Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-milestone-2-chatroom-2024/grade/am3485

IT114-002-S2024 - [IT114] Milestone 2 Chatroom 2024

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 4:36:35 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 checkboxes are for your own tracking

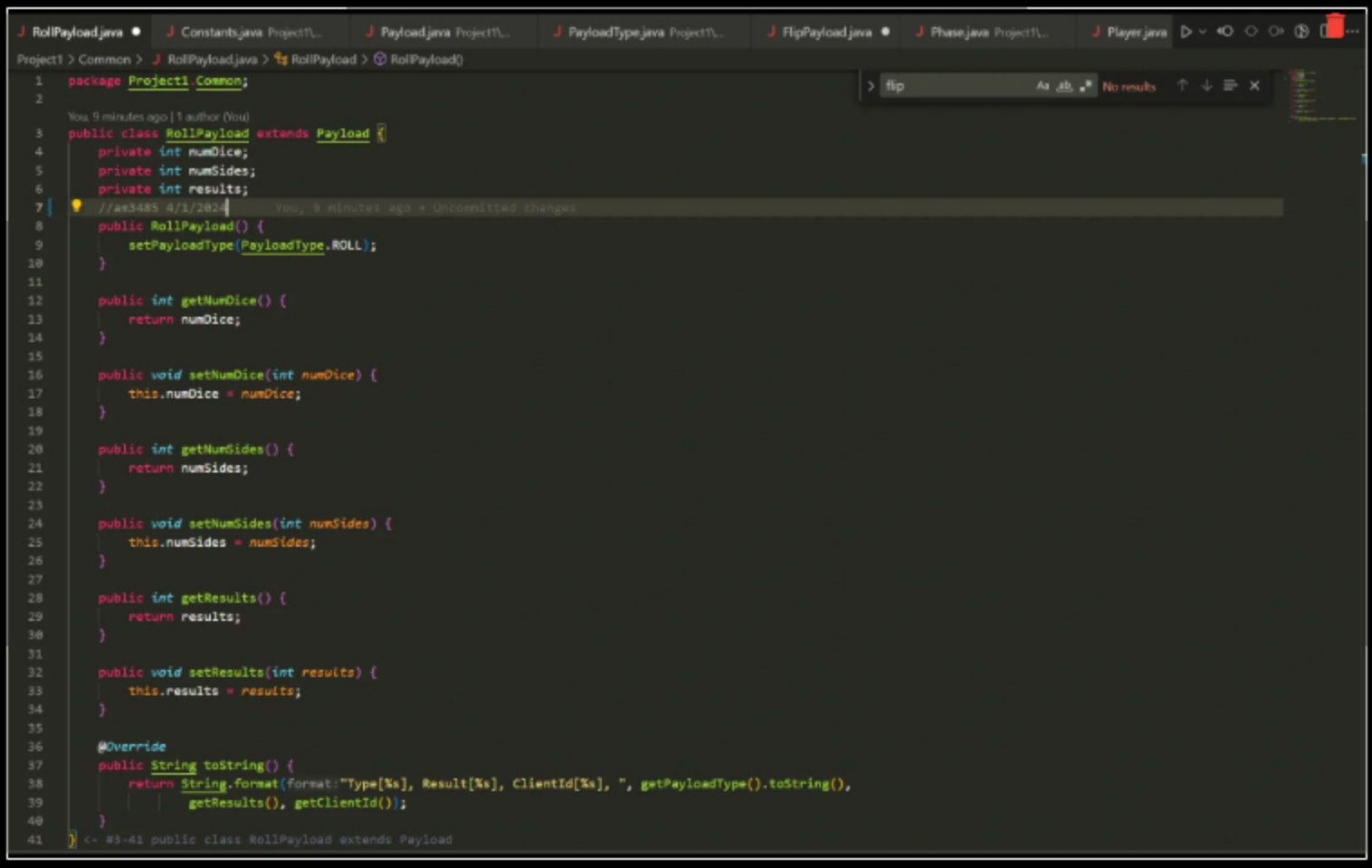
#	Points	Details

#	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

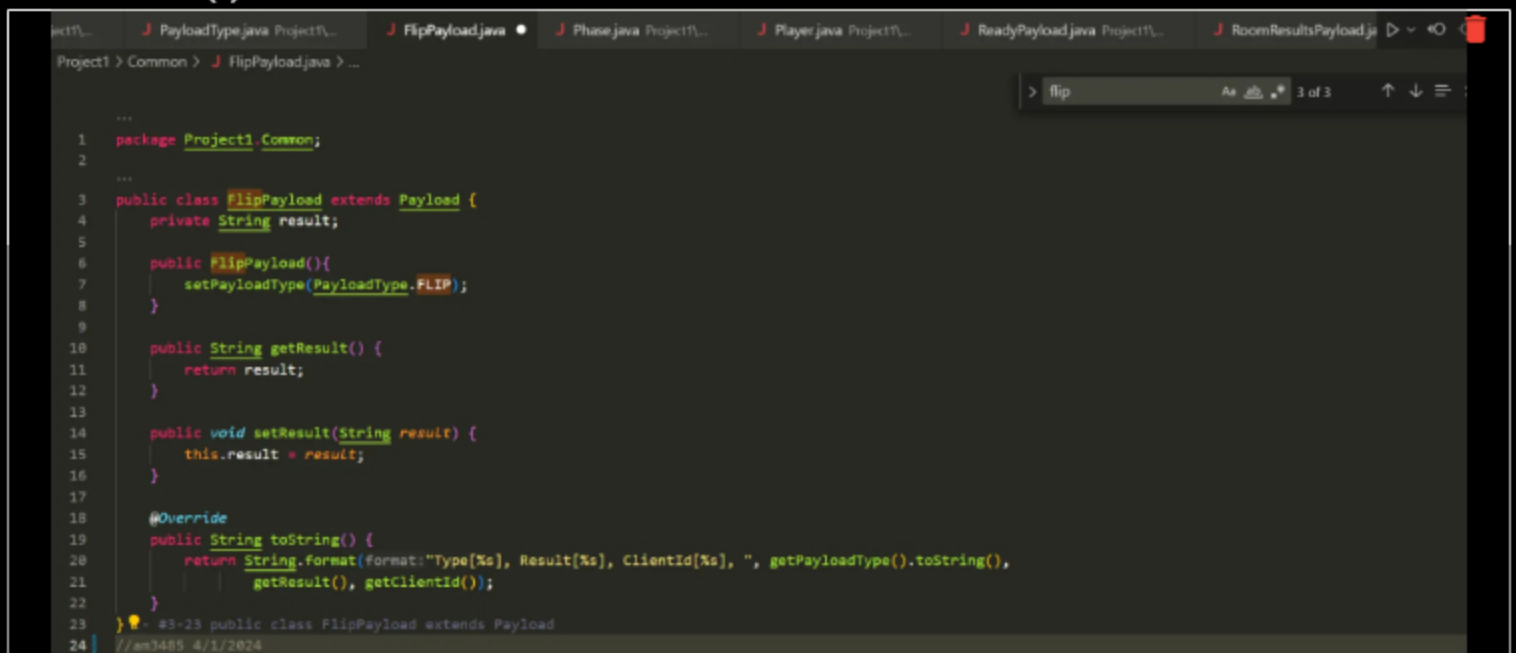
Small Medium Large



```java
package Project1.Common;

public class RollPayload extends Payload {
    private int numDice;
    private int numSides;
    private int results;
    //am3485 4/1/2024
    public RollPayload() {
        setPayloadType(PayloadType.ROLL);
    }

    public int getNumDice() {
        return numDice;
    }

    public void setNumDice(int numDice) {
        this.numDice = numDice;
    }

    public int getNumSides() {
        return numSides;
    }

    public void setNumSides(int numSides) {
        this.numSides = numSides;
    }

    public int getResults() {
        return results;
