Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-003-F2024/it114-module-4-sockets-part-1-3/grade/js2637

Course: IT114-003-F2024
Assigment: [IT114] Module 4 Sockets Part 1-3
Student: Jack S. (js2637)

Submissions:

Submission Selection

1 Submission [submitted] 10/6/2024 4:40:26 PM

Instructions

COLLAPSE

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

1. Create a new branch for this assignment
2. If you haven't, go through the socket lessons and get each part implemented (parts 1-3)
    1. You'll probably want to put them into their own separate folders/packages (i.e., Part1, Part2, Part3) These are for your reference
3. Part 3, below, is what's necessary for this HW
    3. https://github.com/MattToegel/IT114/tree/M24-Sockets-Part3
4. Create a new folder called Part3HW (copy of Part3)
5. Make sure you have all the necessary files from Part3 copied here and fix the package references at the top of each file
    1. Add/commit/push the branch
    2. Create a pull request to main and keep it open
6. 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
        1. Have a /start command that activates the game allowing guesses to be interpreted
        2. 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*)
            1. Guess should only be considered when the game is active
            2. 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)
            3. No need to implement complexities like strikes

2. Coin toss command (random heads or tails)
    1. 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)
    1. 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)
4. 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
    2. 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)
    3. Have an answer command that include a value that is processed to see if it matches the hidden number (i.e., */answer 15*)
        1. 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)
5. Private message (a client can send a message targetting another client where only the two can see the messages)
    1. Command can be /pm, /dm followed by the user's name or an @ preceding the users name (clearly note which)
    2. 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)
    1. Command should be /shuffle or /randomize (clearly mention what you chose) followed by the message to shuffle (i.e., /shuffle hello everybody)
    2. 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: lleho

7. Fill in the below deliverables
8. Save the submission and generated output PDF
9. Add the PDF to the Part3HW folder (local)
10. Add/commit/push your changes
11. Merge the pull request
12. Upload the same PDF to Canvas

Branch name: M4-Sockets3-Homework

Group

100%

Group: Baseline
Tasks: 1
Points: 2

COLLAPSE

Task

100%

Group: Baseline
Task #1: Demonstrate Baseline Code Working
Weight: ~100%
Points: ~2.00

COLLAPSE

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

Columns: 1

Sub-Task

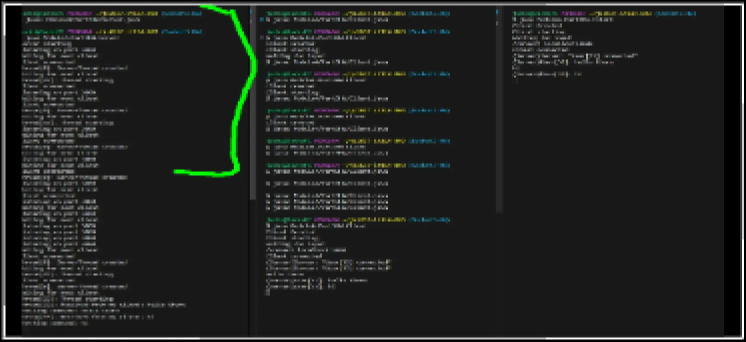
100%

Group: Baseline
Task #1: Demonstrate Baseline Code Working
Sub Task #1: Show and clearly note which terminal is the Server

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
[illegible]
```

Left Terminal is the server

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

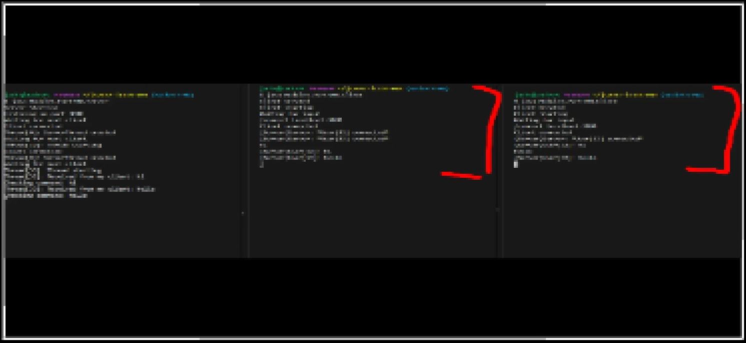
100%

Group: Baseline
Task #1: Demonstrate Baseline Code Working
Sub Task #2: Show and clearly note which terminals are the client

Task Screenshots

Gallery Style: 2 Columns

4 2 1



The client is the right and middle terminal

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Baseline

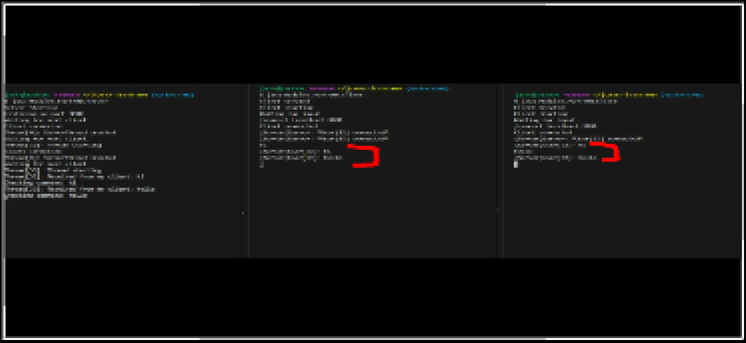
Task #1: Demonstrate Baseline Code Working

Sub Task #3: Show all clients receiving the broadcasted/relayed messages

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Showing Messages

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Baseline

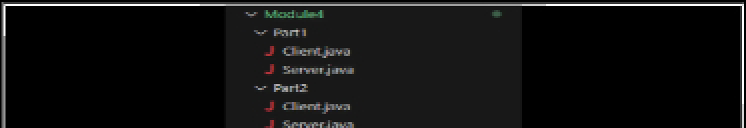
Task #1: Demonstrate Baseline Code Working

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

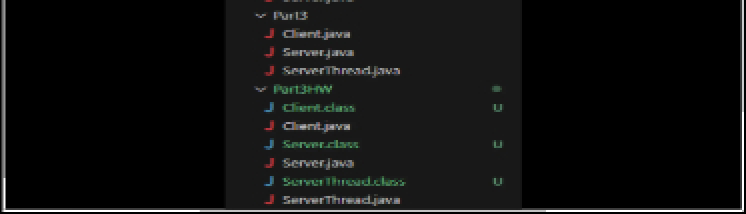
Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
Module4
  Part1
    Client.java
    Server.java
  Part2
    Client.java
    Server.java
```



```
Part3
  Client.java
  Server.java
  ServerThread.java
Part3HW
  Client.class        U
  Client.java
  Server.class        U
  Server.java
  ServerThread.class  U
  ServerThread.java
```

