

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

Assignment: [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/1ONmvEveI97GTFPGfVwwQC96xSsobbSbk56145XizQG4/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



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



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

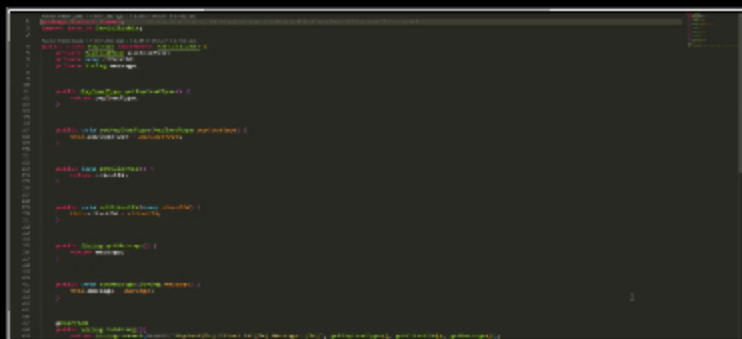
Task Screenshots

Gallery Style: 2 Columns

4

2

1



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

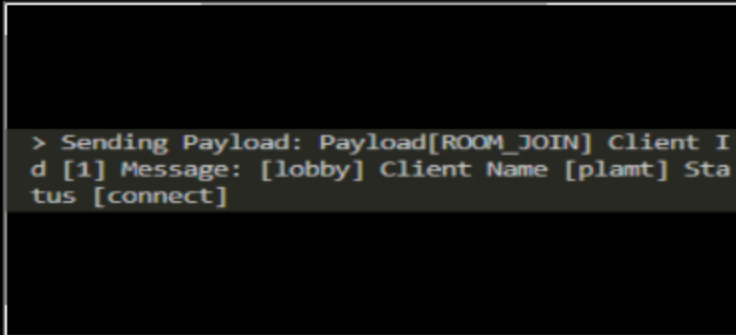


Group: Payloads
Task #1: Base Payload Class
Sub Task #2: Show screenshot examples of the terminal output for base Payload objects

Task Screenshots

Gallery Style: 2 Columns

4 2 1



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



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

^ COLLAPSE ^

i Details:

All code screenshots must have ucid/date visible.



Columns: 1

Sub-Task

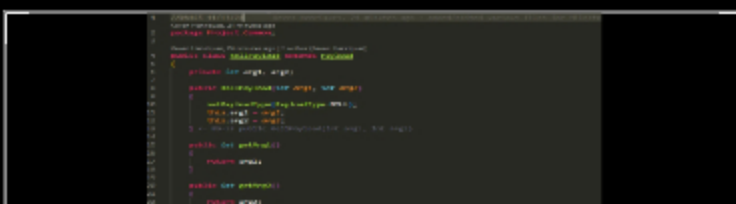


Group: Payloads
Task #2: RollPayload Class
Sub Task #1: Show screenshot of the RollPayload.java (or equivalent)

