Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-451-M2024/it114-milestone-2-chatroom-2024-m24/grade/sa2796

IT114-451-M2024 - [IT114] Milestone 2 Chatroom 2024 (M24)

Submissions:

Submission Selection

1 Submission [active] 6/29/2024 11:14:17 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

Tasks: 8 Points: 10.00

● Payloads (2 pts.)

COLLAPSE

● COLLAPSE

Task #1 - Points: 1

Text: Base Payload Class

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

#1) Show screenshot of the

Caption (required) ✓
Describe/highlight what's being shown
Displaying screenshot of the Payload.java with UCID and Date

Explanation (required) ✓
Briefly explain the purpose of each property and serialization

PREVIEW RESPONSE

The properties in Payload.java are "private PayloadType payloadType;", "private long clientId;", "private String message;". The purpose for the first property is it holds the types of variables which can be connect, disconnect, etc. The purpose of the second property is so it can hold a unique identifier for the client. The purpose for the third property is so it can hold the message of the payload an is a bridge for

#2) Show screenshot examples

Caption (required) ✓
Describe/highlight what's being shown
Displaying screenshot examples of the terminal output for base Payload objects

communication between the server and client.

As for the serialization, it allows, "Payload" objects to be converted to byte stream and is passed on to a network.

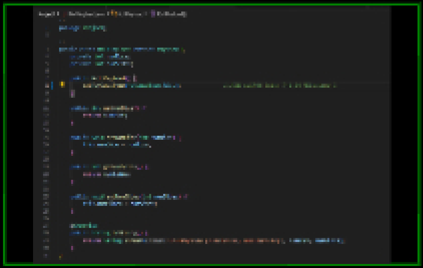
^COLLAPSE ^

Task #2 - Points: 1

Text: RollPayload Class

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

#1) Show screenshot of the RollPayload.java (or



Caption (required) ✓
Describe/highlight what's being shown
Displaying screenshot of the RollPayload.java

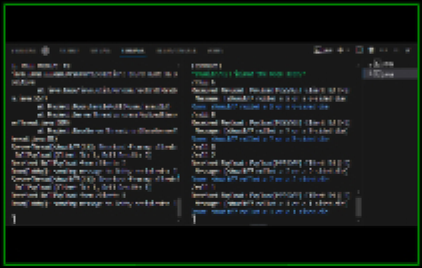
Explanation (required) ✓
Briefly explain the purpose of each property

PREVIEW RESPONSE

rollResult's purpose is it is a private integer field that stores the result of a roll operation. It holds the value that client could send/recieve in payload.

RollPayload()'s purpose is it

#2) Show screenshot examples of the terminal output for



Caption (required) ✓
Describe/highlight what's being shown
Displaying screenshot examples of the terminal output for base RollPayload objects

initializes payload tyoe to "ROLL" using the method "setPayloadType(PayloadType.ROLL);"

getRollRedult(): The public method returns the current calue of rollResult.

setRollResult(int rollResult): This public method sets value of roll result to the provided integer.

● Client Commands (4 pts.)

^COLLAPSE ^

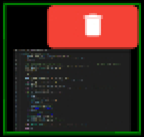
● ^COLLAPSE ^

Task #1 - Points: 1

Text: Roll Command

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.

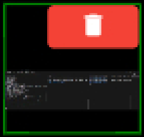
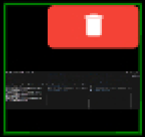
#1) Show the client side code



Caption (required) ✓
Describe/highlight what's being shown
Displaying the client side code for handling /roll #

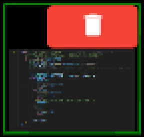
Explanation (required) ✓
Briefly explain the logic
PREVIEW RESPONSE
Essentially, the code

#2) Show the output of a few

Caption (required) ✓
Describe/highlight what's being shown
Displaying output of a few examples of /roll # (related payload output should be visible)