    }

    public void setResults(int results) {
        this.results = results;
    }

    @Override
    public String toString() {
        return String.format("Type[%s], Result[%s], ClientId[%s], ", getPayloadType().toString(),
                getResults(), getClientId());
    }
}
```

Roll payload code

Checklist Items (0)



```java
package Project1.Common;

public class FlipPayload extends Payload {
    private String result;

    public FlipPayload(){
        setPayloadType(PayloadType.FLIP);
    }

    public String getResult() {
        return result;
    }

    public void setResult(String result) {
        this.result = result;
    }

    @Override
    public String toString() {
        return String.format("Type[%s], Result[%s], ClientId[%s], ", getPayloadType().toString(),
                getResult(), getClientId());
    }
}
```

FlipPayload() code

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

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

Checklist

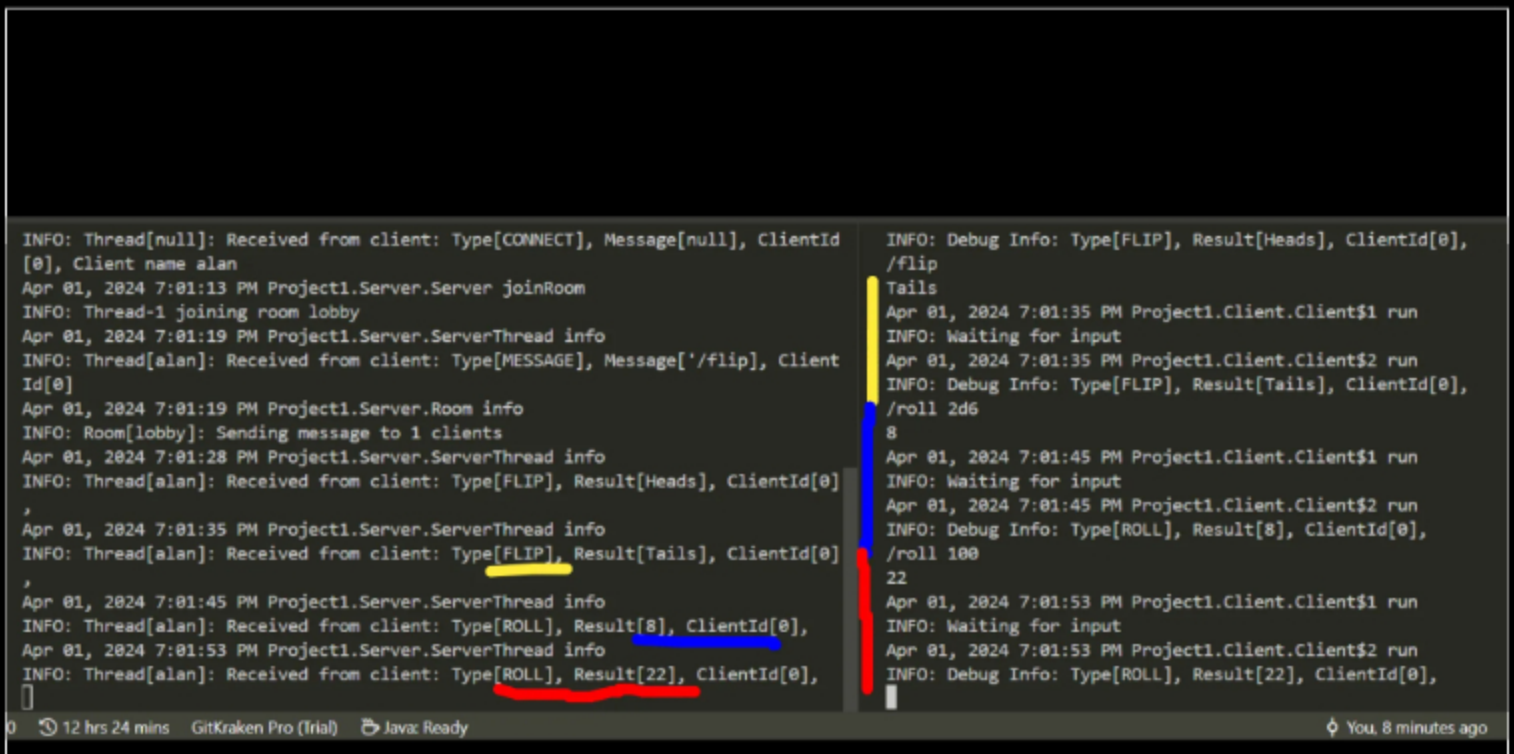
*The checkboxes are for your own tracking

