Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-milestone-2-chatroom-2024/grade/fj28

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

Submissions:

Submission Selection

1 Submission [active] 4/2/2024 4:47:19 AM

Instructions

COLLAPSE

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

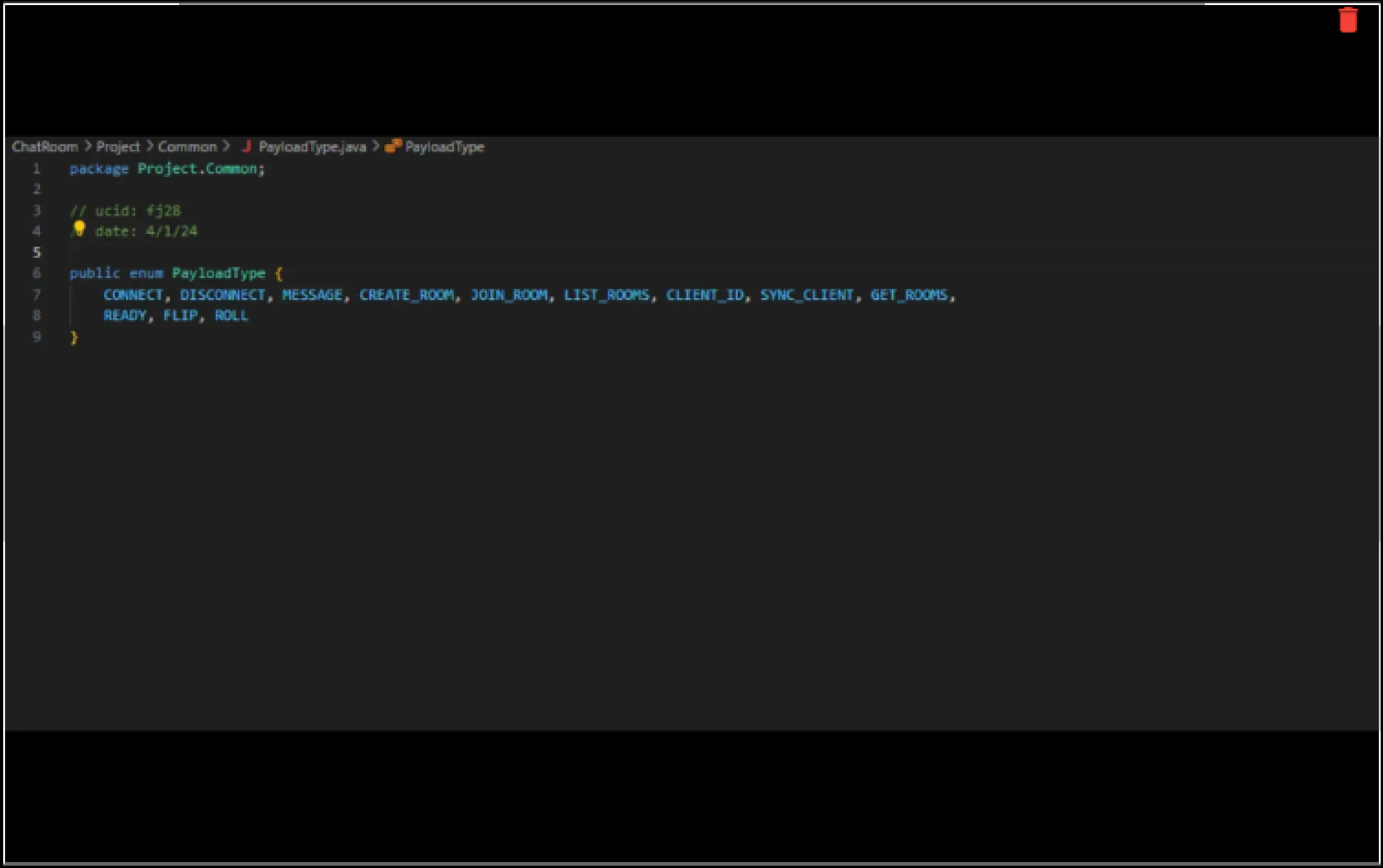
Checklist *The checkboxes are for your own tracking

#	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


ChatRoom > Project > Common > PayloadType.java > PayloadType

```java
package Project.Common;

// ucid: fj28
// date: 4/1/24

public enum PayloadType {
    CONNECT, DISCONNECT, MESSAGE, CREATE_ROOM, JOIN_ROOM, LIST_ROOMS, CLIENT_ID, SYNC_CLIENT, GET_ROOMS,
    READY, FLIP, ROLL
}
```

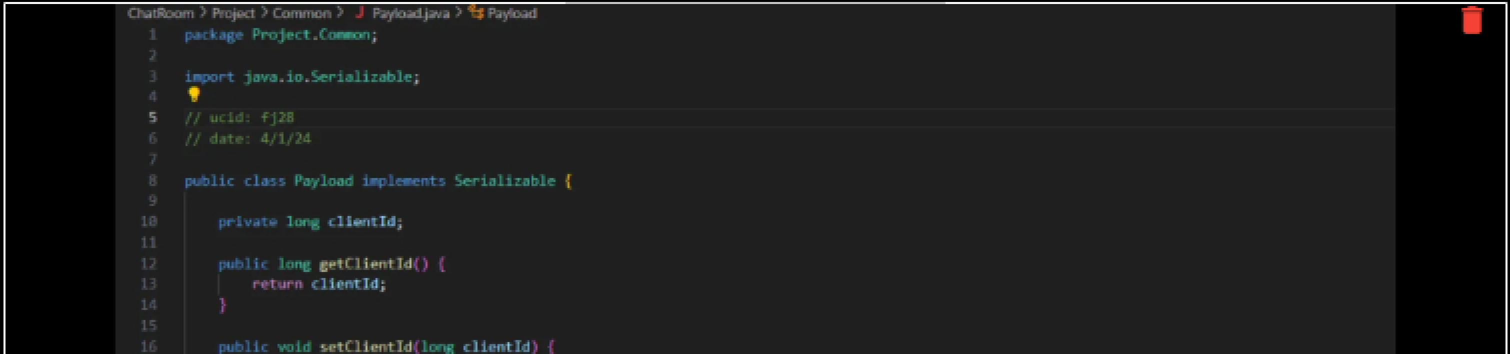
PayloadType

Checklist Items (3)

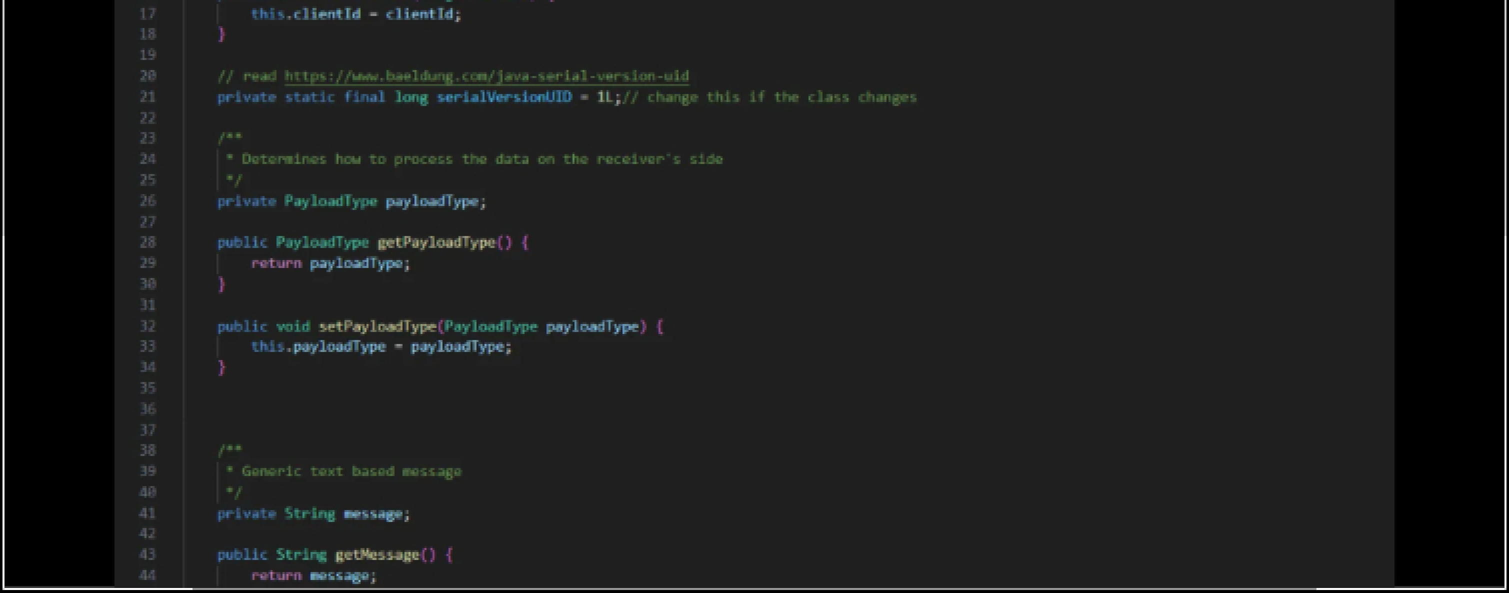
#1 Payload, equivalent of RollPayload, and any others

