Deduplication- Removing Duplicate Elements

This worksheet continues your practice with manipulating data within arrays. If there are library methods you might find useful to carry out an exercise, they'll be listed, along with a link to the API for that method. Note that there are library methods that do a lot of the manipulation we're looking at, but the point of this course is to learn how to write them for yourself!

Removing Duplicates from Unsorted Arrays Using Array Growth

1) Removing duplicates from a numeric array using array growth

Write a **static method** (in NumberUtilities.java) that:

- Takes in a numeric array (nums)
- Iterates through that array and removes all duplicates from the array (each time an element is found that *has not been seen before*, the new array should increase in size by one slot, and the new element should be added to the end)
- Returns the new array

Write a **new program (NumberDuplicateRemoval.java)** that:

- Asks the user how large they'd like their array and creates an array of that size
- Takes numbers in from the user, fills the array and displays it to the user
- Removes all duplicates from the array (using the array growth approach) & displays the new version
- Displays how many duplicates were removed from the original array

2) Removing duplicates from a String array using array growth

Note: The split() method from the String class will be useful here

Syntax: textToBeSplit.split(" "); – This will create an array of words by breaking up the line of text to be split based on where spaces occur.

Write a **static method** (in StringUtilities.java) that:

- Takes in an array of Strings (text)
- Iterates through that array and removes all duplicate elements using the array growth approach
- Returns the new array

Write a new program (StringDuplicateRemoval.java) that:

- Takes a line of text in from the user
- Splits the single line of text up into words (the split method will be useful here)
- Removes all duplicate words from the array using the method you wrote in the previous part of the exercise
- Displays the contents of the array as a single line of text (i.e. at the end you should see your original sentence without duplicates)

Removing Duplicates from Unsorted Arrays Using Blank Deletion

3) Removing duplicates from a numeric array using "blank deletion"

Write a static method (in NumberUtilities.java) that:

- Takes in a numeric array (nums)
- Creates a new array to hold all elements from the original
- Iterates through that array to remove all duplicates from the array
- Removes all blanks from the new array
- Returns the new array (after it has had its blanks removed)

Add the following logic to your NumberDuplicateRemoval.java program:

- Asks the user how large they'd like their next array to be and creates another array of that size
- Takes numbers in from the user, fills the array and displays it to the user
- Removes all duplicates from the array (using the blank deletion approach) & displays the new version
- Displays how many duplicates were removed from the original array

4) Removing duplicates from an array of Person objects using "blank deletion"

Note: For this exercise, you will need a Person class. It should include the following components:

- firstName
- lastName
- age

Two Person objects are considered equal if they have the same firstName, lastName and age

Write a **static method** (in PersonUtilities.java) to count the number of blank spaces in an array of Persons:

- Take in an array of Person objects
- Return the count of all blank spaces within the array (what does a blank in an array of objects look like?)

Write a **static method** (in PersonUtilities.java) to resize an array by removing all blank spaces within the array:

- Take in an array of Person objects
- Count the number of blank spaces in the array (You should use the method you created in the previous part to do this)
- Create a new array to fit only the valid information from the original array
- Copy all valid elements from the original array to the new array
- Return the new array

Write a **static method** (in PersonUtilities.java) to:

- Take in an array of Person objects
- Create a new deduplicated version of the array using blank deletion
- Return the new array

Write a **new program (PersonDuplicateRemoval.java)** that:

- Asks the user how many Persons they'd like to create and creates an array of that size to store them
- Creates that many Person objects based on user input and stores them in the array. For each Person, you will need to:
  - Ask them to enter the first name, last name and age
  - Make a Person out of those pieces
  - Put that Person into the array
- Displays the array of Person objects (in a nicely formatted manner)
- Eliminates all duplicate Persons (based solely on their first name information)
- Displays the new version of the array.

Removing Duplicates from Sorted Arrays

Useful method: sort() from the Arrays class.
Functionality: This method is the Arrays version of Collections.sort(), and will sort the provided array in *natural order*. Where the array provided is an array of Strings, it will be sorted in case-sensitive ascending order.
Usage: Arrays.sort() (note: This does not RETURN the array, it changes it directly)

5) Removing duplicates from a sorted numeric array

Write a **static method** (in NumberUtilities.java) that:

- Takes in a sorted numeric array (nums)
- Iterates through that array and removes all duplicates from the array – this should carry out the action in an optimised manner based on the sorted nature of the data
- Returns the new array

Add the following logic **to NumberDuplicateRemoval**.java that:

- Asks the user for the name of a file to read from
- Creates a numeric array from the information in this file
- Sorts the array (consider using Arrays.sort() to do this)
- Removes all duplicates from the array & displays the new version
- Displays how many duplicates were removed from the original array

6) Removing duplicates from a sorted Object array (an array of Persons)

Before attempting this exercise, you will need to implement the Comparable interface in your Person class. This requires you to:

1) Implement Comparable for Person type data: public class Person implements Comparable<Person>
2) Write the compareTo method: public int compareTo(Person other)
    a. The natural order for Person is by first name (ascending)
3) compareTo's logic works as follows:
    a. The int returned by compareTo reflects the order of the data
    b. If **`this`** object comes before other object, compareTo returns < 0
    c. If **`this`** object comes after other object, compareTo returns > 0
    d. If **`this`** object is equal to other object, compareTo returns 0

Write a **static method** (in PersonUtilities.java) that:

- Takes in a **sorted** array of Person objects
- Iterates through that array and removes all duplicate Person objects
- Returns the new array

Add the following logic **to** your **PersonDuplicateRemoval**.java that:

- Asks the user how many Persons they'd like to create and creates an array of that size to store them
- Creates that many Person objects based on user input and stores them in the array.
- Sorts the array in order of the Person's first name (Consider using Arrays.sort() to do this, it will automatically sort them in ascending order of first name because of the Person's compareTo method).
- Displays the array of Person objects (in a nicely formatted manner)
- Eliminates all duplicate Persons (based solely on their first name information)
- Displays the new version of the array.