

Submission Worksheet

Submission Data

Course: IT114-003-F2025

Assignment: IT114 Module 3 User Input Challenges

Student: Rayyan K. (rk975)

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 4:36:10 PM

Updated: 10/13/2025 7:29:48 PM

Grading Link: <https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-module-3-user-input-challenges/grading/rk975>

View Link: <https://learn.ethereallab.app/assignment/v3/IT114-003-F2025/it114-module-3-user-input-challenges/view/rk975>

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)

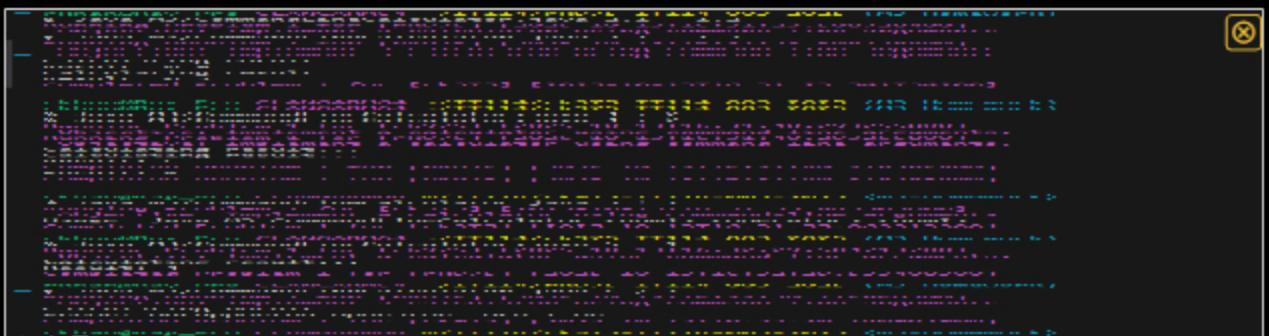
📁 Part 1:

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)



terminal output



```
1 // This program takes two numbers and an operator as input and calculates the result.
2 // It uses floating-point math to handle both integers and decimals.
3 // The result is formatted to match the precision of the input.
4 // Invalid inputs are caught and reported.
5
6 import java.util.Scanner;
7
8 public class Calculator {
9     public static void main(String[] args) {
10         Scanner scanner = new Scanner(System.in);
11         System.out.print("Enter two numbers and an operator: ");
12         String input = scanner.nextLine();
13         String[] tokens = input.split(" ");
14         double num1 = Double.parseDouble(tokens[0]);
15         double num2 = Double.parseDouble(tokens[1]);
16         String operator = tokens[2];
17         double result = 0;
18         if (operator.equals("+")) {
19             result = num1 + num2;
20         } else if (operator.equals("-")) {
21             result = num1 - num2;
22         } else if (operator.equals("*")) {
23             result = num1 * num2;
24         } else if (operator.equals("/")) {
25             result = num1 / num2;
26         } else {
27             System.out.println("Invalid operator");
28             return;
29         }
30         // Format the result to match the precision of the input
31         int decimalPlaces = 0;
32         if (num1 % 1 != 0 || num2 % 1 != 0) {
33             decimalPlaces = Math.max(num1.toString().split(".")[1].length(), num2.toString().split(".")[1].length());
34         }
35         result = String.format("%.f", result).replace(".", "");
36         System.out.println("Result: " + result);
37     }
38 }
```

code



Saved: 10/13/2025 4:37:14 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/rayk101/rk975-IT114/blob/10/13/2025/Homework/M3/CommandLineCalculator.java>



URL

<https://github.com/rayk101/rk975>



Saved: 10/13/2025 4:37:14 PM

Part 3:

Progress: 100%

Details:

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

Your Response:

The code breaks down the user's input into two numbers and an operator, then checks if the operator is valid. It calculates the result using floating-point math to handle both integers and decimals. To match the precision of the input, it analyzes how many decimal places were used and formats the output accordingly. This ensures the result looks clean and accurate, while also catching and reporting any invalid inputs.



Saved: 10/13/2025 4:37:14 PM

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

Progress: 100%

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

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/rayk101/rk975-IT114/blob/2025/Homework/M3/SlashCommandHandler.java>



URL

<https://github.com/rayk101/rk975>



Saved: 10/13/2025 6:14:18 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 listens for user input and checks if it matches one of the supported slash commands. It breaks the input into parts and uses string methods to figure out what the user wants to do. Each command has its own logic block that handles the action and checks for errors like missing arguments or wrong formats. The loop keeps running until the user types /quit, and any unknown command gets a clear error message. This setup makes the program flexible, easy to extend, and responsive to different command styles.