Task Screenshots

Gallery Style: 2 Columns

4 2 1





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

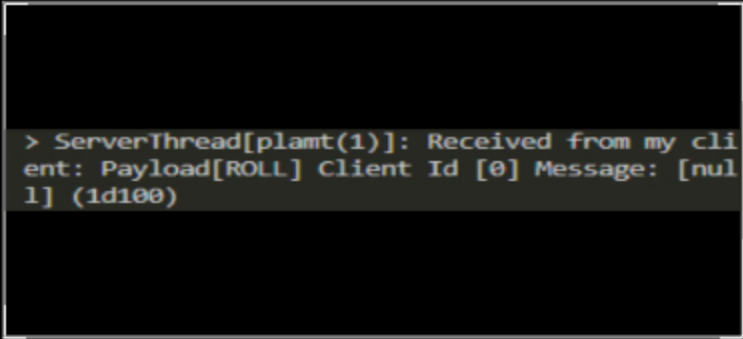
100%

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



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

Group: Client Commands

Task #1: Roll Command

Weight: ~50%

Points: ~2.00

^ COLLAPSE ^

i 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

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

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

```

1 def __getitem__(self, item):
2     try:
3         if isinstance(item, slice):
4             # slicing: parse the start/stop/step
5             start = self._convert(item.start)
6             stop = self._convert(item.stop)
7             step = self._convert(item.step)
8             return self._getitem(slice(start, stop, step))
9         else:
10             # indexing: get the value
11             return self._getitem(item)
12     except:
13         # if the item is not a slice or an integer, raise an error
14         raise ValueError("Invalid index: %s" % item)
15
16 def _convert(self, item):
17     """Convert a value to an integer or a float, or raise an error if it's not a number"""
18     try:
19         return int(item)
20     except:
21         try:
22             return float(item)
23         except:
24             raise ValueError("Invalid index: %s" % item)
25
26 def __setitem__(self, item, value):
27     """Set the value of the element at the given index"""
28     try:
29         if isinstance(item, slice):
30             # slicing: set the value of the slice
31             self._setitem(slice(item.start, item.stop, item.step), value)
32         else:
33             # indexing: set the value of the element
34             self._setitem(item, value)
35     except:
36         # if the item is not a slice or an integer, raise an error
37         raise ValueError("Invalid index: %s" % item)
38
39 def __delitem__(self, item):
40     """Delete the element at the given index"""
41     try:
42         if isinstance(item, slice):
43             # slicing: delete the slice
44             self._delitem(slice(item.start, item.stop, item.step))
45         else:
46             # indexing: delete the element
47             self._delitem(item)
48     except:
49         # if the item is not a slice or an integer, raise an error
50         raise ValueError("Invalid index: %s" % item)
51
52 def __iter__(self):
53     """Return an iterator over the elements of the array"""
54     return self._iter()
55
56 def __len__(self):
57     """Return the length of the array"""
58     return self._len()
59
60 def __repr__(self):
61     """Return a string representation of the array"""
62     return self._repr()
63
64 def __str__(self):
65     """Return a string representation of the array"""
66     return self._str()
67
68 def __add__(self, other):
69     """Add the other array to this array"""
70     return self._add(other)
71
72 def __sub__(self, other):
73     """Subtract the other array from this array"""
74     return self._sub(other)
75
76 def __mul__(self, other):
77     """Multiply this array by the other array"""
78     return self._mul(other)
79
80 def __div__(self, other):
81     """Divide this array by the other array"""
82     return self._div(other)
83
84 def __mod__(self, other):
85     """Return the remainder of the division of this array by the other array"""
86     return self._mod(other)
87
88 def __pow__(self, other):
89     """Return the power of this array to the other array"""
90     return self._pow(other)
91
92 def __matmul__(self, other):
93     """Return the matrix product of this array and the other array"""
94     return self._matmul(other)
95
96 def __radd__(self, other):
97     """Return the other array plus this array"""
98     return self._radd(other)
99
100 def __rsub__(self, other):
101     """Return the other array minus this array"""
102     return self._rsub(other)
103
104 def __rmul__(self, other):
105     """Return the other array multiplied by this array"""
106     return self._rmul(other)
107
108 def __rdiv__(self, other):
109     """Return the other array divided by this array"""
110     return self._rdiv(other)
111
112 def __rmod__(self, other):
113     """Return the remainder of the division of the other array by this array"""
114     return self._rmod(other)
115
116 def __rpow__(self, other):
117     """Return the power of the other array to this array"""
118     return self._rpow(other)
119
120 def __rmatmul__(self, other):
121     """Return the matrix product of the other array and this array"""
122     return self._rmatmul(other)
123
124 def __iadd__(self, other):
125     """Add the other array to this array in place"""
126     return self._iadd(other)
127
128 def __isub__(self, other):
129     """Subtract the other array from this array in place"""
130     return self._isub(other)
131
132 def __imul__(self, other):
133     """Multiply this array by the other array in place"""
134     return self._imul(other)
135
136 def __idiv__(self, other):
137     """Divide this array by the other array in place"""
138     return self._idiv(other)
139
140 def __imod__(self, other):
141     """Return the remainder of the division of this array by the other array in place"""
142     return self._imod(other)
143
144 def __ipow__(self, other):
145     """Return the power of this array to the other array in place"""
146     return self._ipow(other)
147
148 def __imatmul__(self, other):
149     """Return the matrix product of this array and the other array in place"""
150     return self._imatmul(other)
151
152 def __iand__(self, other):
153     """Return the bitwise AND of this array and the other array"""
154     return self._iand(other)
155
156 def __ior__(self, other):
157     """Return the bitwise OR of this array and the other array"""
158     return self._ior(other)
159
160 def __ixor__(self, other):
161     """Return the bitwise XOR of this array and the other array"""
162     return self._ixor(other)
163
164 def __ilshift__(self, other):
165     """Return the left shift of this array by the other array"""
166     return self._ilshift(other)
167
168 def __irshift__(self, other):
169     """Return the right shift of this array by the other array"""
170     return self._irshift(other)
171
172 def __iandl__(self, other):
173     """Return the left shift of this array by the other array, with a mask"""
174     return self._iandl(other)
175
176 def __iorl__(self, other):