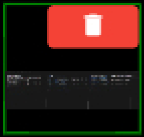
#3) Show the client side code



Caption (required) ✓
Describe/highlight what's being shown
Displaying the client side code for handling /roll #d# (related payload output should be visible)

Explanation (required) ✓
Briefly explain the logic
PREVIEW RESPONSE

#4) Show the output of a few



Caption (required) ✓
Describe/highlight what's being shown
Displaying the output of a few examples of /roll #d#

with /roll and if it doesn't have "d" it goes to one die and parses numSides. As for handling errors, the code utilizes try-catch to handle NumberFormatException which makes sure only numbers are processed. In the event the input is invalid, an error message is printed.

The code sees if the text begins with /roll and if it has "d" it splits the string to get numDice and numSides. The first part being parsed is numDice while the second part being parsed is numSides.

#5) Show the ServerThread

Caption (required) ✓

Describe/highlight what's being shown

Displaying the ServerThread code receiving the RollPayload

Explanation (required) ✓

Briefly explain the logic

PREVIEW RESPONSE

Switch statement sees the type of the payload using "payload.getPayloadType()) "case ROLL" handles the RollPayload types. The method sees if payload is instance of RollPayload to make sure the correct type is processed. Then, the payload is cast to

#6) Show the Room code that

Caption (required) ✓

Describe/highlight what's being shown

Displaying the Room code that processes both Rolls and sends the response

Explanation (required) ✓

Briefly explain the logic

PREVIEW RESPONSE

For handling the roll, it extracts numDIce and numSides from RollPayload and generates a random roll and appends every result to the message. It then calls "sendMessage" to showcase the result to the clients in the room.

For the sendMessage method, it makes sure

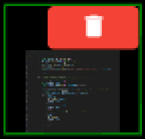
RollPayload which has a log message printed to show that RollPayload has been recieved from client.
Finally, the method calls "currentRoom.handleRoll(th rollPayload)" and the hanedRoll method is the one in charge of the logic of processing the roll and showcasing the result to all the clients in the room.

method, it makes sure the message iterates to all clients in room and attempts to send the message to every client. It removes clients from room if message fails to send

Task #2 - Points: 1

Text: Flip Command

#1) Show the client side code



Caption (required) ✓

Describe/highlight what's being shown

Displaying the client side code for handling /flip

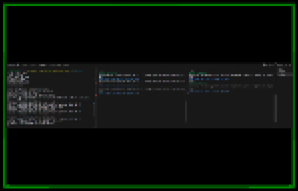
Explanation (required) ✓

Briefly explain the logic

PREVIEW RESPONSE

FlipPayload is created when /flip is detected and the constructor "new FlipPayload() initializes the payload and sets it to "FLIP" type.
The payload is sent to the server using the

#2) Show the output of a few



Caption (required) ✓

Describe/highlight what's being shown

Displaying the output of a few examples of /flip (related payload output should be visible) the server using the send method and the method serializes payload which sends it to the server.

The client gets a printed confirmation message that shows that FlipPayload is sent "System.out.println(TextFX.colorize("Sending FlipPayload", Color.GREEN));"

● Text Formatting (3 pts.)

^COLLAPSE ^

● ^COLLAPSE ^

Task #1 - Points: 1

Text: Text Formatting

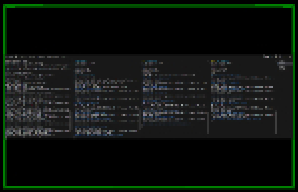
Details:

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
Note: Each text trigger must wrap the text that you want to affect
Note: Slash commands are not an accepted solution, the text must be transformed
Note: You do not need to use the same symbols in the below example, it's just an example, also, the below example doesn't show the "correct" output for colors, I'm leaving the proper conversion up to research on your own.
See proposal for an example.

#1) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
Displaying the code processing special characters for bold,

#2) Show examples of each:



Caption (required) ✓
Describe/highlight what's being shown
Displaying examples of bold, italic, underline, colors (red, green, blue)

italic, underline, colors, and converting them to other characters

and combination of bold, italic, underline and a color

Explanation (required)

✓

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

PREVIEW RESPONSE

Essentially, "processTextEffects" method converts special characters into HTML tags for formatting text.