#2 Screenshots should include ucid and date comment

#3 Each screenshot should be clearly captioned


ChatRoom > Project > Common > Payload.java > Payload

```java
package Project.Common;

import java.io.Serializable;

// ucid: fj28
// date: 4/1/24

public class Payload implements Serializable {

    private long clientId;

    public long getClientId() {
        return clientId;
    }

    public void setClientId(long clientId) {
```



```
        this.clientId = clientId;
    }

    // read https://www.baeldung.com/java-serial-version-uid
    private static final long serialVersionUID = 1L;// change this if the class changes

    /**
     * Determines how to process the data on the receiver's side
     */
    private PayloadType payloadType;

    public PayloadType getPayloadType() {
        return payloadType;
    }

    public void setPayloadType(PayloadType payloadType) {
        this.payloadType = payloadType;
    }



    /**
     * Generic text based message
     */
    private String message;

    public String getMessage() {
        return message;
```

Payload Class

Checklist Items (0)

●
^COLLAPSE ^

Task #2 - Points: 1

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

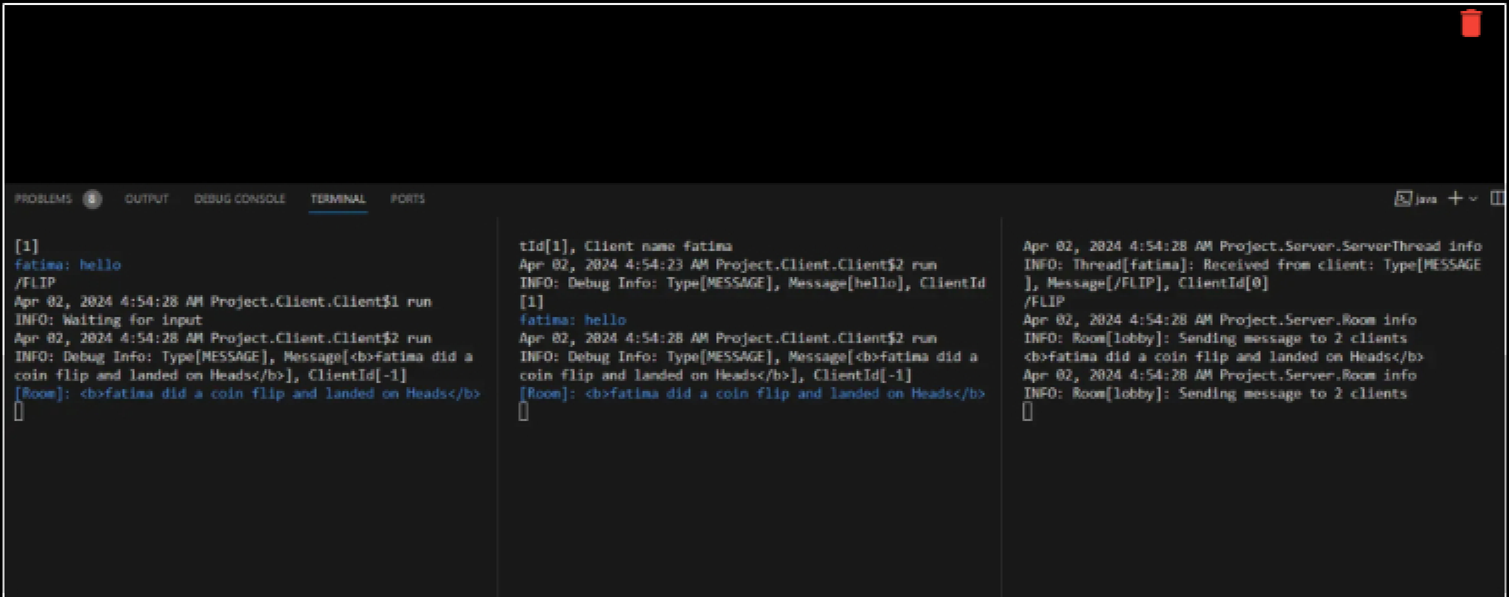
Checklist *The checkboxes are for your own tracking

#	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

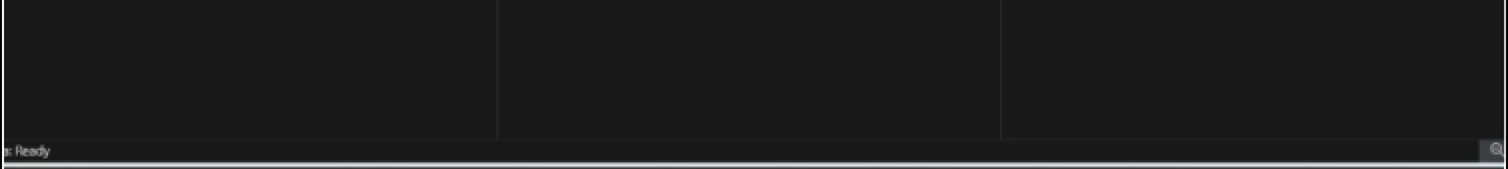
Small Medium Large



```
[1]
fatima: hello
/FLIP
Apr 02, 2024 4:54:28 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:54:28 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
coin flip and landed on Heads</b>], ClientId[-1]
[Room]: <b>fatima did a coin flip and landed on Heads</b>
▯
```

```
tId[1], Client name fatima
Apr 02, 2024 4:54:23 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[hello], ClientId
[1]
fatima: hello
Apr 02, 2024 4:54:28 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
coin flip and landed on Heads</b>], ClientId[-1]
[Room]: <b>fatima did a coin flip and landed on Heads</b>
▯
```

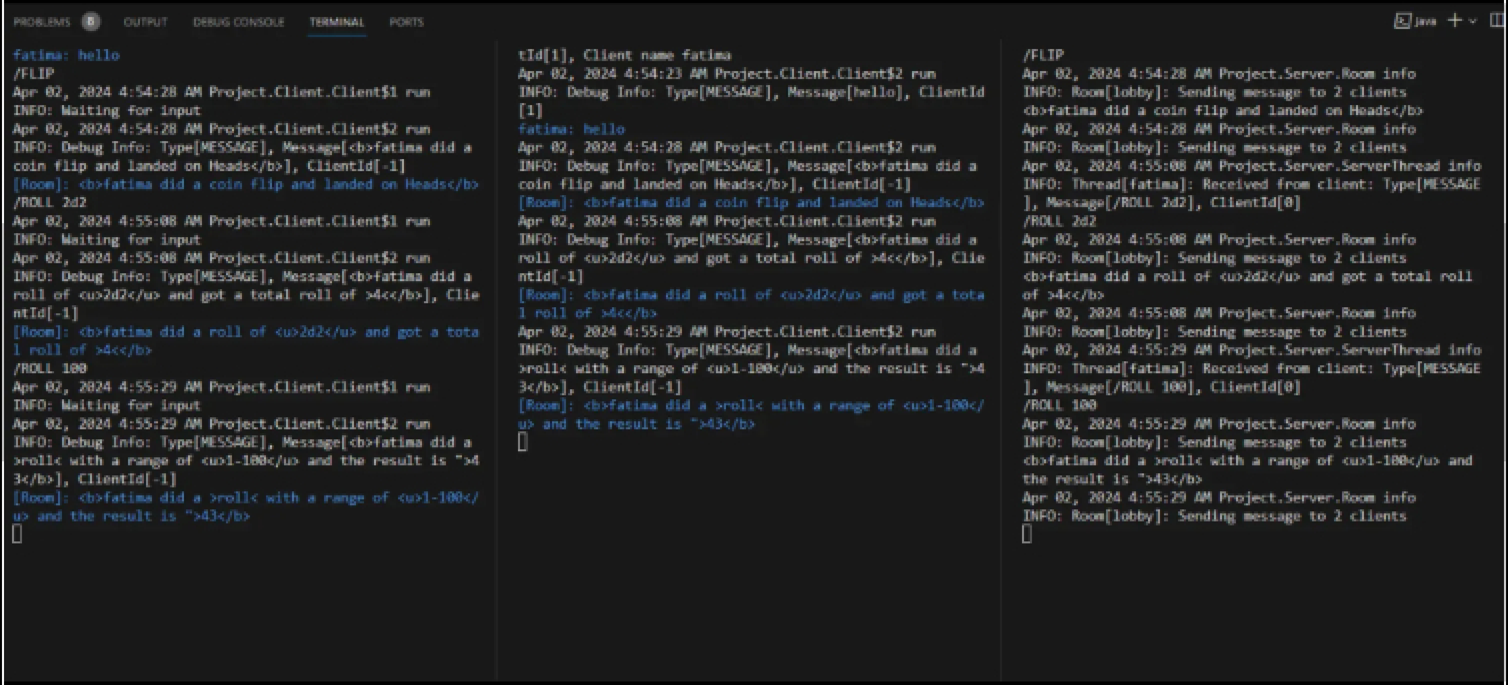
```
Apr 02, 2024 4:54:28 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE
], Message[/FLIP], ClientId[0]
/FLIP
Apr 02, 2024 4:54:28 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
<b>fatima did a coin flip and landed on Heads</b>
Apr 02, 2024 4:54:28 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
▯
```



