Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-milestone-2-chatroom-2024/grade/ms75

IT114-006-S2024 - [IT114] Milestone 2 Chatroom 2024

Submissions:

Submission Selection

1 Submission [active] 4/4/2024 1:56:03 AM

Instructions

COLLAPSE

1. Implement the Milestone 2 features from the project's proposal document: https://docs.google.com/document/d/1ONmvEvel97GTFPGfVwwQC96xSsobbSbk56145XizQG4/view
2. Make sure you add your ucid/date as code comments where code changes are done
3. All code changes should reach the Milestone2 branch
4. Create a pull request from Milestone2 to main and keep it open until you get the output PDF from this assignment.
5. Gather the evidence of feature completion based on the below tasks.
6. Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
7. Run the necessary git add, commit, and push steps to move it to GitHub
8. Complete the pull request that was opened earlier
9. Upload the same output PDF to Canvas

Branch name: Milestone2

Tasks: 12 Points: 10.00

Demonstrate Usage of Payloads (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of your Payload class and subclasses and PayloadType

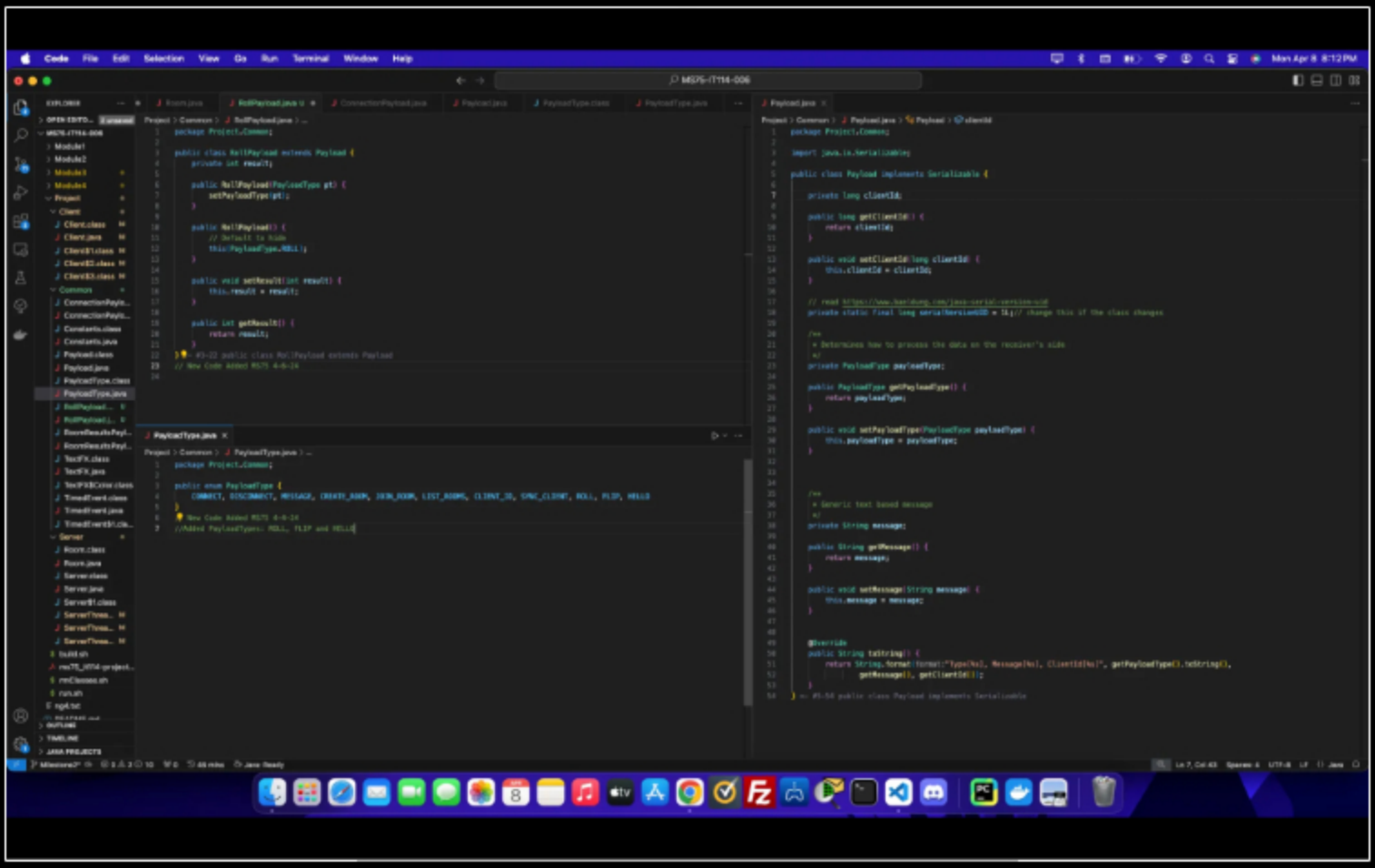
Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Payload, equivalent of RollPayload, and any others
☐ #2	1	Screenshots should include ucid and date comment
☐ #3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Screenshot Shows my RollPayload.java on the top left. My PayloadType.java on the bottom left and my Payload.java class on the right. There were no changes to my Payload.java class.