Asteriks are used for Bold and Italics and is intuitive for users.

Underscores are used for Underlines and is chosen for being simple and distinct from astericks.

Hash is used for colors as it is a simple way to specify colors without utilizing other placeholders.

The method uses expressions to find specific patterns and replace them with HTML tags

● 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/SHUAIB2796/sa2796-IT114-451/pull/10

COLLAPSE

Task #2 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

During this assignment I had trouble with getting the /roll command to work but with the assistance from Lucas I was able to get it to function and run properly. Other than that, I think I did pretty good overall with the /flip and TextFx parts.

COLLAPSE

Task #3 - Points: 1

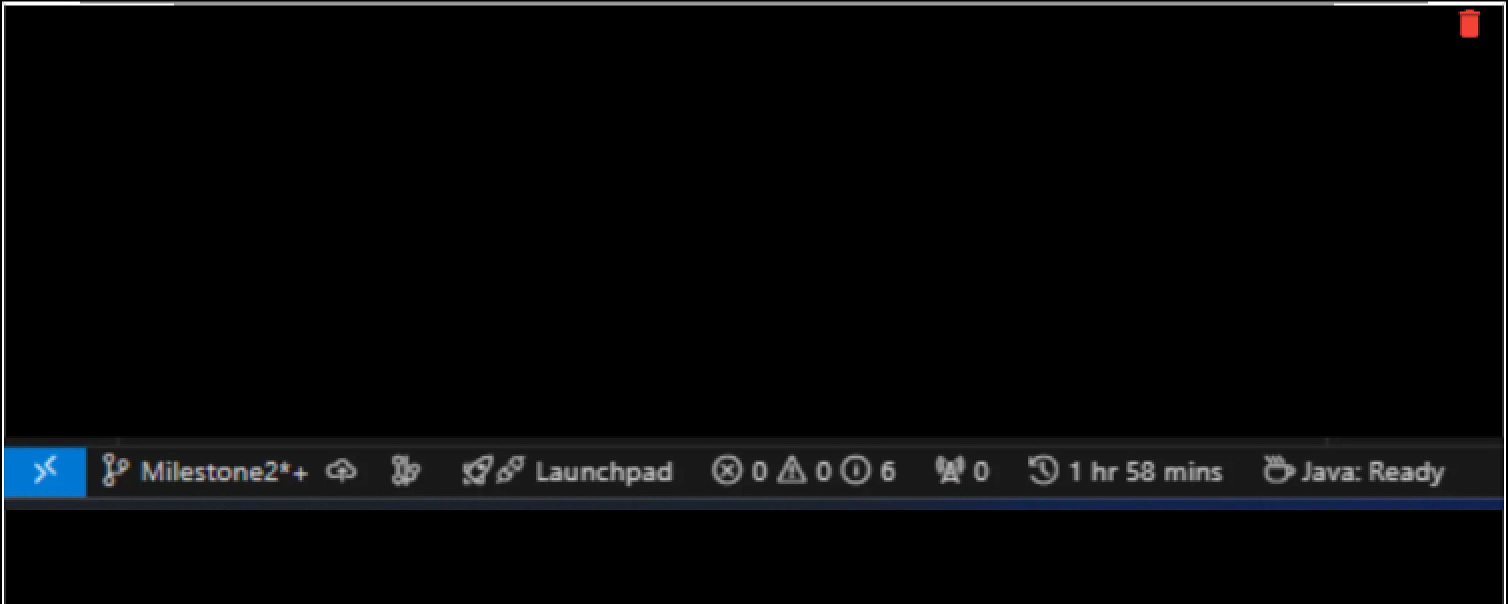
Text: WakaTime Screenshot

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

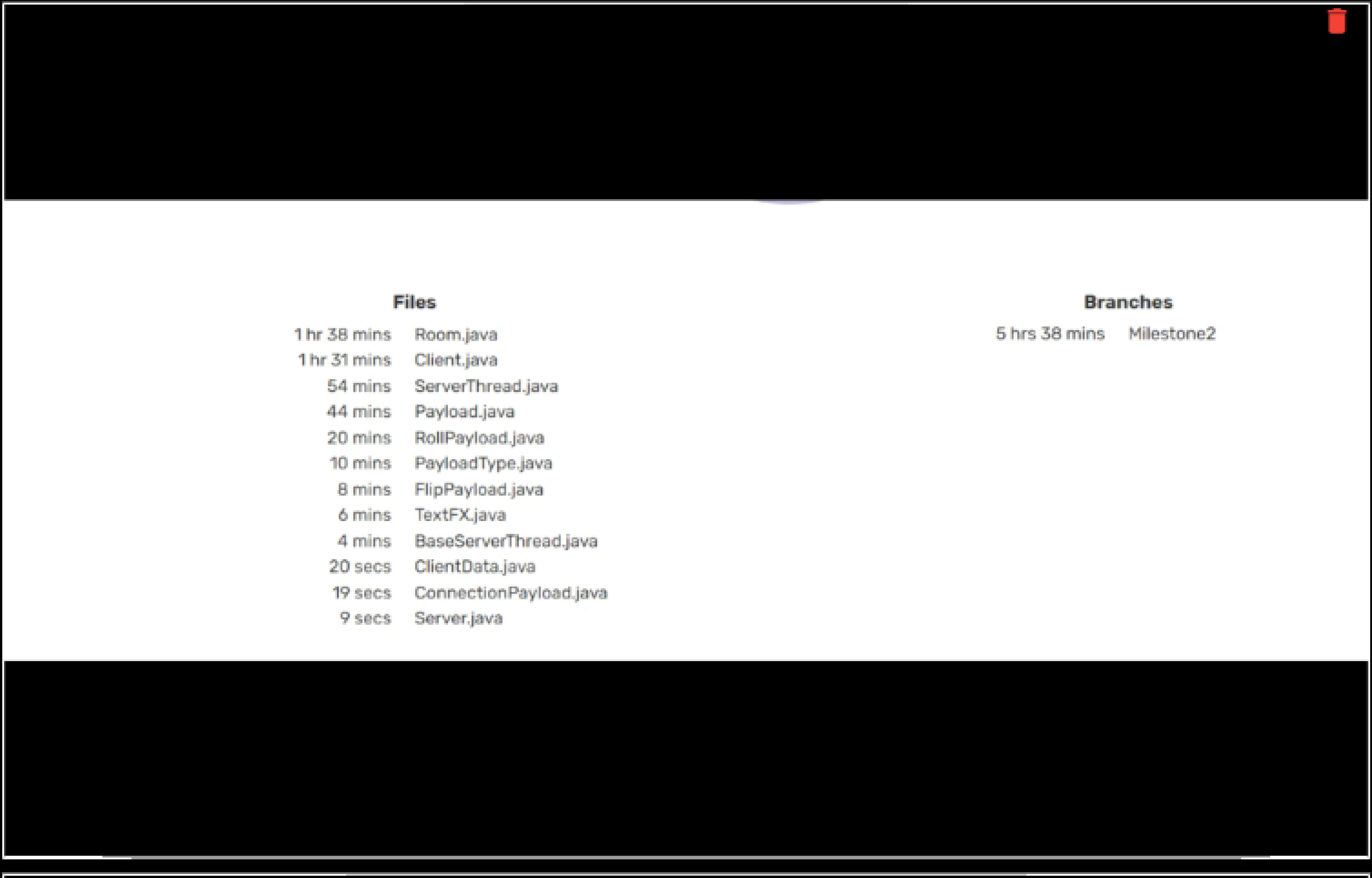
Task Screenshots:

Gallery Style: Large View

Small Medium Large


Milestone2*+ Launchpad 0 0 6 0 1 hr 58 mins Java: Ready

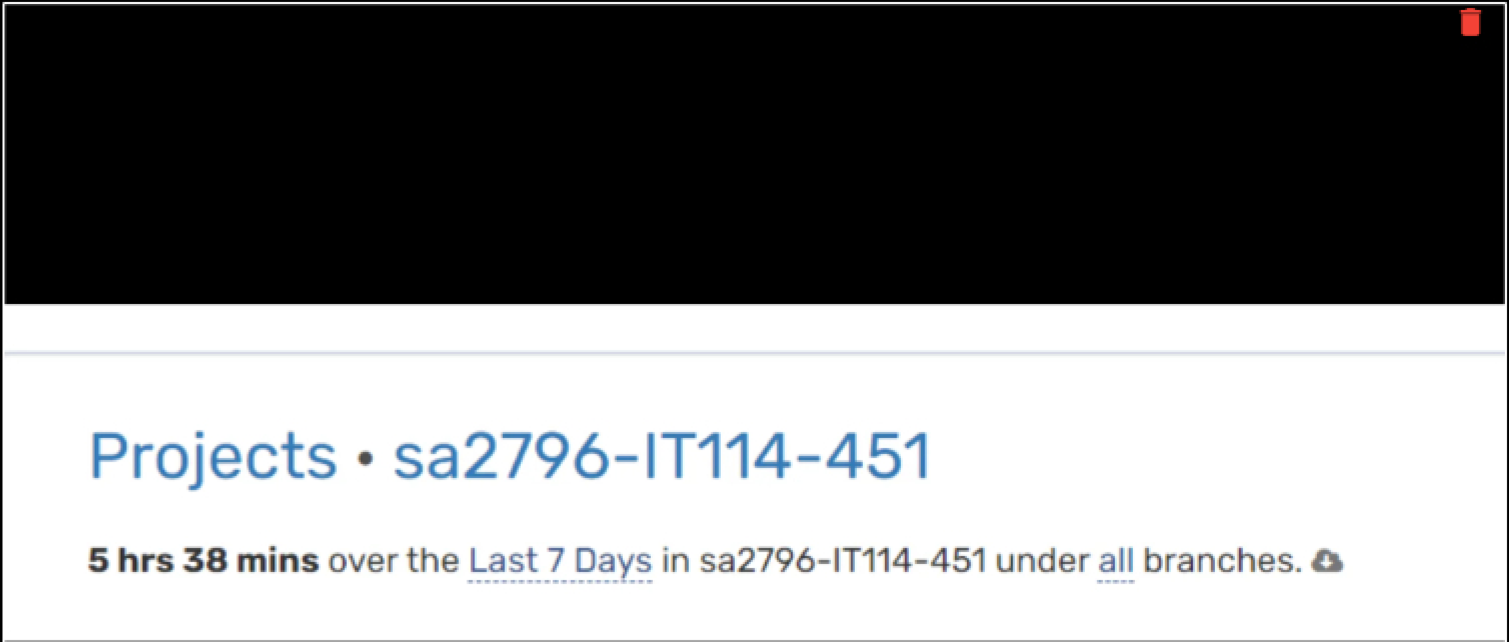
On VSCODE WakaTime



Files	
1 hr 38 mins	Room.java
1 hr 31 mins	Client.java
54 mins	ServerThread.java
44 mins	Payload.java
20 mins	RollPayload.java
10 mins	PayloadType.java
8 mins	FlipPayload.java
6 mins	TextFX.java
4 mins	BaseServerThread.java
20 secs	ClientData.java
19 secs	ConnectionPayload.java
9 secs	Server.java

Branches	
5 hrs 38 mins	Milestone2

Milestone 2 time and individual file times



Projects • sa2796-IT114-451

5 hrs 38 mins over the Last 7 Days in sa2796-IT114-451 under all branches.

Overall time

End of Assignment