Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT114-002-S2024/it114-m2-java-problems/grade/ob75

IT114-002-S2024 - [IT114] M2 Java Problems

Submissions:

Submission Selection

1 Submission [active] 2/6/2024 10:08:42 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

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)

This contains my code. My code is below the comment with my UCID and date.

Terminal demonstrates output.

COLLAPSE

Task #2 - Points: 1

COLLAPSE

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:

I created a for loop that goes through the index's and increments by one checking for odd numbers and looks to see if that elements in the array is not able to be % by 2 into 0 and therefore prints the output of the element which would be an odd number. Even numbers would be if it was equal to 0 so I used a != to show not.

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 (0)

Checklist Items (0)

Code is demonstrated under UCID and date.

Terminal output shows each number being added and then gives the total sum.

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:

I created a for loop which increments each time for the array index once the array is added onto the total. I used += operand that allows the array element to be added on to the the total each time it goes up and repeats for each of the arrays. Then I printed the total by changing it from a double to string and rounding it by using Math.round. The 2 decimal places was done by multiplying the total by 100 and then dividing that by 100.0. I did the same for totalOutput since I was getting more decimal places there as well so I rounded that too.

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

#2		
#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)

Code is under UCID and date.

Checklist Items (0)

Continuation of first screenshot to show entire code.

Checklist Items (0)

Terminal showing the array and result of the array being turned into positive values.

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:

I created 3 if-statement loops inside of a for-statement loop which for each index of the array, it will check the data type and if it matches, it will do the absolute value of that index and loop back to do it for each element incrementing by one in the array. I created an if statement for each of the datatypes, and they would all execute Math.abs to their respective data type. String was the only one I had to convert to Integer, take the absolute value, and then convert it back to String so it doesn't lose its data type. Output[i] created a new list and would contain the new positive values of the old array.

Reflection (1 pt.)

COLLAPSE

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:

One issue I had was figuring out how to created a loop for problem 3 that would check a datatype. Since the Integer, Double, and String were not primitive types, I couldn't just use Math.abs on them directly so I had to figure out what data type generic T was and did that by making an if loop using instanceof which would check the data type of the array and do the method inside. This is what took me the longest to understand but after learning about instanceof, I was able to implement it into the code.

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.