Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-450-M2024/it114-module-4-sockets-part-1-3/grade/yh68

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

Submissions:

Submission Selection

1 Submission [active] 6/16/2024 11:31:49 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

Tasks: 6 Points: 10.00

Baseline (2 pts.)

^COLLAPSE ^

Task #1 - Points: 1

^COLLAPSE ^

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

Caption (required) ✓
Describe/highlight what's being shown
Showing the Server terminal. Leftmost

#2) Show and clearly note which

Caption (required) ✓
Describe/highlight what's being shown
Showing the client terminals. Rightmost

#3) Show all clients receiving the

Caption (required) ✓
Describe/highlight what's being shown
Showing all clients receiving messages.

#4) Include a screenshot showing you

Caption (required) ✓
Describe/highlight what's being shown
Showing parts 1-3 + partHW

Feature 1 (3 pts.)

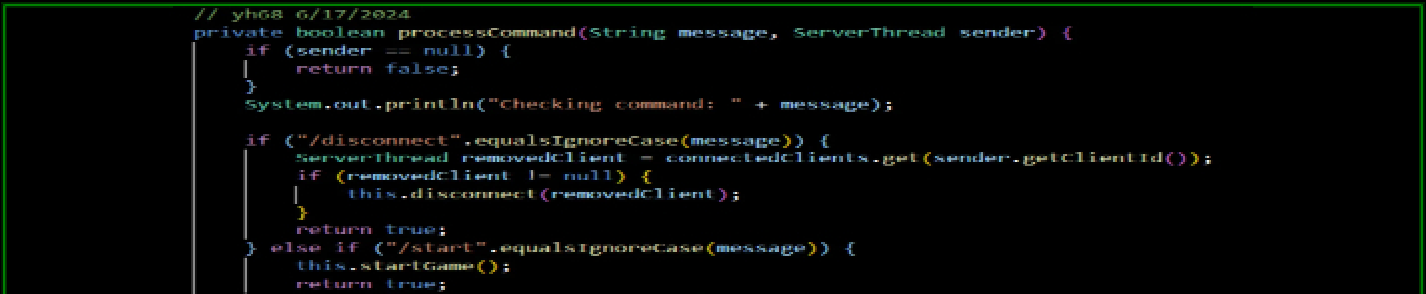
^COLLAPSE ^

Task #1 - Points: 1

^COLLAPSE ^

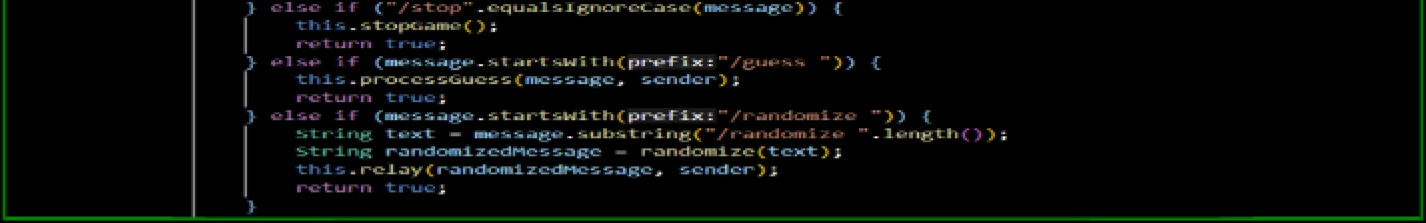
Text: Solution

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



```java
// yh68 6/17/2024
private boolean processCommand(String message, ServerThread sender) {
    if (sender == null) {
        return false;
    }
    System.out.println("Checking command: " + message);

    if ("/disconnect".equalsIgnoreCase(message)) {
        ServerThread removedClient = connectedClients.get(sender.getClientId());
        if (removedClient != null) {
            this.disconnect(removedClient);
        }
        return true;
    } else if ("/start".equalsIgnoreCase(message)) {
        this.startGame();
        return true;
```



```
} else if ("/stop".equalsIgnoreCase(message)) {
   this.stopGame();
   return true;
} else if (message.startsWith(prefix:"/guess ")) {
   this.processGuess(message, sender);
   return true;
} else if (message.startsWith(prefix:"/randomize ")) {
   String text = message.substring("/randomize ".length());
   String randomizedMessage = randomize(text);
   this.relay(randomizedMessage, sender);
   return true;
}
```