Flip

Checklist Items (1)

#1 Demonstrate flip



```
fatima: hello
/FLIP
Apr 02, 2024 4:54:28 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:54:28 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
coin flip and landed on Heads</b>], ClientId[-1]
[Room]: <b>fatima did a coin flip and landed on Heads</b>
/ROLL 2d2
Apr 02, 2024 4:55:08 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:55:08 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
roll of <u>2d2</u> and got a total roll of >4<</b>], Clie
ntId[-1]
[Room]: <b>fatima did a roll of <u>2d2</u> and got a tota
l roll of >4<</b>
/ROLL 100
Apr 02, 2024 4:55:29 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:55:29 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
>roll< with a range of <u>1-100</u> and the result is ">4
3</b>], ClientId[-1]
[Room]: <b>fatima did a >roll< with a range of <u>1-100</
u> and the result is ">43</b>
```

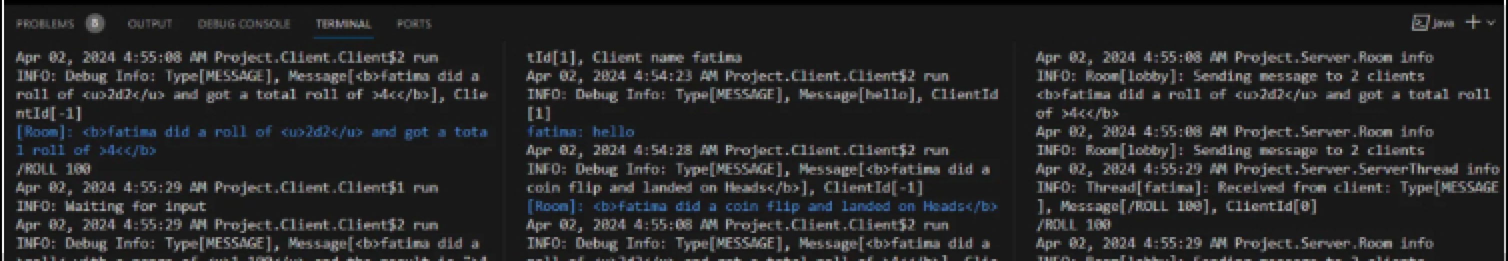
```
tId[1], Client name fatima
Apr 02, 2024 4:54:23 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[hello], ClientId
[1]
fatima: hello
Apr 02, 2024 4:54:28 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
coin flip and landed on Heads</b>], ClientId[-1]
[Room]: <b>fatima did a coin flip and landed on Heads</b>
Apr 02, 2024 4:55:08 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
roll of <u>2d2</u> and got a total roll of >4<</b>], Clie
ntId[-1]
[Room]: <b>fatima did a roll of <u>2d2</u> and got a tota
l roll of >4<</b>
Apr 02, 2024 4:55:29 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
>roll< with a range of <u>1-100</u> and the result is ">4
3</b>], ClientId[-1]
[Room]: <b>fatima did a >roll< with a range of <u>1-100</
u> and the result is ">43</b>
```

```
/FLIP
Apr 02, 2024 4:54:28 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
<b>fatima did a coin flip and landed on Heads</b>
Apr 02, 2024 4:54:28 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 4:55:08 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE
], Message[/ROLL 2d2], ClientId[0]
/ROLL 2d2
Apr 02, 2024 4:55:08 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
<b>fatima did a roll of <u>2d2</u> and got a total roll
of >4<</b>
Apr 02, 2024 4:55:08 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 4:55:29 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE
], Message[/ROLL 100], ClientId[0]
/ROLL 100
Apr 02, 2024 4:55:29 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
<b>fatima did a >roll< with a range of <u>1-100</u> and
the result is ">43</b>
Apr 02, 2024 4:55:29 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
```

Roll

Checklist Items (1)

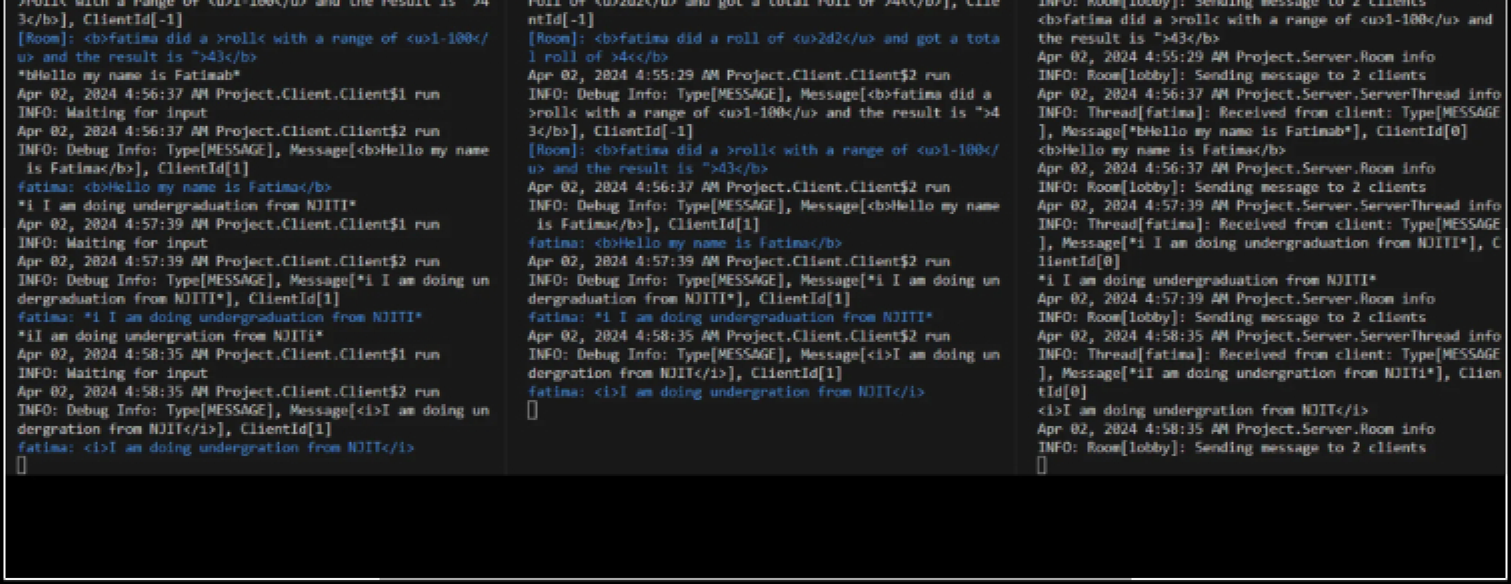
#2 Demonstrate roll (both versions)



```
Apr 02, 2024 4:55:08 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
roll of <u>2d2</u> and got a total roll of >4<</b>], Clie
ntId[-1]
[Room]: <b>fatima did a roll of <u>2d2</u> and got a tota
l roll of >4<</b>
/ROLL 100
Apr 02, 2024 4:55:29 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:55:29 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
```

```
tId[1], Client name fatima
Apr 02, 2024 4:54:23 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[hello], ClientId
[1]
fatima: hello
Apr 02, 2024 4:54:28 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
coin flip and landed on Heads</b>], ClientId[-1]
[Room]: <b>fatima did a coin flip and landed on Heads</b>
Apr 02, 2024 4:55:08 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
```

```
Apr 02, 2024 4:55:08 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
<b>fatima did a roll of <u>2d2</u> and got a total roll
of >4<</b>
Apr 02, 2024 4:55:08 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 4:55:29 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE
], Message[/ROLL 100], ClientId[0]
/ROLL 100
Apr 02, 2024 4:55:29 AM Project.Server.Room info
```



```
>roll< with a range of <u>1-100</u> and the result is ">4
3</b>], ClientId[-1]
[Room]: <b>fatima did a >roll< with a range of <u>1-100</
u> and the result is ">43</b>
*bHello my name is Fatimab*
Apr 02, 2024 4:56:37 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:56:37 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>Hello my name
 is Fatima</b>], ClientId[1]
fatima: <b>Hello my name is Fatima</b>
*i I am doing undergraduation from NJITI*
Apr 02, 2024 4:57:39 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:57:39 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[*i I am doing un
dergraduation from NJITI*], ClientId[1]
fatima: *i I am doing undergraduation from NJITI*
*iI am doing undergration from NJITi*
Apr 02, 2024 4:58:35 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 4:58:35 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<i>I am doing un
dergration from NJIT</i>], ClientId[1]
fatima: <i>I am doing undergration from NJIT</i>
▯
```

```
roll of <u>2d2</u> and got a total roll of >4<</b>], Clie
ntId[-1]
[Room]: <b>fatima did a roll of <u>2d2</u> and got a tota
l roll of >4<</b>
Apr 02, 2024 4:55:29 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>fatima did a
>roll< with a range of <u>1-100</u> and the result is ">4
3</b>], ClientId[-1]
[Room]: <b>fatima did a >roll< with a range of <u>1-100</
u> and the result is ">43</b>
Apr 02, 2024 4:56:37 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<b>Hello my name
 is Fatima</b>], ClientId[1]
fatima: <b>Hello my name is Fatima</b>
Apr 02, 2024 4:57:39 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[*i I am doing un
dergraduation from NJITI*], ClientId[1]
fatima: *i I am doing undergraduation from NJITI*
Apr 02, 2024 4:58:35 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[<i>I am doing un
dergration from NJIT</i>], ClientId[1]
fatima: <i>I am doing undergration from NJIT</i>
▯
```

```
INFO: Room[lobby]: Sending message to 2 clients
<b>fatima did a >roll< with a range of <u>1-100</u> and
the result is ">43</b>
Apr 02, 2024 4:55:29 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 4:56:37 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE
], Message[*bHello my name is Fatimab*], ClientId[0]
<b>Hello my name is Fatima</b>
Apr 02, 2024 4:56:37 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 4:57:39 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE
], Message[*i I am doing undergraduation from NJITI*], C
lientId[0]
*i I am doing undergraduation from NJITI*
Apr 02, 2024 4:57:39 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 4:58:35 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE
], Message[*iI am doing undergration from NJITi*], Clien
tId[0]
<i>I am doing undergration from NJIT</i>
Apr 02, 2024 4:58:35 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
▯
```