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the payloads being debugged/output to the terminal

Checklist

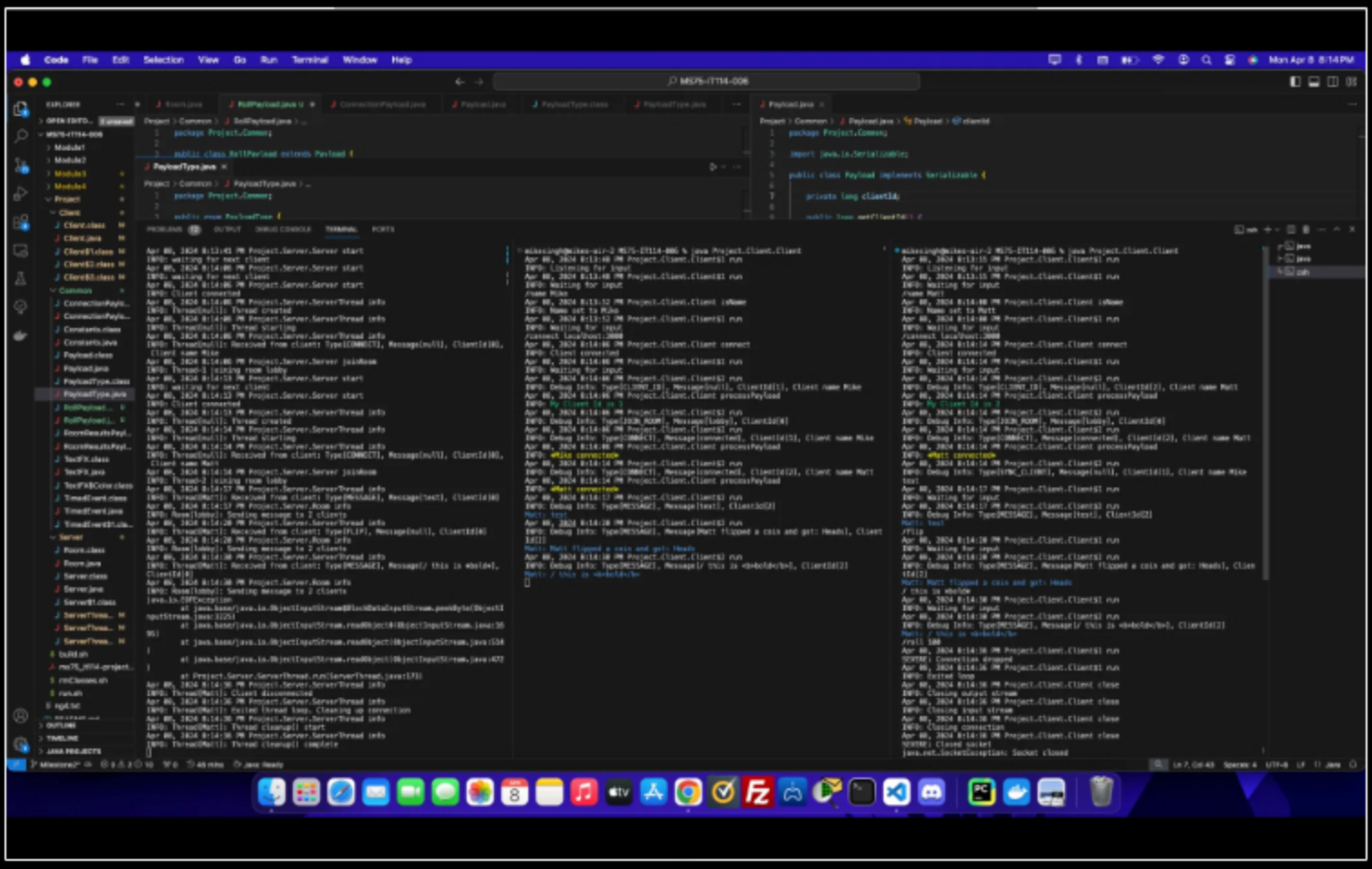
*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Demonstrate flip
☐ #2	1	Demonstrate roll (both versions)
☐ #3	1	Demonstrate formatted message along with any others
☐ #4	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Screenshot shows mike and matt connected to the server where matt calls the /flip command and types "this is *bold*" and the message sent is "Matt Flipped a coin and got: Heads" and "this is bold". The /flip and special message formatting are working. I could not figure out the /roll command.

