Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-005-F2024/it114-milestone-2-chatroom-2024-m24/grade/kh465

Course: IT114-005-F2024
Assigment: [IT114] Milestone 2 Chatroom 2024 (M24)
Student: Keven H. (kh465)

Submissions:

Submission Selection

1 Submission [submitted] 11/14/2024 12:28:30 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

Group

100%

Group: Payloads
Tasks: 2
Points: 2

COLLAPSE

Task

100%

Group: Payloads
Task #1: Base Payload Class
Weight: ~50%
Points: ~1.00

COLLAPSE

Details:
All code screenshots must have ucid/date visible.

Columns: 1

Sub-Task

100%

Group: Payloads
Task #1: Base Payload Class
Sub Task #1: Show screenshot of the Payload.java

Task Screenshots

Gallery Style: 2 Columns

4 2 1

```
[illegible]
```
Code from Payload.java

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the purpose of each property and serialization

Response:

A private object of PayloadType is created, as well as a long clientId and String message. getPayloadType() returns the type of payload from PayloadType (an enum). setPayloadType sets the current payloadType to whatever payloadType is needed from the enum. get/setClientId returns the clientId and sets the clientId to an incoming long respectively. get/setMessage returns the message and sets the message to an incoming String message respectively. toString overrides toString to give more information about the payload, the client, and the message.

Sub-Task

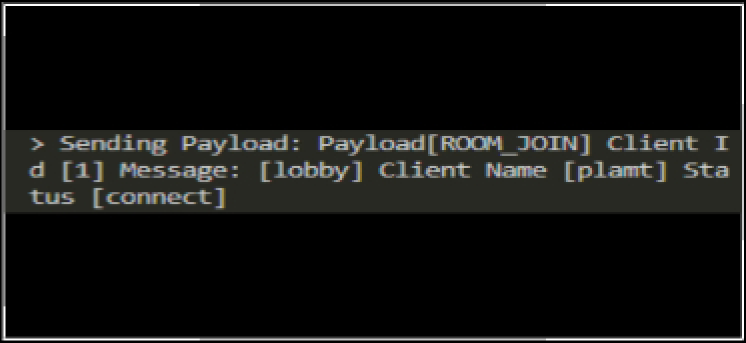
100%

Group: Payloads
Task #1: Base Payload Class
Sub Task #2: Show screenshot examples of the terminal output for base Payload objects

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
> Sending Payload: Payload[ROOM_JOIN] Client I
d [1] Message: [lobby] Client Name [plamt] Sta
tus [connect]
```

Terminal output for Payload showing user joining the lobby

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 1

Task

100%

Group: Payloads
Task #2: RollPayload Class
Weight: ~50%
Points: ~1.00

COLLAPSE

Details:

All code screenshots must have ucid/date visible.

Columns: 1

Sub-Task

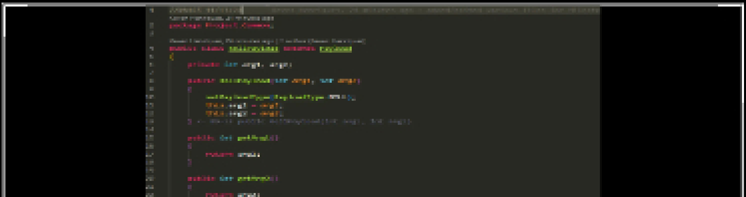
100%

Group: Payloads
Task #2: RollPayload Class
Sub Task #1: Show screenshot of the RollPayload.java (or equivalent)

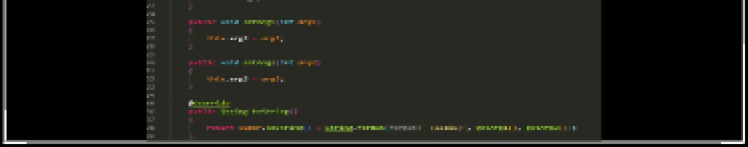
Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
[illegible]
```



```
    }

    public void setArg1(int arg1)
    {
        this.arg1 = arg1;
    }

    public void setArg2(int arg2)
    {
        this.arg2 = arg2;
    }

    @Override
    public String toString()
    {
        [illegible]
    }
```

RollPayload.java with UCID and date visible

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the purpose of each property

Response:

Two private ints, arg1 and arg2 are initialised. An object of RollPayload is created, setting the PayloadType to ROLL and this.arg1 and this.arg2 to incoming ints arg1 and arg2 respectively. get/setArg1(2) retrieves arg1 or arg2, or sets arg1 or arg2 to an incoming int arg1/arg2 respectively.