#	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



yellow = flip red = /roll 100 blue = /roll 2d6

Checklist Items (0)

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 have the server communicate with the client and vice versa. My payloads where made using the parent payloads as a guide and then edited to show the more important information.

COLLAPSE

Demonstrate Roll Command (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

Checklist

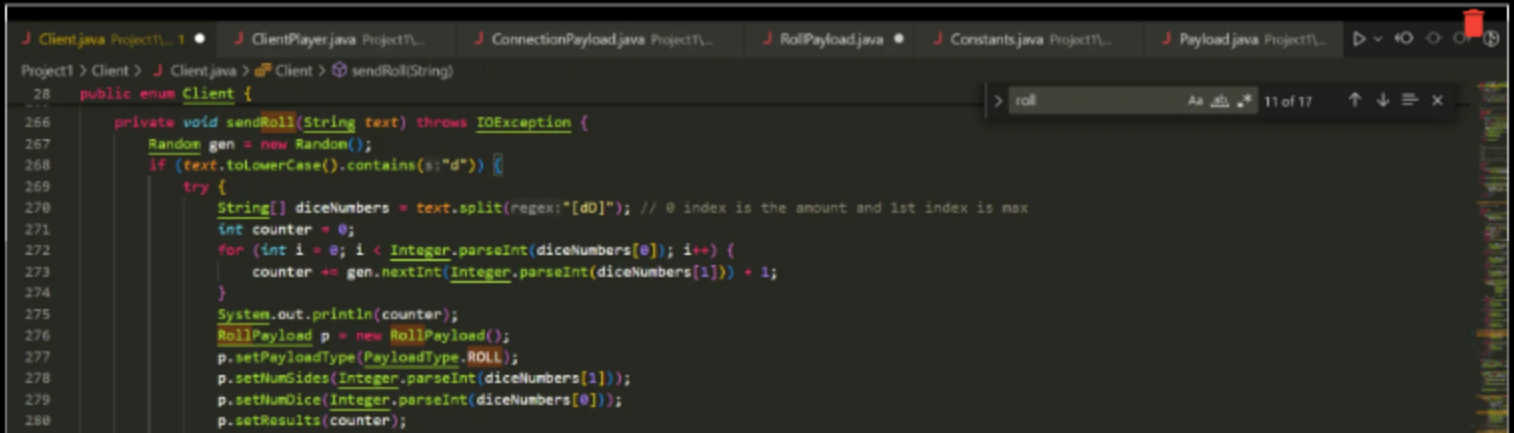
*The checkboxes are for your own tracking

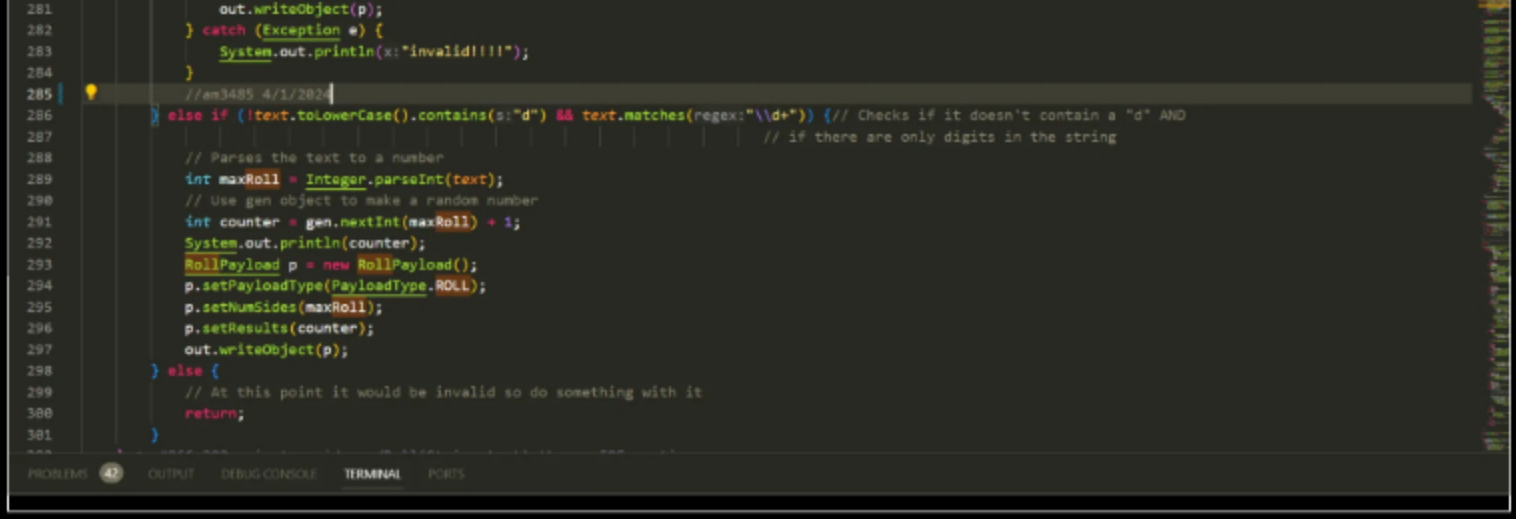
#	Points	Details
☐ #1	1	Client code that captures the command and converts it to a RollPayload (or equivalent) for both scenarios /roll # and /roll #d#
☐ #2	1	ServerThread code receiving the payload and passing it to the Room
☐ #3	1	Room handling the roll action correctly for both scenarios (/roll # 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





```
            out.writeObject(p);
        } catch (Exception e) {
            System.out.println(x:"invalid!!!!");
        }
        //am3485 4/1/2024
    } else if (!text.toLowerCase().contains(s:"d") && text.matches(regex:"\\d+")) {// Checks if it doesn't contain a "d" AND
                                                                                   // if there are only digits in the string
        // Parses the text to a number
        int maxRoll = Integer.parseInt(text);
        // Use gen object to make a random number
        int counter = gen.nextInt(maxRoll) + 1;
        System.out.println(counter);
        RollPayload p = new RollPayload();
        p.setPayloadType(PayloadType.ROLL);
        p.setNumSides(maxRoll);
        p.setResults(counter);
        out.writeObject(p);
    } else {
        // At this point it would be invalid so do something with it
        return;
    }
```