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Explain the purpose of payloads and how your flip/roll payloads were made

Response:

Payloads are used to transfer the relevant information, a coin flip result or dice roll result are sent through the payload to allow each clients data to be processed smoothly and check for errors in the command lines. Payloads are used to extract data, in this example when the client does /roll 100 the payload extracts the 100 if the /roll command is called and processes it into the roll() method allowing for an easier communication between client and server.

Demonstrate Roll Command (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Client code that captures the command and converts it to a RollPayload (or equivalent) for both scenarios /roll # and /roll #d#
☐ #2	1	ServerThread code receiving the payload and passing it to the Room
☐ #3	1	Room handling the roll action correctly for both scenarios (/roll # and /roll #d#) including the message going back out to all clients
☐ #4	1	Code screenshots should include ucid and date comment
☐ #5	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
    private boolean processClientCommand(String text) throws IOException {
//      Here I added the /roll command into the processCommand() method
        else if (text.startsWith(ROLL_COMMAND)) {
            String rollString = text.replace(target:"/roll", replacement:"").trim();
            try {
                int result = roll(rollString);
                sendRoll(result);
            } catch (IOException e) {
    sendMessage(message:"Wrong Format. Use : '/roll x' or '/roll xdy'.");
}
return true;
        } <- #223-230 else if (text.startsWith(ROLL_COMMAND))


//  4-2-24
//  MS75
//  New Code Ends
        return false;
    } <- #233-237 else if (text.equalsIgnoreCase(FLIP_COMMAND))
// New Code MS75 4-8-24 I moved my roll command here as it was previously in the ServerThread.java class
    private int roll(String roll) {
        roll = roll.trim().substring("/roll".length()).trim();
        String[] parts = roll.split(regex:"\\s+");

        int result = 0;

        if (parts.length == 1 && parts[0].matches(regex:"\\d+")) {
            result = (int) (Math.random() * (Integer.parseInt(parts[0]) + 1));
        } else if (parts.length == 2 && parts[0].matches(regex:"\\d+") && parts[1].matches(regex:"\\d+d\\d+")) {
            String[] diceParams = parts[1].split(regex:"d");
            int diceCount = Integer.parseInt(diceParams[0]);
            int faceCount = Integer.parseInt(diceParams[1]);
            for (int i = 0; i < diceCount; i++) {
                result += (int) (Math.random() * faceCount) + 1;
            }
        } else {
            throw new IllegalArgumentException(s:"Wrong Format. Use : '/roll x' or '/roll xdy'.");
        }

        return result;
    } <- #260-290 private int roll(String roll)

    // Send methods

//  New Code Begins
//  MS75
//  4-2-24
//      Here I added the send methods for payloadTypes: HELLO, FLIP and ROLL
//      to allow the payloads to be created in the serverThreads
    private void sendHello() throws IOException {
        Payload p = new Payload();
        p.setPayloadType(PayloadType.HELLO);
        out.writeObject(p);
    } <- #268-272 private void sendHello() throws IOException
    private void sendFlip() throws IOException {
        Payload p = new Payload();
        p.setPayloadType(PayloadType.FLIP);
        out.writeObject(p);
    } <- #273-277 private void sendFlip() throws IOException

    private void sendRoll(int result) throws IOException {
        RollPayload rp = new RollPayload();
        rp.setResult(result);
        out.writeObject(rp);
    } <- #279-283 private void sendRoll(int result) throws IOException