Caption (required) ✓

Describe/highlight what's being shown

Showing my number guessing feature

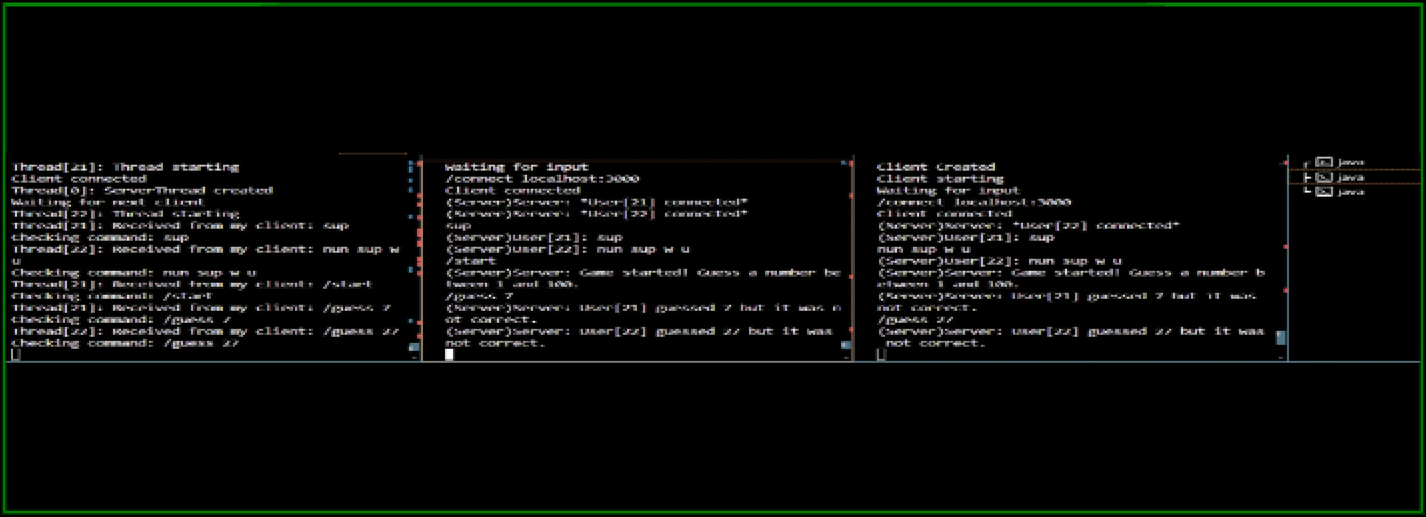
Explanation (required) ✓

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

PREVIEW RESPONSE

This feature is used by using the command /start which calls the method startGame() and allows for the client to start guessing. The game is stopped by using the /stop command which calls the stopGame() method and closes out the game.

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



```
Thread[21]: Thread starting
Client connected
Thread[0]: ServerThread created
Waiting for next client
Thread[22]: Thread starting
Thread[21]: Received from my client: sup
Checking command: sup
Thread[22]: Received from my client: nun sup w
u
Checking command: nun sup w u
Thread[21]: Received from my client: /start
Checking command: /start
Thread[21]: Received from my client: /guess 7
Checking command: /guess 7
Thread[22]: Received from my client: /guess 27
Checking command: /guess 27
```

```
Waiting for input
/connect localhost:3000
Client connected
(Server)Server: *User[21] connected*
(Server)Server: *User[22] connected*
sup
(Server)User[21]: sup
(Server)User[22]: nun sup w u
/start
(Server)Server: Game started! Guess a number be
tween 1 and 100.
/guess 7
(Server)Server: User[21] guessed 7 but it was n
ot correct.
(Server)Server: User[22] guessed 27 but it was
not correct.
```

```
Client Created
Client starting
Waiting for input
/connect localhost:3000
Client connected
(Server)Server: *User[22] connected*
(Server)User[21]: sup
nun sup w u
(Server)User[22]: nun sup w u
(Server)Server: Game started! Guess a number b
etween 1 and 100.
(Server)Server: User[21] guessed 7 but it was
not correct.
/guess 27
(Server)Server: User[22] guessed 27 but it was
 not correct.
```

Caption (required) ✓

Describe/highlight what's being shown

showing the working number guessing feature.

Feature 2 (3 pts.)