PROBLEMS 42 OUTPUT DEBUG CONSOLE TERMINAL PORTS

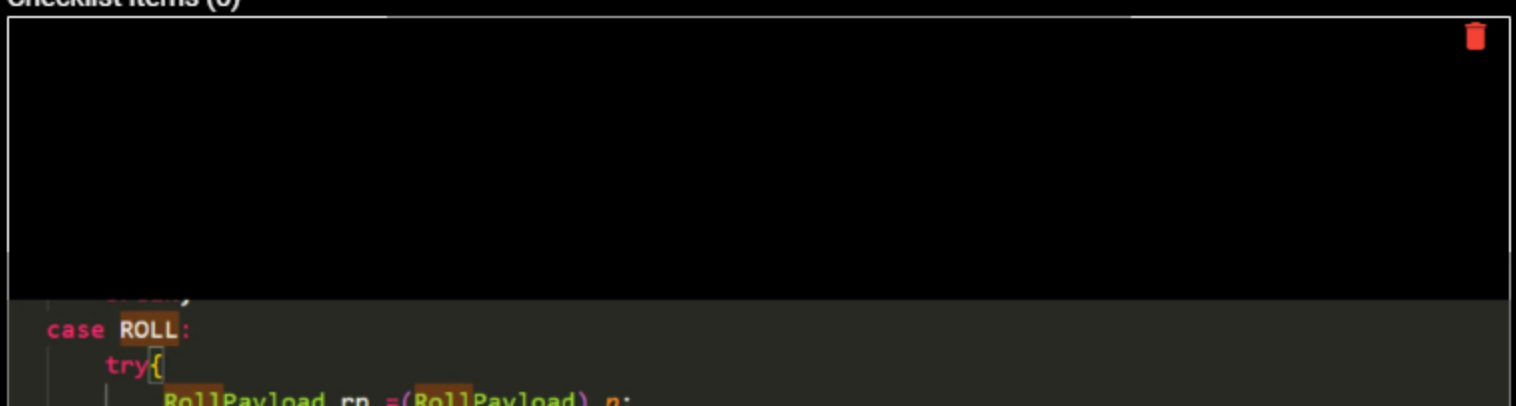
code that converts into rollpayload

Checklist Items (0)

```
} else if (text.startsWith(ROLL)) {
    try { //am3485 4/1/2024
        String searchQuery = text.replace(ROLL, replacement:"").trim();
        sendRoll(searchQuery);        You, 22 minutes ago • i gave up on go fis
    } catch (Exception e) {
        e.printStackTrace();
    }
    return true;
```

code that captures /roll and sends it to be converted

Checklist Items (0)

