Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-004-S2024/it114-m2-java-problems/grade/mbh3

IT114-004-S2024 - [IT114] M2 Java Problems

Submissions:

Submission Selection

1 Submission [active] 2/7/2024 10:43:32 PM

Instructions

COLLAPSE

Guide:

1 .Make sure you're in the main branch locally and `git pull origin main` any pending changes
2 .Make a new branch per the recommended branch name below (git checkout -b ...)
3 .Grab the template code from https://gist.github.com/MattToegel/fdd2b37fa79a06ace9dd259ac82728b6
4 .Create individual Java files for each problem and save the files inside a subfolder of your choice
    1 .The should end with the file extension in lowercase .java
5 .Move the unedited template files to github
    1 .`git add .`
    2 .`git commit -m "adding template files`
    3 .`git push origin <homework branch>` (see below and don't include the < >)
    4 .Create and open a pull request from the homework branch to main (leave it open until later steps)
6 .Note: As you work, it's recommended to add/commit at least after each solution is done (i.e., 3+ times in this case)
    1 .Make sure the files are saved before doing this
7 .Fill in the items in the worksheet below (save as often as necessary)
8 .Once finished, export the worksheet
9 .Add the output file to any location of your choice in your repository folder (i.e., a Module2 folder)
10Check that git sees it via `git status`
11If everything is good, continue to submit
    1 .Track the file(s) via `git add`
    2 .Commit the changes via `git commit` (don't forget the commit message)
    3 .Push the changes to GitHub via `git push` (don't forget to refer to the proper branch)
    4 .Create a pull request from the homework related branch to main (i.e., main <- "homework branch")
    5 .Open and complete the merge of the pull request (it should turn purple)
    6 .Locally checkout main and pull the latest changes (to prepare for future work)
12Take the same output file and upload it to Canvas
    1 .*This step is new since GitHub renders the PDF as an image the links aren't clickable so this method works better
    2 .*Remember, the github process of these files are encouragement for your tracking of your progress

Branch name: M2-Java-Problems

Tasks: 8 Points: 10.00

● **Problem 1** (3 pts.)

COLLAPSE

● COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the Problem 1 Solved Code and Output

Details:
Only make edits where the template code mentions.

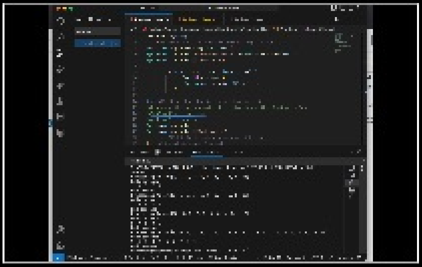
Solution should ensure that any passed in array will have only the odd values output.
Requires at least 2 screenshots (code + output from terminal)

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Edits were done only in the processArray() method and original template code/comments remain untouched
#2	1	Only arr is used (no direct usage of a1, a2, a3, a4)
#3	5	Only odd values output (not odd indexes/keys)
#4	1	Includes code comments with student's ucid and date
#5	1	Terminal output is fully visible

Task Screenshots:

Large Gallery



Checklist Items (5)

#1 Edits were done only in the processArray() method and original template code/comments remain untouched

#2 Only arr is used (no direct usage of a1, a2, a3, a4)

#3 Only odd values output (not odd indexes/keys)

#4 Includes code comments with student's ucid and date

#5 Terminal output is fully visible

Problem 1 output and code

●
^ COLLAPSE ^

Task #2 - Points: 1

Text: Explain your solution

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Clearly explains how the code/logic solves the problem (mentions how the odd values are determined)

Response:

My approach was pretty straightforward. I looped through each array, checking every number to see [redacted], which is like asking, "Does this number leave any leftovers when divided by 2?" If the answer was yes (meaning there was a remainder of 1), then I knew I had an odd number on my hands. This simple yet effective method helped me filter out and display all the odd numbers from the arrays.

●
^ COLLAPSE ^

Problem 2 (3 pts.)

●
^ COLLAPSE ^

Task #1 - Points: 1

Text: Screenshot of the Problem 2 Solved Code and Output

Details:
Only make edits where the template code mentions.

