

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-milestone-2-chatroom-2024/grade/rn364>

IT114-004-S2024 - [IT114] Milestone 2 Chatroom 2024

Submissions:

Submission Selection

1 Submission [active] 4/28/2024 9:06:00 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 ^

● Task #1 - Points: 1

^ COLLAPSE ^

Text: Screenshots of your Payload class and subclasses and PayloadType

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Payload, equivalent of RollPayload, and any others
<input type="checkbox"/> #2	1	Screenshots should include ucid and date comment
<input type="checkbox"/> #3	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

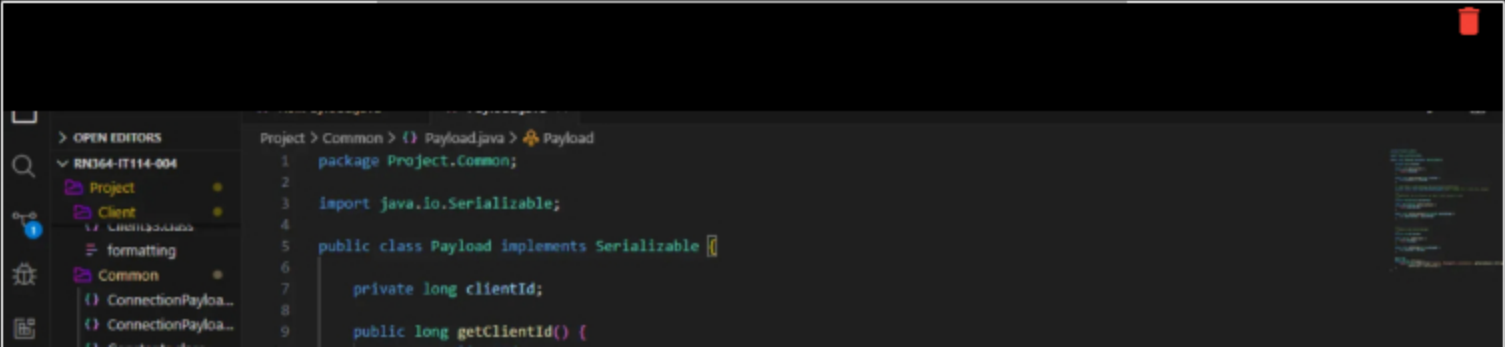
Small Medium Large

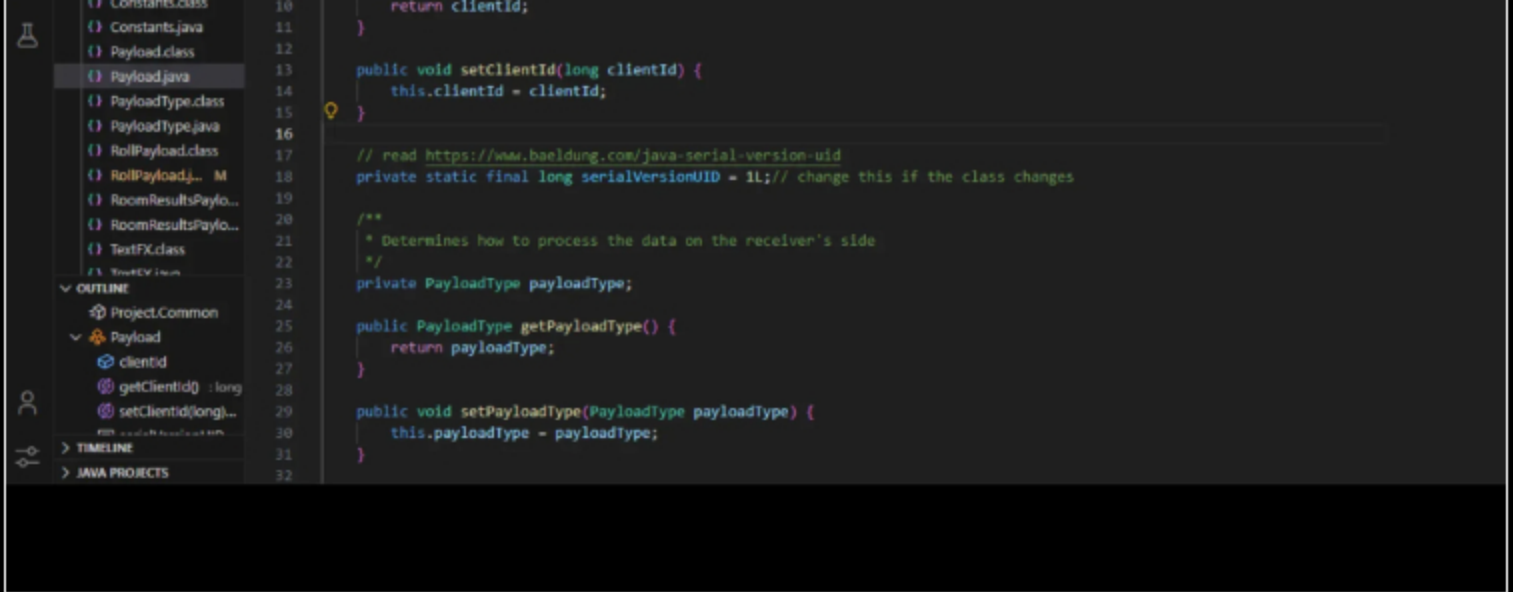


RollPayload, payload and payloadtype. does include my ucid //rn364

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





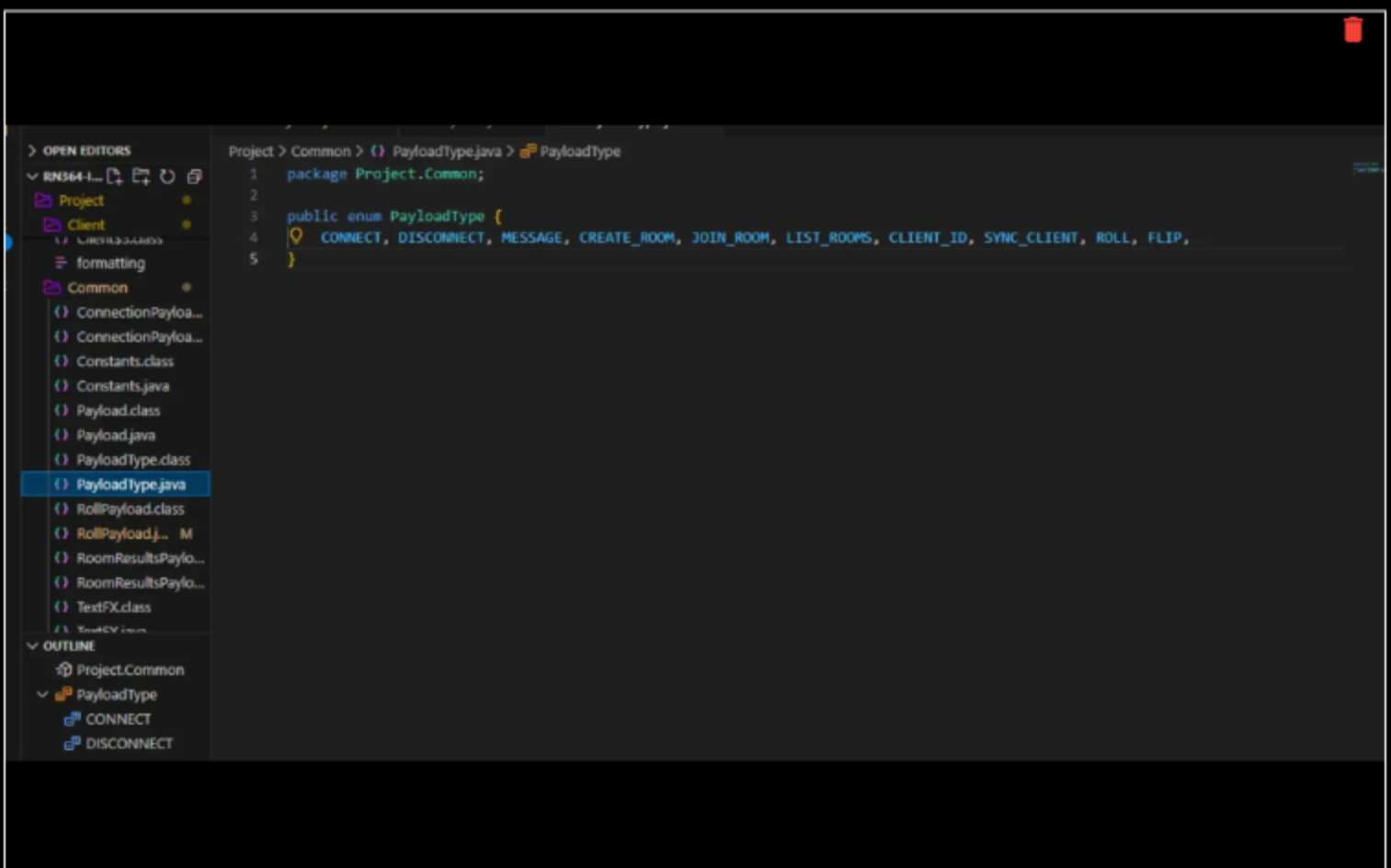
RollPayload, payload and payloadtype. does include my uuid //rn364