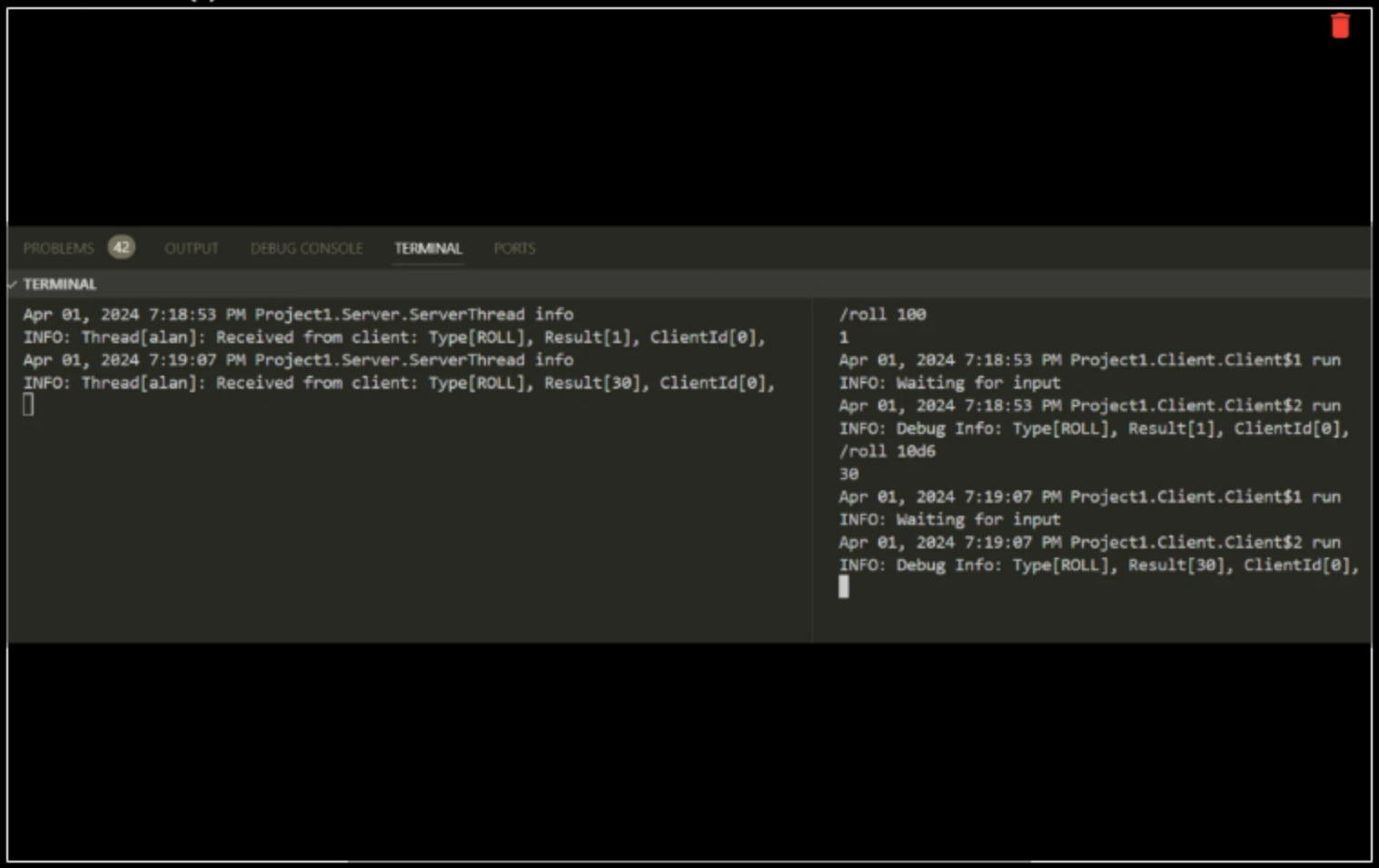


```
case ROLL:
    try{
        RollPayload rp =(RollPayload) p;
```

```
        rp.toString();
        send(rp);//am3485 4/1/2024
    }catch (Exception e){
        this.sendMessage(Constants.DEFAULT_CLIENT_ID,
                message:"You can only use the /turn commmand in a GameRoom and not the Lobby");
    }
    break;
```

code from server thread that handles the payload and sends it back to the client

Checklist Items (0)



room handling the roll action correctly for both scenarios including messages going back to client

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

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

Response:

the two roll formats are handle by checking if the string after /roll contains a d and if it does it runs the code that will roll a certain number of certain sided die and if it doesn't it generates a number between 1 and the roll number. its shared to the room using processPayload and back to the clients using the send method.

Demonstrate Flip Command (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the following items

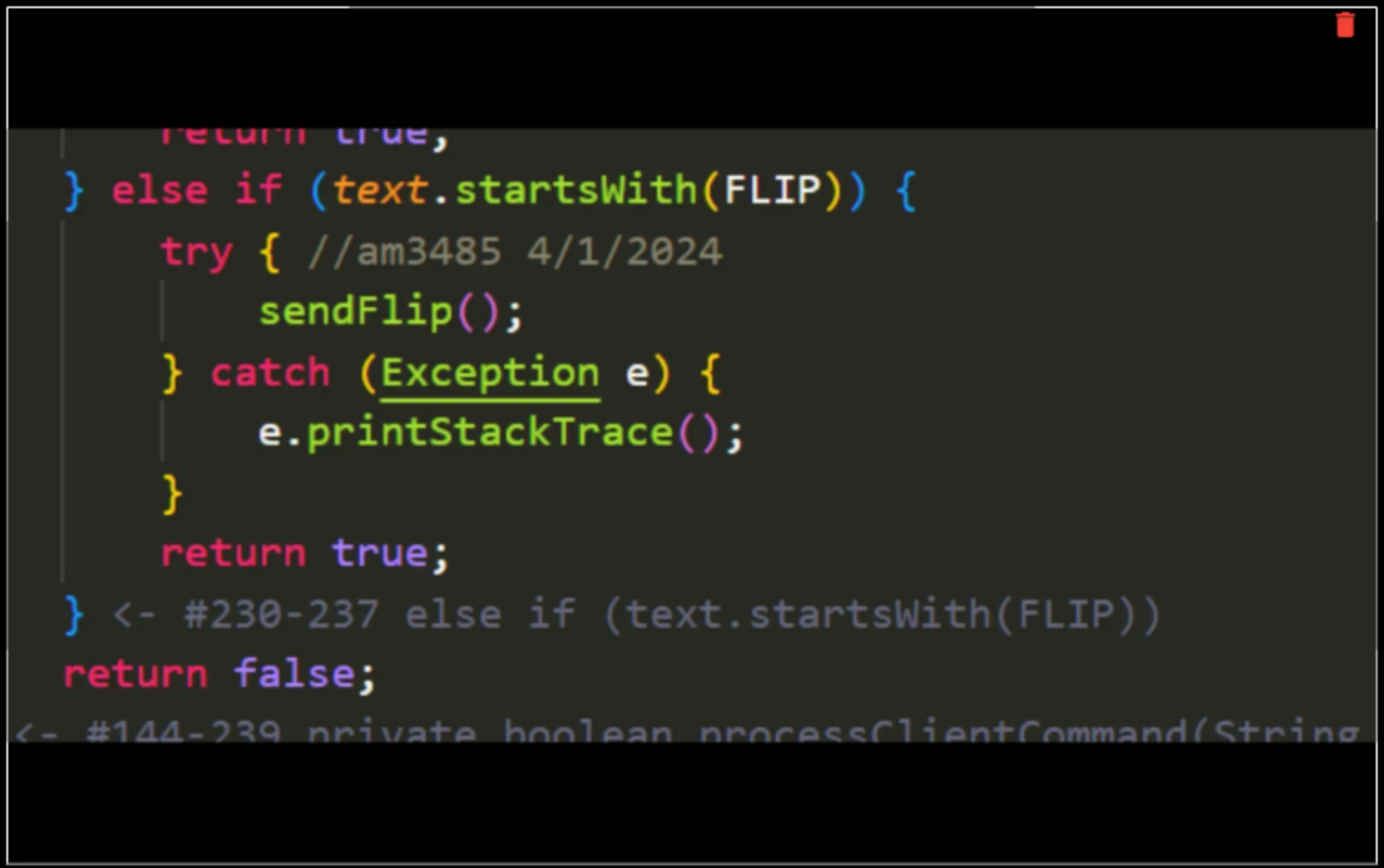
Checklist *The checkboxes are for your own tracking

#	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



catches /flip and sends it to be converted to a payload

Checklist Items (0)

```java
private void sendFlip() throws IOException {
    Random gen = new Random();
    FlipPayload p = new FlipPayload();
    int c = gen.nextInt(bound:1000) % 2;
    if (c == 1) {//am3485 4/1/2024      You, 57 minutes ago • U
        System.out.println(x:"Heads");
        p.setResult(result:"Heads");
    } else {
        System.out.println(x:"Tails");
        p.setResult(result:"Tails");
    }

    p.setPayloadType(PayloadType.FLIP);
    out.writeObject(p);

} <- #249-264 private void sendFlip() throws IOException
```

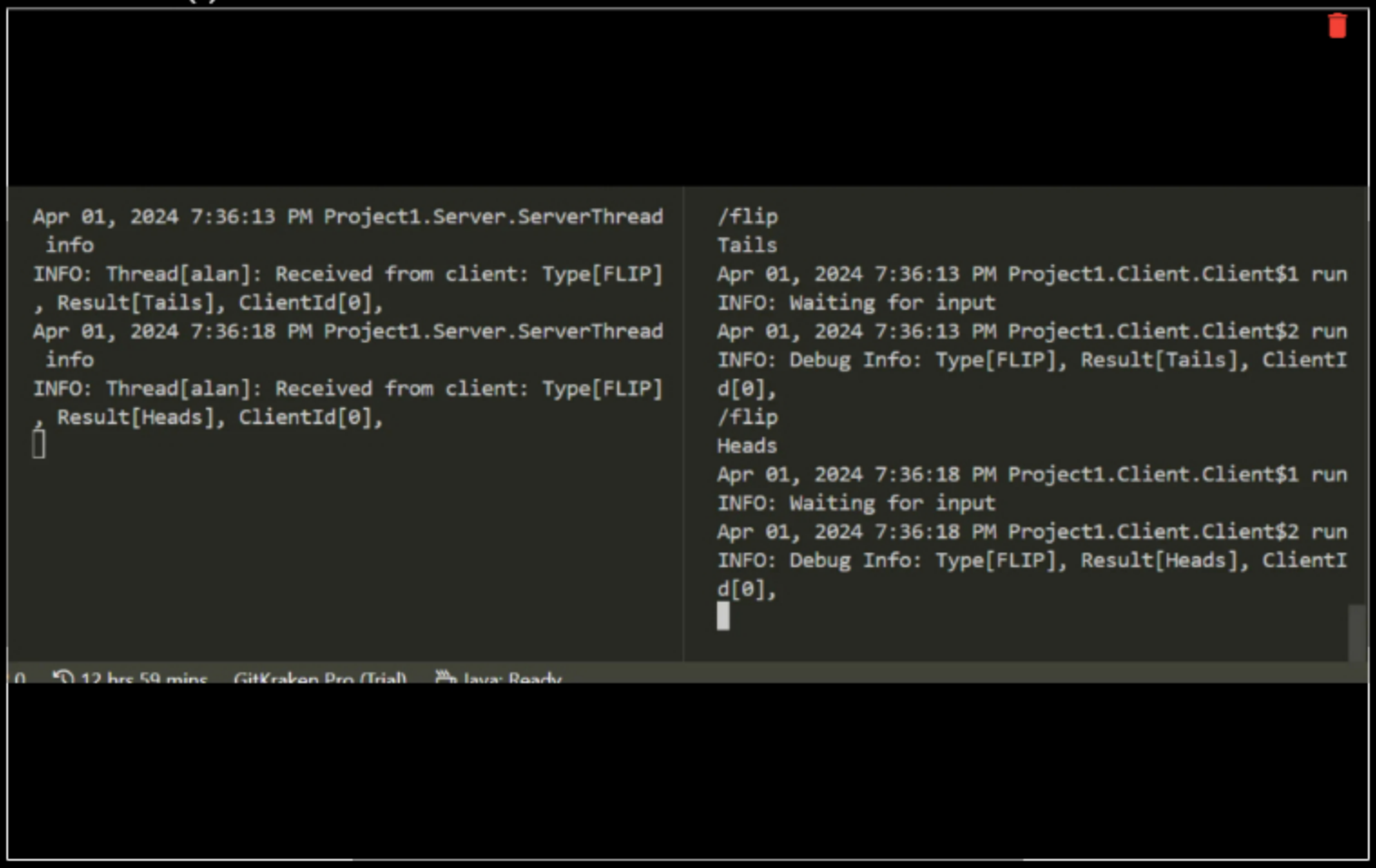
converts to a payload and forwards onto the server thread

Checklist Items (0)

```java
case FLIP:
    try{
        FlipPayload fp = (FlipPayload) p;
        fp.toString();//am3485 4/1/2024
        send(fp);
    }catch (Exception e){
        this.sendMessage(Constants.DEFAULT_CLIENT_ID,
                message:"You can only use the /turn commmand in a GameRoom and not the Lobby");
    }
    break;
```

serverthread gets the payload and sends it to the room using send()

Checklist Items (0)



room correctly handling flip

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

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

Response:

the flip command is handle and processed in client it is then sent as a payload to server where it is processed. its shared to the room using processPayload and back to the clients using the send method.

