Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-milestone-2-chatroom-2024/grade/sb57

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

Submissions:

Submission Selection

1 Submission [active] 4/30/2024 7:40:42 PM

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

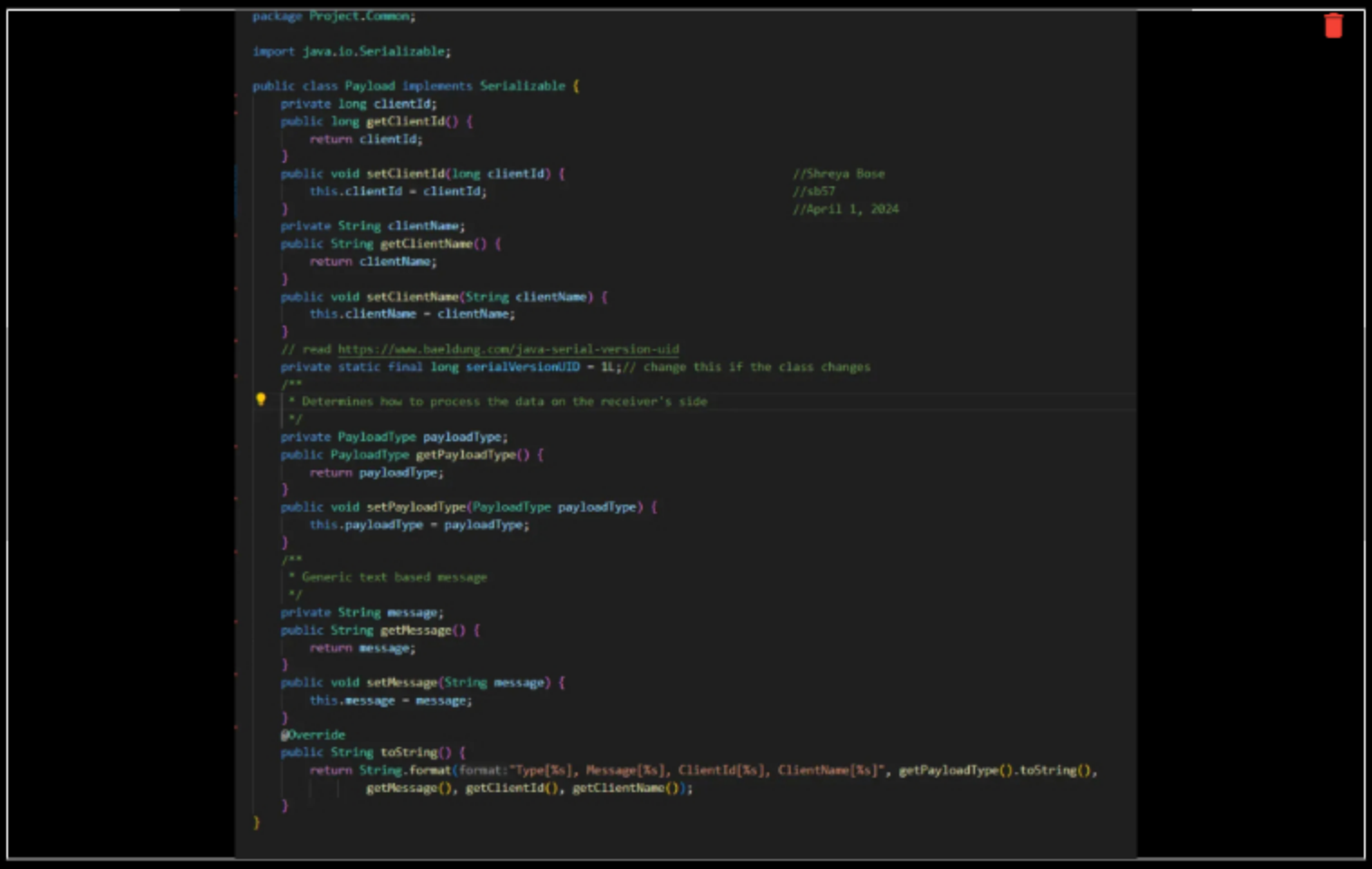
Checklist *The checkbox are for your own tracking