Checklist Items (3)

#1 Payload, equivalent of RollPayload, and any others

#2 Screenshots should include uuid and date comment

#3 Each screenshot should be clearly captioned



RollPayload, payload and payloadtype. does include my uuid //rn364

Checklist Items (3)

#1 Payload, equivalent of RollPayload, and any others

#2 Screenshots should include uuid and date comment

#3 Each screenshot should be clearly captioned

Task #2 - Points: 1

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

Checklist

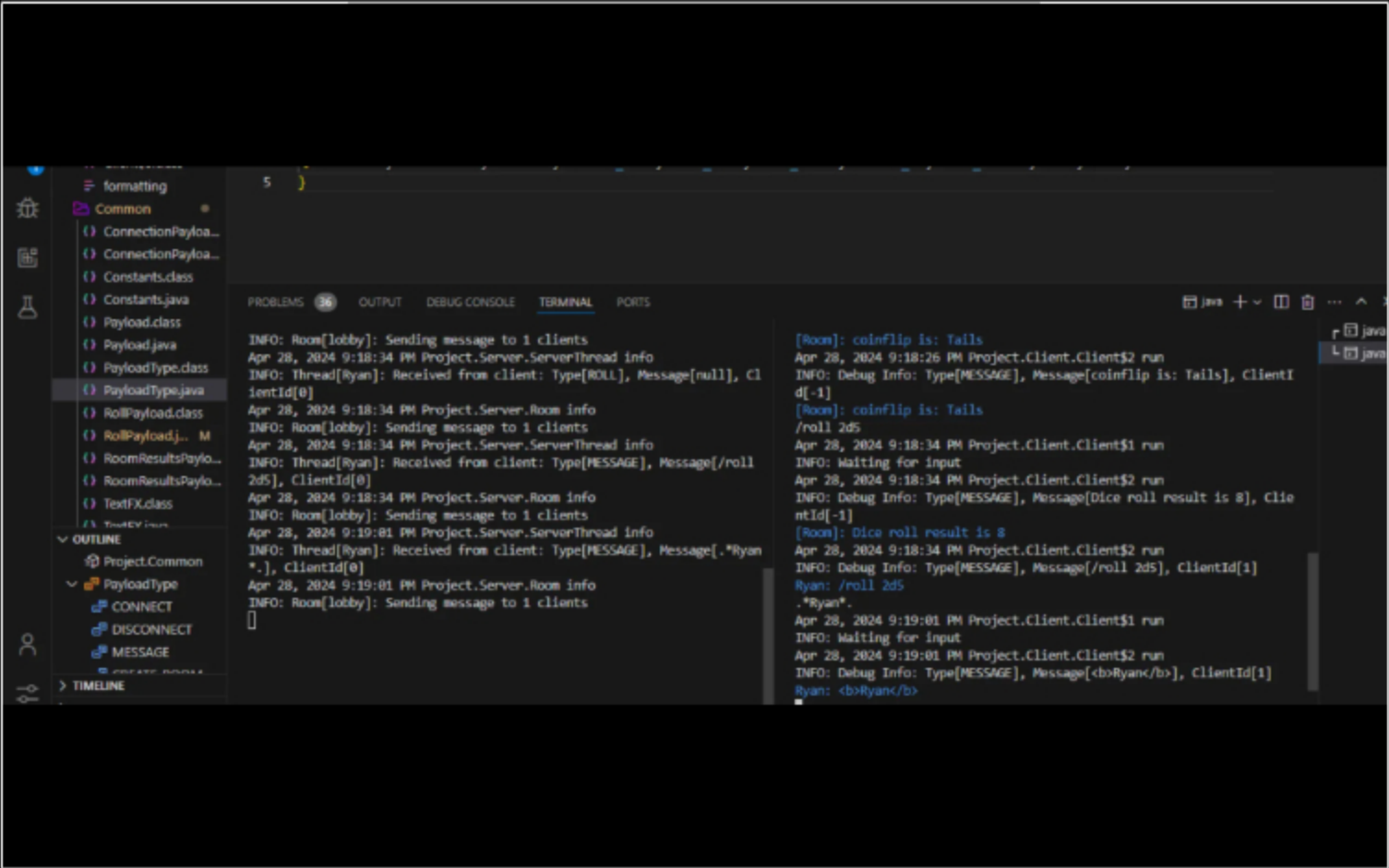
*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Demonstrate flip
<input type="checkbox"/> #2	1	Demonstrate roll (both versions)
<input type="checkbox"/> #3	1	Demonstrate formatted message along with any others
<input type="checkbox"/> #4	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Everything in the checklist