Saved: 10/13/2025 6:14:18 PM

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 given 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 2: Extract each line into a collection (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 ucid/date comment)
2. Full output of executing the program (Capture the process for at least 2 stories)

```

import java.util.Scanner;
import java.util.ArrayList;
import java.util.Collections;

public class Story {
    private String title;
    private ArrayList<String> characters;
    private ArrayList<String> plot;

    public Story(String title, ArrayList<String> characters, ArrayList<String> plot) {
        this.title = title;
        this.characters = characters;
        this.plot = plot;
    }

    public void displayStory() {
        System.out.println("Title: " + title);
        System.out.println("Characters: " + characters);
        System.out.println("Plot: " + plot);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a title:");
        String title = scanner.nextLine();

        ArrayList<String> characters = new ArrayList<>();
        System.out.println("Enter characters (separated by commas):");
        String input = scanner.nextLine();
        String[] charArray = input.split(",");
        for (String char : charArray) {
            characters.add(char.trim());
        }

        ArrayList<String> plot = new ArrayList<>();
        System.out.println("Enter plot (separated by commas):");
        String input2 = scanner.nextLine();
        String[] plotArray = input2.split(",");
        for (String plotLine : plotArray) {
            plot.add(plotLine.trim());
        }

        Story story = new Story(title, characters, plot);
        story.displayStory();
    }
}

```

code

```

$ java Main
Enter a title:
The Great Adventure
Enter characters (separated by commas):
Alice, Bob, Charlie
Enter plot (separated by commas):
Alice and Bob go on a journey to find a treasure. Charlie joins them. They face many challenges and eventually find the treasure.
Title: The Great Adventure
Characters: Alice, Bob, Charlie
Plot: Alice and Bob go on a journey to find a treasure. Charlie joins them. They face many challenges and eventually find the treasure.

```

terminal

Saved: 10/13/2025 7:24:00 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/rayk101/rk975-IT114-100-2025/Homework/M3/MadLibsGenerator.java>



URL

<https://github.com/rayk101/rk975>



Saved: 10/13/2025 7:24:00 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 dynamically selecting a random story file from a predefined directory using `File[]` and `Random`, ensuring variability across runs. It reads each line into an `ArrayList<String>`, enabling indexed access and in-place updates. Placeholders enclosed in angle brackets are identified using `indexOf` and replaced with user input collected via `Scanner`, with underscores converted to spaces for readability. The modified lines are reassigned back into the same list slot to preserve structure. Finally, the story is reconstructed and printed using a `StringBuilder`, completing the transformation from template to personalized output.



Saved: 10/13/2025 7:24:00 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

The screenshot shows a GitHub Pull Request interface for a repository named 'M3 homework #7'. The 'Commits' tab is selected, displaying a list of 10 commits. Each commit entry includes a commit hash, a description, and a status indicator (green circle for successful, red circle for failed). The commits are ordered chronologically from top to bottom.

Commit Hash	Description	Status
7f1b1b1	Initial commit	Success
6d0b1b1	Added initial code	Success
5c0b1b1	Added initial code	Success
4b0b1b1	Added initial code	Success
3a0b1b1	Added initial code	Success
290b1b1	Added initial code	Success
180b1b1	Added initial code	Success
070b1b1	Added initial code	Success
f60b1b1	Added initial code	Success
e50b1b1	Added initial code	Success

pr commit

another if statement with a for loop inside

abstract the part of the if statement inside

fixed syntax errors

Added missing line

Fixed syntax errors

state and word

Steps to achieve

Added comment and state of for loop

Finalized the my assessment

did the catch statements

Fixed syntax errors

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

100%

second commit pr tab

Saved: 10/13/2025 7:26:33 PM

Part 2:

Progress: 100%

Details:

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

URL #1

<https://github.com/rayk101/rk975-IT114-0018-2025/commits>



URL

<https://github.com/rayk101/rk975>

Saved: 10/13/2025 7:26:33 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

SlashCommandHandler.java
1 to 4 mins

MadLibsGenerator.java
1 to 2 mins

wakatime1



wakatime2



Saved: 10/13/2025 7:28:02 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:

I learned how to use Java tools like File, Scanner, and ArrayList to read and manipulate text files. I also practiced using loops and string methods like indexOf, substring, and replace to detect and update placeholders in a story. It showed me how to combine user input with file content to create dynamic output. Most importantly, I learned how to structure code that interacts with both the file system and the user in real time



Saved: 10/13/2025 7:29:48 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 was printing the final story using a loop and `System.out.println`. Once the placeholders were replaced, joining the lines together was straightforward. Using `ArrayList` made it simple to store and update each line. The logic for displaying the final result didn't require any complex conditions or extra tools.



Saved: 10/13/2025 7:29:46 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 was managing the placeholder replacement logic inside each line. It was tricky to find the exact position of `<` and `>` and make sure the string updates didn't break the formatting. I also ran into errors when variables weren't declared in the right scope or method names were misspelled, which made debugging frustrating. Understanding how to use `File[]` and `Random` together to pick a story file took some trial and error too.



Saved: 10/13/2025 7:29:44 PM