check the target audience and send the response to the original sender and to the receiver (no one else should get the message)
 - 3. Alternatively (make note if you do this and show evidence) you can add support to private message multiple people at once. Evidence should show a larger number of clients than the target list of the private message to show it works. Note to grader: if this is accomplished add 0.5 to total final grade on Canvas
- 6. Message shuffler (randomizes the order of the characters of the given message)
 - Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)
 - The message should be sent to all clients showing it's from the user but randomized
 - 1. Example: Bob types / command hello and everyone recevies Bob: Ileho
- 7. Fill in the below deliverables
- 8. Save the submission and generated output PDF
- Add the PDF to the Part3HW folder (local)
- Add/commit/push your changes
- Merge the pull request
- 12. Upload the same PDF to Canvas

Branch name: M4-Sockets3-Homework

Tasks: 6 Points: 10.00





Task #1 - Points: 1

Text: Demonstrate Baseline Code Working

Details:

This can be a single screenshot if everything fits, or can be multiple screenshots

#1) Show and clearly note which terminal is the Server







#2) Show and clearly note which terminals are the client



Caption (required) <

Describe/highlight what's being shown
Showing the left most terminal s the server

Caption (required) <

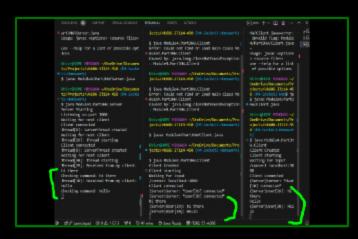
Describe/highlight what's being shown
Showing the middle and right are the clients

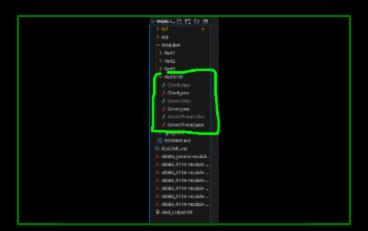
#3) Show all clients receiving the broadcasted/relayed messages



#4) Include a screenshot showing you grabbed Parts 1-3 correctly and have them in your repository alongside Part3HW







Describe/highlight what's being shown
Sowing messages work

Describe/highlight what's being shown
Showing parts 1-3 and their files



Feature 1 (3 pts.)

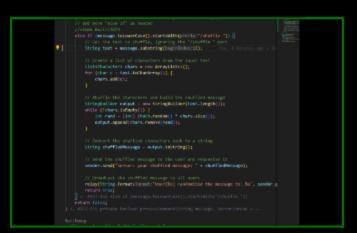


Task #1 - Points: 1

Text: Solution

#1) Show the code related to the feature (ucid and date must be present as a comment)





Caption (required) <

Describe/highlight what's being shown
Showing Server.java file where my code is

Explanation (required) ~

Mention specific feature and explain sufficiently and concisely the implementation (should be aligned with code snippets)

PREVIEW RESPONSE

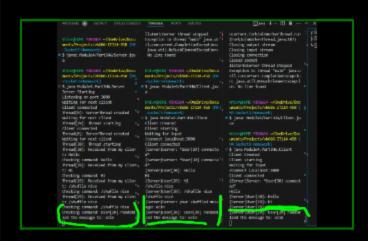
 Message shuffler (randomizes the order of the characters of the given message)
 Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)

The message should be sent to all clients showing it's from the user but randomized Example: Bob types /command hello and everyone recevies Bob: Ileho

When a user sends the command /shuffle

#2) Show the feature working (i.e., all terminals and their related output)





Caption (required) <

Describe/highlight what's being shown
Showing the feature working

nice,the server identifies the command and extracts the text. Converts "nice" into a list of characters and shuffles these characters to create a new string.

while (!chars.isEmpty()) {
 int rand = (int) (Math.random() * chars.size());
 output.append(chars.remove(rand));
}
The code repeatedly selects a random character from the list, removes it, and adds it to a new string by using StringBuilder. This continues until all characters are used,



Feature 2 (3 pts.)

resulting in a shuffled message.



Task #1 - Points: 1

Text: Solution

#1) Show the code related to the feature



#2) Show the feature working (i.e., all terminals and their





Caption (required)

Describe/highlight what's being shown

Missing caption



Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Mention specific feature and explain sufficiently and concisely the implementation (should be aligned with code snippets)s

PREVIEW RESPONSE

Missing tex



Misc (2 pts.)



Task #1 - Points: 1

Text: Reflection

#1) Learn anything new? Face any challenges? How did you overcome any issues?



Explanation (required) <

Provide at least a few logical sentences



I didn't really encounter any bugs or problems.

Everything that was implemented was quite simple and although some things might not work right away. But then I managed to fix it. I needed more time to complete all the tasks completely. It was a very interesting task.



Task #2 - Points: 1

Text: Pull request link



URL should end with /pull/# and be related to this assignment

URL #1

https://github.com/VK686NJ/vk686-IT114-450/pull/8



Task #3 - Points: 1

Text: Waka Time (or related) Screenshot



Screenshot clearly shows what files/project were being worked on (the duration of time doesn't correlated with the grade for this item)

Task Screenshots:

Gallery Style: Large View Small Medium Large Projects • vk686-IT114-***** * total 6 hrs 36 mins 450 2 hrs 37 mins over the Last 7 Days in vk686-IT114-450 under all branches. 🖎 How long it took to get it done

How long it took to get it done

End of Assignment