#	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



```java
package Project.Common;

import java.io.Serializable;

public class Payload implements Serializable {
    private long clientId;
    public long getClientId() {
        return clientId;
    }
    public void setClientId(long clientId) {                    //Shreya Bose
        this.clientId = clientId;                               //sb57
    }                                                           //April 1, 2024
    private String clientName;
    public String getClientName() {
        return clientName;
    }
    public void setClientName(String clientName) {
        this.clientName = clientName;
    }
    // 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;
    }
    public void setMessage(String message) {
        this.message = message;
    }
    @Override
    public String toString() {
        return String.format(format:"Type[%s], Message[%s], ClientId[%s], ClientName[%s]", getPayloadType().toString(),
                getMessage(), getClientId(), getClientName());
    }
}
```

Screenshot of Payload.java

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

```java
package Project.Common;

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

//Shreya Bose
//sb57
//April 1, 2024
```

Screenshot of PayloadType.java

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

```java
package Project.Common;

import java.util.ArrayList;
import java.util.List;

public class RoomResultsPayload extends Payload {
    private List<String> rooms = new ArrayList<String>();
    /**
     * Used to limit the returned result set. Added after video recording
     */
    private int limit = 10;

    public int getLimit() {                         //Shreya Bose
        return limit;                               //sb57
    }                                               //April 1, 2024

    public void setLimit(int limit) {
        this.limit = limit;
    }

    public RoomResultsPayload() {
        setPayloadType(PayloadType.LIST_ROOMS);
    }

    public List<String> getRooms() {
        return rooms;
    }

    public void setRooms(List<String> rooms) {
        this.rooms = rooms;
    }
}
```

Screenshot of RoomResultsPayload.java

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

```java
package Project.Common;

public class ConnectionPayload extends Payload {
                                                    //Shreya Bose
    public ConnectionPayload() {                    //sb57
        setPayloadType(PayloadType.CLIENT_ID);      //April 1, 2024
    }

    public ConnectionPayload(boolean isConnected) {
        setPayloadType(isConnected ? PayloadType.CONNECT : PayloadType.DISCONNECT);
    }

    /**
     * Who the payload is from
     */
    private String clientName;

    public String getClientName() {
        return clientName;
    }

    public void setClientName(String clientName) {
        this.clientName = clientName;
    }

    @Override
    public String toString() {
        return super.toString() + ", Client name " + getClientName();
    }
}
```

