Array Utility

In this activity, we will be working on solving a specific problem using your knowledge of nested loops, arrays, and if-else statements.

Code Provided

You have been provided with a Java file:

ArrayUtil_Test.java

- The main function of this file is to test the class and methods you will create for this assignment. It includes a main method with several tests and a helper method to format the display of the result of the test. You do not need to modify anything in this file. Download and copy it into your project folder in VSCode. There will be errors at first but they'll go away as you complete the project.

Part A – Develop the algorithms (9 points)

The challenge in the project is developing algorithms that accomplishes basic operations over an array of integer numbers.

The three operations are:

- Find the minimum (smallest) value that appears in the array
- Find the maximum (largest) value that appears in the array
- Determine the number of unique values that appear in the array

Develop an algorithm for each of those tasks. The algorithms should work for any size of array. If the array is empty (i.e., doesn't include any elements/values) the algorithm should output a zero.

Keep in mind that like any computational problem there are several approaches to achieve a solution. For this programming checkpoint your will need to utilize loops and conditional statements.

Write the pseudocode or flowchart for the algorithm and store it in a file (file format can be text, pdf or doc) with the name *arrayUtilSteps*.

Part B - Implement the class (21 points)

The goal for this part is to implement a class with the algorithms developed in Part A.

Create a class with the following requirements:

1. Class name is ArrayUtil
2. Has an integer array field named intArray
3. A default constructor
4. A constructor that accepts one argument for the array field and assigns that argument to the appropriate field. You may assume that only valid values will be used to test the constructor (i.e., you don't have to worry about input validation).
5. An accessor for the field.
6. A mutator for the field.
7. A method called minValue that does not accept any arguments and finds the minimum (smallest) value in the integer array and returns that value.
8. A method called maxValue that does not accept any arguments and finds the maximum (largest) value in the integer array and returns that value.
9. A method called countUniqueIntegers that does not accept any arguments and determines the number of unique values that appear in the array and returns that value. (i.e., count of the number of unique groups of integer values in an array). For example, given this array of integers {1,2,2,4,3,5,4,3,8,10} we have 7 unique groups of values 1, 2, 3, 4 ,5, 8, 10.

NOTE: Remember to follow the Java naming conventions and the rules for encapsulation.

NOTE 2: You must use simple arrays and conditionals and not any helper classes or packages.

Keep in mind that the algorithms for each of the methods should work for any size of array. If the array is empty (i.e., doesn't include any elements/values) they should return zero.

You can test your code by running the ArrayUtil_Test program. The main method includes creating an object of your ArrayUtil class and several calls (tests) to each of the operations methods with varying arrays (length and content). If your logic is correct your output should look like this:

```
*****************************
     Array Utility Project
*****************************
Test 0 - min value: The result 0  match expected: 0 --->> OK
Test 0 - max value: The result 0  match expected: 0 --->> OK
Test 0 - count unique: The result 0  match expected: 0 --->> OK
----------------------
Test 1 - min value: The result -3  match expected: -3 --->> OK
Test 1 - max value: The result 4  match expected: 4 --->> OK
Test 1 - count unique: The result 5  match expected: 5 --->> OK
----------------------
Test 2 - min value: The result -5  match expected: -5 --->> OK
Test 2 - max value: The result 4  match expected: 4 --->> OK
Test 2 - count unique: The result 7  match expected: 7 --->> OK
----------------------
Test 3 - min value: The result -3  match expected: -3 --->> OK
Test 3 - max value: The result 2  match expected: 2 --->> OK
Test 3 - count unique: The result 5  match expected: 5 --->> OK
----------------------
Test 4 - min value: The result -3  match expected: -3 --->> OK
Test 4 - max value: The result 9  match expected: 9 --->> OK
Test 4 - count unique: The result 9  match expected: 9 --->> OK
----------------------
Test 5 - min value: The result 1  match expected: 1 --->> OK
Test 5 - max value: The result 10  match expected: 10 --->> OK
Test 5 - count unique: The result 10  match expected: 10 --->> OK
----------------------
Test 6 - min value: The result -3  match expected: -3 --->> OK
Test 6 - max value: The result 2  match expected: 2 --->> OK
Test 6 - count unique: The result 6  match expected: 6 --->> OK
----------------------
Test 7 - min value: The result -4  match expected: -4 --->> OK
Test 7 - max value: The result 6  match expected: 6 --->> OK
Test 7 - count unique: The result 5  match expected: 5 --->> OK
----------------------
Test 8 - min value: The result -1  match expected: -1 --->> OK
Test 8 - max value: The result 0  match expected: 0 --->> OK
Test 8 - count unique: The result 2  match expected: 2 --->> OK
----------------------
Test 9 - min value: The result -5  match expected: -5 --->> OK
Test 9 - max value: The result 40  match expected: 40 --->> OK
Test 9 - count unique: The result 9  match expected: 9 --->> OK
----------------------
Test 10 - min value: The result -5  match expected: -5 --->> OK
Test 10 - max value: The result 5  match expected: 5 --->> OK
Test 10 - count unique: The result 7  match expected: 7 --->> OK
----------------------
```

If a specific call fails that means that the returned value did not match the correct (expected) value. Looking at the example that failed will help you understand what went wrong in order to fix this.

For example, this indicates the method for finding the minimum value did not pass the first test case (Test 0) we have.

```
*********************************
Test 0 - min value: The result 0 does not match expected: 1 --->> Failed
```

Remember that the test cases are just samples used to check if your code works. You should expect that your code will be tested using other input values.

Coding Style – (3 points)

- As we mentioned in class, formatting your code makes a difference when someone else has to look at it and review it. It is important to practice good formatting habits. Here are some guidelines to follow:

    - Appropriate variable names. In general, do not use single letters like X or create strange abbreviations. Make your variables names descriptive (e.g., hrlyWage, rent, ...). Always use camelCase!
    - Proper indentation.
    - Good commenting explains what code is doing.
    - The solution is well-organized, possibly even elegant, and "reads" like a story that another programmer can follow

Possible Bonus Opportunities (2 points each)

- Add a method that will return an array that will only contain the unique values of the original array (i.e., remove duplicates)
- Add a method that given two element positions (indices) in the array swaps the two elements with each other.
- Add a method that determines whether the array is sorted or not. This method should allow for specifying whether the sorting to be determined is in descending or ascending order.

In the submission documentation include a section describing any bonuses implemented. While it is not required you can update ArrayUtil_Test with cases to test any additional methods you've implemented. You may include the pseudocode as well.

Submission Requirements (2 points)

1. Submit the pseudocode/flowchart file (named ***arrayUtilSteps***) you developed in Part A.
2. Submit the code file for part B (**ArrayUtil**).
3. Submit a PDF document with the name **project4Info.pdf** that includes the following assignment information:
    1. List of references used to acquire information, if any.
    2. Explanation of any special or additional features you added, if any.
    3. Explanation of status, stopping point, and issues if incomplete.
    4. Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?
    5. Did you learn anything that will help you perform better on future assignments?
    6. If you completed any bonuses, include the documentation described in the Bonus section. If you updated the test program include that as well.

PreviousNext