177     """Return the right shift of this array by the other array, with a mask"""
178     return self._iorl(other)
179
180 def __ixorl__(self, other):
181     """Return the bitwise XOR of this array and the other array, with a mask"""
182     return self._ixorl(other)
183
184 def __ilshl__(self, other):
185     """Return the left shift of this array by the other array, with a mask and a shift"""
186     return self._ilshl(other)
187
188 def __irshl__(self, other):
189     """Return the right shift of this array by the other array, with a mask and a shift"""
190     return self._irshl(other)
191
192 def __iandl__(self, other):
193     """Return the left shift of this array by the other array, with a mask and a shift"""
194     return self._iandl(other)
195
196 def __iorl__(self, other):
197     """Return the right shift of this array by the other array, with a mask and a shift"""
198     return self._iorl(other)
199
200 def __ixorl__(self, other):
201     """Return the bitwise XOR of this array and the other array, with a mask and a shift"""
202     return self._ixorl(other)
203
204 def __ilshl__(self, other):
205     """Return the left shift of this array by the other array, with a mask and a shift"""
206     return self._ilshl(other)
207
208 def __irshl__(self, other):
209     """Return the right shift of this array by the other array, with a mask and a shift"""
210     return self._irshl(other)
211
212 def __iandl__(self, other):
213     """Return the left shift of this array by the other array, with a mask and a shift"""
214     return self._iandl(other)
215
216 def __iorl__(self, other):
217     """Return the right shift of this array by the other array, with a mask and a shift"""
218     return self._iorl(other)
219
220 def __ixorl__(self, other):
221     """Return the bitwise XOR of this array and the other array, with a mask and a shift"""
222     return self._ixorl(other)
223
224 def __ilshl__(self, other):
225     """Return the left shift of this array by the other array, with a mask and a shift"""
226     return self._ilshl(other)
227
228 def __irshl__(self, other):
229     """Return the right shift of this array by the other array, with a mask and a shift"""
230     return self._irshl(other)
231
232 def __iandl__(self, other):
233     """Return the left shift of this array by the other array, with a mask and a shift"""
234     return self._iandl(other)
235
236 def __iorl__(self, other):
237     """Return the right shift of this array by the other array, with a mask and a shift"""
238     return self._iorl(other)
239
240 def __ixorl__(self, other):
241     """Return the bitwise XOR of this array and the other array, with a mask and a shift"""
242     return self._ixorl(other)
243
244 def __ilshl__(self, other):
245     """Return the left shift of this array by the other array, with a mask and a shift"""
246     return self._ilshl(other)
247
248 def __irshl__(self, other):
249     """Return the right shift of this array by the other array, with a mask and a shift"""
250     return self._irshl(other)
251
252 def __iandl__(self, other):
253     """Return the left shift of this array by the other array, with a mask and a shift"""
254     return self._iandl(other)
255
256 def __iorl__(self, other):
257     """Return the right shift of this array by the other array, with a mask and a shift"""
258     return self._iorl(other)
259
260 def __ixorl__(self, other):
261     """Return the bitwise XOR of this array and the other array, with a mask and a shift"""
262     return self._ixorl(other)
263
264 def __ilshl__(self, other):
265     """Return the left shift of this array by the other array, with a mask and a shift"""
266     return self._ilshl(other)
267
268 def __irshl__(self, other):
269     """Return the right shift of this array by the other array, with a mask and a shift"""
270     return self._irshl(other)
271
272 def __iandl__(self, other):
273     """Return the left shift of this array by the other array, with a mask and a shift"""
274     return self._iandl(other)
275
276 def __iorl__(self, other):
277     """Return the right shift of this array by the other array, with a mask and a shift"""
278     return self._iorl(other)
279
280 def __ixorl__(self, other):
281     """Return the bitwise XOR of this array and the other array, with a mask and a shift"""
282     return self._ixorl(other)
283
284 def __ilshl__(self, other):
285     """Return the left shift of this array by the other array, with a mask and a shift"""
286     return self._ilshl(other)
287
288 def __irshl__(self, other):
289     """Return the right shift of this array by the other array, with a mask and a shift"""
290     return self._irshl(other)
291
292 def __iandl__(self, other):
293     """Return the left shift of this array by the other array, with a mask and a shift"""
294     return self._iandl(other)
295
296 def __iorl__(self, other):
297     """Return the right shift of this array by the other array, with a mask and a shift"""
298     return self._iorl(other)
299
300 def __ixorl__(self, other):
301     """Return the bitwise XOR of this array and the other array, with a mask and a shift"""
302     return self._ixorl(other)
303
304 def __ilshl__(self, other):
305     """Return the left shift of this array by the other array, with a mask and a shift"""
306     return self._ilshl(other)
307
308 def __irshl__(self, other):
309     """Return the right shift of this array by the other array, with a mask and a shift"""
310     return self._irshl(other)
311
312 def __iandl__(self, other):
313     """Return the left shift of this array by the other array, with a mask and a shift"""
314     return self._iandl(other)
315
316 def __iorl__(self, other):
317     """Return the right shift of this array by the other array, with a mask and a shift"""
318     return self._iorl(other)
319
320 def __ixorl__(self, other):
321     """Return the bitwise XOR of this array and the other array, with a mask and a shift"""
322     return self._ixorl(other)
323
324 def __ilshl__(self, other):
325     """Return the left shift of this array by the other array, with a mask and a shift"""
326     return self._ilshl(other)
327
328 def __irshl__(self, other):
329     """Return the right shift of this array by the other array, with a mask and a shift"""
330     return self._irshl(other)
331
332 def __iandl__(self, other):
333     """Return the left shift of this array by the other array, with a mask and a shift"""
334     return self._i
```