Sub-Task

100%

Group: Payloads

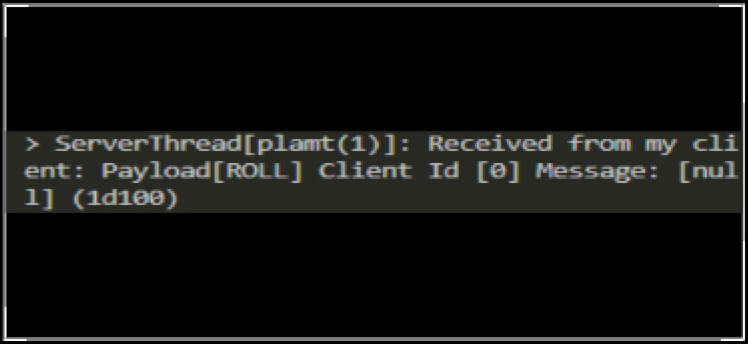
Task #2: RollPayload Class

Sub Task #2: Show screenshot examples of the terminal output for base RollPayload objects

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
> ServerThread[plamt(1)]: Received from my cli
ent: Payload[ROLL] Client Id [0] Message: [nul
l] (1d100)
```

Terminal output for RollPayload object

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 2

End of Group: Payloads

Task Status: 2/2

Group

100%

Group: Client Commands

Tasks: 2

Points: 4

COLLAPSE

Task

100%

Group: Client Commands
Task #1: Roll Command
Weight: ~50%
Points: ~2.00

COLLAPSE

Details:
All code screenshots must have ucid/date visible.
Any output screenshots must have at least 3 connected clients able to see the output.
All commands must show who triggered it, what they did (specifically) and what the outcome was.

Columns: 1

Sub-Task

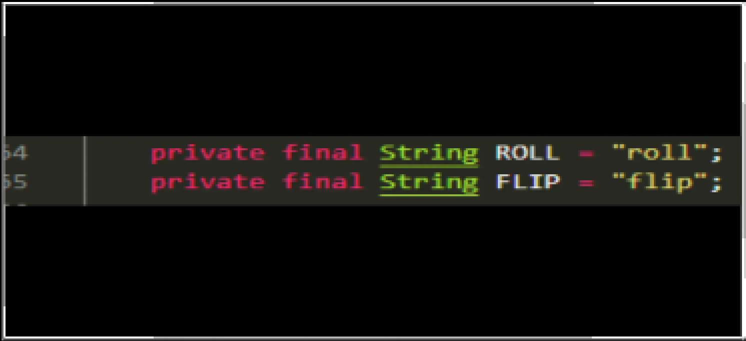
100%

Group: Client Commands
Task #1: Roll Command
Sub Task #1: Show the client side code for handling /roll #

Task Screenshots

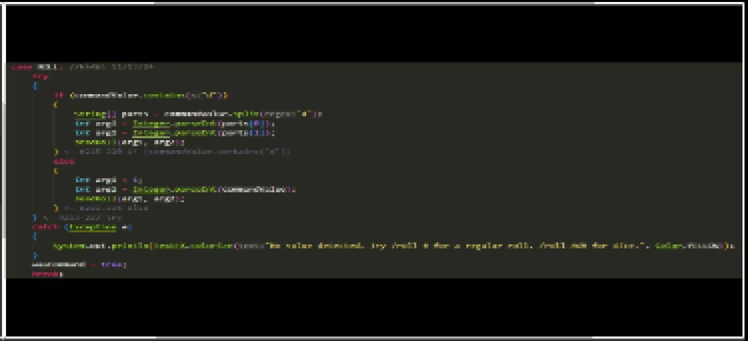
Gallery Style: 2 Columns

4 2 1



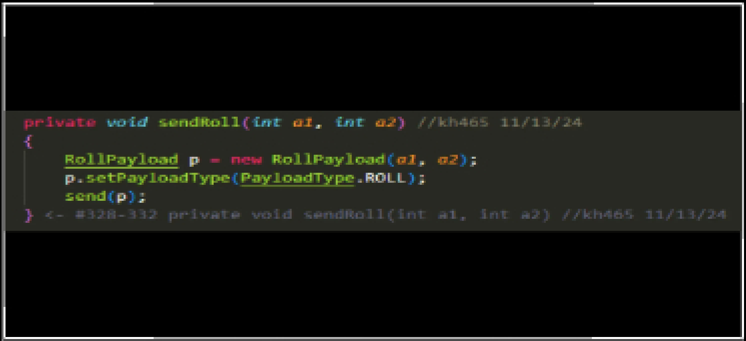
```
    private final String ROLL = "roll";
    private final String FLIP = "flip";
```

Roll being added to constants list