Checklist Items (4)

#1 Demonstrate flip

#2 Demonstrate roll (both versions)

#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:

Payload pretty much carries to messages between the clients and the server. It sends the commands, the messages and the status. When the command of the flip is made the client creates a payload for the command then send it to server. It the gets the command and processes it. Same with roll



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

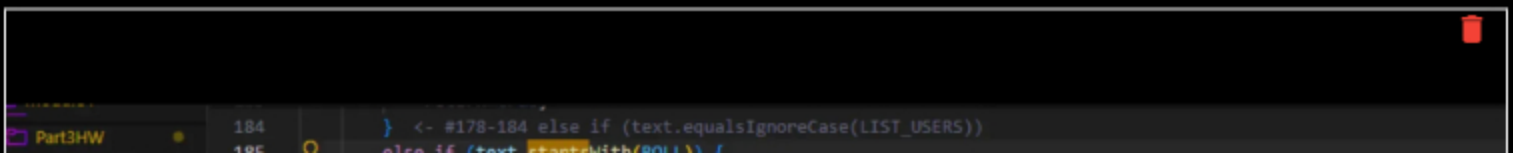
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



```

185 } else if (text.startsWith(ROLL)) {
186     String roll = text.replace(ROLL, replacement: "").trim();
187     String[] rollParts = roll.split(regex: "d");
188     if (rollParts.length >= 2) {
189         try {
190             int lower = Integer.parseInt(rollParts[0]);
191             int upper = Integer.parseInt(rollParts[1]);
192             sendRoll(lower, upper);
193         } catch (IOException e) {
194             System.out.println(TextFX.colorize(text: "Socket error", Color.RED));
195         } catch (Exception e) {
196             e.printStackTrace();
197         }
198     } <- #188-198 if (rollParts.length >= 2)
199     else {
200         try {
201             int lower = 1;
202             int upper = Integer.parseInt(rollParts[0]);
203             sendRoll(lower, upper);
204         } <- #200-204 try
205         catch (IOException e) {
206             System.out.println(TextFX.colorize(text: "Socket error", Color.RED));
207         }
208         catch (Exception e) {
209             e.printStackTrace();
210         }
211     } <- #199-211 else
212 } <- #185-212 else if (text.startsWith(ROLL))

```

Client code

Checklist Items (1)