Solution should ensure that any passed in array will have its values summed AND the final result converted to two decimal places (i.e., 0.10, 1.00, 1.01).
Requires at least 2 screenshots (code + output from terminal)

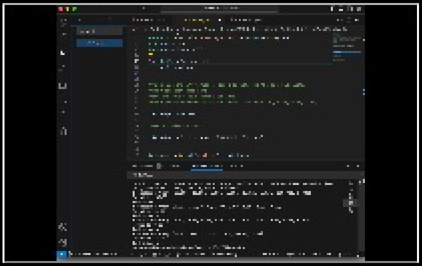
Checklist *The checkboxes are for your own tracking

#	Points	Details
■ #1	1	Edits were done only in the getTotal() method and original template code/comments remain untouched (unless noted)

#2	1	Only arr is used (no direct usage of a1, a2, a3, a4)
#3	5	Passed in array's values get summed AND rounded to two decimal places like currency (i.e., 0.00, 0.10, 1.10)
#4	1	Includes code comments with student's ucid and date
#5	1	Terminal output is fully visible

Task Screenshots:

Large Gallery



Checklist Items (5)

#1 Edits were done only in the getTotal() method and original template code/comments remain untouched (unless noted)

#2 Only arr is used (no direct usage of a1, a2, a3, a4)

#3 Passed in array's values get summed AND rounded to two decimal places like currency (i.e., 0.00, 0.10, 1.10)

#4 Includes code comments with student's ucid and date

#5 Terminal output is fully visible

Output of Problem2 and code.

COLLAPSE

Task #2 - Points: 1

Text: Explain your solution

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Clearly explains how the code/logic solves the problem (mentions both how the values get summed and how the rounding is solved correctly)

Response:

So first, I looped through each array, adding every number together to get the total sum. The real magic happened when I needed to round these totals accurately. I turned to (String.format ("%.2f, a clean trick that formats the sum into a string with exactly two digits after the decimal point. This step was crucial because it ensured that every total I reported was precise and easy to read. It was a whole number or a fractional part. This method proved to be a straightforward and effective way to handle both the summation and the rounding-off parts of the problem.

● ^ COLLAPSE ^

Problem 3 (3 pts.)

● ^ COLLAPSE ^

Task #1 - Points: 1

Text: Screenshot of the Problem 2 Solved Code and Output

Details:
Only make edits where the template code mentions.

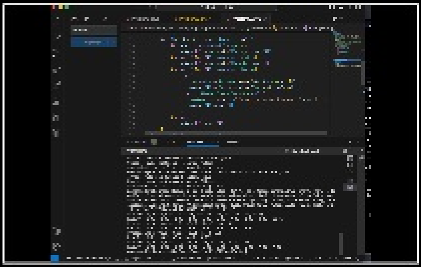
Solution should ensure that any passed in array will have its values converted to a positive version of the value AND converted back to the original data type.
Requires at least 2 screenshots (code + output from terminal)

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Edits were done only in the bePositive() method and original template code/comments remain untouched
☐ #2	1	Only arr is used (no direct usage of a1, a2, a3, a4)
☐ #3	5	Passed in array's values will get converted to a positive version AND converted back to the original data type
☐ #4	1	Includes code comments with student's ucid and date
☐ #5	1	Terminal output is fully visible

Task Screenshots:

Large Gallery



Checklist Items (5)

#1 Edits were done only in the bePositive() method and original template code/comments remain untouched

#2 Only arr is used (no direct usage of

a1, a2, a3, a4)

#3 Passed in array's values will get converted to a positive version AND converted back to the original data type

#4 Includes code comments with student's ucid and date

#5 Terminal output is fully visible

Problem 3 - code and output

● ^ COLLAPSE ^

Task #2 - Points: 1

Text: Explain your solution

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Clearly explains how the code/logic solves the problem (mentions both the conversion to positive and conversion to original data type)

Response:

My challenge was to convert every value in arrays of different types to their positive counterparts while keeping their original data types intact. My strategy involved iterating through each array, regardless of whether it contained integers, doubles, or strings. For numerical values, whether they [redacted] function to flip any negatives into positives. The trickier part came with strings representing numbers. Here, I first tried to parse them into doubles, apply [redacted] them, and then convert them back to strings. My method was all about ensuring that no matter the data type, the output stayed true to its original form—just positively so. This approach allowed me to maintain the integrity of the data while effectively meeting the goal of the problem.

● ^ COLLAPSE ^ Reflection (1 pt.)