//  4-2-24
//  MS75
//  New Code Ends
```

Screenshot shows adding else if statement into processClientCommand() in the Client.java class. It also shows me adding the logic for the roll() command and the sendRoll() send method all in the Client.java class

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic in how the two different roll formats are handled and how the message flows from the client, to the Room, and shared with all other users

Response:

The client first types their command of /roll x or /roll xdx, where x are integer values. In the Client.java class the processClientCommand() checks for if the clients message starts with the roll_command string which is "/roll" and if it does it calls the roll() method. This roll method checks if the clients format is correct and if not it sends an error back at the client telling them to fix their format. The new payload with payload type RollPayload is created in the sendRoll() method and the integers x and y or just x are sent through the payload depending on which format the client wrote their /roll command in. Once the roll() method is called the sendMessage() method is used to print out the clients roll result to the room.

Demonstrate Flip Command (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

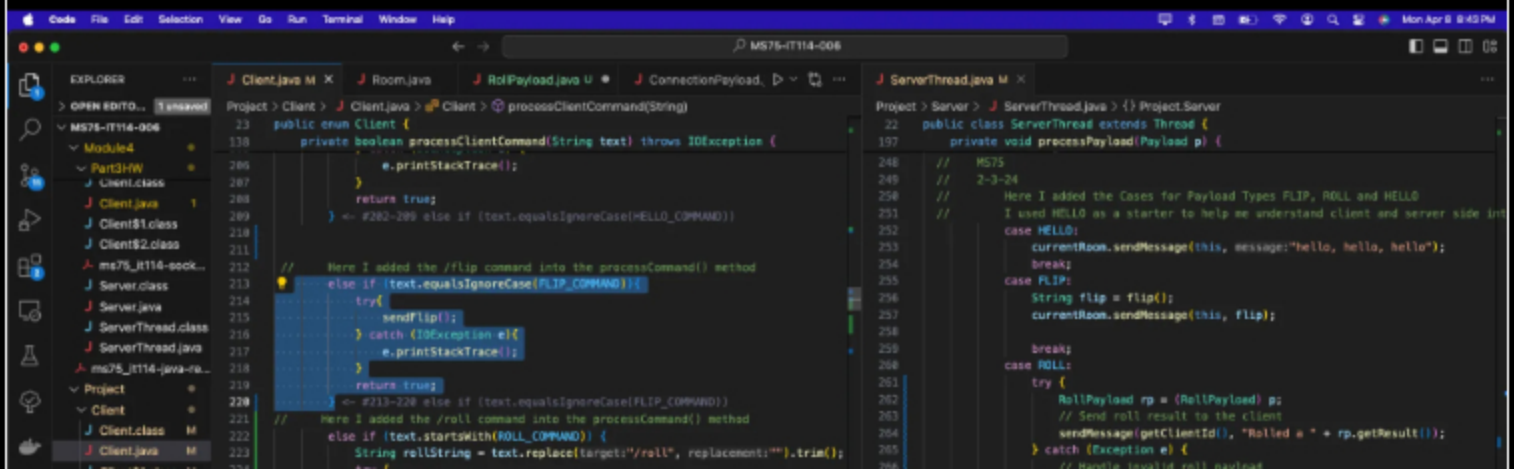
Checklist *The checkboxes are for your own tracking

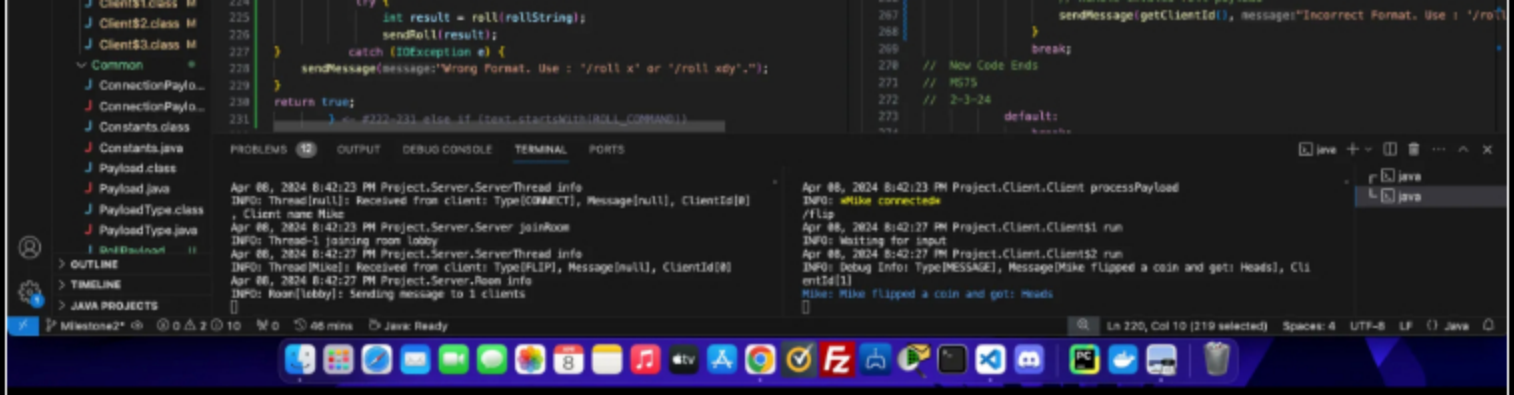
#	Points	Details
☐ #1	1	Client code that captures the command and converts it to a payload