Showing part1-3

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 1

End of Group: Baseline

Task Status: 1/1

Group

100%

Group: Feature 1

Tasks: 1

Points: 3

COLLAPSE

Task

100%

Group: Feature 1

Task #1: Solution

Weight: ~100%

Points: ~3.00

COLLAPSE

Columns: 1

Sub-Task

100%

Group: Feature 1

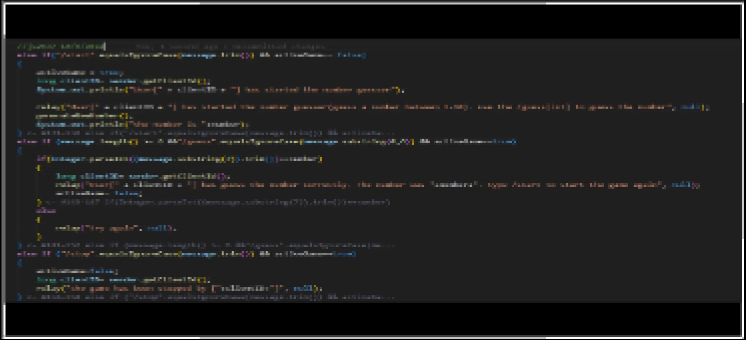
Task #1: Solution

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
//[illegible]
else if("/start".equalsIgnoreCase(message.trim()) && activeGame== false)
{
    activeGame = true;
    long clientID= sender.getClientId();
    System.out.println("User[" + clientID + "] has started the number guesser");

    relay("User[" + clientID + "] has started the number guesser[illegible]", null);
    generateRandomNumber();
    System.out.println("the number is "+number);
} [illegible]
else if (message.length() >= 6 && "/guess".equalsIgnoreCase(message.substring(0,6)) && activeGame==true)
{
    if(Integer.parseInt((message.substring(7)).trim())==number)
    {
        long clientID= sender.getClientId();
        relay("User[" + clientID + "] has guess the number correctly. the number was "+number+". type /start to start the game again", null);
        activeGame= false;
    } [illegible]
    else
    {
        relay("try again", null);
    }
} [illegible]
else if ("/stop".equalsIgnoreCase(message.trim()) && activeGame==true)
{
    activeGame=false;
    long clientID= sender.getClientId();
    relay("the game has been stopped by ["+clientID+"]", null);
} [illegible]
```

code for the number quesser

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

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

Response:

I used some of the code from last week's homework with the number guesser to generate the Number and then I used a basic if statement to look for the commands, for the /guess command I had to use the str. substring method so that the user can type their guess with a number on the same line. the if statement grabs just the command and the the code inside the if statement grabs the int and compares it to the random number generated in the /start command. I also added a game state so that the /guess,/start, and /stop commands would only work when appropriate.

Sub-Task

100%

Group: Feature 1

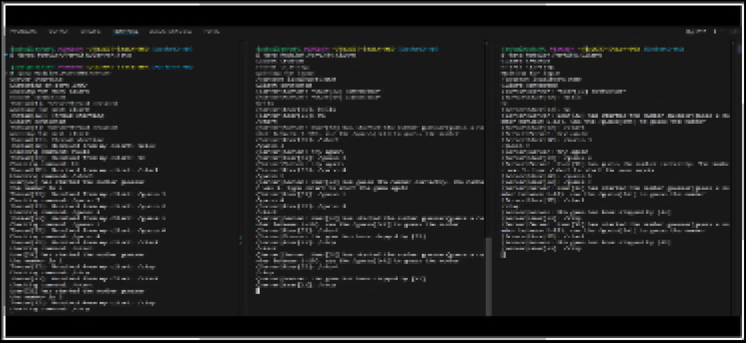
Task #1: Solution

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
[illegible]
```

