Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-sockets-part-1-3-checkpoint/grade/zb64

IT114-004-S2024 - [IT114] Sockets Part 1-3-Checkpoint

Submissions:

Submission Selection

1 Submission [active] 2/24/2024 10:01:29 PM

1.

1. Instructions

2.

COLLAPSE

Create a new branch for this assignment
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
    https://github.com/MattToegel/IT114/tree/Module4/Module4/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
    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)
    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)
        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
    Coin toss command (random heads or tails)
        Command should be something logical like /flip or /toss or /coin or similar
        The result should mention *who* did *what* and got what *result* (i.e., Bob Flipped a coin and got heads)
    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)
        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)
        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 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)
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
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: lleho
Fill in the below deliverables
Save the submission and generated output PDF
Add the PDF to the Part3HW folder (local)
Add/commit/push your changes
Merge the pull request
Upload the same PDF to Canvas

Branch name: M4-Sockets3-Homework

Tasks: 7 Points: 10.00

Baseline (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Demonstrate Baseline Code Working

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

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Server terminal/instance is clearly shown/noted
#2	1	At least 3 client terminals should be visible and noted
#3	1	Each client should correctly receive all broadcasted/shared messages
	1	Captions clearly explain what each screenshot is showing

#4	1	Captions clearly explain what each screenshot is showing
#5	1	Include a screenshot showing you grabbed Parts 1-3 correctly and have them in your repository alongside Part3HW

Task Screenshots:

Gallery Style: Large View

Small Medium Large

This picture shows that on the left there is a folder named Part3HW will parts 1-3 added. This picture also shows at the bottom that there are at least 3 clients and 1 server connected in the terminal.

Checklist Items (5)

#1 Server terminal/instance is clearly shown/noted

#2 At least 3 client terminals should be visible and noted

#3 Each client should correctly receive all broadcasted/shared messages

#4 Captions clearly explain what each screenshot is showing

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

Feature 1 (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: What feature did you pick? Briefly explain how you implemented it

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Feature is clearly stated (best to copy/paste it from above)
#2	1	Explanation sufficiently and concisely describes implementation (should be aligned with code snippets in related task)

Response:

Coin toss command (random heads or tails)
Command should be something logical like /flip or /toss or /coin or similar
The result should mention *who* did *what* and got what *result* (i.e., Bob Flipped a coin and got heads)

I started working on the proccessCommand method by adding a statement of when the client types the word "flip" followed by making a message to confirm that the client wanted to flip a coin in the server—followed by using Random random = new Random() to generate between two integers. Then stating that if the variable results = 0 as its first integer the new object message would print out that the user has flipped heads, and if the generator lands on a 1 then the message would print out the user flipped tails—followed by a string that formats the final message to include the client ID and the outcome of the coin flip. Then I followed the line of codes to iterate the loop through the clients collection.

COLLAPSE

Task #2 - Points: 1

Text: Add screenshot(s) showing the implemented feature working (code and output)

Details:
Add screenshots of the relevant code changes AND relevant output during runtime