☐ #2	1	ServerThread receiving the payload and passing it to the Room
☐ #3	1	Room handling the flip action correctly
☐ #4	1	Code screenshots should include ucid and date comment
☐ #5	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large





This screenshot shows on the top left my processClientCommand() in the client.java file which checks if the client text calls the /flip command. If it does then a new payload is created with the sendFlip() method. In the top right the ServerThread.java class shows the processPayload() method processing the /flip command and it calls the flip method in the ServerThread class which prints out the heads or tails.

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic in how the flip command is handled and processed and how the message flows from the client, to the Room, and shared with all other users

Response:

The flip command is typed by the client and first caught in the processClientCommand() method in the Client.java class. It then calls a new instance of the flip payload in the sendFlip() method. The serverThread of that client actually flips the coin with the flip() class (lines 284 -288 in the ServerThread.java class) and then finds which room "this" client is in and prints out the result of the flip() method.

Demonstrate Formatted Messages (4 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of Room how the following formatting is processed from a message

Details:

Note: this processing is server-side

Slash commands are not valid solutions for this and will receive 0 credit

Checklist

*The checkboxes are for your own tracking

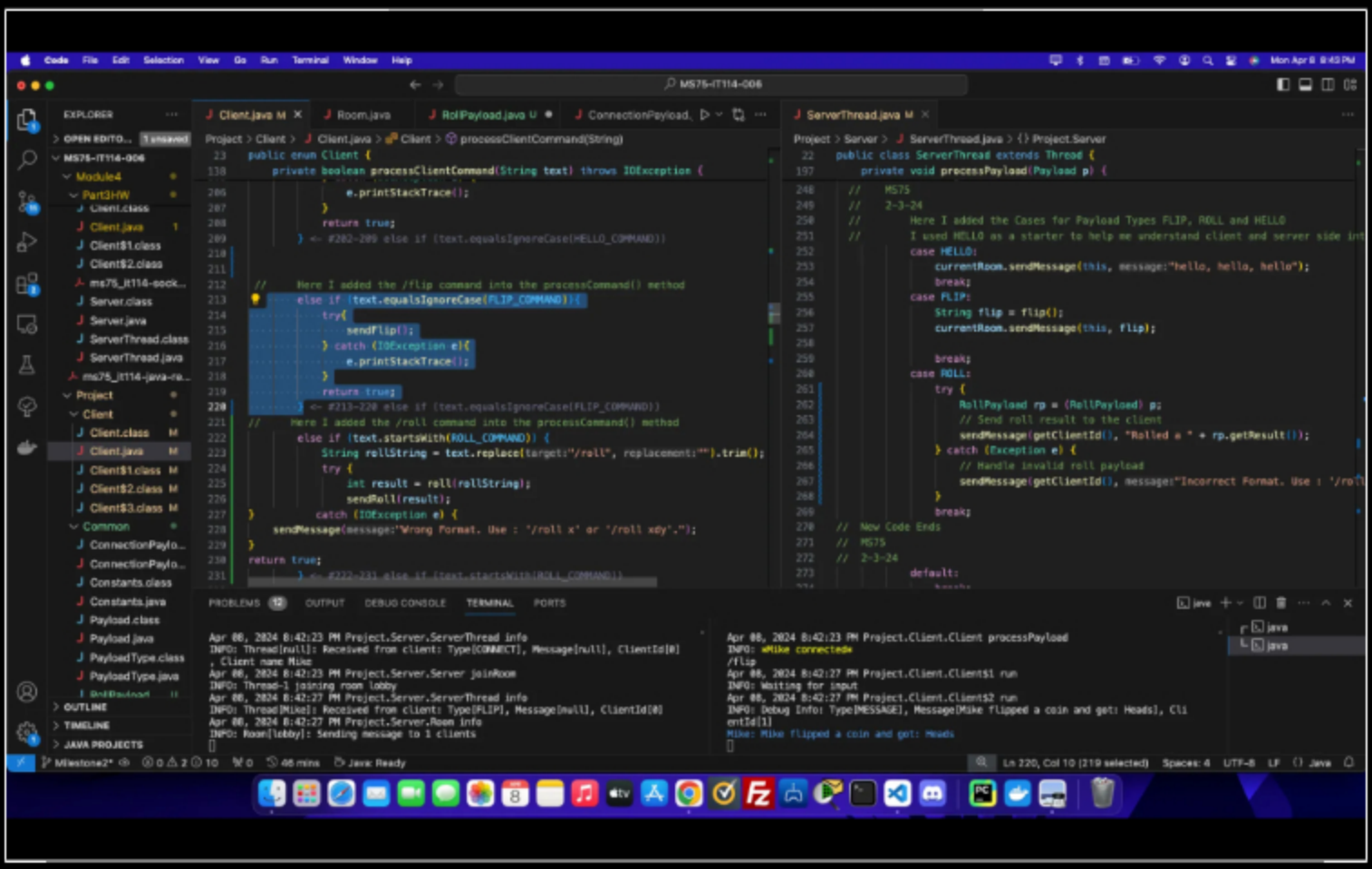
#	Points	Details
☐ #1	1	Room code processing for bold

☐ #2	1	Room code processing for italic
☐ #3	1	Room code processing for underline
☐ #4	1	Room code processing for color (at least R, G, B or support for hex codes)
☐ #5	1	Show each one working individually and one showing a combination of all of the formats and 1 color from the terminal
☐ #6	1	Must not rely on the user typing html characters, but the output can be html characters
☐ #7	1	Code screenshots should include ucid and date comment
☐ #8	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large



In this screenshot my case in the switch case in the processPayload() method shows the case MESSAGE where all the clients text is scanned through before it is processed in other cases or even checked for commands. The MESSAGE case calls the processedMessage() method which checks for special characters and replaces them with html tags. I could not figure out how to upload more than 1 screenshot for this response but the terminals shows a client type "this is *bold* *-_#rthis is bold, italic, underlined and red r#_-* " and the response printed out was "Mike: this is bold <u>this is bold, italic, underlined and red r</u>" this shows all of the special character requirements working which are - italics -bold -underline -color changer of text

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the following