```
case ROLL: //kh465 11/13/24
    try
    {
        if (commandValue.contains(s:"d"))
        {
            String[] parts = commandValue.split(regex:"d");
            int arg1 = Integer.parseInt(parts[0]);
            int arg2 = Integer.parseInt(parts[1]);
            sendRoll(arg1, arg2);
        } <- [illegible] if (commandValue.contains("d"))
        else
        {
            int arg1 = 1;
            int arg2 = Integer.parseInt(commandValue);
            sendRoll(arg1, arg2);
        } <- [illegible] else
    } <- [illegible] try
    catch (Exception e)
    {
        System.out.println(TextFX.colorize(text:"No value detected. Try /roll # for a regular roll. /roll #d# for dice.", Color.YELLOW));
    }
    wasCommand = true;
    break;
```

case ROLL added to processClientCommand, with logic



```
private void sendRoll(int a1, int a2) //kh465 11/13/24
{
    RollPayload p = new RollPayload(a1, a2);
    p.setPayloadType(PayloadType.ROLL);
    send(p);
} <- #328-332 private void sendRoll(int a1, int a2) //kh465 11/13/24
```

New method sendRoll handling the sending of the command

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the logic

Response:

case ROLL checks to see if the commandValue passed into it contains "d" which denotes whether it's a straight numerical roll or a dice roll. If it contains "d", create a String array named parts, and add the value on either side of "d" to it. Set ints arg1 and arg2 to parts[0] and [1], then send it via sendRoll(arg1, arg2). If it does not contain "d", set arg1 to 1, arg2 to commandValue and send it via sendRoll(arg1, arg2). sendRoll creates a new RollPayload p, sets a1 and a2 to incoming arg1 and arg2, sets the payloadType to ROLL, and uses send(p).

Sub-Task

100%

Group: Client Commands

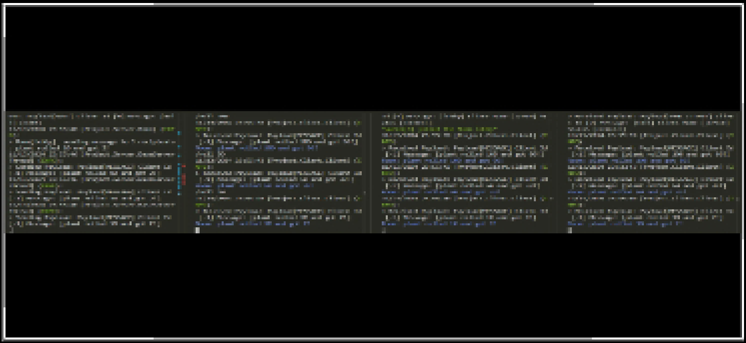
Task #1: Roll Command

Sub Task #2: Show the output of a few examples of /roll # (related payload output should be visible)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
[illegible]
```