bold and italic

Checklist Items (1)

#3 Demonstrate formatted message along with any others

COLLAPSE

Task #3 - Points: 1

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

Response:

Payloads refer to the data transmitted over a network or exchanged between different components of a system. The payload contains the actual instructions that need to be delivered or processed. It's the essential data that is being carried within a message.

Flip Payload:

The flip method is responsible for simulating a coin flip (either heads or tails).
The payload for the flip operation is the result of the coin flip, which is either "heads" or "tails".
For example, when a client triggers the flip command, the server generates a random number (0 or 1) to represent heads or tails. The resulting payload sent back to the client is the outcome of this coin flip, such as "flipped heads" or "flipped tails".

Roll Payload:

The roll method handles rolling dice based on the client's input.
The payload for the roll operation is the result of the dice roll, which is a random number within a specified range.
If the client rolls a single die (e.g., /roll 6), the payload would be the specific number rolled (e.g., "rolled 4").
If the client rolls multiple dice with different numbers of sides (e.g., /roll 2d6), the payload would be the total value obtained from rolling those dice (e.g., "rolled 8" if the two dice rolled show 3 and 5).

Demonstrate Roll Command (2 pts.)

COLLAPSE

^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshot of the following items

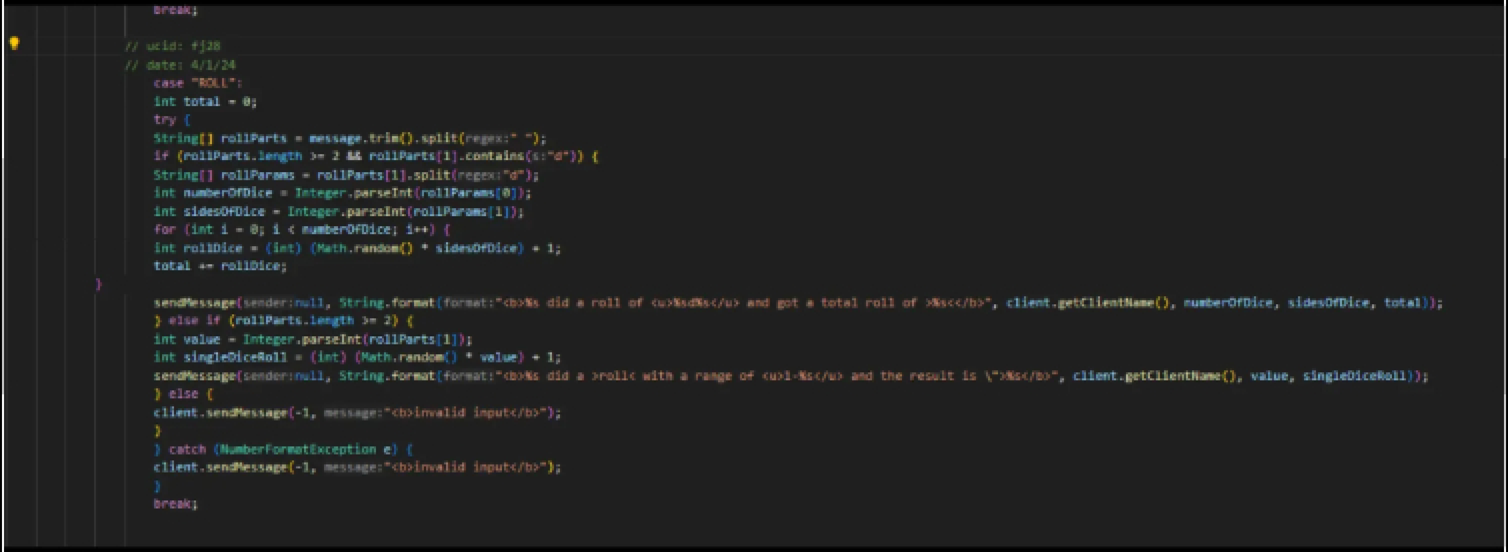
Checklist *The checkboxes are for your own tracking

#	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

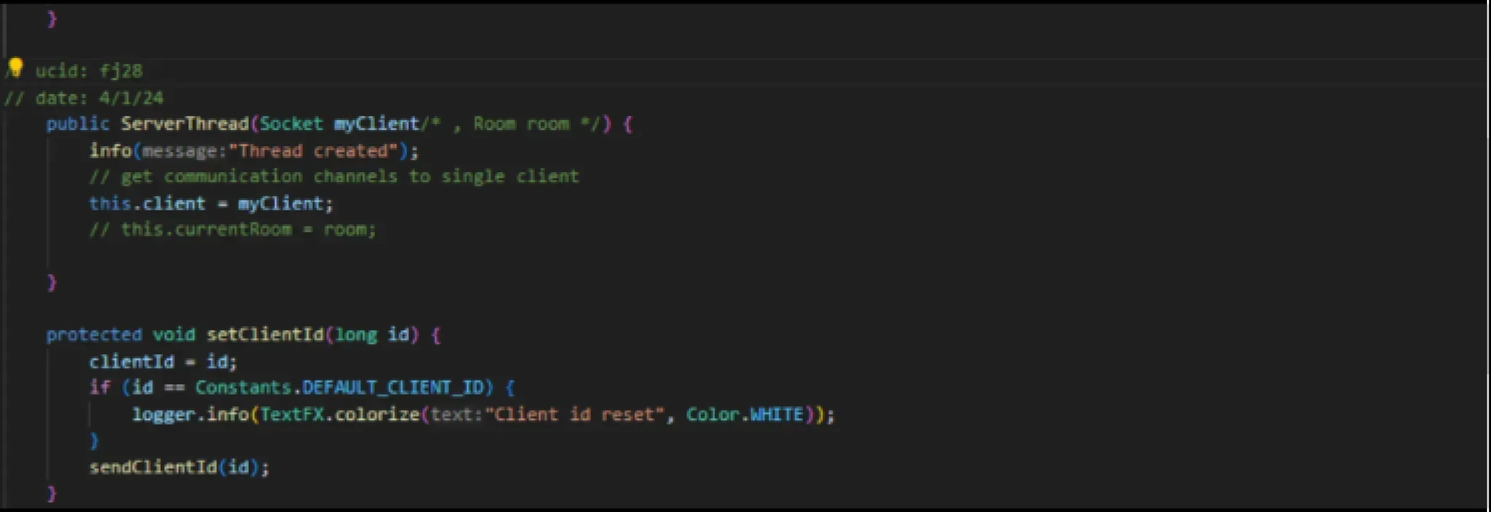


```
        break;

    // ucid: fj28
    // date: 4/1/24
        case "ROLL":
        int total = 0;
        try {
        String[] rollParts = message.trim().split(regex:" ");
        if (rollParts.length >= 2 && rollParts[1].contains(s:"d")) {
        String[] rollParams = rollParts[1].split(regex:"d");
        int numberOfDice = Integer.parseInt(rollParams[0]);
        int sidesOfDice = Integer.parseInt(rollParams[1]);
        for (int i = 0; i < numberOfDice; i++) {
        int rollDice = (int) (Math.random() * sidesOfDice) + 1;
        total += rollDice;
}
        sendMessage(sender:null, String.format(format:"<b>%s did a roll of <u>%sd%s</u> and got a total roll of >%s<</b>", client.getClientName(), numberOfDice, sidesOfDice, total));
        } else if (rollParts.length >= 2) {
        int value = Integer.parseInt(rollParts[1]);
        int singleDiceRoll = (int) (Math.random() * value) + 1;
        sendMessage(sender:null, String.format(format:"<b>%s did a >roll< with a range of <u>1-%s</u> and the result is \">%s</b>", client.getClientName(), value, singleDiceRoll));
        } else {
        client.sendMessage(-1, message:"<b>invalid input</b>");
        }
        } catch (NumberFormatException e) {
        client.sendMessage(-1, message:"<b>invalid input</b>");
        }
        break;
```

Room handling the roll action correctly for both scenarios (/roll # and /roll #d#) including the message going back out to all clients

Checklist Items (1)

#3 Room handling the roll action correctly for both scenarios (/roll # and /roll #d#) including the message going back out to all clients



```
    }

/ ucid: fj28
// date: 4/1/24
    public ServerThread(Socket myClient/* , Room room */) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
        // this.currentRoom = room;

    }

    protected void setClientId(long id) {
        clientId = id;
        if (id == Constants.DEFAULT_CLIENT_ID) {
            logger.info(TextFX.colorize(text:"Client id reset", Color.WHITE));
        }
        sendClientId(id);
    }
```

ServerThread code receiving the payload and passing it to the Room

Checklist Items (1)

#2 ServerThread code receiving the payload and passing it to the Room

^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:

Message Flow from Client to Room:

The client sends a message with the roll command (e.g., /FLIP or /ROLL 2d6).
The ServerThread representing the client receives the message and forwards it to the Room object associated with the client's current room.
Handling FLIP Command:

When the message starts with /FLIP, the processCommands method in the Room class recognizes it as a command.
It generates a random number (coin) between 0 and 1.
Based on the random value, it constructs a message indicating whether the flip landed on "Heads" or "Tails".
The sendMessage method is then called to broadcast this message to all clients in the room.
Handling ROLL Command:

When the message starts with /ROLL, the processCommands method in the Room class recognizes it as a command.
It parses the message to extract the roll parameters (e.g., 2d6 means rolling 2 six-sided dice).
It generates random numbers for each die roll based on the parameters.
It calculates the total roll and constructs a message indicating the result, including the individual dice rolls if applicable.
The sendMessage method is called to broadcast this message to all clients in the room.
Broadcasting Messages:

The sendMessage method formats the message based on certain symbols in the message (e.g., *b for bold, #r for red color).
It checks if the message contains special formatting symbols and processes them accordingly to add HTML tags for formatting.
After processing the message, it broadcasts it to all clients in the room using the ServerThread objects stored in the clients list.
Each client receives the message and displays it in their interface, applying any formatting specified in the message.

● Demonstrate Flip Command (1 pt.)

^COLLAPSE ^

● ^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshot of the following items

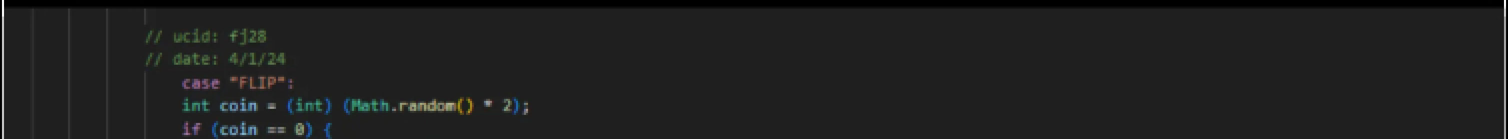
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Client code that captures the command and converts it to a 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

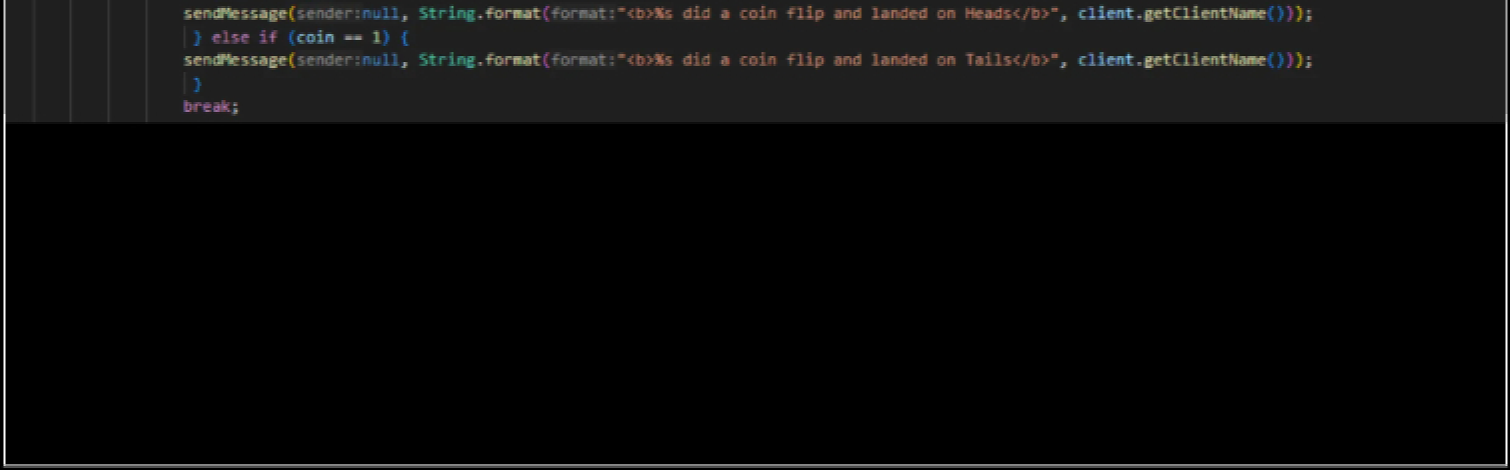
Task Screenshots:

Gallery Style: Large View

Small Medium Large



```
// ucid: fj28
// date: 4/1/24
    case "FLIP":
    int coin = (int) (Math.random() * 2);
    if (coin == 0) {
```

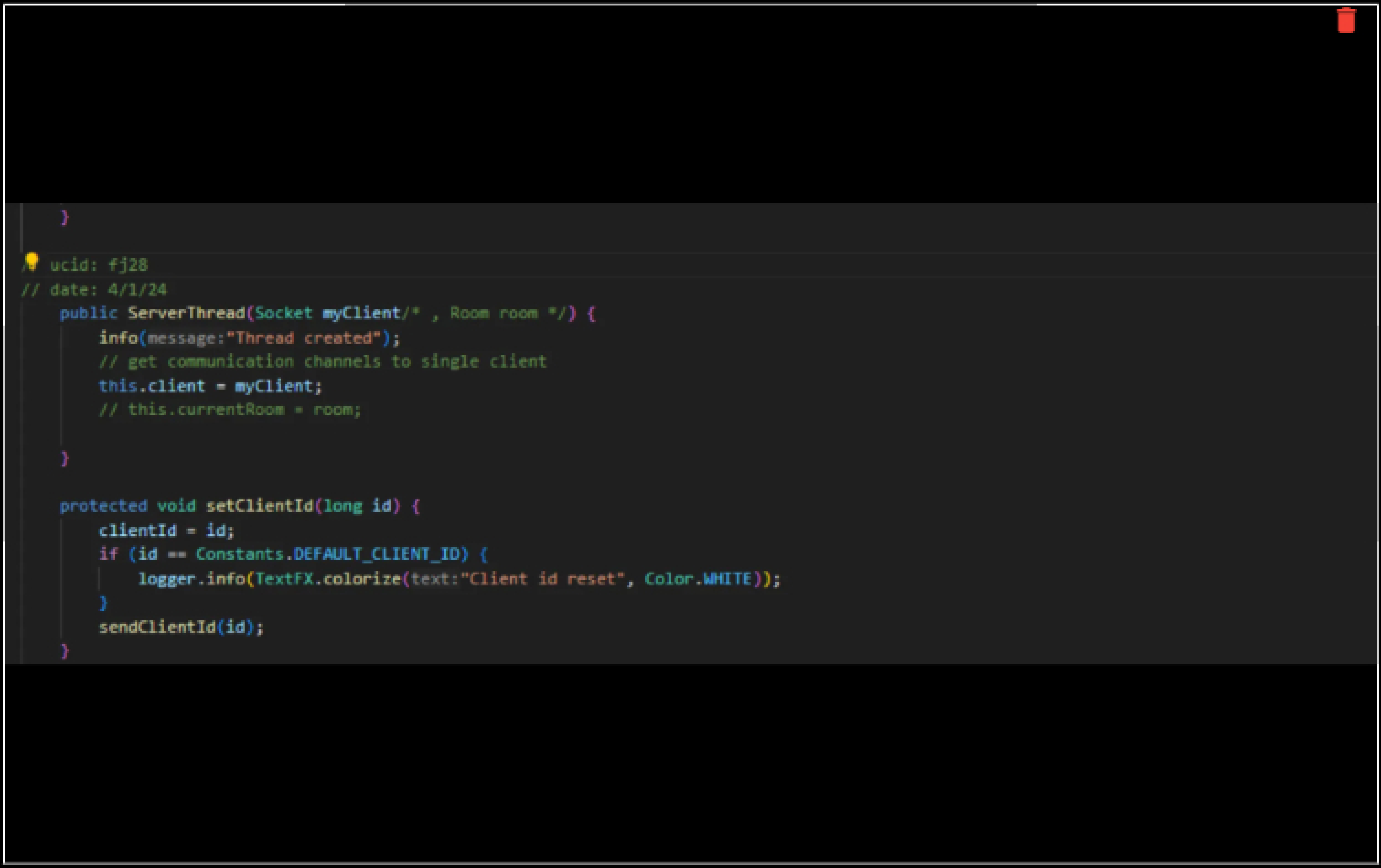


```
                sendMessage(sender:null, String.format(format:"<b>%s did a coin flip and landed on Heads</b>", client.getClientName()));
                } else if (coin == 1) {
                sendMessage(sender:null, String.format(format:"<b>%s did a coin flip and landed on Tails</b>", client.getClientName()));
                }
                break;
```

Room handling the flip action correctly

Checklist Items (1)

#3 Room handling the flip action correctly



```
    }

// ucid: fj28
// date: 4/1/24
    public ServerThread(Socket myClient/* , Room room */) {
        info(message:"Thread created");
        // get communication channels to single client
        this.client = myClient;
        // this.currentRoom = room;

    }

    protected void setClientId(long id) {
        clientId = id;
        if (id == Constants.DEFAULT_CLIENT_ID) {
            logger.info(TextFX.colorize(text:"Client id reset", Color.WHITE));
        }
        sendClientId(id);
    }
```

ServerThread receiving the payload and passing it to the Room

Checklist Items (4)

#1 Client code that captures the command and converts it to a payload

#2 ServerThread receiving the payload and passing it to the Room

#4 Code screenshots should include ucid and date comment

#5 Each screenshot should be clearly captioned

● 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:

Client Initiates the Flip Command:

The client sends a message containing the flip command (e.g., /FLIP) to the server.
Server Receives and Routes the Command:

The server, specifically the Room object associated with the client's current room, receives the flip command message.
Processing the Flip Command in the Room:

The processCommands method in the Room class identifies the received message as a command due to the /FLIP prefix.
It generates a random number (coin) between 0 and 1, simulating a coin flip where 0 represents "Heads" and 1 represents "Tails".
Constructing the Flip Result Message:

Based on the random value generated, the Room constructs a message indicating the result of the coin flip (Heads or Tails).
Broadcasting the Flip Result:

The sendMessage method is called to broadcast this result message to all clients in the room.
Message Flow to Other Users:

The broadcasted message containing the result of the coin flip is received by all users connected to the same room, including the client who initiated the flip command.
Each client's interface displays the flip result message, allowing all users to see the outcome of the coin flip.

● Demonstrate Formatted Messages (4 pts.)

COLLAPSE

● 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

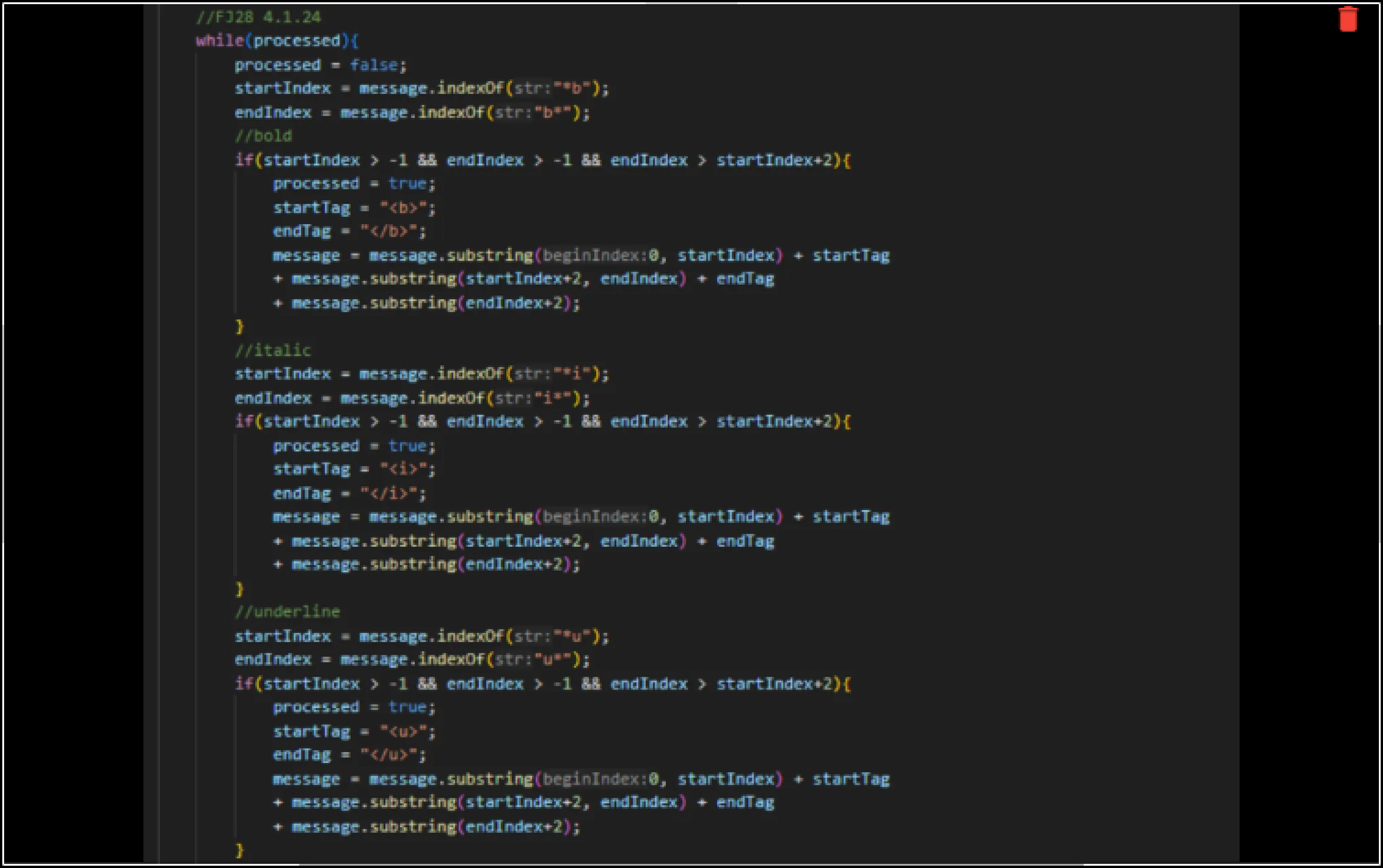
Checklist *The checkboxes are for your own tracking

#	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



```
//FJ28 4.1.24
while(processed){
    processed = false;
    startIndex = message.indexOf(str:"*b");
    endIndex = message.indexOf(str:"b*");
    //bold
    if(startIndex > -1 && endIndex > -1 && endIndex > startIndex+2){
        processed = true;
        startTag = "<b>";
        endTag = "</b>";
        message = message.substring(beginIndex:0, startIndex) + startTag
        + message.substring(startIndex+2, endIndex) + endTag
        + message.substring(endIndex+2);
    }
    //italic
    startIndex = message.indexOf(str:"*i");
    endIndex = message.indexOf(str:"i*");
    if(startIndex > -1 && endIndex > -1 && endIndex > startIndex+2){
        processed = true;
        startTag = "<i>";
        endTag = "</i>";
        message = message.substring(beginIndex:0, startIndex) + startTag
        + message.substring(startIndex+2, endIndex) + endTag
        + message.substring(endIndex+2);
    }
    //underline
    startIndex = message.indexOf(str:"*u");
    endIndex = message.indexOf(str:"u*");
    if(startIndex > -1 && endIndex > -1 && endIndex > startIndex+2){
        processed = true;
        startTag = "<u>";
        endTag = "</u>";
        message = message.substring(beginIndex:0, startIndex) + startTag
        + message.substring(startIndex+2, endIndex) + endTag
        + message.substring(endIndex+2);
    }
```

Room code processing for bold Room code processing for italic Room code processing for underline

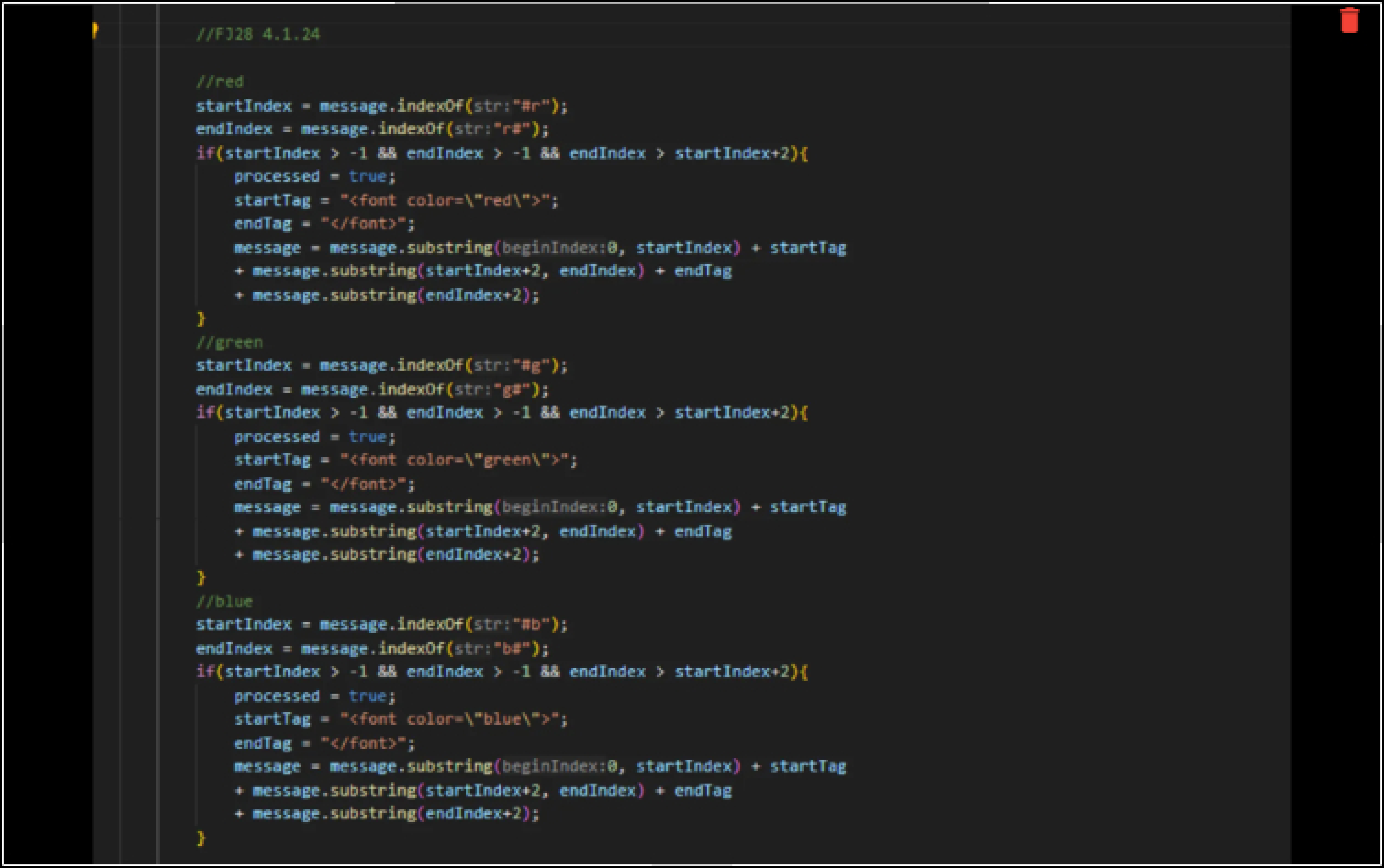
Checklist Items (3)

#1 Room code processing for bold

#2 Room code processing for italic

#3 Room code processing for underline

#3 Room code processing for underline



```java
//FJ28 4.1.24

//red
startIndex = message.indexOf(str:"#r");
endIndex = message.indexOf(str:"r#");
if(startIndex > -1 && endIndex > -1 && endIndex > startIndex+2){
    processed = true;
    startTag = "<font color=\"red\">";
    endTag = "</font>";
    message = message.substring(beginIndex:0, startIndex) + startTag
    + message.substring(startIndex+2, endIndex) + endTag
    + message.substring(endIndex+2);
}
//green
startIndex = message.indexOf(str:"#g");
endIndex = message.indexOf(str:"g#");
if(startIndex > -1 && endIndex > -1 && endIndex > startIndex+2){
    processed = true;
    startTag = "<font color=\"green\">";
    endTag = "</font>";
    message = message.substring(beginIndex:0, startIndex) + startTag
    + message.substring(startIndex+2, endIndex) + endTag
    + message.substring(endIndex+2);
}
//blue
startIndex = message.indexOf(str:"#b");
endIndex = message.indexOf(str:"b#");
if(startIndex > -1 && endIndex > -1 && endIndex > startIndex+2){
    processed = true;
    startTag = "<font color=\"blue\">";
    endTag = "</font>";
    message = message.substring(beginIndex:0, startIndex) + startTag
    + message.substring(startIndex+2, endIndex) + endTag
    + message.substring(endIndex+2);
}
```