Roll being added to constants list

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);
} // 3328-332 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

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)

100%

Task Screenshots

Gallery Style: 2 Columns

4

2

1



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

Caption(s) (required) ✓

Caption Hint: Describe/highlight what's being shown

Sub-Task

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)

100%

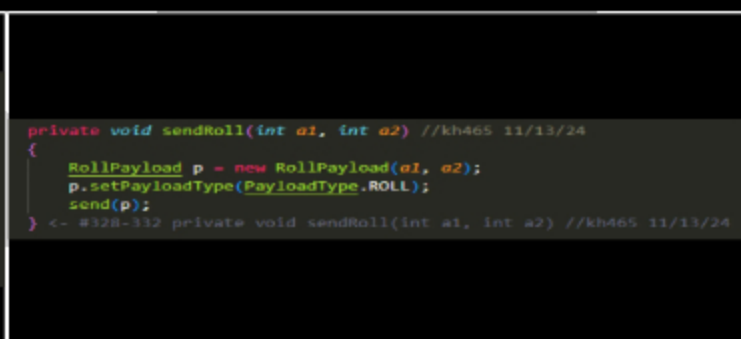
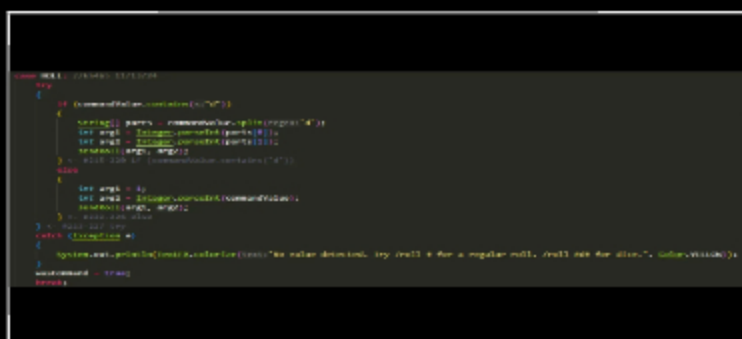
Task Screenshots

Gallery Style: 2 Columns

4

2

1



case ROLL added to processClientCommand with logic

New method sendRoll handling the sending of the

case ROLE added to processorenterpointmain, with logic

New method sensoron handling the sending of the command

```
54 private final String ROLL = "roll";
55 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

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