/roll being used. Left to right: server, plamt (used /roll), client2, client3

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

Group: Client Commands

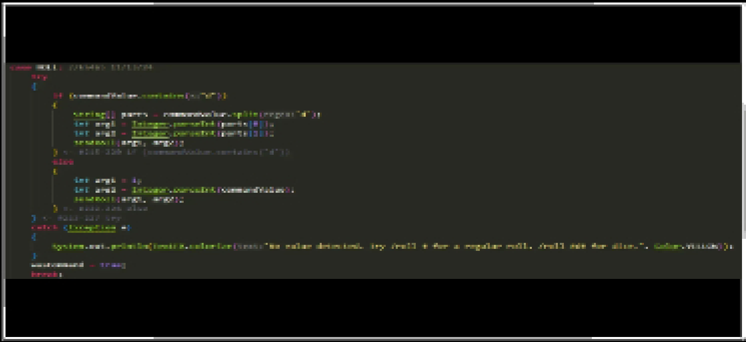
Task #1: Roll Command

Sub Task #3: Show the client side code for handling /roll #d# (related payload output should be visible)

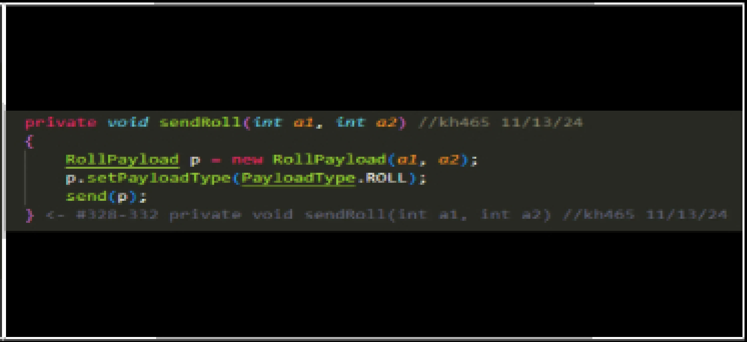
Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
case ROLL: //kh465 11/13/24
    try
    {
        if (commandValue.contains("d"))
        {
            String[] parts = commandValue.split("d");
            int arg1 = Integer.parseInt(parts[0]);
            int arg2 = Integer.parseInt(parts[1]);
            sendRoll(arg1, arg2);
        } <- [illegible] if (commandValue.contains("d"))
        else
        {
            int arg1 = 1;
            int arg2 = Integer.parseInt(commandValue);
            sendRoll(arg1, arg2);
        } <- [illegible] else
    } <- [illegible] try
    catch (Exception e)
    {
        System.out.println(TextFX.colorize("No value detected. Try /roll # for a regular roll, /roll #d# for dice.", Color.YELLOW));
    }
    wasCommand = true;
    break;
```

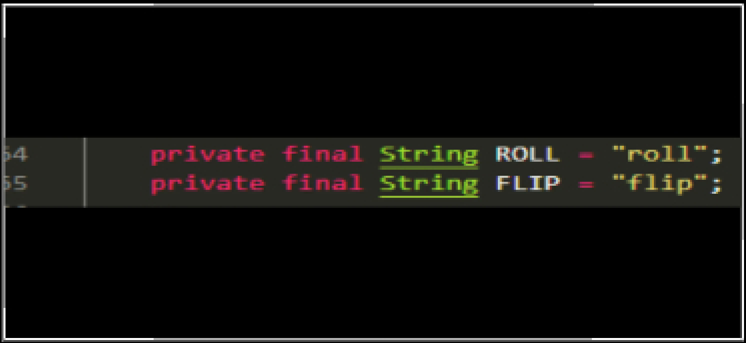


```
private void sendRoll(int a1, int a2) //kh465 11/13/24
{
    RollPayload p = new RollPayload(a1, a2);
    p.setPayloadType(PayloadType.ROLL);
    send(p);
} <- #328-332 private void sendRoll(int a1, int a2) //kh465 11/13/24
```

case ROLL added to processClientCommand, with logic

New method sendRoll handling the sending of the case ROLL added to processClientCommand, with logic

New method sendRoll handling the sending of the command