Screenshot of ConnectionPayload.java

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

COLLAPSE

Task #2 - Points: 1

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

Checklist *The checkboxes are for your own tracking

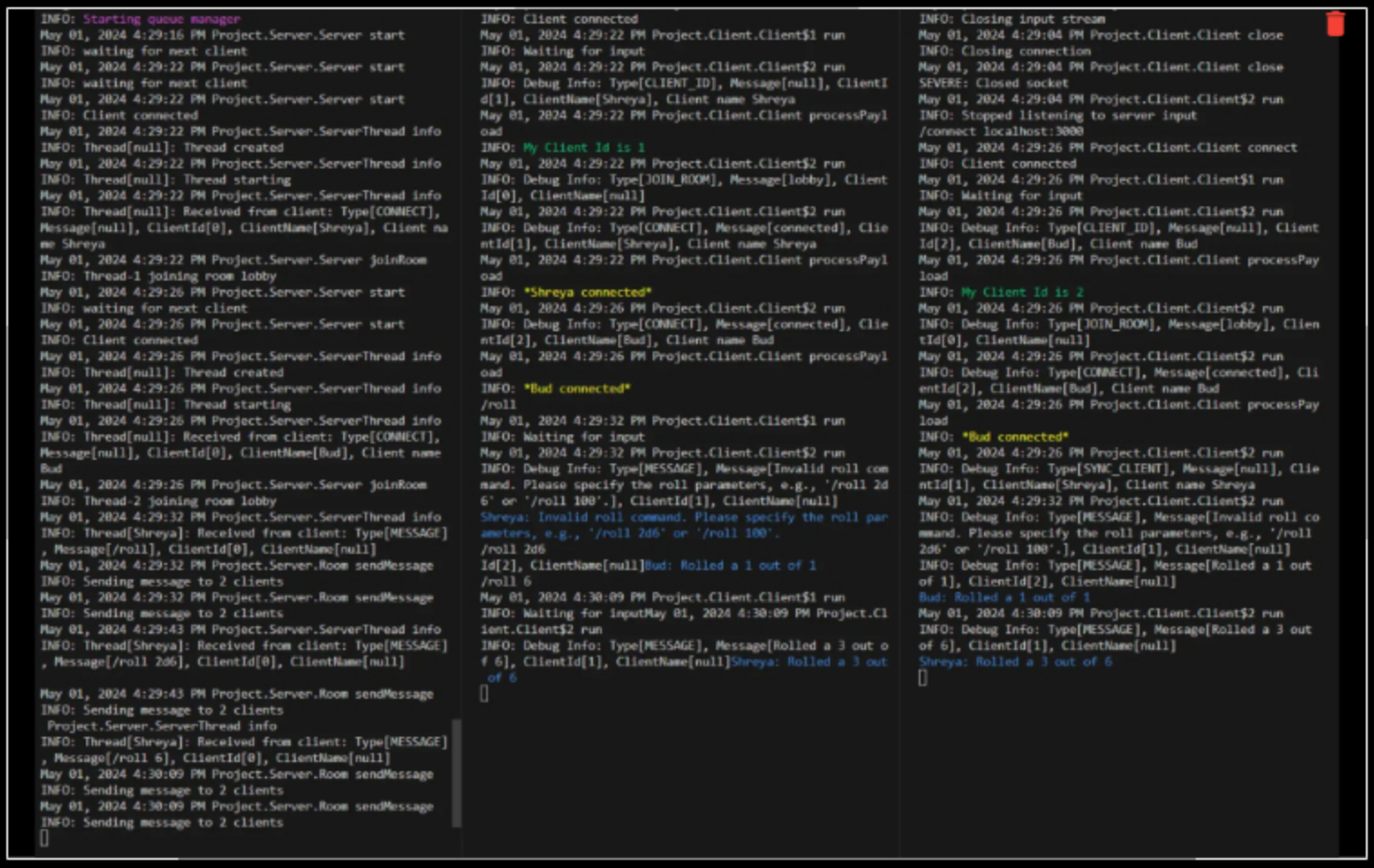
#	Points	Details

#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



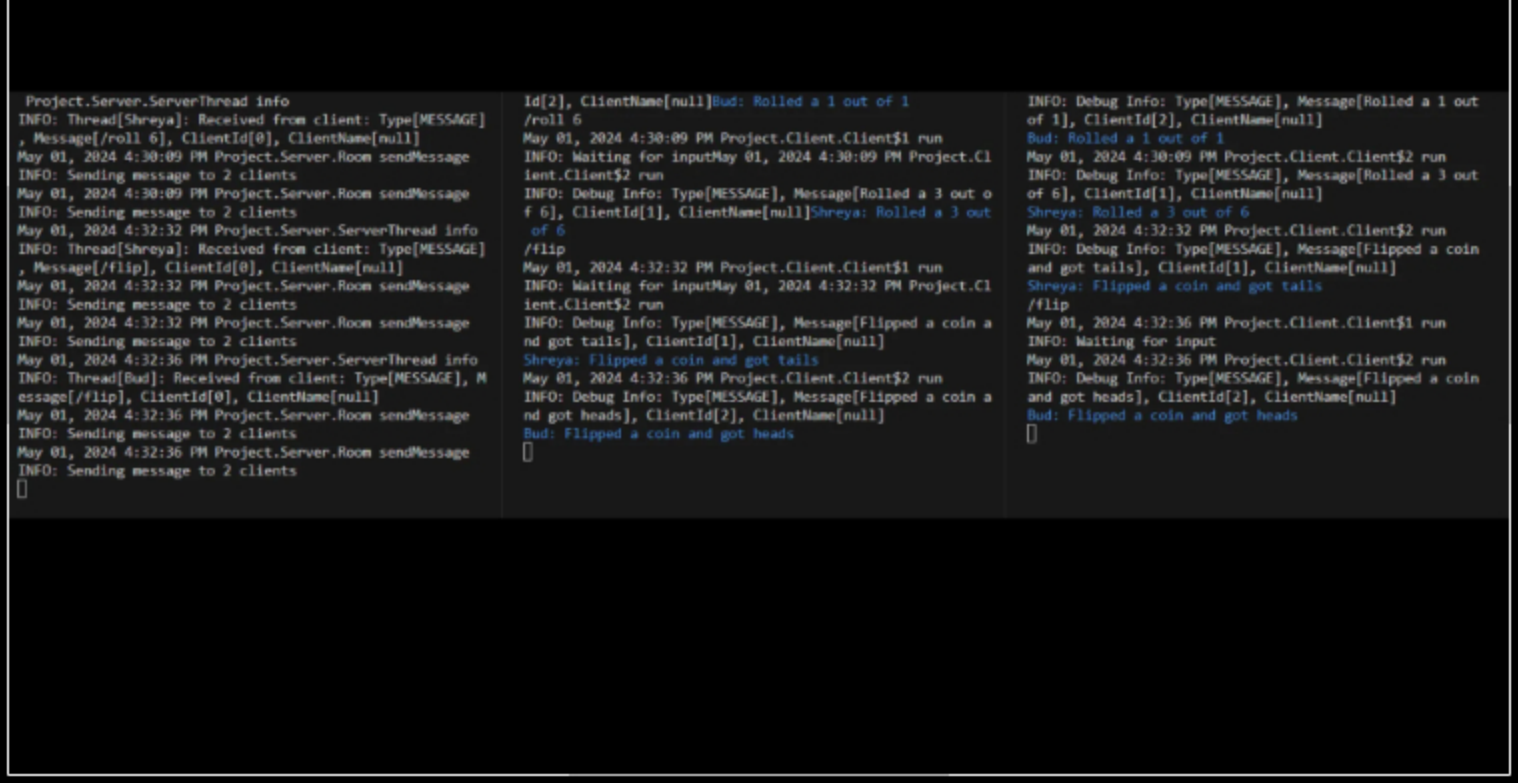
Screenshot demonstrating both versions of Roll.

Checklist Items (3)

#2 Demonstrate roll (both versions)

#3 Demonstrate formatted message along with any others

#4 Each screenshot should be clearly captioned



Screenshot demonstrating Flip.

Checklist Items (3)

#1 Demonstrate flip

#3 Demonstrate formatted message along with any others

#4 Each screenshot should be clearly captioned

COLLAPSE

Task #3 - Points: 1

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

Response:

The purpose of payloads is to handle the data and make sure it is communicated correctly between client and server. The flip payload only handles heads or tails while the roll payload handles the two different formats.

COLLAPSE

Demonstrate Roll Command (2 pts.)

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

```
} else if (text.equalsIgnoreCase(anotherString:"/flip")) {
    try {
        Payload p = new Payload();
        p.setPayloadType(PayloadType.FLIP);
        sendMessage(p);                                    //Shreya Bose
    } catch (IOException e) {                              //sb57
        e.printStackTrace();                               //April 1, 2024
    }
    return true;
} else if (text.startsWith(prefix:"/roll ")) {
    try {
        Payload p = new Payload();
        p.setPayloadType(PayloadType.ROLL);
        p.setMessage(text.substring(beginIndex:6));
        sendMessage(p);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return true;
```

Screenshot of Client.java

Checklist Items (3)

