

Submission Worksheet

Submission Data

Course: IT114-003-F2025

Assignment: IT114 Module 3 User Input Challenges

Student: Laura L. (lsl8)

Status: Submitted | **Worksheet Progress:** 100%

Potential Grade: 10.00/10.00 (100.00%)

Received Grade: 0.00/10.00 (0.00%)

Started: 10/13/2025 6:45:44 PM

Updated: 10/13/2025 8:13:56 PM

Grading Link: <https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-module-3-user-input-challenges/grading/lsl8>

View Link: <https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-module-3-user-input-challenges/view/lsl8>

Instructions

- Overview Link: <https://youtu.be/iowHMCKuj5o>
- 1. Ensure you read all instructions and objectives before starting.
- 2. Create a new branch from main called M3-Homework
 - 1. `git checkout main` (ensure proper starting branch)
 - 2. `git pull origin main` (ensure history is up to date)
 - 3. `git checkout -b M3-Homework` (create and switch to branch)
- 3. Copy the template code from here: [GitHub Repository - M3 Homework](#)
 - It includes CommandLineCalculator, SlashCommandHandler, MadLibsGenerator, a BaseClass and a stories folder with 5 stories (used for MadLibsGenerator). Put all into an M3 folder or similar (adjust package reference at the top if you chose a different folder name).
 - Immediately record to history
 - `git add .`
 - `git commit -m "adding M3 HW baseline files"`
 - `git push origin M3-Homework`
 - Create a Pull Request from M3-Homework to main and keep it open
- 4. Fill out the below worksheet
 - Each Problem requires the following as you work
 - Ensure there's a comment with your UCID, date, and brief summary of how the problem was solved
 - Update the `ucid` variable
 - Code solution (add/commit periodically as needed)
- 5. Once finished, click "Submit and Export"
- 6. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
 - 1. `git add .`
 - 2. `git commit -m "adding PDF"`
 - 3. `git push origin M3-Homework`
 - 4. On Github merge the pull request from M3-Homework to main

7. Upload the same PDF to Canvas
8. Sync Local
 1. git checkout main
 2. git pull origin main

Section #1: (3 pts.) Challenge 1 - Command Line Calculator (Add/sub)

Progress: 100%

≡ Task #1 (3 pts.) - Edit the `main` method to solve the requirements

Progress: 100%

Details:

- Don't adjust the give code unless noted
- Challenge 1: Accept two numbers and an operator as command-line arguments (+ and -)
- Challenge 2: Allow integer and floating-point numbers
 - Ensure correct decimal places in output based on input (e.g., $0.1 + 0.2 \rightarrow 1$ decimal place)
- Display an error for invalid inputs or unsupported operators
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:

Two screenshots are expected

1. Snippet of relevant code showing solution (with ucid/date comment)
2. Full output of executing the program (Capture 5 variations of tests)

```
24 //System.out.println("Calculating result...");
25 //Task 1: Accept two numbers and an operator as command-line arguments
26 //Extract the equation (format is usually: number1 operator number2)
27
28 String number1 = args[0];
29 String operator = args[1];
30 String number2 = args[2];
31
32 //Check if operator is addition or subtraction
33 if (operator.equals("+") || operator.equals("-")) {
34     System.out.println("Error: Unsupported operator. Use + or - only.");
35     return;
36 }
37
38 //Check the type of each number and choose appropriate parsing
39 double num1 = Double.parseDouble(number1);
40 double num2 = Double.parseDouble(number2);
41
42 //Generate the equation result (important: ensure decimal's display as the
43 //lowest decimal count)
44 //e.g., 0.1 + 0.2 would show as one decimal place (0.3), 0.33 + 0.2 would show
45 //as two (0.53), etc.
46 int decimal1 = getDecimalPlaces(number1);
47 int decimal2 = getDecimalPlaces(number2);
48 int maxDecimals = Math.max(decimal1, decimal2);
49
50 double result;
51 if (operator.equals("+")) {
52     result = num1 + num2;
53 } else {
54     result = num1 - num2;
55 }
```

Code

```
1 //Task 1: Accept two numbers and an operator as command-line arguments
2 //Extract the equation (format is usually: number1 operator number2)
3
4 //Challenge 1: Accept two numbers and an operator as command-line arguments (+ and -)
5 //Challenge 2: Allow integer and floating-point numbers
6 //Ensure correct decimal places in output based on input (e.g., 0.1 + 0.2 -> 1 decimal place)
7 //Display an error for invalid inputs or unsupported operators
8 //Add code to solve the problem (add/commit as needed)
9
10 //System.out.println("Calculating result...");
11 //Task 1: Accept two numbers and an operator as command-line arguments
12 //Extract the equation (format is usually: number1 operator number2)
13
14 String number1 = args[0];
15 String operator = args[1];
16 String number2 = args[2];
17
18 //Check if operator is addition or subtraction
19 if (operator.equals("+") || operator.equals("-")) {
20     System.out.println("Error: Unsupported operator. Use + or - only.");
21     return;
22 }
23
24 //Check the type of each number and choose appropriate parsing
25 double num1 = Double.parseDouble(number1);
26 double num2 = Double.parseDouble(number2);
27
28 //Generate the equation result (important: ensure decimal's display as the
29 //lowest decimal count)
30 //e.g., 0.1 + 0.2 would show as one decimal place (0.3), 0.33 + 0.2 would show
31 //as two (0.53), etc.
32 int decimal1 = getDecimalPlaces(number1);
33 int decimal2 = getDecimalPlaces(number2);
34 int maxDecimals = Math.max(decimal1, decimal2);
35
36 double result;
37 if (operator.equals("+")) {
38     result = num1 + num2;
39 } else {
40     result = num1 - num2;
41 }
```

Output

Code 2

➡ Part 2:

Details:

URL #1



URL

≡ Part 3:

Details:

Your Response:

 Saved: 10/13/2025 7:08:14 PM

Section #2: (3 pts.) Challenge 2 - Slash Command Handler

Progress: 100%

Task #1 (3 pts.) - Edit the `main` method to solve the requirements

Progress: 100%

Details:

- Don't adjust the give code unless noted
- Challenge 1: Accept user input as slash commands (Commands are case-insensitive)
 - `"/greet <name>"` → Prints "Hello, <name>!"
 - `"/roll <num>d<sides>"` → Roll <num> dice with <sides> and returns a
 - `"/echo <message>"` → Prints the message back
 - `"/quit"` → Exits the program
- Challenge 2: Print an error for unrecognized commands
- Challenge 3: Print errors for invalid command formats (when applicable)
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:

Two screenshots are expected

1. Snippet of relevant code showing solution (with uid/date comment)
2. Full output of executing the program (Capture 3 variations of each command except `"/quit"`)

```
24 public class SlashCommandHandler extends MainClass {
25     public static void main(String[] args) {
26         Scanner scanner = new Scanner(System.in);
27         Random rand = new Random();
28         //uid 10/13/21
29         while (true) {
30             System.out.print("Enter command: ");
31             String input = scanner.nextLine().trim();
32
33             // greet
34             if (input.equalsIgnoreCase("/greet")) {
35                 System.out.println("Error: use /greet <name>");
36             } else if (input.toLowerCase().startsWith("/greet ")) {
37                 String name = input.substring(10).trim();
38                 if (name.isEmpty()) {
39                     System.out.println("Error: use /greet <name>");
40                 } else {
41                     System.out.println("Hello, " + name + "!");
42                 }
43             }
44
45             // check if roll
46             // parse roll
47             // handle invalid format
48             if (input.toLowerCase().startsWith("/roll ")) {
49                 System.out.println("Error: use /roll <num>d<sides>");
50             } else if (input.toLowerCase().startsWith("/roll ")) {
51                 String roll = input.substring(10).trim();
52                 int num = 1;
53                 int sides = 20;
54                 if (roll.contains("d")) {
55                     String[] parts = roll.split("d");
56                     num = Integer.parseInt(parts[0].trim());
57                     sides = Integer.parseInt(parts[1].trim());
58                 }
59                 if (num < 1 || sides < 2) {
60                     System.out.println("Error: num must be >= 1 and sides >= 2");
61                 } else {
62                     int total = 0;
63                     for (int i = 0; i < num; i++) {
64                         total += rand.nextInt(sides) + 1;
65                     }
66                     System.out.println("Rolled " + num + "d" + sides + " = " + total + "!");
67                 }
68             }
69             // echo
70             if (input.toLowerCase().startsWith("/echo ")) {
71                 String message = input.substring(10).trim();
72                 System.out.println(message);
73             }
74             // quit
75             if (input.toLowerCase().equals("/quit")) {
76                 System.out.println("Exiting...");
77                 break;
78             }
79             // unrecognized command
80             if (input.toLowerCase().startsWith("/") || input.toLowerCase().startsWith("/")) {
81                 System.out.println("Error: unrecognized command");
82             }
83         }
84     }
85 }
```

Code 1

```
1 int num = Integer.parseInt(roll.substring(0, 0).trim());
2 int sides = Integer.parseInt(roll.substring(0, 0).trim());
3 if (num < 1 || sides < 2) {
4     System.out.println("Error: num must be >= 1 and sides >= 2");
5 } else {
6     int total = 0;
7     for (int i = 0; i < num; i++) {
8         total += rand.nextInt(sides) + 1;
9     }
10    System.out.println("Rolled " + num + "d" + sides + " = " + total + "!");
11 }
```

```

    echo $? echo
    /// process echo
    else if (input.equalsIgnoreCase("/echo")) {
        System.out.println("Error: Use /echo <message>");
    } else if (input.toLowerCase().startsWith("/echo ")) {
        String msg = input.substring(7).trim();
        if (msg.isEmpty()) System.out.println("Error: Use /echo <message>");
        else System.out.println(msg);
    }

    // check if quit
    /// process quit
    else if (input.equalsIgnoreCase("/quit")) {
        System.out.println("Goodbye!");
        break;
    }
}

```


Code 2

```

laura@laura-laptop MINGW64 ~/lsl8-1111-000 (MJ-Homework)
$ java MS-SlashCommandHandler
Running Problem 2 for [1111] [2025-10-13T19:38:37.617676100]
Objective: Implement a simple slash command parser.
Enter command: /greet Laura
Hello, Laura!
Enter command: /greet
Error: Use /greet <name>
Enter command: /greet Bob
Hello, Bob!
Enter command: /roll
Error: Use /roll <num> or <id>
Enter command: /roll 4d6
Rolled 4d6 and got 24!
Enter command: /roll 7-5d4
Error: Numbers must be valid integers.
Enter command: /echo
Error: Use /echo <message>
Enter command: /echo Hello
Hello
Enter command: /echo How are you?
How are you?
Enter command: /quit
Goodbye!
Completed Problem 2 for [1111] [2025-10-13T19:37:11.395699500]

```

Output

 Saved: 10/13/2025 7:42:30 PM

Part 2:

Progress: 100%

Details:

Direct link to the file in the homework related branch from Github (should end in `.java`)


URL #1

<https://github.com/laurasofia544/lsl8-IT1111003M3-Homework/M3/SlashCommandHandler.java>



URL

<https://github.com/laurasofia544/>

 Saved: 10/13/2025 7:42:30 PM

Part 3:

Progress: 100%

Details:

Briefly explain `how` the code solves the challenges (note: this isn't the same as `what` the code does)

Your Response:

The code solves the challenge by letting the user type commands that start with a slash and then handling each one separately. It checks what command the user entered, makes sure it's written correctly, and gives clear feedback if something is missing or invalid. This solves the problem by processing the input so that the program can respond properly to different actions like greeting, rolling dice, echoing text, or quitting.



Section #3: (3 pts.) Challenge 3 - Mad Libs Generator

Progress: 100%

Task #1 (3 pts.) - Edit the `main` method to solve the challenges

Progress: 100%

Details:

- Don't adjust the give code unless noted
- Ensure you have the `stories` folder with the 5 stories
- Challenge 1: Load a random story from the "stories" folder
- Challenge 2: Extract **each line** into a collection (i.e., ArrayList)
- Challenge 3: Prompts user for each placeholder (i.e., `<adjective>`)
 - Any word the user types is acceptable, no need to verify if it matches the placeholder type
 - Any placeholder with underscores should display with spaces instead
- Challenge 4: Replace placeholders with user input (assign back to original slot in collection)
- Add code to solve the problem (add/commit as needed)

Part 1:

Progress: 100%

Details:

Two screenshots are expected

1. Snippet of relevant code showing solution (with uid/date comment)
2. Full output of executing the program (Capture the process for at least 2 stories)

```
// Start with
// Load a random story file
File f = Files.find(Path.of(".").toAbsolutePath().getParent(), 1,
    (p, attr) -> Files.isRegularFile(p));
File story = Files.find(f, Files.isRegularFile);

// parse the story lines
try (Scanner scanner = new Scanner(new FileReader(story))) {
    while (scanner.hasNextLine()) {
        lines.add(scanner.nextLine());
    }
} catch (Exception e) {
    System.out.println("Error reading story file.");
}

// Iterate through the lines
Pattern pattern = Pattern.compile("<[a-z]+>");
for (int i = 0; i < lines.size(); i++) {
    String line = lines.get(i);

    // prompt the user for each placeholder (note: there may be more than one)
    // placeholder: i = 0
    Matcher matcher = pattern.matcher(line);
    StringBuffer sb = new StringBuffer();
    while (matcher.find()) {
        String placeholder = matcher.group(1).replace("<", " ");
        System.out.print("Enter " + placeholder + ": ");
        String userWord = scanner.nextLine();

        // apply the update to the same collection slot
        matcher.appendReplacement(sb, userWord);
    }
    lines.set(i, sb.toString());
}
```

Code 1

```
// apply the update to the same collection slot
matcher.appendReplacement(sb, userWord);
}
<- #65-72 while (matcher.find())
```



```

        matcher.appendTail(sb);
        lines.set(i, sb.toString());
    } <- #25-75 for (int i = 0; i < lines.size(); i++)

    // End edita
    System.out.println("\nyour completed Mad Libs story:\n");
    StringBuilder finalStory = new StringBuilder();
    for (String line : lines) {
        finalStory.append(line).append("\n");
    }
    System.out.println(finalStory.toString());

    printFooter(uuid, 3);
    scanner.close();
} <- #26-87 public static void main(String[] args)
} <- #22-88 public class MadLibsGenerator extends BaseClass

```

Code 2

```

laura@Laura-Laptop MINGW64 ~/1s18-IT114-003 (M3-Homework)
$ java M3.MadLibsGenerator
Running Problem 3 for [1s18] [2025-10-13T19:56:46.046550000]
Objective: Implement a Mad Libs generator that replaces placeholders dynamically.
Enter adjective: pretty
Enter adjective: scary
Enter object: mirror
Enter adjective: smart
Enter verb ending in ing: running
Enter adjective: slow


Your Completed Mad Libs Story:

A pretty witch gave me a potion that would make me scary.
She told me to drink it while standing on a mirror under the smart moon.
As soon as I drank it, I started running uncontrollably.
From that day forward, I became the most slow person in town.

Completed Problem 3 for [1s18] [2025-10-13T19:57:34.003726000]

```

Output

 Saved: 10/13/2025 8:01:56 PM

Part 2:

Progress: 100%

Details:

Direct link to the file in the homework related branch from Github (should end in `.java`)


URL #1

<https://github.com/laurasofia544/1s18-IT114-003M3-Homework/M3/MadLibsGenerator.java>



URL

<https://github.com/laurasofia544/>

 Saved: 10/13/2025 8:01:56 PM

Part 3:

Progress: 100%

Details:

Briefly explain `how` the code solves the challenges (note: this isn't the same as `what` the code does)

Your Response:

The code solves the challenge by using a simple way to pick a random story, read it, and let the user fill in the blanks. It looks for each placeholder inside the story and replaces it with whatever the user types.



Saved: 10/13/2025 8:01:56 PM

Section #4: (1 pt.) Misc

Progress: 100%

≡ Task #1 (0.33 pts.) - Github Details

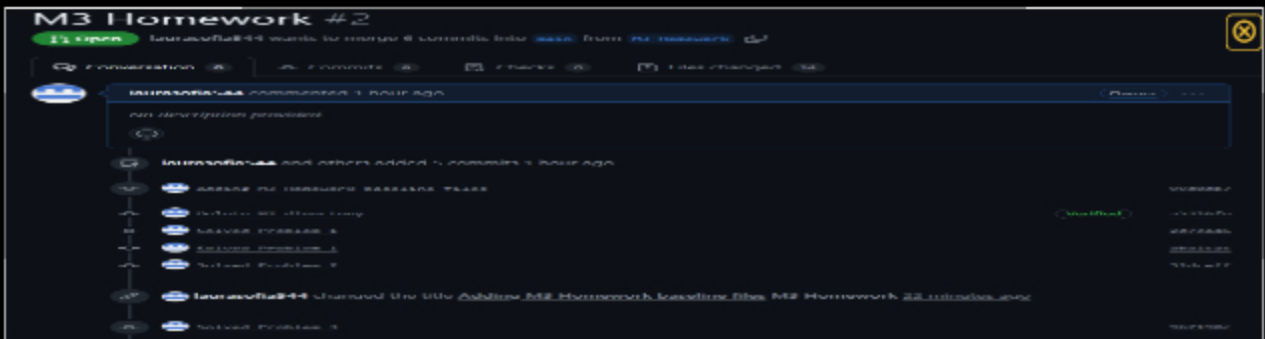
Progress: 100%

📁 Part 1:

Progress: 100%

Details:

From the Commits tab of the Pull Request screenshot the commit history Following minimum should be present



Git commits



Saved: 10/13/2025 8:04:30 PM

🔗 Part 2:

Progress: 100%

Details:

Include the link to the Pull Request (should end in `/pull/#`)

URL #1

<https://github.com/laurasofia544/IsI8-IT114@03/>



URL

<https://github.com/laurasofia544/>



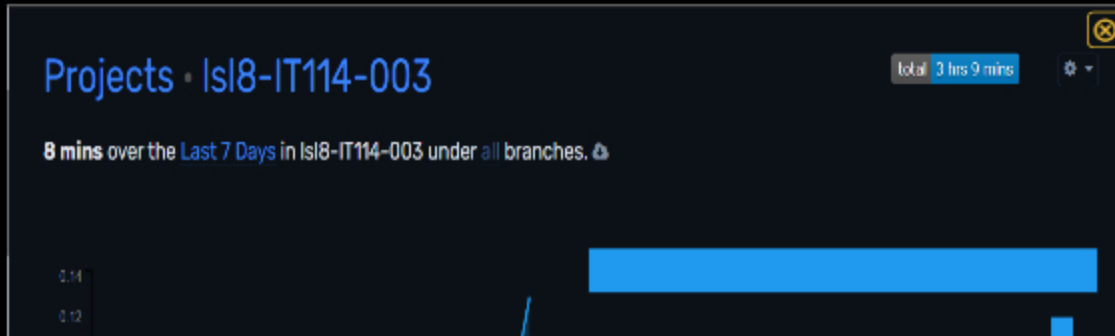
Saved: 10/13/2025 8:04:30 PM

📁 Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

Details:

- Visit the WakaTime.com Dashboard
- Click **Projects** and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary



I accidentally had my default profile open on VS code and I don't have wakatime installed on there.

Files		Branches	
8 mins	M3/MadLibsGenerator.java	8 mins	M3-Homework

That's why it says 8 minutes and only one file. I realized when I was almost done.



Saved: 10/13/2025 8:07:24 PM

≡ Task #3 (0.33 pts.) - Reflection

Progress: 100%

⇒ Task #1 (0.33 pts.) - What did you learn?

Progress: 100%

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

Through all the problems, I learned how to handle user input in different ways, from command-line arguments, to slash commands, to reading from files. I got better at using conditionals, loops, and basic error checking to make my programs

work more smoothly. I also learned how to organize code with methods and follow a structure using classes.



Saved: 10/13/2025 8:11:05 PM

⇒ Task #2 (0.33 pts.) - What was the easiest part of the assignment?

Progress: 100%

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

The easiest part of the assignment was importing the baseline files from github to vs code. As well as reading through the code to understand what was missing and what I needed to complete



Saved: 10/13/2025 8:12:55 PM

⇒ Task #3 (0.33 pts.) - What was the hardest part of the assignment?

Progress: 100%

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

The hardest part of the assignment was honestly the second problem with the roll dice and the last problem with the madlibs. It took me a while to figure out how to exactly start and what it was that I needed to do. Thankfully the comments helped me figure it out a little bit.



Saved: 10/13/2025 8:13:56 PM