Demonstrate Formatted Messages (4 pts.)

COLLAPSE

Task #1 - Points: 1

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

Details:

Note: this processing is server-side

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

Checklist

*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

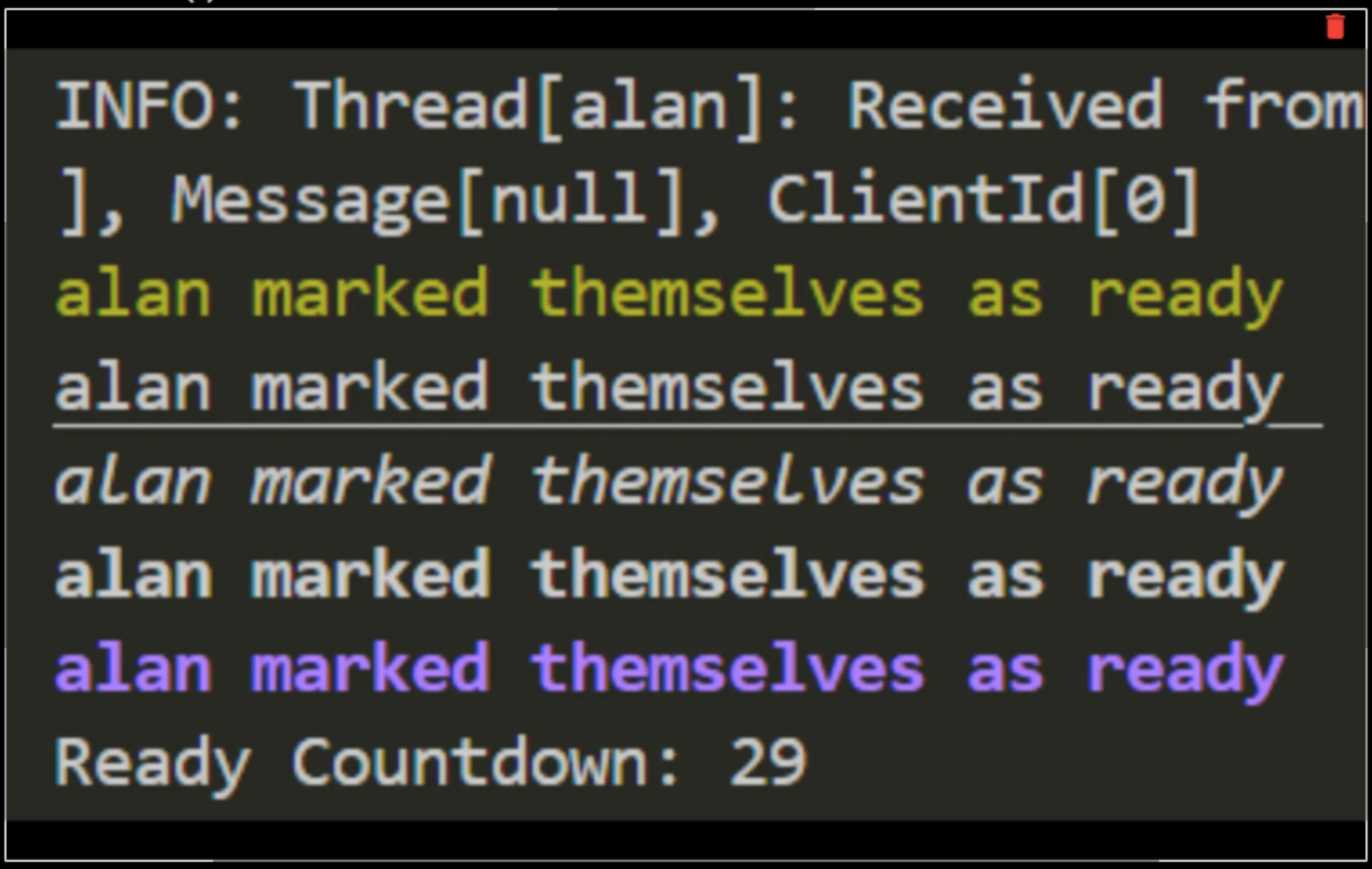
Small Medium Large

```
System.out.println(TextFX.colorize(players.get(playerId).getClientName() + " marked themselves as ready ",
    Color.YELLOW)); //am3485 4/3/2024
System.out.println(TextFX.underline(players.get(playerId).getClientName() + " marked themselves as ready "));
System.out.println(TextFX.italize(players.get(playerId).getClientName() + " marked themselves as ready "));
System.out.println(TextFX.embolden(players.get(playerId).getClientName() + " marked themselves as ready "));
System.out.println(TextFX.colorize(TextFX.embolden(players.get(playerId).getClientName() + " marked themselves as ready "), Color.PURPLE));
readyCheck();
```

Red = room code for colorize Green = room code for underline Blue = room code for italics Yellow = room code for bold

White = room code for bold and color working together

Checklist Items (0)



All of the code working

Checklist Items (0)

^COLLAPSE^

Task #2 - Points: 1

Text: Explain the following

Checklist

*The checkboxes are for your own tracking

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

Response:

for the code to work you have to use ansi codes for colors BLACK("\033[0;30m"), RED("\033[0;31m"), GREEN("\033[0;32m"), YELLOW("\033[0;33m"), BLUE("\033[0;34m"), PURPLE("\033[0;35m"), CYAN("\033[0;36m"), WHITE("\033[0;37m"); and then reset, bold, italic, and underline each had "\033[0m", "\033[1m", "\033[3m", and "\033[4m" respectively and then after you end it with a reset

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/alanpear/am3485-it114-002/pull/12

^COLLAPSE ^

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

I faced issues understanding payloads how to send them back and forth and what to send. I solved them by reading the code very carefully and trying to google everything i didnt understand.