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Which special characters translate to the desired effect
☐ #2	1	How the logic works that converts the message to its final format

Response:

The replaceAll method is used to replace findings of patterns in the message with their corresponding HTML tags. The string is scanned to look for these patterns which is finding text that is enclosed within the following characters "-*_" and either "#r" or "#g" or "#b". Once the entire string is checked and replaced the string message is returned and then in the MESSAGE case switch the current room of the serverThread is found the the string message is sent using the sendMessage method.

● Misc (1 pt.)

COLLAPSE

● COLLAPSE

Task #1 - Points: 1

Text: Add the pull request link for the branch

Details:

Note: the link should end with /pull/#

URL #1

https://github.com/mikes1302/MS75-IT114-006/pull/13

● COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Still having issues with getting my /roll command working. I am going to just submit and try to get it to work by milestone 3. I also just had slight issues distinguishing which files I needed for my specific project choice. I spent a lot of time reading through text that did not pertain to my project directly.

● COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

Details:

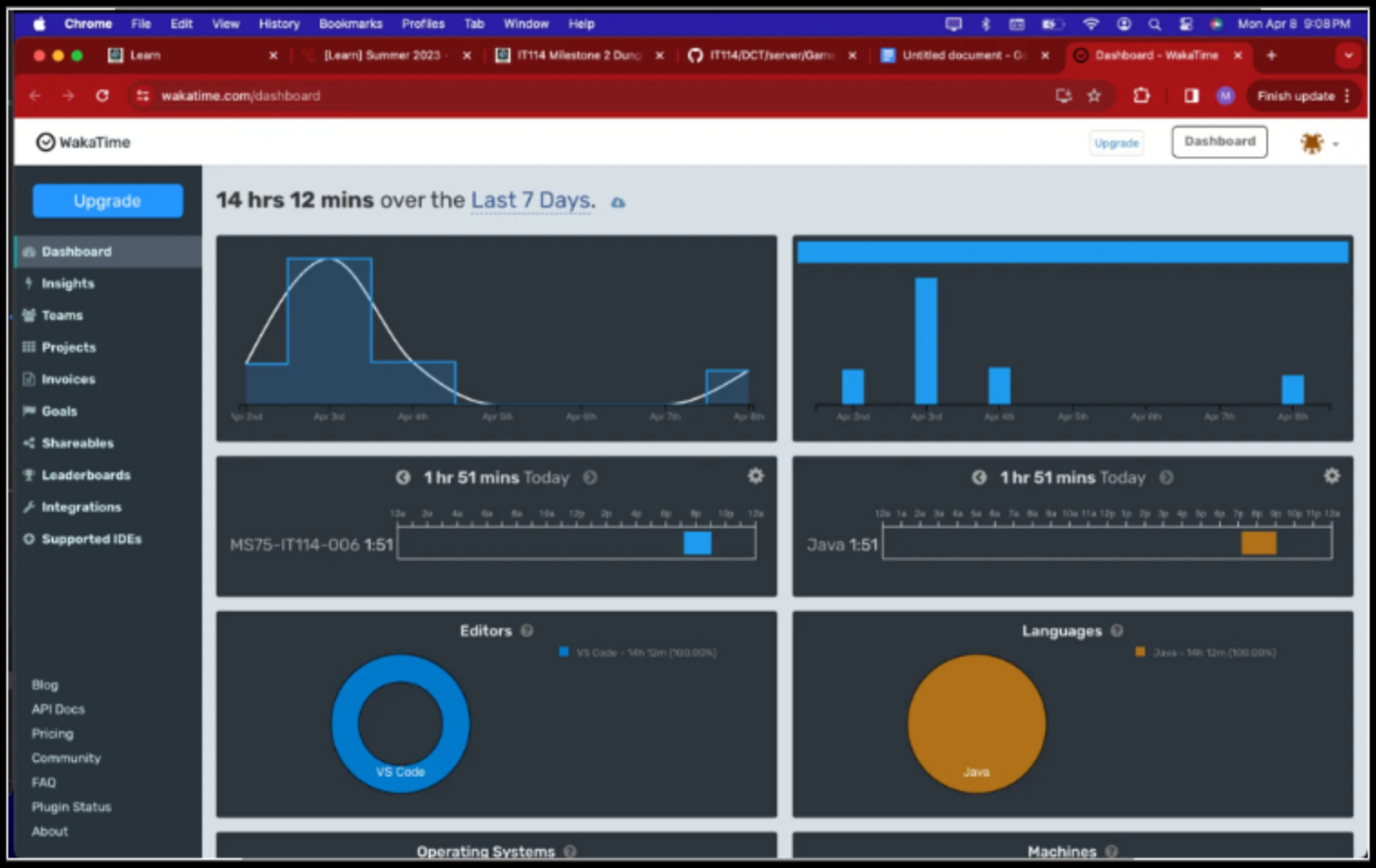
Details:

Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Screenshot shows wakatime screenshot for work on IT 114 repository within the last week.

End of Assignment