number guesser terminal

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 1

End of Group: Feature 1

Task Status: 1/1

Group

100%

Group: Feature 2

Tasks: 1

Points: 3

COLLAPSE

Task

100%

Group: Feature 2
Task #1: Solution
Weight: ~100%
Points: ~3.00

COLLAPSE

Columns: 1

Sub-Task

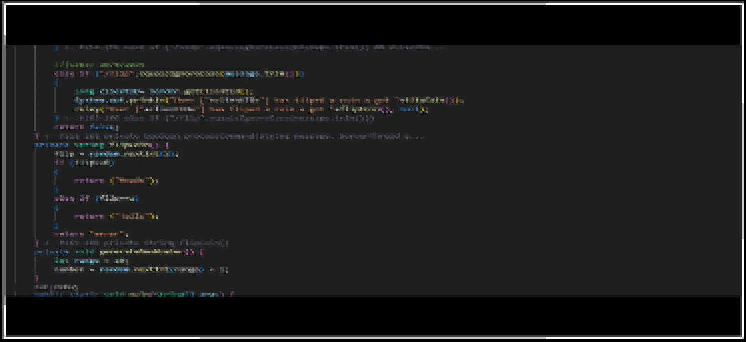
100%

Group: Feature 2
Task #1: Solution
Sub Task #1: Show the code related to the feature (ucid and date must be present as a comment)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
} // [illegible]

//[illegible]
else if ("/flip".equalsIgnoreCase(message.trim()))
{
    long clientID = [illegible];
    System.out.println("User ["+clientID+"] has fliped a coin & got "+flipCoin());
    relay("User ["+clientID+"] has fliped a coin & got "+flipCoin(), null);
} // [illegible]
return false;
} // [illegible]
private String flipCoin() {
    flip = random.nextInt(2);
    if (flip==0)
    {
        return ("Heads");
    }
    else if (flip==1)
    {
        return ("Tails");
    }
    return "error";
} // [illegible] private String flipCoin()
private void generateRandNumber() {
    int range = 10;
    number = random.nextInt(range) + 1;
}
[illegible]
public static void main(String[] args) {
```

The Flip coin code

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

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

Response:

I created a flipcoin method that generates a number either 0-1 and made an if statement so if it gets 0 it returns heads and if it gets 1 it returns tails. then I added the if statement in the processcommand method so that if the user types /flips it will relay their user ID and if they got heads or tails.

Sub-Task

100%

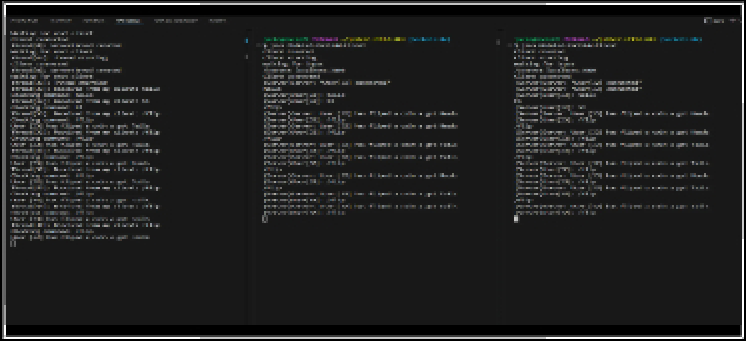
Group: Feature 2
Task #1: Solution
Sub Task #2: Show the feature working (i.e., all terminals and their related output)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

4 2 1


The Terminal of the flip code

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 1

End of Group: Feature 2
Task Status: 1/1

Group

91%

Group: Misc
Tasks: 3
Points: 2

COLLAPSE

Task

100%

Group: Misc
Task #1: Reflection
Weight: ~33%
Points: ~0.67

COLLAPSE

Sub-Task

100%

Group: Misc
Task #1: Reflection
Sub Task #1: Learn anything new? Face any challenges? How did you overcome any issues?

Task Response Prompt

Provide at least a few logical sentences

Response:

I learn to use the substring method in the String class. This method is quite helpful to get a certain part of the string in the long text. The only challenge i face was finding out where i needed to put the code for it to work the way i wanted.

End of Task 1

Task

100%

Group: Misc
Task #2: Pull request link
Weight: ~33%
Points: ~0.67

COLLAPSE

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

Task URLs

URL #1
https://github.com/Jackshii/Js2637-IT114-003/pull/8

URL
https://github.com/Jackshii/Js2637-IT114-003/p

End of Task 2

Task

75%

Group: Misc
Task #3: Waka Time (or related) Screenshot
Weight: ~33%
Points: ~0.67

COLLAPSE

Details:
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: 2 Columns

4 2 1

Missing Caption

End of Task 3

End of Group: Misc
Task Status: 2/3

End of Assignment