```
    private final String ROLL = "roll";
    private final String FLIP = "flip";
```

Roll being added to constants list

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the logic

Response:

case ROLL checks to see if the commandValue passed into it contains "d" which denotes whether it's a straight numerical roll or a dice roll. If it contains "d", create a String array named parts, and add the value on either side of "d" to it. Set ints arg1 and arg2 to parts[0] and [1], then send it via sendRoll(arg1, arg2). If it does not contain "d", set arg1 to 1, arg2 to commandValue and send it via sendRoll(arg1, arg2). sendRoll creates a new RollPayload p, sets a1 and a2 to incoming arg1 and arg2, sets the payloadType to ROLL, and uses send(p).

Sub-Task 100%

Group: Client Commands

Task #1: Roll Command

Sub Task #4: Show the output of a few examples of /roll #d#

Task Screenshots

Gallery Style: 2 Columns

4 2 1

/roll #d# being used. Left to right: server, plamt (used /roll #d#), client2, client3

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task 100%

Group: Client Commands

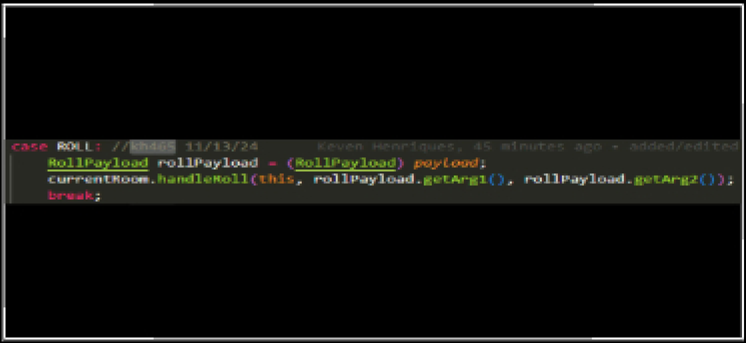
Task #1: Roll Command

Sub Task #5: Show the ServerThread code receiving the RollPayload

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
case ROLL: //kh465 11/13/24        Keven Henriques, 45 minutes ago • added/edited
    RollPayload rollPayload = (RollPayload) payload;
    currentRoom.handleRoll(this, rollPayload.getArg1(), rollPayload.getArg2());
    break;
```

ServerThread's processPayload receiving RollPayload, getting arg1 and arg2 for Room.

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the logic

Response:

Payload is being casted as a RollPayload, then the data is sent off to Room to be handled via a method in Room called handleRoll which receives the sender, arg1 and arg2.

