ARRAY OPERATIONS using Java

Note: Throughout this document, you will find questions that appear in bold italics. You must address these questions in a separate file that will be the report you must submit along with the working set of programs required for this assignment. These reports will be an important portion of the grade for these assignments as they show your understanding of the programs you submit and the underlying rationale for the strategy you have adopted to solve these programming problems. Devise your answers carefully and in an organized manner. The working programs without justifying report to prove your understanding of your solution will not be granted a passing grade.

Here is the description of this two-part assignment:

PART I:
Instantiate two integer arrays of size of 50: *ListA* and *ListB*. Generate a random number in the range 1-49, use this as the number of elements that you will store in each of these arrays. For example, if the random number is 18, that means that you need to randomly generate 18 numbers in the range 0-100 and store them in *ListA*. You repeat this process for *ListB*. Notice that you MUST make sure that there are NO DUPLICATE values in either of these arrays. However, the two array lists can have items in common.
Please note that in the process of populating these arrays you SHOULD NOT implement a strategy to prevent these arrays from having common elements. We would like these arrays to be "realistically" populated. We would just want to avoid duplicate values in a single array. Read the requirements for this part of the assignment in full before getting started on it. You would want to know ALL the requirements to take into consideration in your strategy.

a. Find the common elements in these arrays and remove them from both arrays. You must attempt to come up with an efficient way to implement this, therefore you are asked to describe your algorithm in your report:
i. Describe your strategy in finding common elements across the two arrays. Mention the number of iterations along with both arrays as a result of your strategy and any other "cost" that you believe your strategy may incur.
b. Display the common elements of these arrays IN ORDER.

ii. Does this requirement play an important role in your strategy in finding and eliminating the common elements? Explain your answer.
c. Combine/merge these two arrays into a single sorted array called TheList. A side note: your original strategy is going to play an important role in the efficiency of this part.

iii. Explain your steps in accomplishing this part. Don’t forget to discuss your view of the cost of your strategy.
An example for Part I would be:

ListA: | 13 7 42 2 9 11 38 21 >>> 42 2 9 21 |

ListB: | 7 11 40 91 82 8 13 38 72 12 6 >>> 40 91 82 8 72 6 |

TheList: | 2 6 8 9 21 40 42 72 82 91|

The required output of this part of the assignment would be:

• Display original ListA, and *ListB*.
• Display the common elements in order

• Display both arrays: ListA, and ListB after deleting the common elements

• Display the merged sorted array TheList

Part II.
This is an int-array game on the **"Part I’s" ListA and ListB**.
Here are the steps of this game:

1. Choose the shorter array as the key. In case of equal length, pick ListA as the key.

2. Use every element of the key array as an index into the longer array and sum the values you find in those indices. So for example, if the first element in the key array is 5, look for the value in index 5 of the second array and keep the running total.

A very important point is to avoid array index out of bound exception, if the content of the key array translates in an out of bound index in the second array, ignore that item and move to the next element in the key array, and continue the process.

The sum becomes the target value we are looking for.

A very simple example would be:

ListA: |8 12 4 90|

ListB: |10 6 32 87 50 61 77 39 45|

We pick ListA as the key, we iterate through its elements starting with 8, check to see whether 8 is in the valid index range for ListB, if it is, look for ListB[8] which is 45. Next is 12, this index will not work for ListB, so skip it. Go to 4, ListB[4] is 50, we add this to 45. Next is 90, another invalid index, and skipping it takes us to the end of the key array. So 45 + 50 = 95 is our target sum. We would also like to keep the count of elements that we skipped.

The output for this part would be:

- Display both arrays: ListA, and ListB in their current state (common elements removed).
- Display the target sum.
- Display the count of the elements that you needed to skip in order to avoid array index out of bounds exception.

Note: Do not use Java in-built classes. If you use any built-in Java Classes for any portion of this program, your submission will be considered invalid.