#1 Client code that captures the command and converts it to a RollPayload (or equivalent) for both scenarios /roll # and /roll #d#

#4 Code screenshots should include ucid and date comment

#5 Each screenshot should be clearly captioned

```java
case FLIP:                                          //Shreya Bose
case ROLL:                                          //sb57
    if (currentRoom != null) {                      //April 1, 2024
        String command = (p.getPayloadType() == PayloadType.FLIP ? "/flip" : "/roll " + p.getMessage());
        currentRoom.processCommands(command, this);
    }
    break;
```

Screenshot of ServerThread.java

Checklist Items (3)

#2 ServerThread code 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

```java
case ROLL:
if (comm2.length > 1) {
    diceRoll = comm2[1];
    if (diceRoll.contains(s:"d")) {
        String[] parts = diceRoll.split(regex:"d");                 //Shreya Bose
        int numberOfDice = Integer.parseInt(parts[0]);              //sb57
        int sides = Integer.parseInt(parts[1]);                     //April 1, 2024
        int total = 0;
        StringBuilder results = new StringBuilder("Rolling " + numberOfDice + "d" + sides + ": ");
        for(int i = 0; i < numberOfDice; i++) {
            int roll = (int) (Math.random() * sides) + 1;
            total += roll;
            results.append(roll).append(str:" ");
        }
        results.append(str:"Total: ").append(total);
        sendMessage(client, results.toString());
    } else {
        int max = Integer.parseInt(diceRoll);
        int roll = (int) (Math.random() * max) + 1;
        sendMessage(client, "Rolled a " + roll + " out of " + max);
    }
} else {
    sendMessage(client, message:"Invalid roll command. Please specify the roll parameters, e.g., '/roll 2d6' or '/roll 100'.");
}
break;
```

Screenshot of Room.java

Checklist Items (3)

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

#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 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 sends the /roll payload, specifying whether it is in the 1-100 format or #d# format. It is passed to the Room class whether it is processed. The server receives the message and uses the format that was processed in Room.

COLLAPSE

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

Gallery Style: Large View

Small Medium Large

```
} else if (text.equalsIgnoreCase(anotherString:"/flip")) {
    try {
        Payload p = new Payload();
        p.setPayloadType(PayloadType.FLIP);
        sendMessage(p);                                    //Shreya Bose
    } catch (IOException e) {                              //sb57
        e.printStackTrace();                               //April 1, 2024
    }
    return true;
```