```

1 # Import the random module and set a seed value
2 import random
3 random.seed(42)
4
5 # Create a list of 10 random integers between 1 and 100
6 random_list = [random.randint(1, 100) for _ in range(10)]
7
8 # Print the random list
9 print(random_list)
10
11 # Create a dictionary with 5 random key-value pairs
12 random_dict = {}
13 for _ in range(5):
14     key = chr(random.randint(65, 90))
15     value = random.randint(1, 100)
16     random_dict[key] = value
17
18 # Print the random dictionary
19 print(random_dict)
20
21 # Create a tuple with 3 random floats between 0.1 and 0.9
22 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
23
24 # Print the random tuple
25 print(random_tuple)
26
27 # Create a set of 5 random integers between 1 and 100
28 random_set = set([random.randint(1, 100) for _ in range(5)])
29
30 # Print the random set
31 print(random_set)
32
33 # Create a list of 10 random strings of length 5
34 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
35
36 # Print the random strings
37 print(random_strings)
38
39 # Create a dictionary with 5 random key-value pairs
40 random_dict = {}
41 for _ in range(5):
42     key = chr(random.randint(65, 90))
43     value = random.randint(1, 100)
44     random_dict[key] = value
45
46 # Print the random dictionary
47 print(random_dict)
48
49 # Create a tuple with 3 random floats between 0.1 and 0.9
50 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
51
52 # Print the random tuple
53 print(random_tuple)
54
55 # Create a set of 5 random integers between 1 and 100
56 random_set = set([random.randint(1, 100) for _ in range(5)])
57
58 # Print the random set
59 print(random_set)
60
61 # Create a list of 10 random strings of length 5
62 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
63
64 # Print the random strings
65 print(random_strings)
66
67 # Create a dictionary with 5 random key-value pairs
68 random_dict = {}
69 for _ in range(5):
70     key = chr(random.randint(65, 90))
71     value = random.randint(1, 100)
72     random_dict[key] = value
73
74 # Print the random dictionary
75 print(random_dict)
76
77 # Create a tuple with 3 random floats between 0.1 and 0.9
78 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
79
80 # Print the random tuple
81 print(random_tuple)
82
83 # Create a set of 5 random integers between 1 and 100
84 random_set = set([random.randint(1, 100) for _ in range(5)])
85
86 # Print the random set
87 print(random_set)
88
89 # Create a list of 10 random strings of length 5
90 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
91
92 # Print the random strings
93 print(random_strings)
94
95 # Create a dictionary with 5 random key-value pairs
96 random_dict = {}
97 for _ in range(5):
98     key = chr(random.randint(65, 90))
99     value = random.randint(1, 100)
100    random_dict[key] = value
101
102 # Print the random dictionary
103 print(random_dict)
104
105 # Create a tuple with 3 random floats between 0.1 and 0.9
106 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
107
108 # Print the random tuple
109 print(random_tuple)
110
111 # Create a set of 5 random integers between 1 and 100
112 random_set = set([random.randint(1, 100) for _ in range(5)])
113
114 # Print the random set
115 print(random_set)
116
117 # Create a list of 10 random strings of length 5
118 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
119
120 # Print the random strings
121 print(random_strings)
122
123 # Create a dictionary with 5 random key-value pairs
124 random_dict = {}
125 for _ in range(5):
126     key = chr(random.randint(65, 90))
127     value = random.randint(1, 100)
128     random_dict[key] = value
129
130 # Print the random dictionary
131 print(random_dict)
132
133 # Create a tuple with 3 random floats between 0.1 and 0.9
134 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
135
136 # Print the random tuple
137 print(random_tuple)
138
139 # Create a set of 5 random integers between 1 and 100
140 random_set = set([random.randint(1, 100) for _ in range(5)])
141
142 # Print the random set