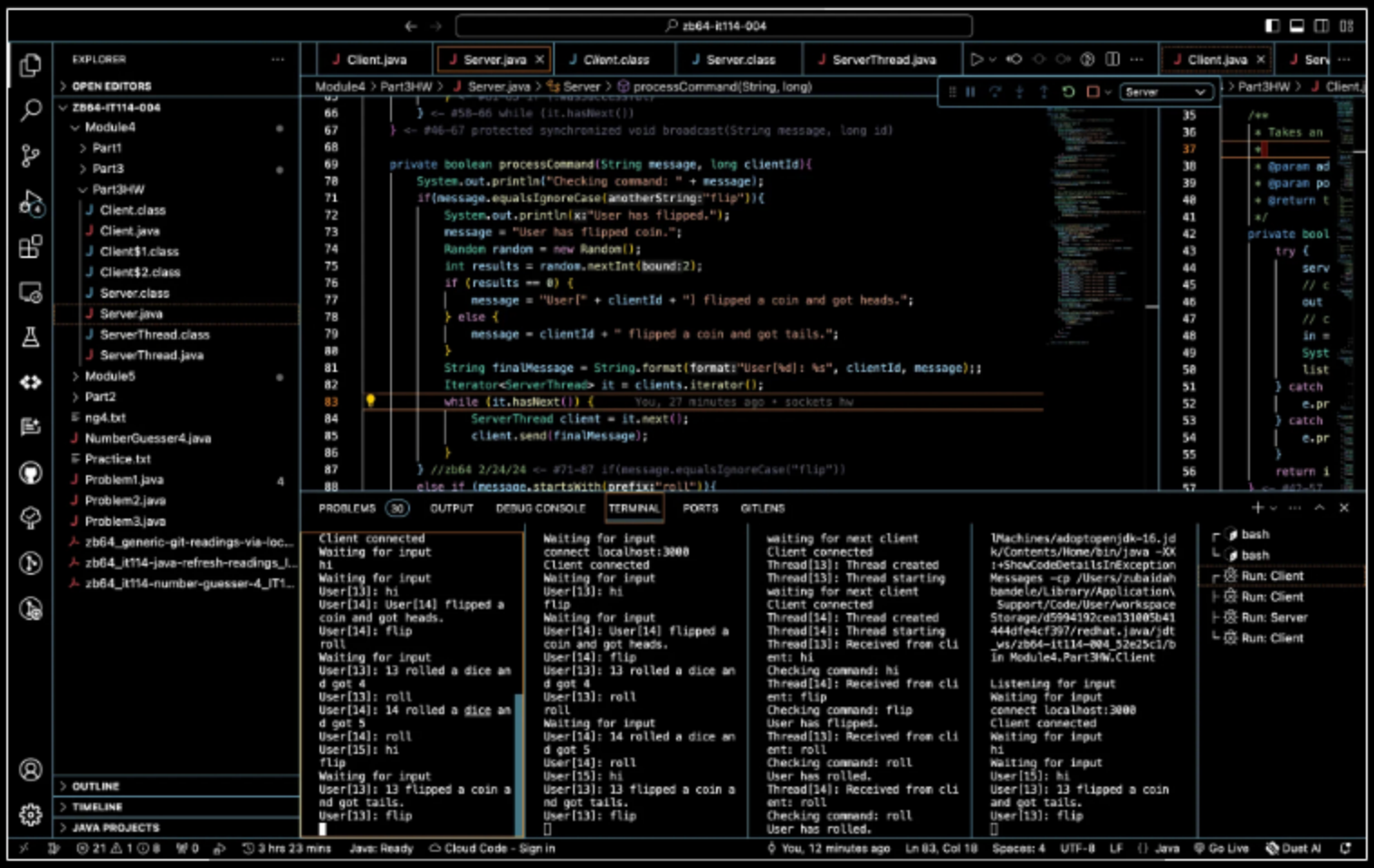
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Output is clearly shown and captioned
#2	1	Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

Task Screenshots:

Gallery Style: Large View

Small Medium Large



At the top is the code for the coin command and at the bottom is the output once the client types flip in the terminal.

Checklist Items (2)

#1 Output is clearly shown and captioned

#2 Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

Feature 2 (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: What feature did you pick? Briefly explain how you implemented it

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Feature is clearly stated (best to copy/paste it from above)
#2	1	Explanation sufficiently and concisely describes implementation (should be aligned with code snippets in related task)

Response:

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)
The result should mention *who* did *what* and got what *result* (i.e., Bob rolled 1d10 and got 7)

I replicated the process used for the 'flip' command, but this time, I utilized the command 'roll'. After a message to confirm that the client wanted to roll a die in the server —followed by using Random random = new Random() to generate This involved generating a random number between 1 and 6 using Random in between there would be a message that prints out that the user has rolled the number that was randomly generated. Followed by a string that formats the final message to include the clientID and the outcome of the rolled dice. Then I followed the iterator to loop through the clients collection from disconnected case that was already in the server.java.

COLLAPSE

Task #2 - Points: 1

Text: Add screenshot(s) showing the implemented feature working (code and output)

Details:
Add screenshots of the relevant code changes AND relevant output during runtime

Checklist

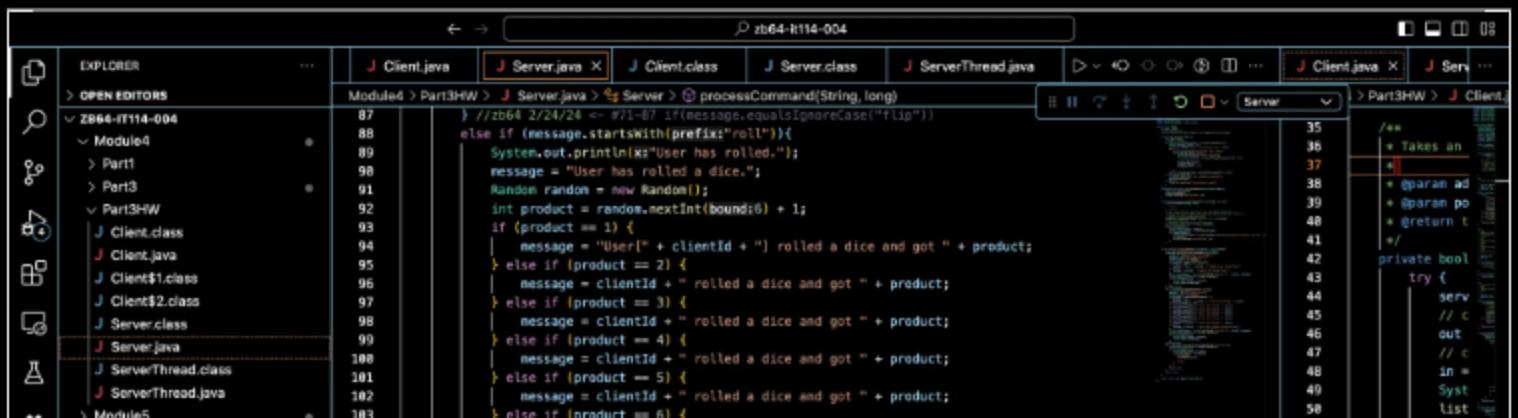
*The checkboxes are for your own tracking