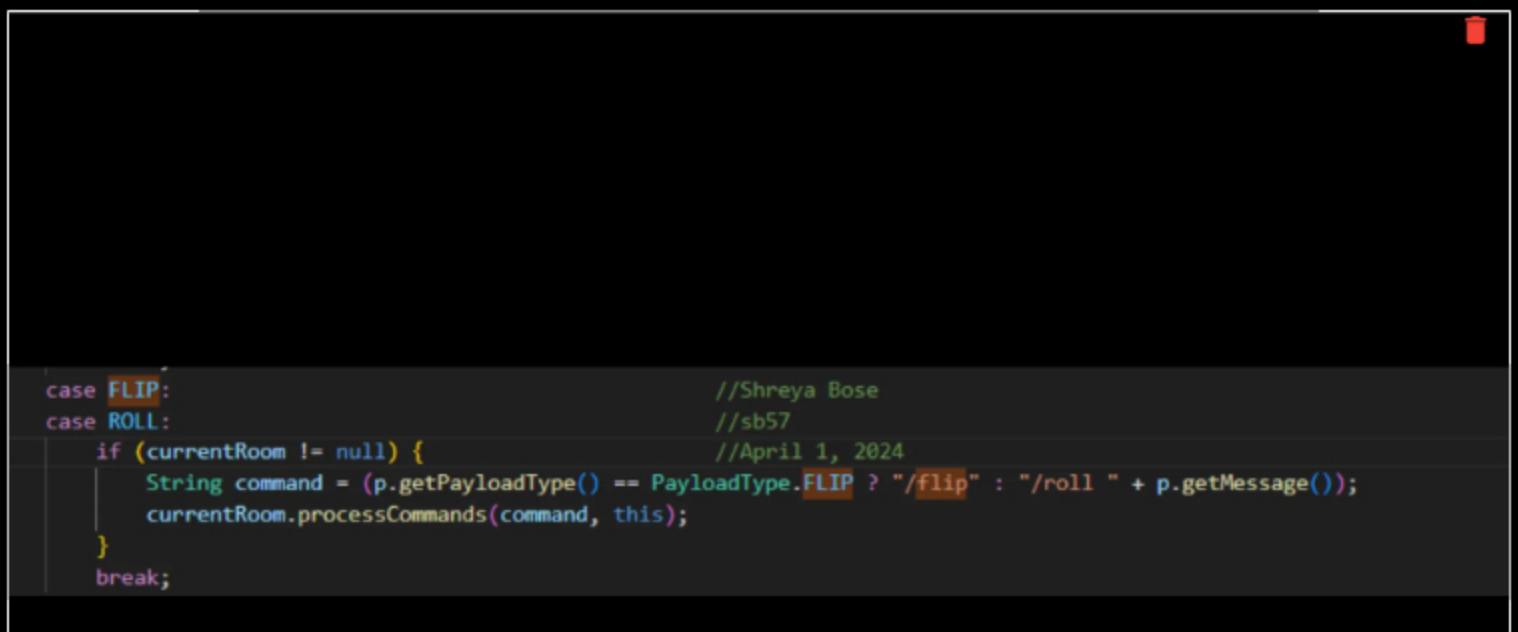
Screenshot of Client.java

Checklist Items (3)

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

#4 Code screenshots should include ucid and date comment

#5 Each screenshot should be clearly captioned



```
case FLIP:                                          //Shreya Bose
case ROLL:                                          //sb57
    if (currentRoom != null) {                      //April 1, 2024
        String command = (p.getPayloadType() == PayloadType.FLIP ? "/flip" : "/roll " + p.getMessage());
        currentRoom.processCommands(command, this);
    }
    break;
```

Screenshot of ServerThread.java

Checklist Items (3)

#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

```
case FLIP:
    Random random = new Random();                                   //Shreya Bose
    String result = random.nextBoolean() ? "heads" : "tails";       //sb57
    sendMessage(client, "Flipped a coin and got " + result);        //April 1, 2024
    break;
```

Screenshot of Room.java

Checklist Items (3)

#3 Room handling the flip action correctly

#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:

The client sends the /flip payload=. It is passed to the Room class whether it is processed. The server receives the message and uses the format that was processed in Room which is straightforward and spits out either heads or tails.

COLLAPSE

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

```
}                                                   //Shreya Bose
                                                    //sb56
                                                    //April 1, 2024

private String formatMessage(String message) {
    message = message.replaceAll(regex:"\\*([^\\*]+)\\*", TextFX.colorize(text:"$1", TextFX.Color.RED)); // Bold as red
    message = message.replaceAll(regex:"_([^_]+)_", TextFX.colorize(text:"$1", TextFX.Color.GREEN)); // Italics as green
    message = message.replaceAll(regex:"~([^~]+)~", TextFX.colorize(text:"$1", TextFX.Color.BLUE)); // Underline as blue
    return message;
}
```

Screenshot of Room.java

Checklist Items (7)