143 print(random_set)
144
145 # Create a list of 10 random strings of length 5
146 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
147
148 # Print the random strings
149 print(random_strings)
150
151 # Create a dictionary with 5 random key-value pairs
152 random_dict = {}
153 for _ in range(5):
154     key = chr(random.randint(65, 90))
155     value = random.randint(1, 100)
156     random_dict[key] = value
157
158 # Print the random dictionary
159 print(random_dict)
160
161 # Create a tuple with 3 random floats between 0.1 and 0.9
162 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
163
164 # Print the random tuple
165 print(random_tuple)
166
167 # Create a set of 5 random integers between 1 and 100
168 random_set = set([random.randint(1, 100) for _ in range(5)])
169
170 # Print the random set
171 print(random_set)
172
173 # Create a list of 10 random strings of length 5
174 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
175
176 # Print the random strings
177 print(random_strings)
178
179 # Create a dictionary with 5 random key-value pairs
180 random_dict = {}
181 for _ in range(5):
182     key = chr(random.randint(65, 90))
183     value = random.randint(1, 100)
184     random_dict[key] = value
185
186 # Print the random dictionary
187 print(random_dict)
188
189 # Create a tuple with 3 random floats between 0.1 and 0.9
190 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
191
192 # Print the random tuple
193 print(random_tuple)
194
195 # Create a set of 5 random integers between 1 and 100
196 random_set = set([random.randint(1, 100) for _ in range(5)])
197
198 # Print the random set
199 print(random_set)
200
201 # Create a list of 10 random strings of length 5
202 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
203
204 # Print the random strings
205 print(random_strings)
206
207 # Create a dictionary with 5 random key-value pairs
208 random_dict = {}
209 for _ in range(5):
210     key = chr(random.randint(65, 90))
211     value = random.randint(1, 100)
212     random_dict[key] = value
213
214 # Print the random dictionary
215 print(random_dict)
216
217 # Create a tuple with 3 random floats between 0.1 and 0.9
218 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
219
220 # Print the random tuple
221 print(random_tuple)
222
223 # Create a set of 5 random integers between 1 and 100
224 random_set = set([random.randint(1, 100) for _ in range(5)])
225
226 # Print the random set
227 print(random_set)
228
229 # Create a list of 10 random strings of length 5
230 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
231
232 # Print the random strings
233 print(random_strings)
234
235 # Create a dictionary with 5 random key-value pairs
236 random_dict = {}
237 for _ in range(5):
238     key = chr(random.randint(65, 90))
239     value = random.randint(1, 100)
240     random_dict[key] = value
241
242 # Print the random dictionary
243 print(random_dict)
244
245 # Create a tuple with 3 random floats between 0.1 and 0.9
246 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
247
248 # Print the random tuple
249 print(random_tuple)
250
251 # Create a set of 5 random integers between 1 and 100
252 random_set = set([random.randint(1, 100) for _ in range(5)])
253
254 # Print the random set
255 print(random_set)
256
257 # Create a list of 10 random strings of length 5
258 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
259
260 # Print the random strings
261 print(random_strings)
262
263 # Create a dictionary with 5 random key-value pairs
264 random_dict = {}
265 for _ in range(5):
266     key = chr(random.randint(65, 90))
267     value = random.randint(1, 100)
268     random_dict[key] = value
269
270 # Print the random dictionary
271 print(random_dict)
272
273 # Create a tuple with 3 random floats between 0.1 and 0.9
274 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
275
276 # Print the random tuple
277 print(random_tuple)
278
279 # Create a set of 5 random integers between 1 and 100
280 random_set = set([random.randint(1, 100) for _ in range(5)])
281
282 # Print the random set