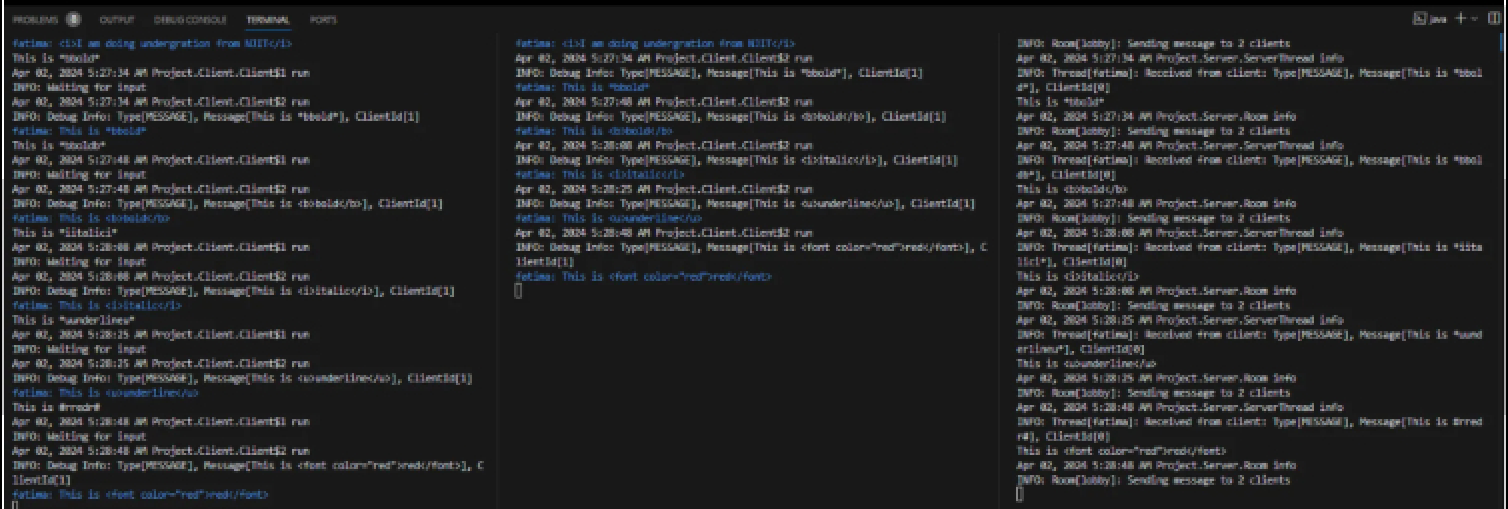
Room code processing for color (at least R, G, B or support for hex codes)

Checklist Items (3)

#4 Room code processing for color (at least R, G, B or support for hex codes)

#7 Code screenshots should include ucid and date comment

#8 Each screenshot should be clearly captioned



```
fatima: <i>I am doing undergration from NJIT</i>
This is *bbold*
Apr 02, 2024 5:27:34 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 5:27:34 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is *bbold*], ClientId[1]
fatima: This is *bbold*
This is *bboldb*
Apr 02, 2024 5:27:48 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 5:27:48 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <b>bold</b>], ClientId[1]
fatima: This is <b>bold</b>
This is *iitalici*
Apr 02, 2024 5:28:08 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 5:28:08 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <i>italic</i>], ClientId[1]
fatima: This is <i>italic</i>
This is *uunderlineu*
Apr 02, 2024 5:28:25 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 5:28:25 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <u>underline</u>], ClientId[1]
fatima: This is <u>underline</u>
This is #rredr#
Apr 02, 2024 5:28:48 AM Project.Client.Client$1 run
INFO: Waiting for input
Apr 02, 2024 5:28:48 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <font color="red">red</font>], C
lientId[1]
fatima: This is <font color="red">red</font>
```

```
fatima: <i>I am doing undergration from NJIT</i>
Apr 02, 2024 5:27:34 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is *bbold*], ClientId[1]
fatima: This is *bbold*
Apr 02, 2024 5:27:48 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <b>bold</b>], ClientId[1]
fatima: This is <b>bold</b>
Apr 02, 2024 5:28:08 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <i>italic</i>], ClientId[1]
fatima: This is <i>italic</i>
Apr 02, 2024 5:28:25 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <u>underline</u>], ClientId[1]
fatima: This is <u>underline</u>
Apr 02, 2024 5:28:48 AM Project.Client.Client$2 run
INFO: Debug Info: Type[MESSAGE], Message[This is <font color="red">red</font>], C
lientId[1]
fatima: This is <font color="red">red</font>
```

```
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 5:27:34 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE], Message[This is *bbol
d*], ClientId[0]
This is *bbold*
Apr 02, 2024 5:27:34 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 5:27:48 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE], Message[This is *bbol
db*], ClientId[0]
This is <b>bold</b>
Apr 02, 2024 5:27:48 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 5:28:08 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE], Message[This is *iita
lici*], ClientId[0]
This is <i>italic</i>
Apr 02, 2024 5:28:08 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 5:28:25 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE], Message[This is *uund
erlineu*], ClientId[0]
This is <u>underline</u>
Apr 02, 2024 5:28:25 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
Apr 02, 2024 5:28:48 AM Project.Server.ServerThread info
INFO: Thread[fatima]: Received from client: Type[MESSAGE], Message[This is #rred
r#], ClientId[0]
This is <font color="red">red</font>
Apr 02, 2024 5:28:48 AM Project.Server.Room info
INFO: Room[lobby]: Sending message to 2 clients
```

Show each one working individually and one showing a combination of all of the formats and 1 color from the terminal

Checklist Items (2)

#5 Show each one working individually and one showing a combination of all of the formats and 1 color from the terminal

#6 Must not rely on the user typing html characters, but the output can be html characters

^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:

Special Characters for Formatting:

Bold Text: *b at the start and b* at the end of the text.
Italic Text: *i at the start and i* at the end of the text.
Underline Text: *u at the start and u* at the end of the text.
Red Text: #r at the start and r# at the end of the text.
Green Text: #g at the start and g# at the end of the text.
Blue Text: #b at the start and b# at the end of the text.
Logic Overview:
The provided code processes a message and applies HTML formatting based on certain patterns in the message. Here's how the logic works:

It initializes variables startTag, endTag, startIndex, endIndex, and processed.
The while loop continues until processed is false.
Inside the loop, it first sets processed to false.
Then, it searches for specific patterns using indexOf method, such as *b and b* for bold text, *i and i* for italic text, and similar patterns for other formatting types.
If a pattern is found (startIndex and endIndex are not -1 and endIndex is greater than startIndex+2), it sets processed to true, determines the appropriate HTML start and end tags (startTag and endTag), and replaces the text within the pattern with the corresponding HTML tags using substring method.
The loop continues until no more patterns are found or until processed remains false.

Misc (1 pt.)

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/fj29/-Milestone-Milestone-1/pull/2

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Issue: Handling multiple formatting options (bold, italic, underline, color) within a message can increase code complexity.
Learning: Breaking down the code into smaller, manageable functions

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

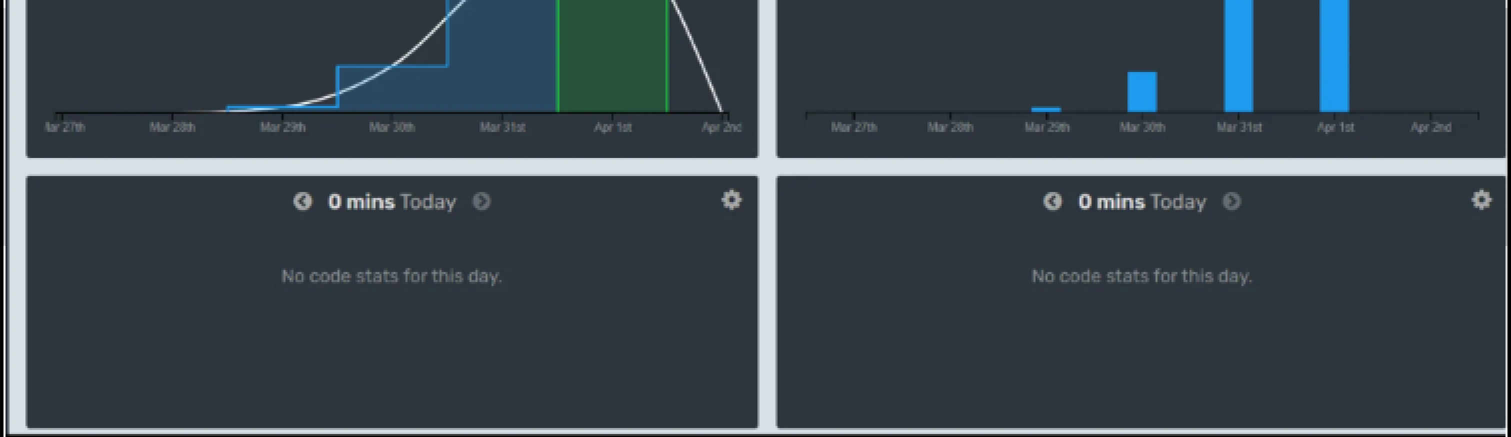
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots:

Gallery Style: Large View

Small Medium Large

6 hrs 14 mins over the Last 7 Days.



0 mins Today

No code stats for this day.

0 mins Today

No code stats for this day.

wakatime

End of Assignment