Sub-Task

100%

Group: Client Commands

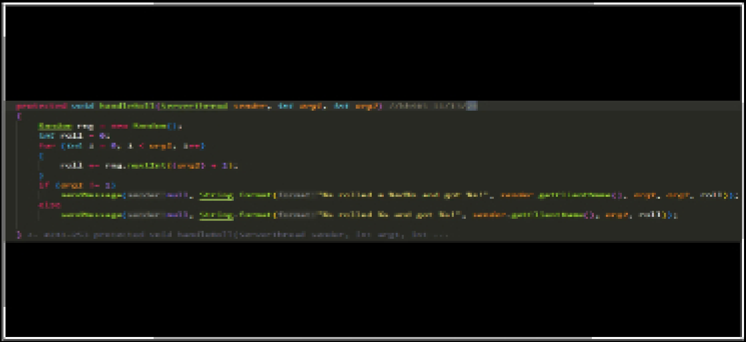
Task #1: Roll Command

Sub Task #6: Show the Room code that processes both Rolls and sends the response

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
protected void handleRoll(ServerThread sender, int arg1, int arg2) //kh465 11/13/24
{
    Random rng = new Random();
    int roll = 0;
    for (int i = 0; i < arg1; i++)
    {
        roll += rng.nextInt(arg2) + 1;
    }
    if (arg1 != 1)
        sendMessage(sender:null, String.format(format:"%s rolled a %dd%d and got %d!", sender.getClientName(), arg1, arg2, roll));
    else
        sendMessage(sender:null, String.format(format:"%s rolled %d and got %d!", sender.getClientName(), arg2, roll));
} // end protected void handleRoll(ServerThread sender, int arg1, int ...
```

handleRoll in Room that handles both roll logics. The logic is the same, only formatting changes.

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the logic

Response:

Random is created, and int roll is initialised to 0. A for loop loops for the length of arg1 (our dice amount), adding and setting roll to rng.nextInt((arg2) + 1) (arg2 is our roll amount, +1 for inclusivity). If arg1 is not 1, sendMessage is altered slightly to include "d" and the numerical value. If arg1 is 1, sendMessage is altered slightly to not include "d", just the numerical value.

End of Task 1

Task

100%

Group: Client Commands
Task #2: Flip Command
Weight: ~50%
Points: ~2.00

COLLAPSE

Columns: 1

Sub-Task

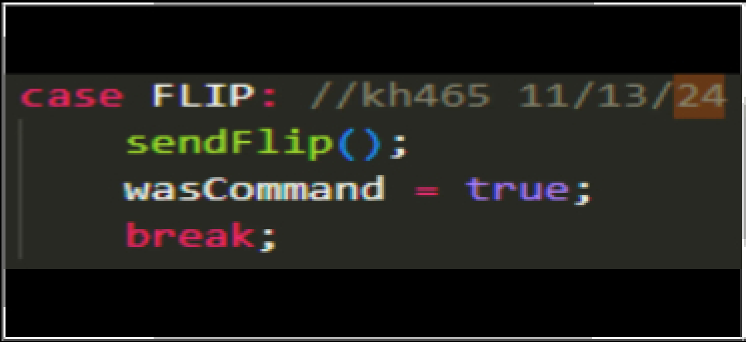
100%

Group: Client Commands
Task #2: Flip Command
Sub Task #1: Show the client side code for handling /flip

Task Screenshots

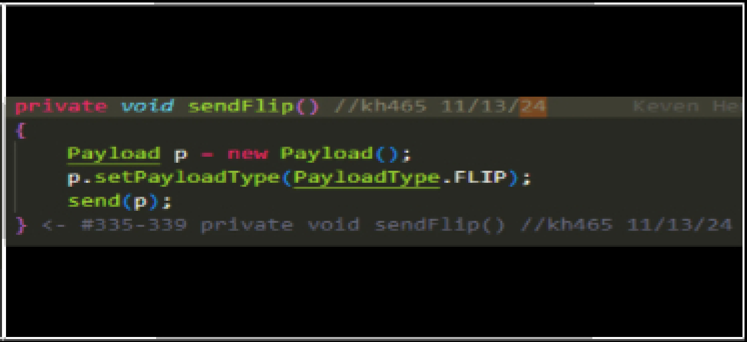
Gallery Style: 2 Columns

4 2 1



```
case FLIP: //kh465 11/13/24
    sendFlip();
    wasCommand = true;
    break;
```

case FLIP, invoking sendFlip()



