

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 ^



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
<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.java • Constants.java Project1\... Payload.java Project1\... PayloadType.java Project1\... FlipPayload.java • Phase.java Project1\... Player.java

Project1 > Common > RollPayload.java > RollPayload > RollPayload()

```
1 package Project1.Common;
2
3 You, 9 minutes ago | 1 author (You)
4 public class RollPayload extends Payload {
5     private int numDice;
6     private int numSides;
7     private int results;
8
9     //am3485 4/1/2024 You, 9 minutes ago • Uncommitted changes
10    public RollPayload() {
11        setPayloadType(PayloadType.ROLL);
12    }
13
14    public int getNumDice() {
15        return numDice;
16    }
17
18    public void setNumDice(int numDice) {
19        this.numDice = numDice;
20    }
21
22    public int getNumSides() {
23        return numSides;
24    }
25
26    public void setNumSides(int numSides) {
27        this.numSides = numSides;
28    }
29
30    public int getResults() {
31        return results;
32    }
33
34    public void setResults(int results) {
35        this.results = results;
36    }
37
38    @Override
39    public String toString() {
40        return String.format(format:"Type[%s], Result[%s], ClientId[%s], ", getPayloadType().toString(),
41                               getResult(), getClientId());
42    }
43 }
44
45 } <- #3-41 public class RollPayload extends Payload
```

Roll payload code

Checklist Items (0)

Project1\... PayloadType.java Project1\... FlipPayload.java • Phase.java Project1\... Player.java Project1\... ReadyPayload.java Project1\... RoomResultsPayload.java

Project1 > Common > FlipPayload.java > ...

```
1 ...
2 package Project1.Common;
3 ...
4 public class FlipPayload extends Payload {
5     private String result;
6
7     public FlipPayload(){
8         setPayloadType(PayloadType.FLIP);
9     }
10
11    public String getResult() {
12        return result;
13    }
14
15    public void setResult(String result) {
16        this.result = result;
17    }
18
19    @Override
20    public String toString() {
21        return String.format(format:"Type[%s], Result[%s], ClientId[%s], ", getPayloadType().toString(),
22                               getResult(), getClientId());
23    }
24 }
25 } <- #3-23 public class FlipPayload extends Payload
26 //am3485 4/1/2024
```

FlipPayload() code

Checklist Items (0)

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

```
INFO: Thread[null]: Received from client: Type[CONNECT], Message[null], ClientId
[0], Client name alan
Apr 01, 2024 7:01:13 PM Project1.Server.Server joinRoom
INFO: Thread-1 joining room lobby
Apr 01, 2024 7:01:19 PM Project1.Server.ServerThread info
INFO: Thread[alan]: Received from client: Type[MESSAGE], Message['/flip'], Client
Id[0]
Apr 01, 2024 7:01:19 PM Project1.Server.Room info
INFO: Room[lobby]: Sending message to 1 clients
Apr 01, 2024 7:01:28 PM Project1.Server.ServerThread info
INFO: Thread[alan]: Received from client: Type[FLIP], Result[Heads], ClientId[0]
,
Apr 01, 2024 7:01:35 PM Project1.Server.ServerThread info
INFO: Thread[alan]: Received from client: Type[FLIP], Result[Tails], ClientId[0]
,
Apr 01, 2024 7:01:45 PM Project1.Server.ServerThread info
INFO: Thread[alan]: Received from client: Type[ROLL], Result[8], ClientId[0],
Apr 01, 2024 7:01:53 PM Project1.Server.ServerThread info
INFO: Thread[alan]: Received from client: Type[ROLL], Result[22], ClientId[0],
[]

INFO: Debug Info: Type[FLIP], Result[Heads], ClientId[0],
/flip
Tails
Apr 01, 2024 7:01:35 PM Project1.Client.Client$1 run
INFO: Waiting for input
Apr 01, 2024 7:01:35 PM Project1.Client.Client$2 run
INFO: Debug Info: Type[FLIP], Result[Tails], ClientId[0],
/roll 2d6
8
Apr 01, 2024 7:01:45 PM Project1.Client.Client$1 run
INFO: Waiting for input
Apr 01, 2024 7:01:45 PM Project1.Client.Client$2 run
INFO: Debug Info: Type[ROLL], Result[8], ClientId[0],
/roll 100
22
Apr 01, 2024 7:01:53 PM Project1.Client.Client$1 run
INFO: Waiting for input
Apr 01, 2024 7:01:53 PM Project1.Client.Client$2 run
INFO: Debug Info: Type[ROLL], Result[22], ClientId[0],
[]
```

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

Checklist Items (0)

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 were made using the parent payloads as a guide and then edited to show the more important information.

Demonstrate Roll Command (2 pts.)

