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:

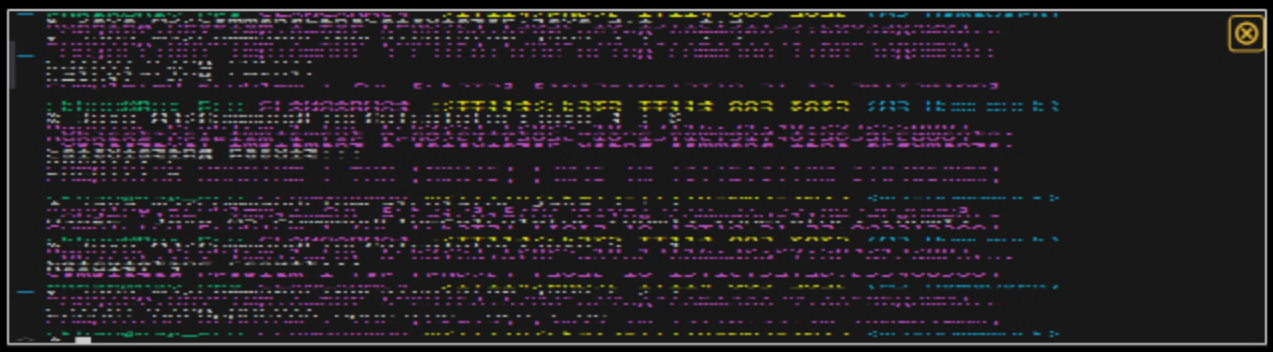
- Don't adjust the give code unless noted
- Challenge 1: Accept two numbers and an operator as command-line arguments (+ and -)
- Challenge 2: Allow integer and floating-point numbers
    - Ensure correct decimal places in output based on input (e.g., 0.1 + 0.2 → 1 decimal place)
- Display an error for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

Part 1:

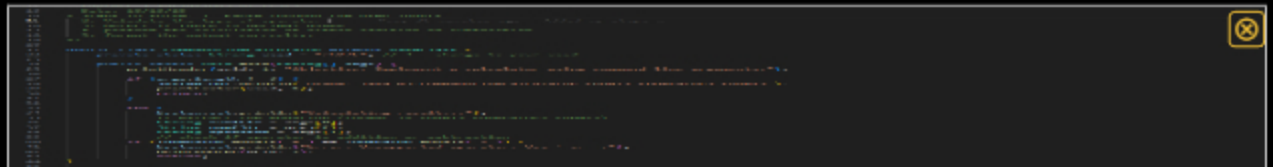
Progress: 100%

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



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-IT114-003-2025/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

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)

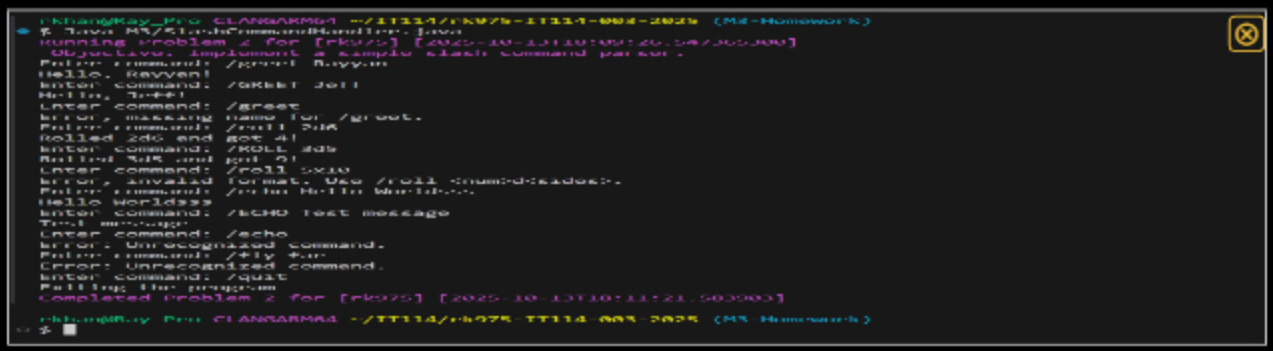
Part 1:

Progress: 100%

Details:
Two screenshots are expected

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

code



terminal output

Saved: 10/13/2025 6:14:18 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-IT114-003-2025/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 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 ucid/date comment)
2. Full output of executing the program (Capture the process for at least 2 stories)

code

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-IT114-003-2025/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%

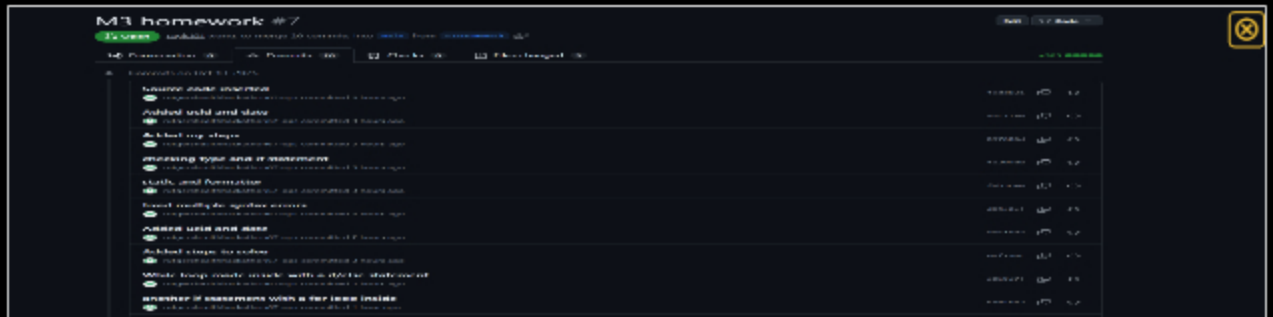
Task #1 (0.33 pts.) - Github Details

Progress: 100%

Part 1:

Progress: 100%

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



pr commit

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-IT114-003-2025/commits

URL
https://github.com/rayk101/rk975

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

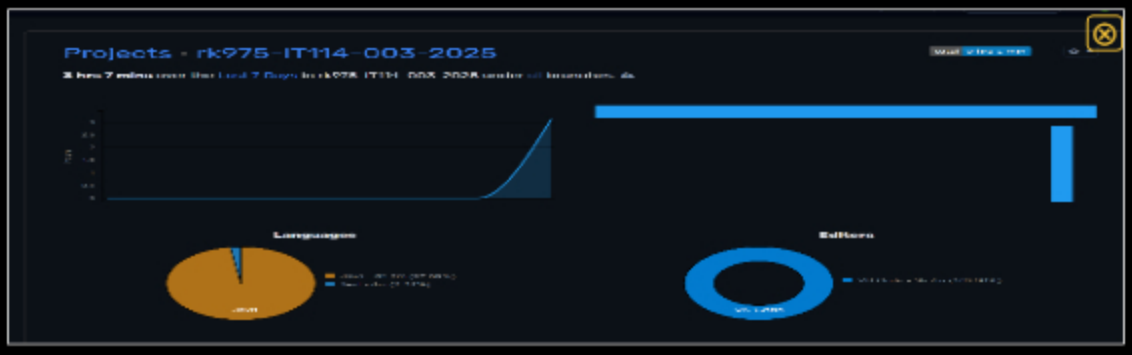
Task #2 (0.33 pts.) - WakaTime - Activity

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

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