```
private void sendFlip() //kh465 11/13/24        Keven He
{
    Payload p = new Payload();
    p.setPayloadType(PayloadType.FLIP);
    send(p);
} <- #335-339 private void sendFlip() //kh465 11/13/24
```

sendFlip() creating a payload and sending it via send(p)

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the logic

Response:

case FLIP only acts as a trigger, no data is passed along. It invokes sendFlip, which creates a payload, sets its payloadType to FLIP, and sends it via send(p)

Sub-Task

100%

Group: Client Commands
Task #2: Flip Command
Sub Task #2: Show the output of a few examples of /flip (related payload output should be visible)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

/flip being used. From left to right: server, plamt (used /flip), client2, client3

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

End of Task 2

End of Group: Client Commands
Task Status: 2/2

Group

100%

Group: Text Formatting
Tasks: 1
Points: 3

COLLAPSE

Task

100%

Group: Text Formatting
Task #1: Text Formatting
Weight: ~100%
Points: ~3.00

COLLAPSE

Details:
All code screenshots must have ucid/date visible.
Any output screenshots must have at least 3 connected clients able to see the output.
Note: Having the user type out html tags is not valid for this feature, instead treat it like WhatsApp, Discord, Markdown, etc

Columns: 1

Sub-Task

100%

Group: Text Formatting

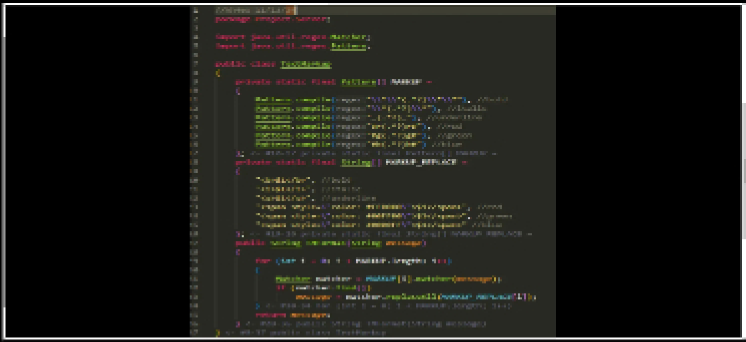
Task #1: Text Formatting

Sub Task #1: Show the code related to processing the special characters for bold, italic, underline, and colors, and converting them to other characters (should be in Room.java)

Task Screenshots

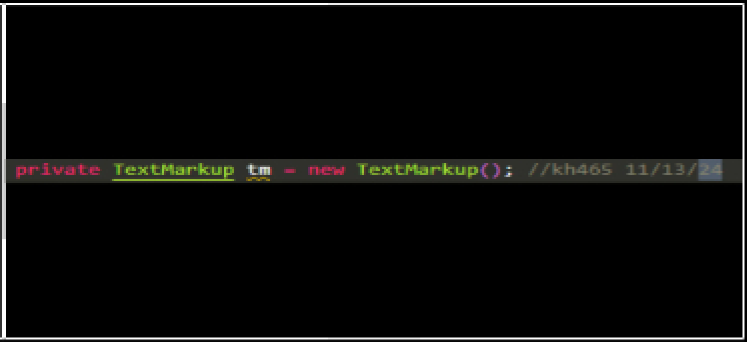
Gallery Style: 2 Columns

4 2 1



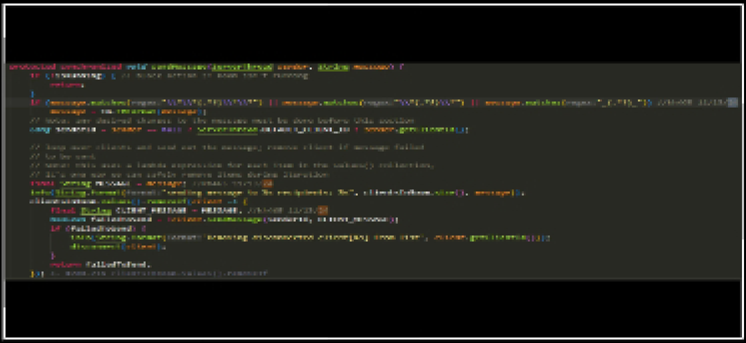
[illegible]

class TextMarkup.java which handles all text markup logic. Room imports this and can invoke it by creating a TextMarkup object.



