Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-sockets-part-1-3-checkpoint/grade/mth39

IT114-002-S2024 - [IT114] Sockets Part 1-3-Checkpoint

Submissions:

Submission Selection

1 Submission [active] 2/19/2024 7:42:40 PM

Instructions

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

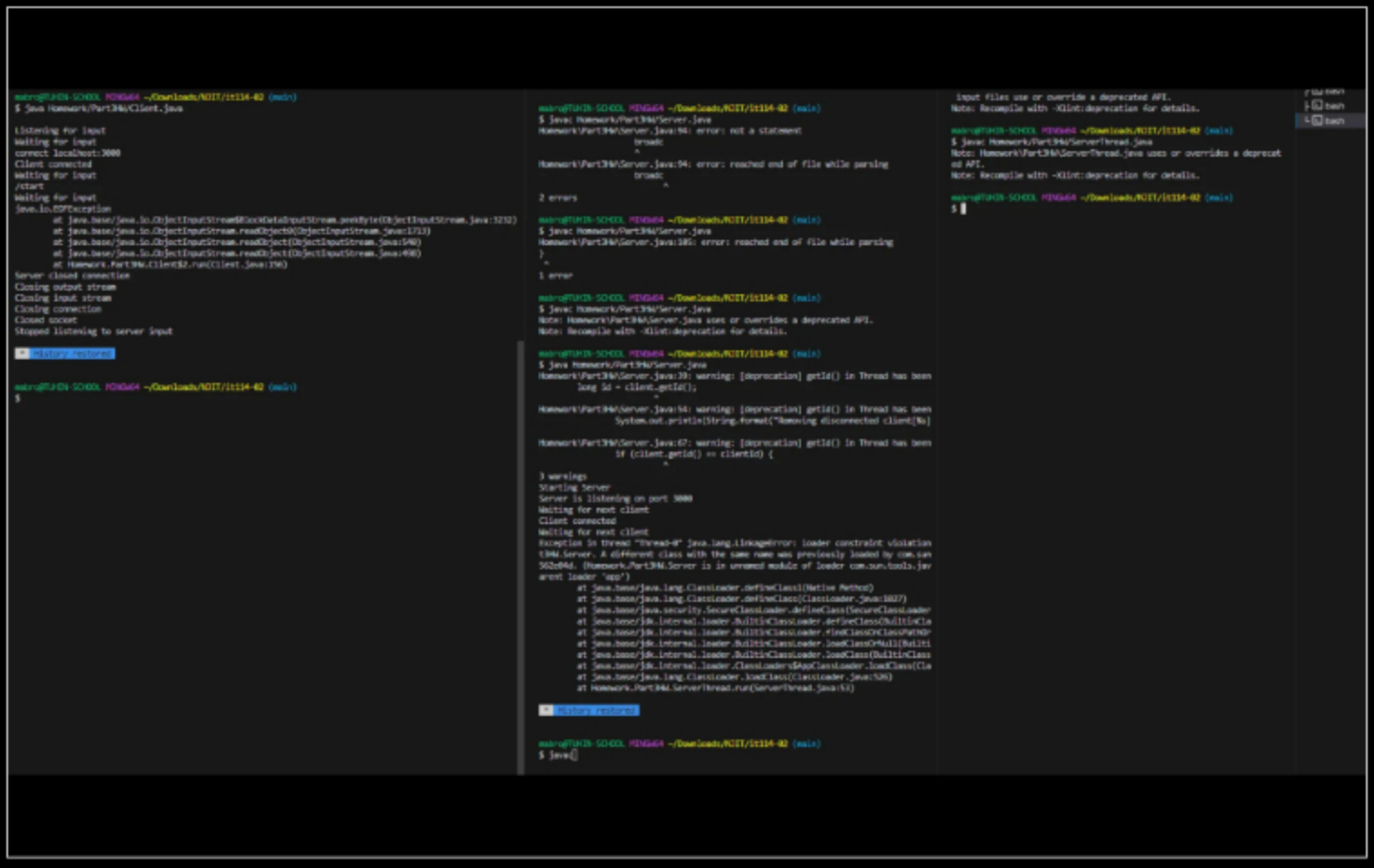
*The checkboxes are for your own tracking

#	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
☐ #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



Screenshot showing client/server terminal output

Checklist Items (0)

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:

The feature I picked was the simple number guesser. I modified the Server class to handle the start, stop, and guess commands. When the game is active, guesses will be checked against a hidden number. The response will be broadcasted to all clients indicating who guessed, what they guessed, and whether it was correct. I added a boolean

variable to track the state of the game (active or inactive). When /start command is received, the game starts, and a random number is generated. When /stop command is received, the game stops, and no more guesses are accepted. When a guess command (/guess) is received and the game is active, it processes the guess and broadcasts whether it's correct or not.

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

```
61      private boolean processCommand(String message, long clientId) {
62          System.out.println("Checking command: " + message);
63          if (message.equalsIgnoreCase("/disconnect")) {
64              Iterator<ServerThread> it = clients.iterator();
65              while (it.hasNext()) {
66                  ServerThread client = it.next();
67                  if (client.getId() == clientId) {      You, 3 hours ago • finished java refresh readings part 2
68                      it.remove();
69                      disconnect(client);
70                      break;
71                  } <- #67-71 if (client.getId() == clientId)
72              } <- #65-72 while (it.hasNext())
73              return true;
74          } else if (message.equalsIgnoreCase("/start")) {
75              if (!gameActive) {
76                  gameActive = true;
77                  // Generate a random number between 1 and 100 as the hidden number
78                  hiddenNumber = new Random().nextInt(100) + 1;
79                  broadcast("Number guesser game started! Guess a number between 1 and 100.", clientId);
80              } <- #75-80 if (!gameActive)
81              return true;
82          } else if (message.equalsIgnoreCase("/stop")) {
83              if (gameActive) {
84                  gameActive = false;
85                  broadcast("Number guesser game stopped. Guesses will be treated as regular messages.", clientId);
86              }
87              return true;
88          } else if (message.toLowerCase().startsWith("/guess")) {
89              if (gameActive) {
90                  try {
91                      int guess = Integer.parseInt(message.substring(7).trim());
92                      String guessResult = (guess == hiddenNumber) ? "correct!" : "incorrect.";
93                      broadcast("User[" + clientId + "] guessed " + guess + ", which was " + guessResult, clientId);
94                  } catch (NumberFormatException e) {
95                      broadcast("Invalid guess format. Please provide a number.", clientId);
96                  }
97              } else {
98                  broadcast("Game is not active. Please start the game first.", clientId);
```

```
            return true;
        } else if (message.equalsIgnoreCase("/flip") || message.equalsIgnoreCase("/toss") || message.equalsIgnoreCase("/coin")) {
            flipCoin(clientId);
            return true;
        }
        return false;
    } <- #61-106 private boolean processCommand(String message, long clientId)
```

Screenshot showing the coding of the simple number guesser option

Checklist Items (0)

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:

The processCommand method checks if the message starts with /flip. If it does, it calls the flipCoin method, which generates a random result (heads or tails) and broadcasts the result to all clients. Finally, it returns true to indicate that the command has been processed.

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

```java
    private void flipCoin(long clientId) {
        // Perform a coin toss
        String result = (Math.random() < 0.5) ? "Heads" : "Tails";
        broadcast("User[" + clientId + "] flipped a coin and got " + result, clientId);
    } <- #108-112 private void flipCoin(long clientId)

    public static void main(String[] args) {
        System.out.println("Starting Server");
        Server server = new Server();
        int port = 3000;
        try {
            port = Integer.parseInt(args[0]);
        } catch (Exception e) {
            // Ignore - will default to the defined value prior to the try/catch
        }
        server.start(port);
        System.out.println("Server Stopped");
    } <- #114-125 public static void main(String[] args)
} <- #11-126 public class Server
```

Screenshot showing the completion of coin flip option

Checklist Items (0)

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 got a 'java.lang.LinkageError' when trying to input something in client for some reason. I tried going over my code to find out what's wrong but found nothing.

COLLAPSE

Task #2 - Points: 1

Text: Pull request link

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

URL #1
https://github.com/MabroorHussan/it114-02/pull/6

End of Assignment