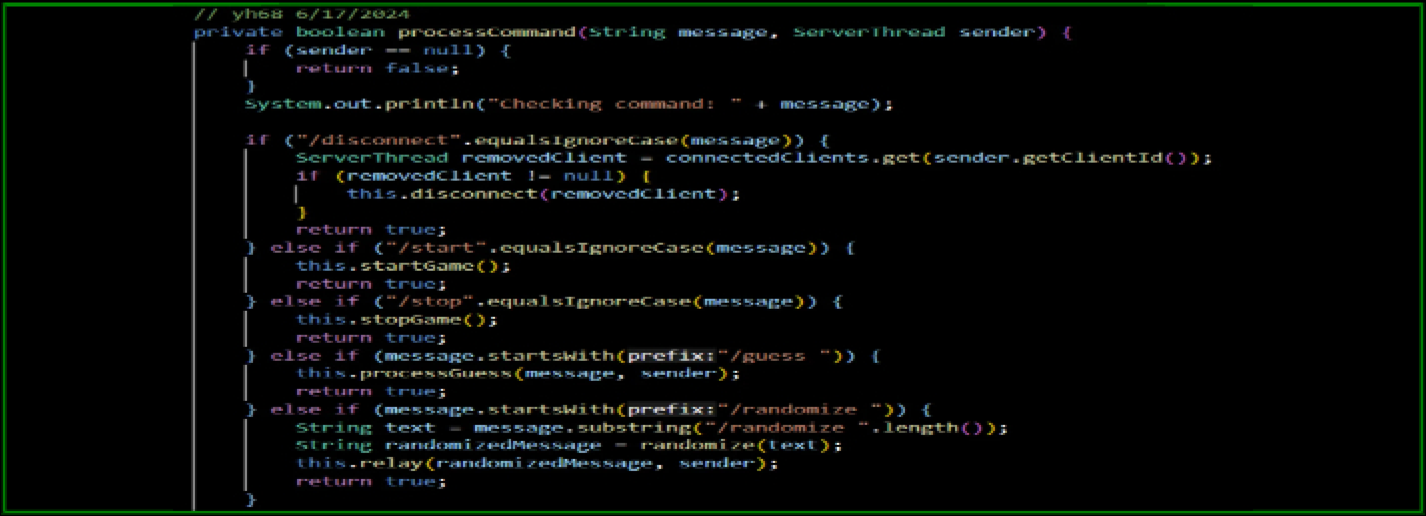
COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Solution

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



```java
// yh68 6/17/2024
private boolean processCommand(String message, ServerThread sender) {
    if (sender == null) {
        return false;
    }
    System.out.println("Checking command: " + message);

    if ("/disconnect".equalsIgnoreCase(message)) {
        ServerThread removedClient = connectedClients.get(sender.getClientId());
        if (removedClient != null) {
            this.disconnect(removedClient);
        }
        return true;
    } else if ("/start".equalsIgnoreCase(message)) {
        this.startGame();
        return true;
    } else if ("/stop".equalsIgnoreCase(message)) {
        this.stopGame();
        return true;
    } else if (message.startsWith(prefix:"/guess ")) {
        this.processGuess(message, sender);
        return true;
    } else if (message.startsWith(prefix:"/randomize ")) {
        String text = message.substring("/randomize ".length());
        String randomizedMessage = randomize(text);
        this.relay(randomizedMessage, sender);
        return true;
    }
```

Caption (required) ✓

Describe/highlight what's being shown

Showing the randomize feature.