#1 Room code processing for bold

#2 Room code processing for italic

#3 Room code processing for underline

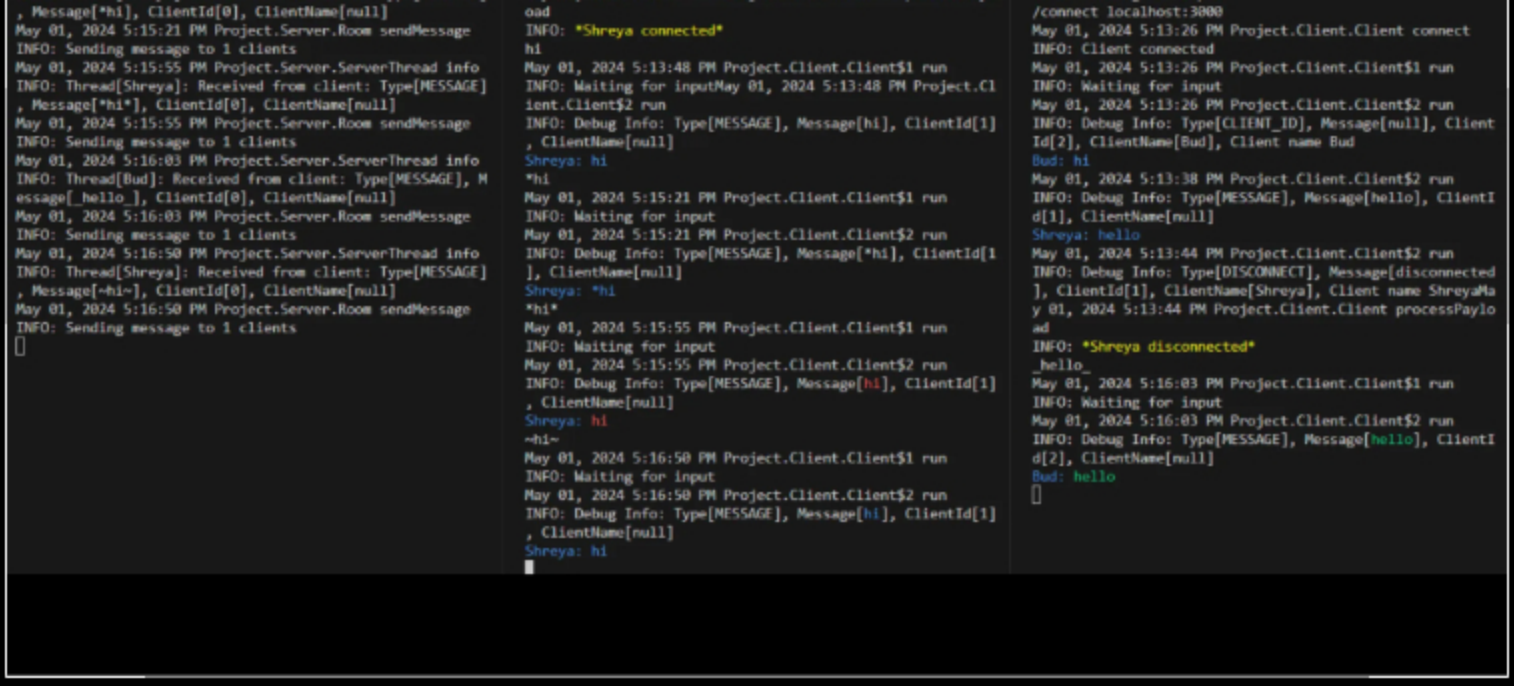
#4 Room code processing for color (at least R, G, B or support for hex codes)

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

#7 Code screenshots should include ucid and date comment

#8 Each screenshot should be clearly captioned

```
Project.Server.Server joinRoom
INFO: Thread-1 joining room test
May 01, 2024 5:13:48 PM Project.Server.ServerThread info
INFO: Thread[Shreya]: Received from client: Type[MESSAGE]
, Message[hi], ClientId[0], ClientName[null]
May 01, 2024 5:13:48 PM Project.Server.Room sendMessage
INFO: Sending message to 1 clients
May 01, 2024 5:15:21 PM Project.Server.ServerThread info
INFO: Thread[Shreya]: Received from client: Type[MESSAGE]
oad
INFO: *Shreya connected*
May 01, 2024 5:13:26 PM Project.Client.Client$2 run
INFO: Debug Info: Type[CONNECT], Message[connected], Clie
ntId[2], ClientName[Bud], Client name Bud
May 01, 2024 5:13:26 PM Project.Client.Client processPayl
CONNECT], Message[connected], ClientId[1], ClientName[Shr
eya], Client name Shreya
May 01, 2024 5:13:44 PM Project.Client.Client processPayl
May 01, 2024 5:12:49 PM Project.Client.Client$1 run
INFO: Listening for input
May 01, 2024 5:12:49 PM Project.Client.Client$1 run
INFO: Waiting for input
/name Bud
May 01, 2024 5:13:21 PM Project.Client.Client isName
INFO: Name set to Bud
May 01, 2024 5:13:21 PM Project.Client.Client$1 run
INFO: Waiting for input
```