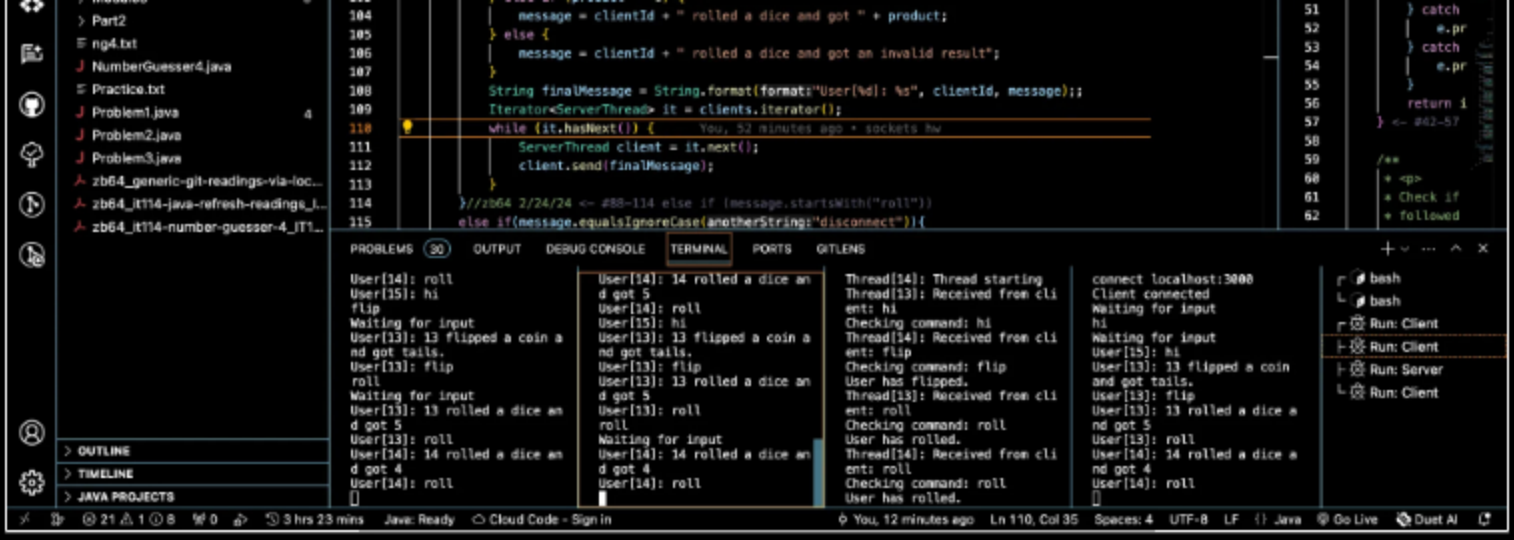
#	Points	Details
#1	1	Output is clearly shown and captioned
#2	1	Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

Task Screenshots:

Gallery Style: Large View

Small Medium Large





At the top of the picture is my added code and at the bottom is the output once the client types roll in the terminal.

Checklist Items (2)

#1 Output is clearly shown and captioned

#2 Code shows relevant snippets that accomplish feature, UCID and date are present in all code screenshots. Relevant captions are included for each screenshot of the code.

Misc (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Reflection: Did you have an issues and how did you resolve them? If no issues, what did you learn during this assignment that you found interesting?

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	An issue or learning is clearly stated
#2	1	Response is a few reasonable sentences

Response:

I encountered a challenge when deciding how to incorporate the new code into the existing server.java file. Initially, I was unsure about which method within server.java would be the appropriate location for the code, ensuring visibility of the server-client interactions in the terminal. To resolve this uncertainty, I fixed this issue by reading line by line of code and reading what each line meant and its function. I realized that the code for the coin and dice commands should be integrated within the processCommand method.

COLLAPSE

Task #2 - Points: 1

Text: Pull request link

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

URL #1
Missing URL

End of Assignment