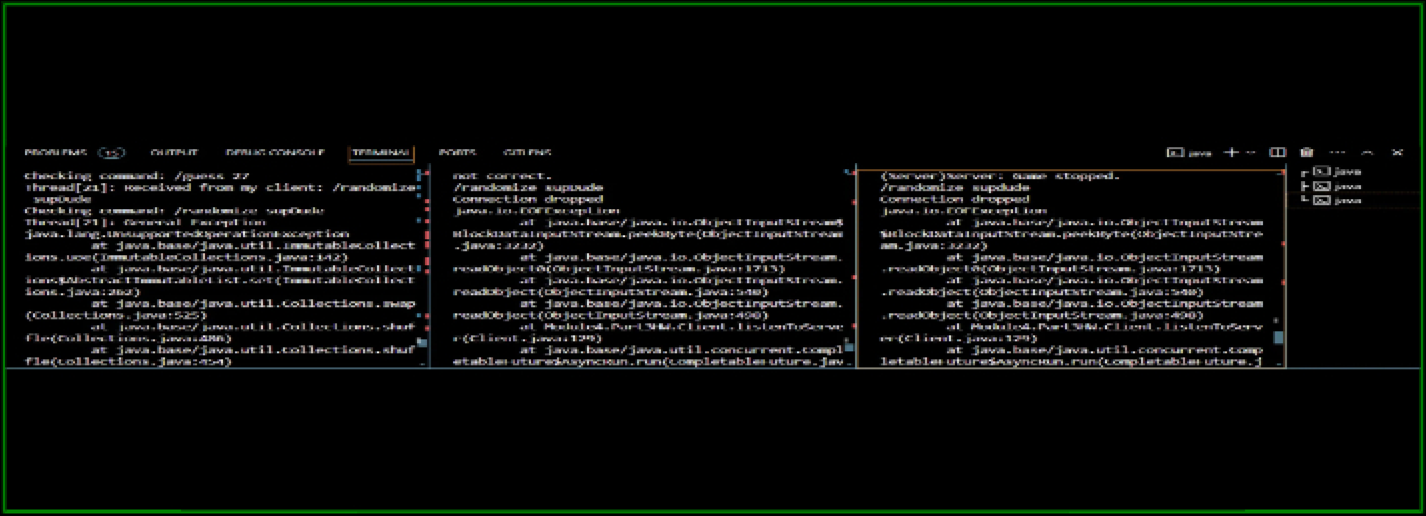
Explanation (required) ✓

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

PREVIEW RESPONSE

I used the command /randomize for this feature which takes the string and shuffles the letters around. When the command /randomize is used the randomize() method is called and shufles the string that is typed with the command.

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



```
checking command: /guess 27
Thread[21]: Received from my client: /randomize
 supDude
Checking command: /randomize supDude
Thread[21]: General Exception
java.lang.UnsupportedOperationException
        at java.base/java.util.ImmutableCollect
ions.uoe(ImmutableCollections.java:142)
        at java.base/java.util.ImmutableCollect
ions$AbstractImmutableList.set(ImmutableCollect
ions.java:262)
        at java.base/java.util.Collections.swap
(Collections.java:525)
        at java.base/java.util.Collections.shuf
fle(Collections.java:488)
        at java.base/java.util.Collections.shuf
fle(Collections.java:454)
```

```
not correct.
/randomize supDude
Connection dropped
java.io.EOFException
        at java.base/java.io.ObjectInputStream$
BlockDataInputStream.peekByte(ObjectInputStream
.java:3232)
        at java.base/java.io.ObjectInputStream.
readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream.
readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream.
readObject(ObjectInputStream.java:498)
        at Module4.Part3HW.Client.listenToServe
r(Client.java:129)
        at java.base/java.util.concurrent.Compl
etableFuture$AsyncRun.run(CompletableFuture.jav
```

```
(Server)Server: Game stopped.
/randomize supdude
Connection dropped
java.io.EOFException
        at java.base/java.io.ObjectInputStream
$BlockDataInputStream.peekByte(ObjectInputStre
am.java:3232)
        at java.base/java.io.ObjectInputStream
.readObject0(ObjectInputStream.java:1713)
        at java.base/java.io.ObjectInputStream
.readObject(ObjectInputStream.java:540)
        at java.base/java.io.ObjectInputStream
.readObject(ObjectInputStream.java:498)
        at Module4.Part3HW.Client.listenToServ
er(Client.java:129)
        at java.base/java.util.concurrent.Comp
letableFuture$AsyncRun.run(CompletableFuture.j
```

Caption (required) ✓

Describe/highlight what's being shown
Showing the output for the feature, it should have worked but I must've messed something up by accident.

● Misc (2 pts.)
COLLAPSE

● COLLAPSE
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
PREVIEW RESPONSE
I learned a lot this module about inputstream and outputstream. I faced a challenge implementing the last feature, but debugging the whole code was a lot so I tried to focus on my actual code but I couldn't figure it out.

● 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/FreePalestine7/yh68-it114-450/pull/12