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

```

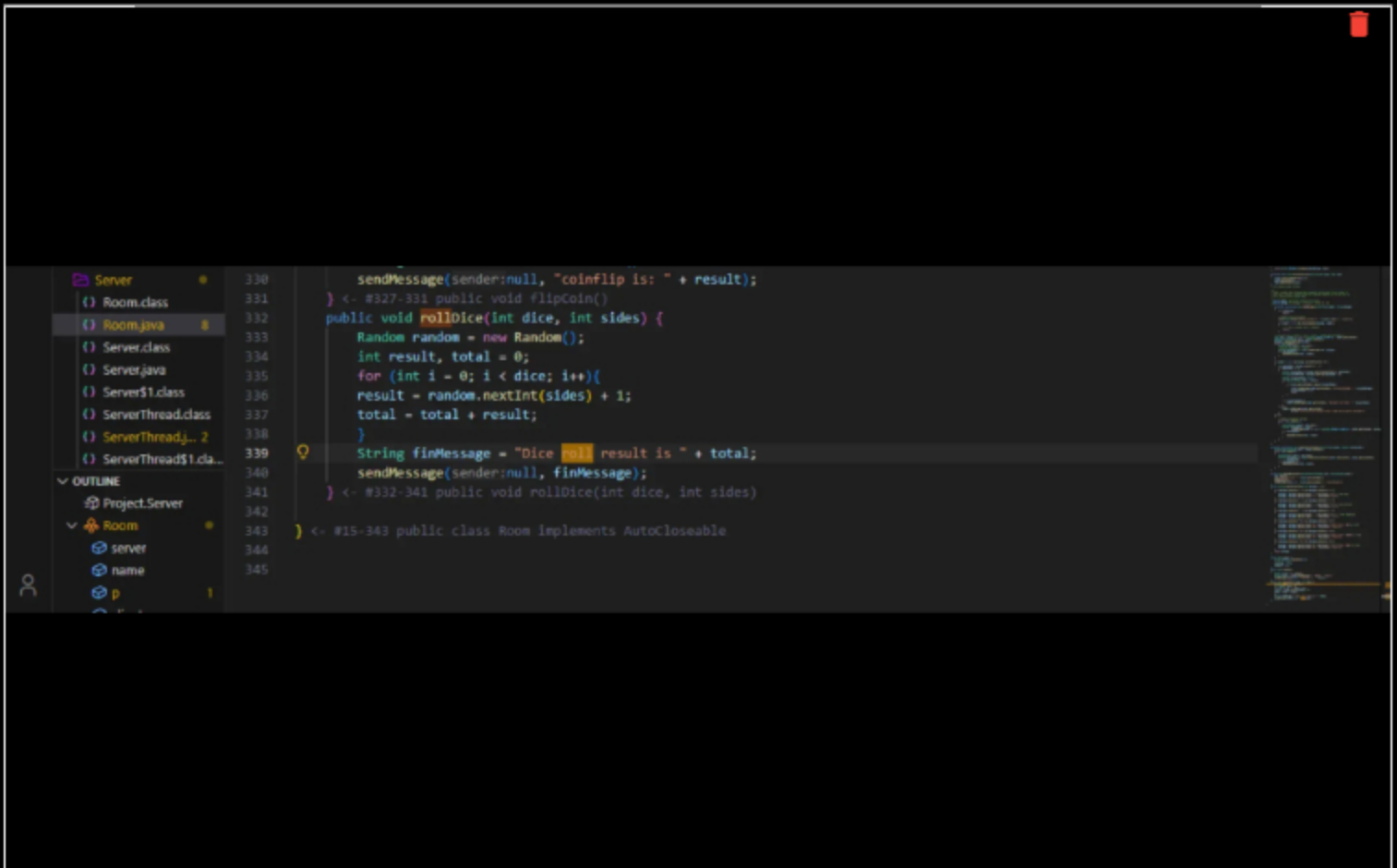
231 case ROLL:
232     RollPayload roller = (RollPayload) p;
233     currentRoom.rollDice(roller.getDice(), roller.getSides());
234     break;
235 case FLIP: //rn364
236     currentRoom.flipCoin();

```

Serverthread code

Checklist Items (1)

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



```
330     sendMessage(sender:null, "coinFlip is: " + result);
331 }
332 } <- #327-331 public void flipCoin()
333 public void rollDice(int dice, int sides) {
334     Random random = new Random();
335     int result, total = 0;
336     for (int i = 0; i < dice; i++){
337         result = random.nextInt(sides) + 1;
338         total = total + result;
339     }
340     String finMessage = "Dice roll result is " + total;
341     sendMessage(sender:null, finMessage);
342 } <- #332-341 public void rollDice(int dice, int sides)
343 }
344 } <- #15-343 public class Room implements AutoCloseable
345
```

Roomcode

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

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 server is notified when a player types anything like "/roll 6" or "/roll 2d6" in the game chat. The number of dice to roll and the number of sides on each die are then determined. All players in the game receive a message with the total roll once the dice have been rolled and the results have been added up

Demonstrate Flip Command (1 pt.)

Task #1 - Points: 1

Text: Screenshot of the following items

Checklist

*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```

185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
}
else if (text.startsWith(ROLL)) {
    String roll = text.replace(ROLL, replacement:"").trim();
    String[] rollParts = roll.split(regex:"d");
    if (rollParts.length >= 2) {
        try {
            int lower = Integer.parseInt(rollParts[0]);
            int upper = Integer.parseInt(rollParts[1]);
            sendRoll(lower, upper);
        } catch (IOException e) {
            System.out.println(TextFX.colorize(text:"Socket error", Color.RED));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}
else {
    try {
        int lower = 1;
        int upper = Integer.parseInt(rollParts[0]);
        sendRoll(lower, upper);
    } catch (IOException e) {
        System.out.println(TextFX.colorize(text:"Socket error", Color.RED));
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

client code

Checklist Items (1)

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

```

Room.java 8

```


() Server.class	228			break;
() Server.java	229			case JOIN_ROOM:
() Server\$1.class	230			Room.joinRoom(p.getMessage(), this);
() ServerThread.class	231			break;
() ServerThread.j... 2	232			case ROLL:
() ServerThread\$1.cla...	233			RollPayload roller = (RollPayload) p;
= sources.txt	234			currentRoom.rollDice(roller.getDice(),roller.getSides());
build.sh	235			break;
= ng4.txt	236			case FLIP: //rn364
() NumberGuesser4.class	237			currentRoom.flipCoin();
OUTLINE	238			break;
Project Server	239			case LIST_ROOMS:
	240			String searchString = p.getMessage() == null ? "" : p.getMessage();
				int limit = 10;

ServerThread receiving the payload and passing it to the Room

Checklist Items (1)

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

```

325 } <- #320-325 public void close()
326 public void flipCoin()
327 {
328     Random random = new Random();
329     String result = random.nextBoolean() ? "Heads" : "Tails";
330     sendMessage(sender:null, "coinflip is: " + result);
331 } <- #327-331 public void flipCoin()
332 public void rollDice(int dice, int sides) {

```

Room handling the flip action correctly

Checklist Items (1)

#3 Room handling the flip action correctly

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 message "/flip" is sent to the server by a client, and the server relays this information to the room. After flipping a coin, the room determines if it is heads or tails and relays the outcome to each and every client there. After receiving the outcome, each client presents it to the user. In essence, the server promotes an entire virtual coin flip among all users!

Demonstrate Formatted Messages (4 pts.)

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

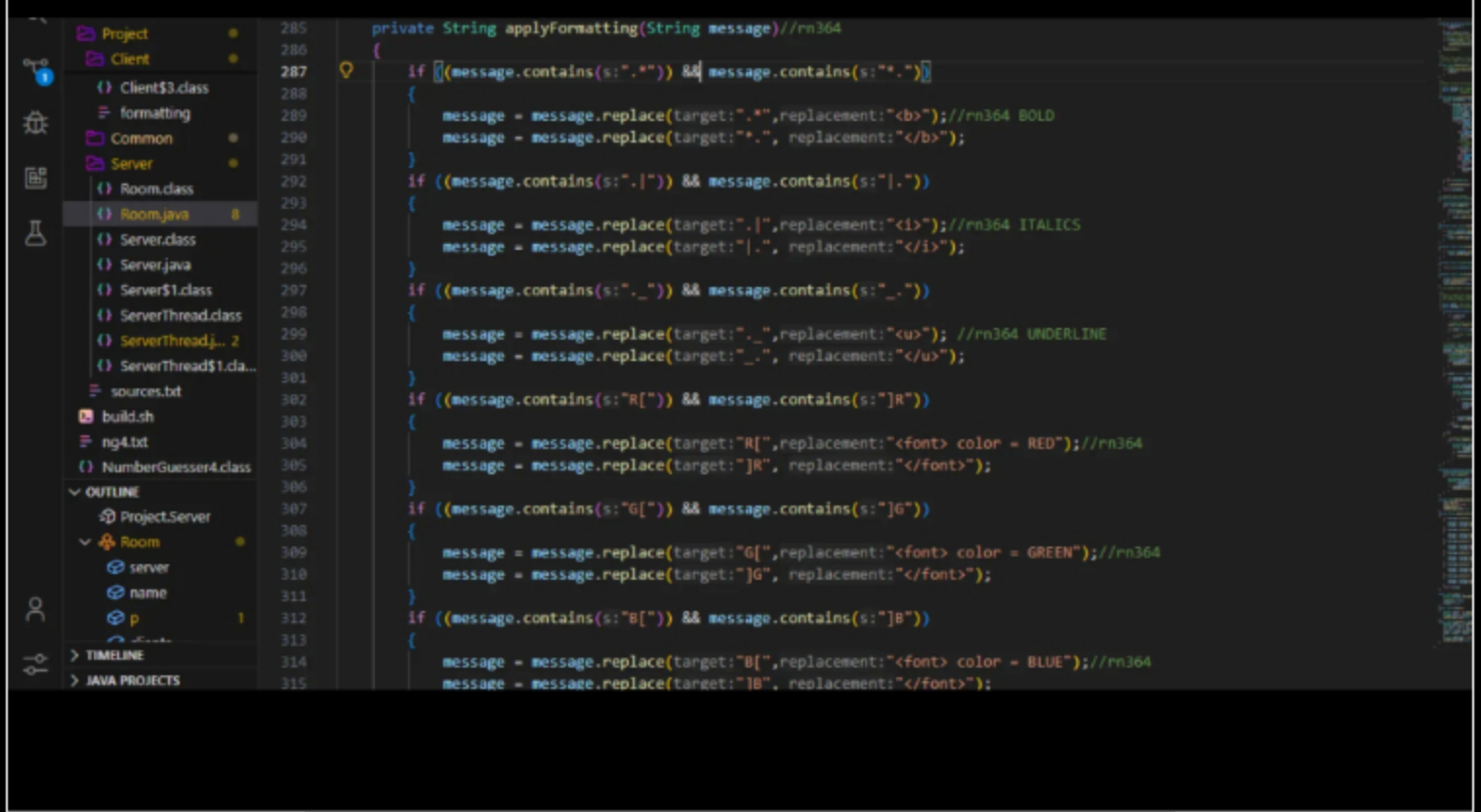
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Done plus the comments on the side

Checklist Items (8)

- #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)
- #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
- #7 Code screenshots should include ucid and date comment
- #8 Each screenshot should be clearly captioned



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:

Using a message as input, this function looks for certain sequences such as ".*", ".|", "._", "R[...]R", "G[...]G", and "B[...]B". When it comes across one of these sequences, it replaces them with HTML elements so that the text can be formatted with bold, italics, underlining, or different colors (red, green, or blue). In this technique, it uses the special characters in the message to convert plain text into formatted text.

Misc (1 pt.)

^COLLAPSE ^



^COLLAPSE ^

Task #1 - Points: 1

Text: Add the pull request link for the branch

i Details:

Note: the link should end with /pull/#

URL #1

<https://github.com/ryann2n/rn364-IT114-004/pull/9>



^COLLAPSE ^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Had a lot of help but was worth it to understand. Learned to format and type code.



^COLLAPSE ^

Task #3 - Points: 1

Text: WakaTime Screenshot

i 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



Waka time

End of Assignment