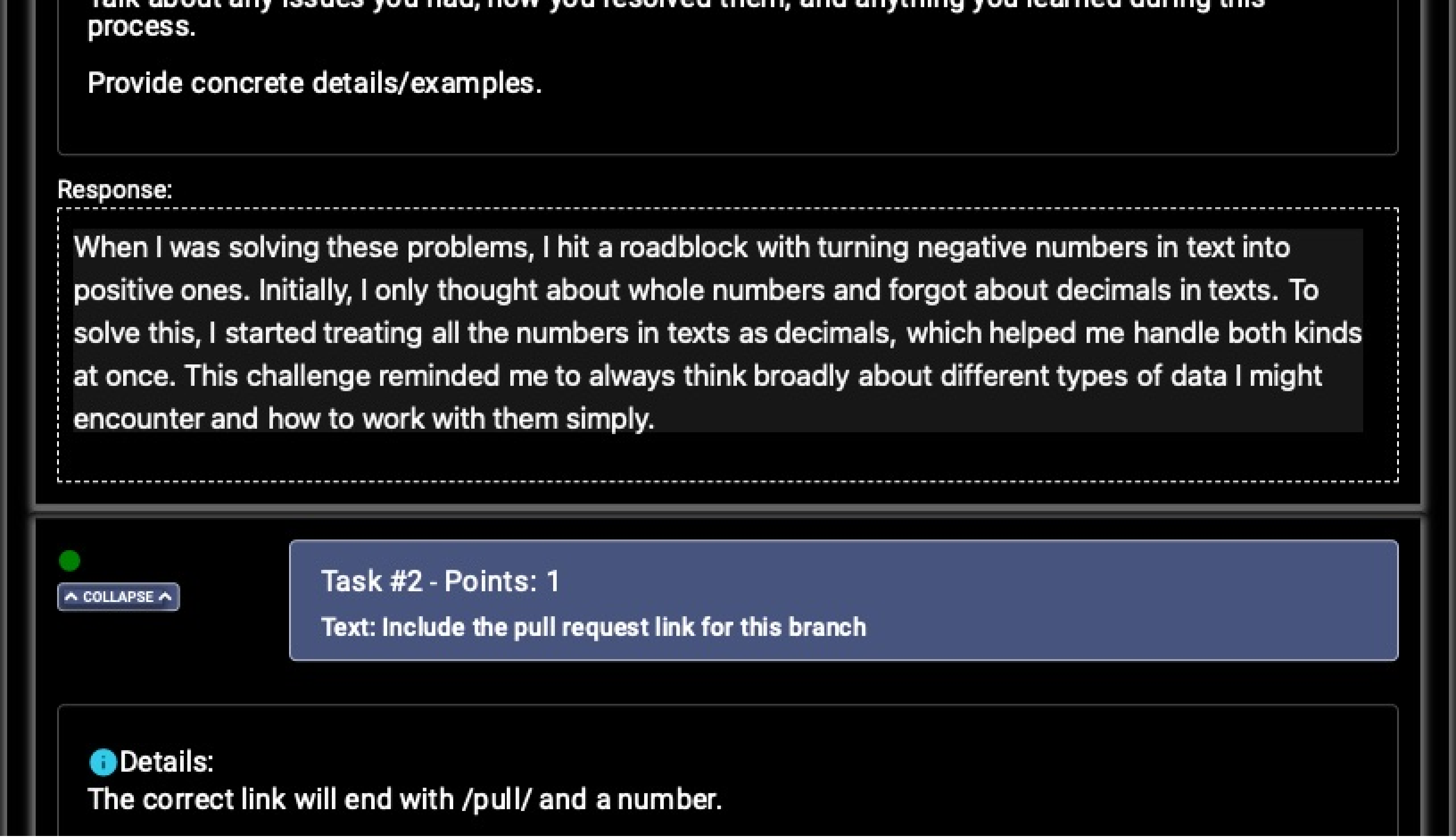
● ^ COLLAPSE ^

Task #1 - Points: 1

Text: Reflect on your experience

Details:

Talk about any issues you had, how you resolved them, and anything you learned during this



Talk about any issues you had, how you resolved them, and anything you learned during this process.

Provide concrete details/examples.

Response:

When I was solving these problems, I hit a roadblock with turning negative numbers in text into positive ones. Initially, I only thought about whole numbers and forgot about decimals in texts. To solve this, I started treating all the numbers in texts as decimals, which helped me handle both kinds at once. This challenge reminded me to always think broadly about different types of data I might encounter and how to work with them simply.

Task #2 - Points: 1

Text: Include the pull request link for this branch

Details:
The correct link will end with /pull/ and a number.