283 print(random_set)
284
285 # Create a list of 10 random strings of length 5
286 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
287
288 # Print the random strings
289 print(random_strings)
290
291 # Create a dictionary with 5 random key-value pairs
292 random_dict = {}
293 for _ in range(5):
294     key = chr(random.randint(65, 90))
295     value = random.randint(1, 100)
296     random_dict[key] = value
297
298 # Print the random dictionary
299 print(random_dict)
300
301 # Create a tuple with 3 random floats between 0.1 and 0.9
302 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
303
304 # Print the random tuple
305 print(random_tuple)
306
307 # Create a set of 5 random integers between 1 and 100
308 random_set = set([random.randint(1, 100) for _ in range(5)])
309
310 # Print the random set
311 print(random_set)
312
313 # Create a list of 10 random strings of length 5
314 random_strings = [''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for _ in range(5)) for _ in range(10)]
315
316 # Print the random strings
317 print(random_strings)
318
319 # Create a dictionary with 5 random key-value pairs
320 random_dict = {}
321 for _ in range(5):
322     key = chr(random.randint(65, 90))
323     value = random.randint(1, 100)
324     random_dict[key] = value
325
326 # Print the random dictionary
327 print(random_dict)
328
329 # Create a tuple with 3 random floats between 0.1 and 0.9
330 random_tuple = (random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1, random.random() * 0.8 + 0.1)
331
332 # Print the random tuple
333 print(random_tuple)
334
335 # Create a set of 5 random integers between 1 and 100
336 random_set = set([random.randint(1, 100) for _ in range(5)])
337
338 # Print the random set
339
```

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

Caption(s) (required) ✓

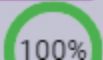
Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: Client Commands

Task #1: Roll Command

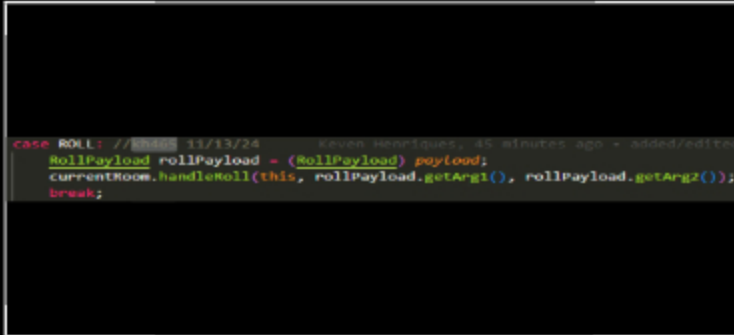
1. *Journal of the American Medical Association*, 1997; 277: 1033-1036.



Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

Group: Client Commands

Task #1: Roll Command

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

100%

Task Screenshots

Gallery Style: 2 Columns

4 2 1



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

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)

100%

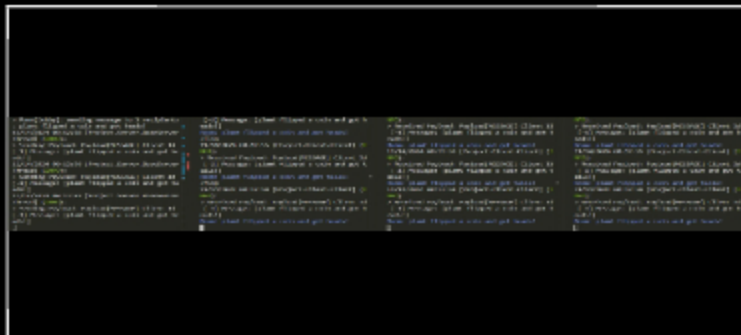
Task Screenshots

Gallery Style: 2 Columns

4

2

1



/flip being used. From left to right: server, plant (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

Group: Text Formatting

Tasks: 1

Points: 3

100%

^ COLLAPSE ^

Task

Group: Text Formatting

Task #1: Text Formatting

Weight: ~100%

Points: ~3.00

100%

^ 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

Sub-Task

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)

100%

Task Screenshots

Gallery Style: 2 Columns

4

2

1



class TextMarkup.java which handles all text markup logic. Object TextMarkup in Room

Room imports this and can invoke it by creating a TextMarkup object.



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 #), 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

Group: Text Formatting

Task #1: Text Formatting

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

100%

Task Screenshots

Gallery Style: 2 Columns

4

2

1



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

Combo of bold, italic, underline and red. Left to right: server, plant (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 ^

i 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



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



Group: Misc

Task #3: WakaTime Screenshot

Weight: ~33%

Points: ~0.33

^ COLLAPSE ^

i Details:

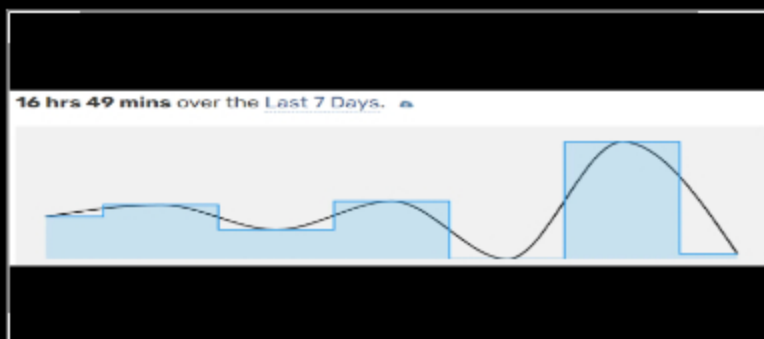
Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved



🖼 Task Screenshots

Gallery Style: 2 Columns

4 2 1



WakaTime from 11/8 to 11/14

End of Task 3

End of Group: Misc
Task Status: 3/3

End of Assignment