Screenshot of terminal

Checklist Items (3)

#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

#8 Each screenshot should be clearly captioned

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 * translates to red bold text, the _ translates to green italic text, and the ~ translates to blue underlined text. The formatMessage function searches for these characters, processes, and then sends the formatted message to all clients.

COLLAPSE

Misc (1 pt.)

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/sb57-shreya/sb57-IT114-006/pull/10

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

This assignment was difficult for me as I am still trying to grasp an understanding of all of the code. Payloads were very confusing for me to understand.

COLLAPSE

Task #3 - Points: 1

Text: WakaTime Screenshot

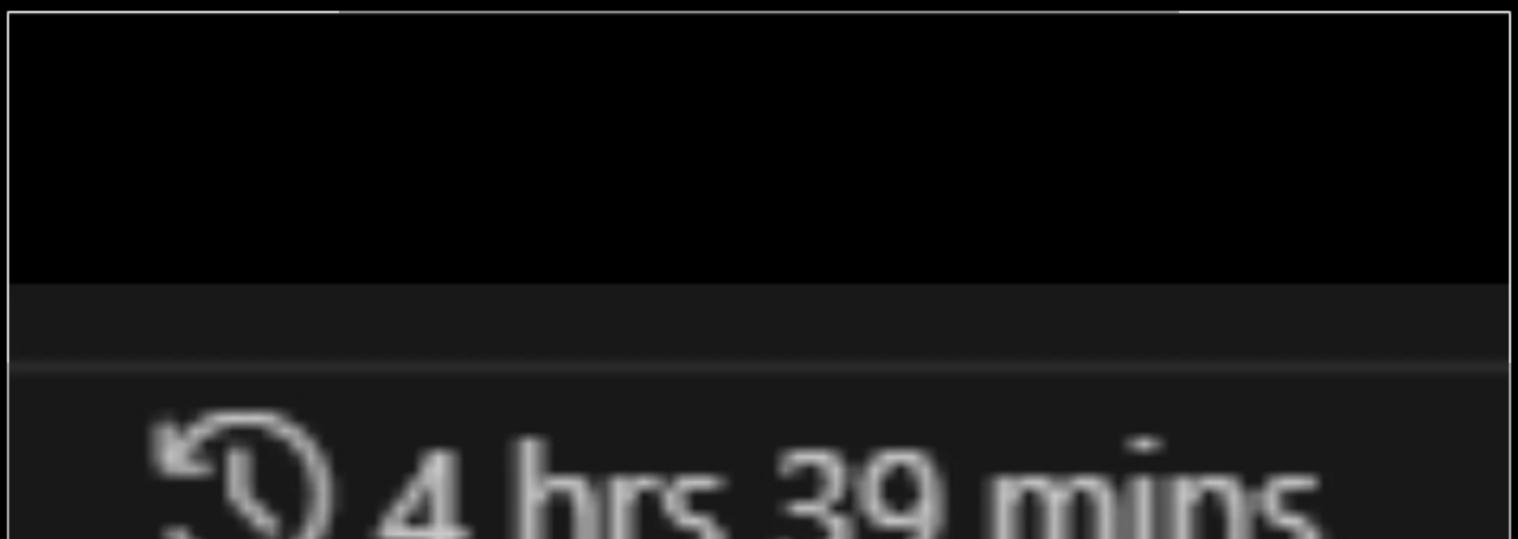
Details:

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



Screenshot of WakaTime

End of Assignment