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 uuid and date comment
<input type="checkbox"/> #5	1	Each screenshot should be clearly captioned

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
Project1 > Client > J Client.java > Client > sendRoll(String)
28 public enum Client {
29     ...
266 private void sendRoll(String text) throws IOException {
267     Random gen = new Random();
268     if (text.toLowerCase().contains("d")) {
269         try {
270             String[] diceNumbers = text.split(regex:"[dD]"); // 0 index is the amount and 1st index is max
271             int counter = 0;
272             for (int i = 0; i < Integer.parseInt(diceNumbers[0]); i++) {
273                 counter += gen.nextInt(Integer.parseInt(diceNumbers[1])) + 1;
274             }
275             System.out.println(counter);
276             RollPayload p = new RollPayload();
277             p.setPayloadType(PayloadType.ROLL);
278             p.setNumDice(Integer.parseInt(diceNumbers[1]));
279             p.setResults(counter);
280             ...

```

```

281         out.writeObject(p);
282     } catch (Exception e) {
283         System.out.println(x:"invalid!!!!");
284     }
285     //am3485 4/1/2024
286 } else if (!text.toLowerCase().contains(s:"d") && text.matches(regex:"\\d+")) { // Checks if it doesn't contain a "d" AND
287     // if there are only digits in the string
288     // Parses the text to a number
289     int maxRoll = Integer.parseInt(text);
290     // Use gen object to make a random number
291     int counter = gen.nextInt(maxRoll) + 1;
292     System.out.println(counter);
293     RollPayload p = new RollPayload();
294     p.setPayloadType(PayloadType.ROLL);
295     p.setNumSides(maxRoll);
296     p.setResults(counter);
297     out.writeObject(p);
298 } else {
299     // At this point it would be invalid so do something with it
300     return;
301 }

```

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);
    } 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;

```

```

        rollBy100 = (rollBy100) * 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 command 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)

PROBLEMS 42

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

✓ TERMINAL

Apr 01, 2024 7:18:53 PM Project1.Server.ServerThread info

INFO: Thread[alan]: Received from client: Type[ROLL], Result[1], ClientId[0],

Apr 01, 2024 7:19:07 PM Project1.Server.ServerThread info

INFO: Thread[alan]: Received from client: Type[ROLL], Result[30], ClientId[0],

[]

/roll 100

1

Apr 01, 2024 7:18:53 PM Project1.Client.Client\$1 run

INFO: Waiting for input

Apr 01, 2024 7:18:53 PM Project1.Client.Client\$2 run

INFO: Debug Info: Type[ROLL], Result[1], ClientId[0],

/roll 10d6

30

Apr 01, 2024 7:19:07 PM Project1.Client.Client\$1 run

INFO: Waiting for input

Apr 01, 2024 7:19:07 PM Project1.Client.Client\$2 run

INFO: Debug Info: Type[ROLL], Result[30], ClientId[0],

[]

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

Checklist Items (0)

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 ^

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

```
return true;
} else if (text.startsWith(FLIP)) {
    try { //am3485 4/1/2024
        sendFlip();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return true;
} <- #230-237 else if (text.startsWith(FLIP))
return false;
<- #144-239 private boolean processClientCommand(String
```

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

Checklist Items (0)

```
private void sendFlip() throws IOException {
    Random gen = new Random();
    FlipPayload p = new FlipPayload();
    int c = gen.nextInt(bound:1000) % 2;
    if (c == 1) {
        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)

```
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 command in a GameRoom and not the Lobby");
    }
    break;
```


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

Checklist Items (0)

```
Apr 01, 2024 7:36:13 PM Project1.Server.ServerThread
info
INFO: Thread[alan]: Received from client: Type[FLIP]
, Result[Tails], ClientId[0],
Apr 01, 2024 7:36:18 PM Project1.Server.ServerThread
info
INFO: Thread[alan]: Received from client: Type[FLIP]
, Result[Heads], ClientId[0],
```

```
/flip
Tails
Apr 01, 2024 7:36:13 PM Project1.Client.Client$1 run
INFO: Waiting for input
Apr 01, 2024 7:36:13 PM Project1.Client.Client$2 run
INFO: Debug Info: Type[FLIP], Result[Tails], ClientI
d[0],
/flip
Heads
Apr 01, 2024 7:36:18 PM Project1.Client.Client$1 run
INFO: Waiting for input
Apr 01, 2024 7:36:18 PM Project1.Client.Client$2 run
INFO: Debug Info: Type[FLIP], Result[Heads], ClientI
d[0],
```

room correctly handling flip

Checklist Items (0)

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.)

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 uuid and date comment
<input type="checkbox"/> #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)); //an3485 4/3/2024
System.out.println(TextFX.underline(players.get(playerId).getClientName() + " marked themselves as ready "));
System.out.println(TextFX.italicize(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

Checklist Items (0)

```
INFO: Thread[alan]: Received from
], Message[null], ClientId[0]
alan marked themselves as ready
alan marked themselves as ready
alan marked themselves as ready
alan marked themselves as ready
alan marked themselves as ready
Ready Countdown: 29
```

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
<input type="checkbox"/> #1	1	Which special characters translate to the desired effect
<input type="checkbox"/> #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 ^

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>

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.

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

Projects • am3485-it114-002

total 20 hrs 4 mins

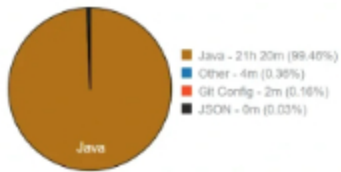


21 hrs 27 mins over the Last 7 Days in am3485-it114-002 under all branches. 





Languages



Editors



WakaTime Screenshot

End of Assignment