Submission Worksheet

CLICK TO GRADE

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

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

Submissions:

Submission Selection

1 Submission [active] 2/5/2024 6:30:27 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

COLLAPSE

Problem 1 (3 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the Problem 1 Solved Code and Output

Details:
Only make edits where the template code mentions.

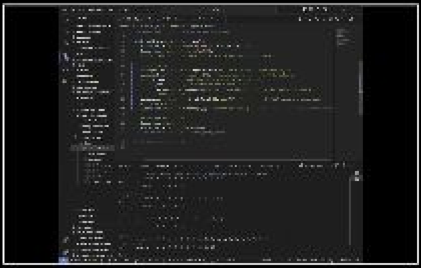
Solution should ensure that any passed in array will have only the odd values output.
Requires at least 2 screenshots (code + output from terminal)

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 (0)

problem 1

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:

Logically, this code executes a single for loop through the array that is passed to the method and uses an if statement to check the parity of each value (using modulus) -- and ultimately prints a string of those odd values.

Problem 2 (3 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of the Problem 2 Solved Code and Output

Details:
Only make edits where the template code mentions.

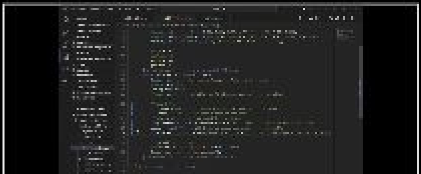
Solution should ensure that any passed in array will have its values summed AND the final result converted to two decimal places (i.e., 0.10, 1.00, 1.01).
Requires at least 2 screenshots (code + output from terminal)

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 (0)

problem 2

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:

The method loops through the array that is passed and adds each of those values to a double. Then, that total is multiplied by a hundred and rounded to a long (more accurate than truncation, avoids floating point instability). This long is then divided by 100 to scale it back to the correct units and then formatted to 2 decimal places in a string (truncated).

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)

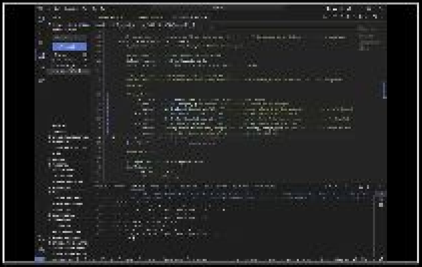
Checklist *The checkboxes are for your own tracking

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

#	Points	Details
#4	1	Includes code comments with student's ucid and date
☐ #5	1	Terminal output is fully visible

Task Screenshots:

Large Gallery



Checklist Items (0)

problem 3

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:

Logically this program is relatively trivial -- the program loops once through the area, checks each value in the area against different types, and then uses takes the absolute value of each value. Using 'instanceof", the program identifies the original datatype, casts it to that original datatype (can't use methods that belong to a object through a general object reference regardless of actual object's type), and then takes the absolute value. For Strings, the object is cast to String, parsed to an integer, has its absolute value taken, and it once again cast back to a String. The outputs for each of these absolute values are ultimately added to the "output" array.

Reflection (1 pt.)

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 process.

Provide concrete details/examples.

Response:

All the problems were relatively trivial, didn't particularly have any challenges, considered making some of them run in parallel for the fun of it but decided not to modify the code templates too much --> also, did not submit this for problem one, but could also solve it using the following:

```
        String processedArray = Arrays.stream(arr)
            .filter(num -> num % 2 != 0)
            .mapToObj(Integer::toString)
            .collect(Collectors.joining(", ", "Processed Array: [", "]"));

        System.out.println(processedArray);
```

Would have to import collectors, but is relatively simple -- although arguably less human readable, and I am unsure how the two solutions compare in time complexity (I am not familiar enough with Java's implementations)

COLLAPSE

Task #2 - Points: 1

Text: Include the pull request link for this branch

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