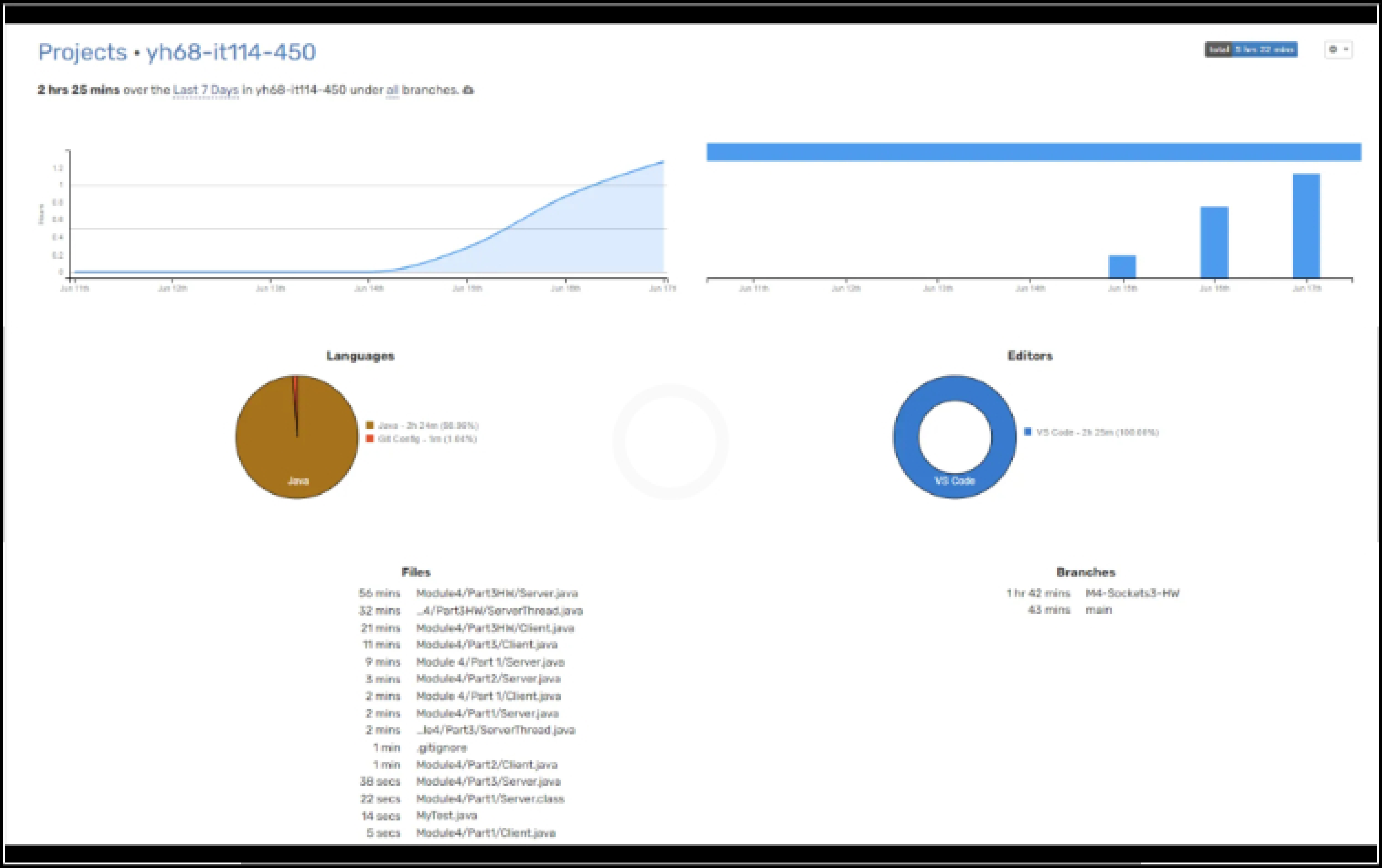
● COLLAPSE
Task #3 - Points: 1
Text: Waka Time (or related) Screenshot

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: Large View

Small Medium Large



Projects • yh68-it114-450

2 hrs 25 mins over the Last 7 Days in yh68-it114-450 under all branches.

Languages

- 🟫 Java - 2h 24m (98.36%)
- 🟥 Git Config - 1m (1.64%)

Editors

- 🟦 VS Code - 2h 25m (100.00%)

Files

56 mins	Module4/Part3HW/Server.java
32 mins	...4/Part3HW/ServerThread.java
21 mins	Module4/Part3HW/Client.java
11 mins	Module4/Part3/Client.java
9 mins	Module 4/Part 1/Server.java
3 mins	Module4/Part2/Server.java
2 mins	Module 4/Part 1/Client.java
2 mins	Module4/Part1/Server.java
2 mins	...le4/Part3/ServerThread.java
1 min	.gitignore
1 min	Module4/Part2/Client.java
38 secs	Module4/Part3/Server.java
22 secs	Module4/Part1/Server.class
14 secs	MyTest.java
5 secs	Module4/Part1/Client.java

Branches

1 hr 42 mins	M4-Sockets3-HW
43 mins	main

Showing overview of wakatime

End of Assignment