Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-006-S2024/it114-m2-java-problems/grade/owe

IT114-006-S2024 - [IT114] M2 Java Problems

Submissions:

Submission Selection

1 Submission [active] 2/5/2024 11:11:09 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)
10. Check that git sees it via `git status`
11. If 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)
12. Take 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

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)

Checklist Items (0)

ss of code

ss of code

Checklist Items (0)

finished 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:

The Java code that is supplied defines a class called Problem1, and its processArray method is intended to recognize and output odd numbers from an integer array. The processArray method is called for each of the four arrays once the main method initializes them with unique integer sequences. The original array is printed inside the processArray method, and then odd numbers are found and printed. The code iterates through the array elements using a simple loop, determining whether each number is odd (via num % 2!= 0). A number is printed clearly and concisely if it is odd. By indicating the process's conclusion, the method comes to an end. Identifying and managing odd values in arrays is made clear and easy with this method.

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

#3	5	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)

ss code 1

Checklist Items (0)

ss code 2

Checklist Items (0)

code 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 how the values get summed and how the rounding is solved correctly)

Response:

The total sum of the elements in each double array is determined by the Java code in the Problem2 class, which rounds the result to two decimal places. Four arrays are initialized with different double values in the main method, and the getTotal method is called for each array. The original array is printed inside the getTotal method, and then a for-each loop iterates through the array, adding the sum of its elements to the total variable. After that, the total is rounded with the Math.round method to two decimal places. Together with a notification that the processing is finished, the outcome is printed. This code offers a short and understandable way to calculate the total sum and round it for every double array.

Problem 3 (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 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 (0)

ss code 1

Checklist Items (0)

ss code 2

Checklist Items (0)

code 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:

The Java code in the Problem3 class defines a method named bePositive that processes an array of generic type T and converts each value to its positive form, creating a new array to store the results. The main method initializes four arrays with different data types and calls the bePositive method for each. Inside the bePositive method, the original array is printed, and a new array (output) is created to store the positive values. The code handles different data types by iterating through the input array's elements using a loop. Math is used to convert numbers to their absolute values. Before conversion, abs, strings are parsed to numbers, and other types are kept unchanged. After that, the result is printed with the positive values displayed. as well as the corresponding data types in a string format. For values of various data types, this code offers a versatile and universal method of converting them to their positive forms.

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 process.

Provide concrete details/examples.

Response:

Well, rounding numbers to two decimal places presented a problem for me when I was working on a project with a similar task. I tried a different method at first, but the results were not what I had anticipated. I had to modify my rounding strategy after doing some debugging. Rather than rounding the total amount directly, I multiplied the total by 100, rounded it, and then divided it by 100 again. The problem was fixed as a result, and the outcomes were precise. Though it was a minor detail, it helped me understand the nuances of Java rounding and the significance of accuracy in numerical computations.

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.

URL #1

https://github.com/WayguBeef5/OWE-114-006/pull/2

End of Assignment