^COLLAPSE ^

Task #3 - Points: 1

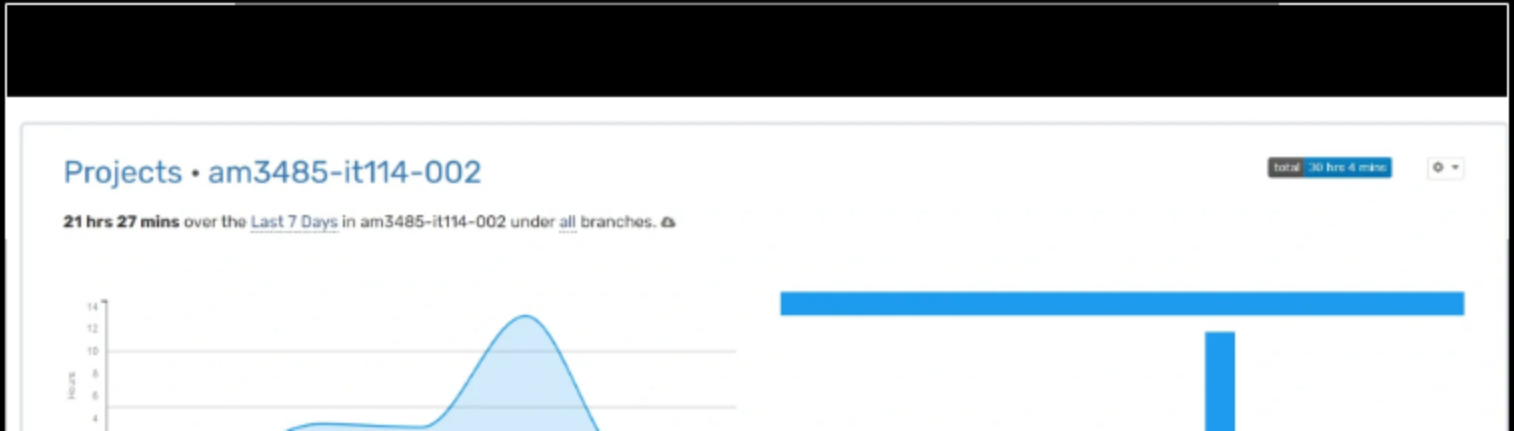
Text: WakaTime Screenshot

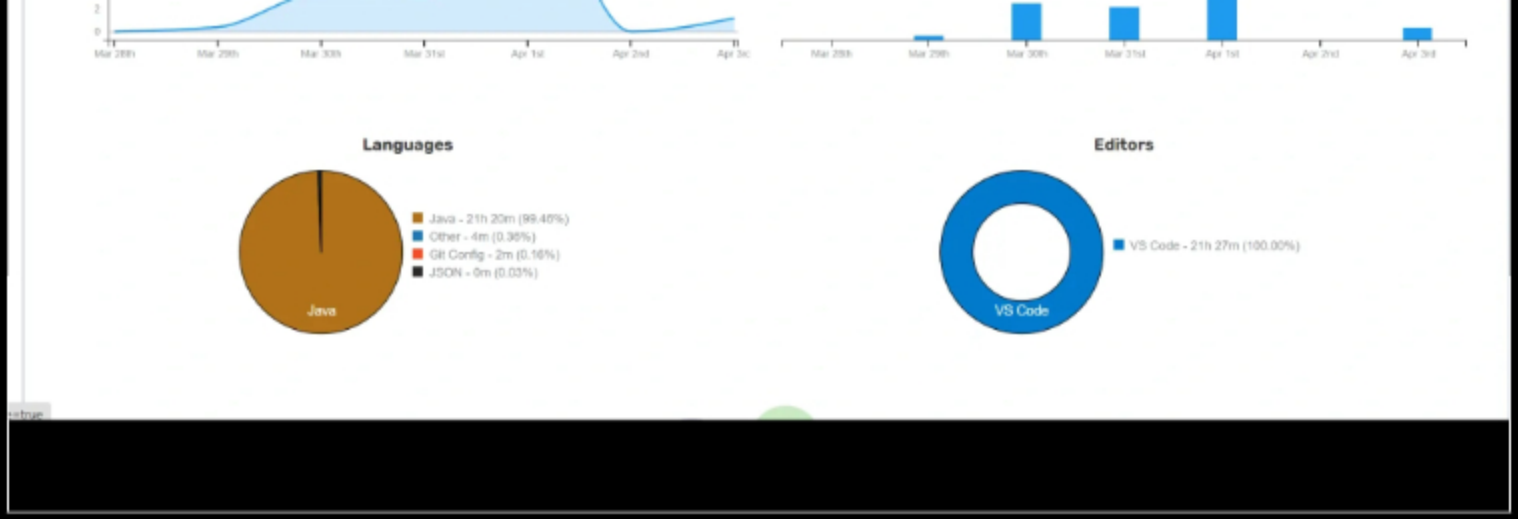
Details:
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots:

Gallery Style: Large View

Small Medium Large





WakaTime Screenshot

End of Assignment