```
private TextMarkup tm = new TextMarkup(); //kh465 11/13/24
```

Object TextMarkup in Room



[illegible]

Logic in Room that handles how message formatting is done. Invokes TMFormat from TextMarkup

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain how it works and the choices of the placeholder characters and the result characters

Response:

TextMarkup.java handles the actual formatting changes. It has a String array named MARKUP which uses regex to check for bold (**), italics (*), underline (_), red (#r r#), green (#g g#) and blue (#b b#). MARKUP_REPLACE replaces the characters above with their respective formats. TMFormat iterates over MARKUP.length and uses Matcher to check MARKUP[i] and see if there's a match in the message. If there is, replace it with the same index from String array MARKUP_REPLACE (which has the changed symbols in the same order)

Sub-Task

100%

Group: Text Formatting

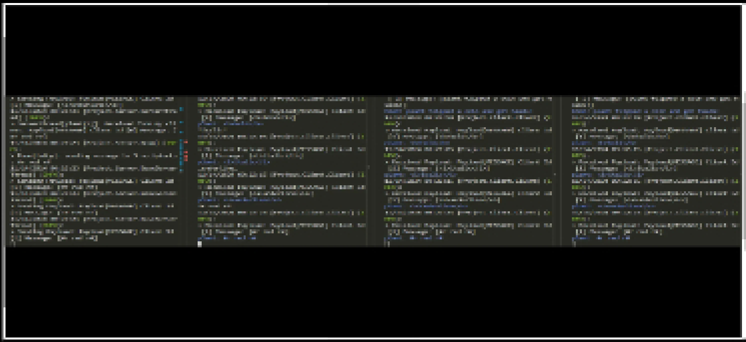
Task #1: Text Formatting

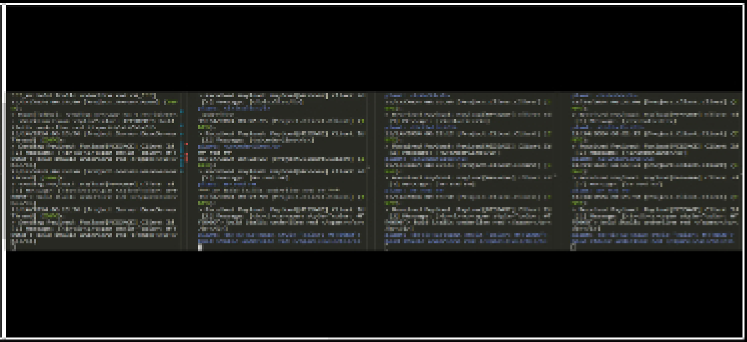
Sub Task #2: Show examples of each: bold, italic, underline, colors (red, green, blue), and combination of bold, italic, underline and a color

Task Screenshots

Gallery Style: 2 Columns

4 2 1


Bold, italic, underline and red. From left to right: server, plamt (demo), client2, client3.


Combo of bold, italic, underline and red. Left to right: server, plamt (demo), client2, client3

Caption(s) (required) ✓
Caption Hint: *Describe/highlight what's being shown*

End of Task 1

End of Group: Text Formatting
Task Status: 1/1

Group

100%

Group: Misc
Tasks: 3
Points: 1

COLLAPSE

Task

100%

Group: Misc
Task #1: Add the pull request link for the branch
Weight: ~33%
Points: ~0.33

COLLAPSE

Details:
Note: the link should end with /pull/#

Task URLs

URL #1
https://github.com/kh465/kh465-IT114-005/pull/13

URL
https://github.com/kh465/kh465-IT114-005/pull/

End of Task 1

Task

100%

Group: Misc
Task #2: Talk about any issues or learnings during this assignment
Weight: ~33%
Points: ~0.33

COLLAPSE

Task Response Prompt

Response:

I faced many issues during this assignment. I felt completely overwhelmed when I first attempted to work on it, but as I learned more about how the files worked, and about various topics that were helpful in my journey to complete Milestone2, I was able to overcome most of these issues and learned quite a bit.

End of Task 2

Task

100%

Group: Misc
Task #3: WakaTime Screenshot
Weight: ~33%
Points: ~0.33

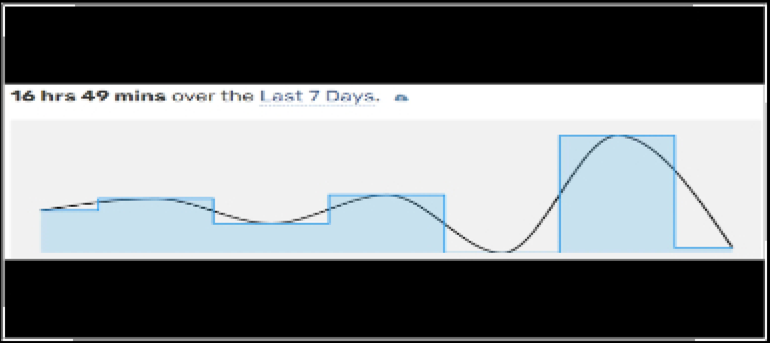
COLLAPSE

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: 2 Columns

4 2 1


16 hrs 49 mins over the Last 7 Days.

WakaTime from 11/8 to 11/14

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment