
**GIFT School of Engineering and Applied Sciences**

**Fall 2020**

**CS-124: Introduction to Programming - Lab**

# Lab-13 Manual

**Methods and Arrays**

v1.2
18/11/2020

# Task #1: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

In a program, write a method that accepts two arguments: an array and a number *n*. Assume that the array contains integers. The method should display all of the numbers in the array that are greater than the number *n*.

You may use the following header for this method:

```
public static void largerThanNumber(int[] array, int number)
```

**NOTE: Write methods and the main() method in separate files.**

1. Create a program called **ArrayMethodsLab13.java** and add this method in this class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Create another file **RunArrayMethodsLab13.java** having the **main()** method. Call the method with appropriate arguments.
4. Display appropriate messages.

```
public static void largerThanNumber(int[] array, int number)
    {
        for(int i = 0; i < array.length; ++i)
        {
            if(array[i] > number)
                System.out.print(array[i] + "   ");
        }//for
        System.out.println();
    }//largerThanNumber
```

# Task #2: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Write a method that returns the index of the **second occurrence** of an integer value in an integer array**.** That is, the method searches for the given value for its second occurrence in an array. If that value does not occur more than once, the method returns an index of **-1.**

You may use the following header for this method:

```
public static int getSecondIndex(int[] array, int value)
```

**HINTS:**

- You will need to count the occurrence ofthe given value in the array. The method will not stop at finding the first occurrence of the value, and at that point, the count becomes **1**.
- You would continue the search, and when the count becomes equal to **2**, you have found your value.

**NOTE: Write methods and the main() method in separate files.**

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display appropriate messages based on the return value.

```
public static int getSecondIndex(int[] array, int value)
    {
          int length = array.length;
          int occurrence = 0;
          for(int i = 0; i < length ; ++i)
          {
               if(array[i] == value)
               {
                    occurrence++;
               }
               if(occurrence == 2){
                    return i;
               }
          }
          return -1;
    }//getSecondIndex
```

# Task #3: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Write a method that returns the index of the ***nth* occurrence** of an integer value in an integer array**.** That is, the method searches for the given value for its *nth* occurrence, where *n* could be any integer **>= 1**. If that value does not occur *n* times, the method returns an index of **-1.**

You may use the following header for this method:

```
public static int getNthIndex(int[] array, int occurrence,
int value)
```

For example, the method call:

```
int index = getNthIndex(array, 4, 100);
```

Searches for the **4th occurrence** in the array for the value **100.** If found, the index of the found value is returned, otherwise **-1** is returned.

**NOTE:**

- **Write methods and the main() method in separate files.**
- You should perform input validation on the **occurrence** argument. It should be greater-than or equal to **1.** If the validation fails, you should print an appropriate error message saying that the **occurrence is invalid!** and return **-1** as the index value.

**HINTS:**

- You will need to count the occurrence of the given value in the array. The method will not stop at finding the first occurrence of the value, and at that point, the count becomes **1**.
- You would continue the search, and when the count becomes equal to ***n***, you have found your value.

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display appropriate messages based on the return value.

```
public static int getNthOccurrence(int[] array, int occurrence,
int value)
     {
          int length = array.length;
          int count = 0;
          for(int i = 0; i < length; ++i)
          {
               if(array[i] == value)
                    count++;
               if(count == occurrence)
                    return i;
          }
          return -1;
     }//getNthOccurrenceIndex
```

# Task #4: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Write a method that returns the index of the **smallest** or the **largest** value in an integer array, based on the given character argument**.** The character argument could be either **'S'** for the smallest, or **'L'** for the largest value.

You may use the following header for this method:

```
public static int getSLIndex(int[] array, char typeOfValue)
```

For example, the method call:

```
int index = getSLIndex(array, 'L');
```

Returns the index of the **largest value** in the array.

**NOTE:**

- **Write methods and the main() method in separate files.**
- You should perform input validation on the typeOfValue argument. It should only one of 'L' or 'S'. If the validation fails, you should print an appropriate error message saying that the Type of Value is invalid! and return -1 as the index value.

**HINTS:**

- To find any one of both type of values, you would need to create a variable that holds the smallest value or the largest value and initialize that variable with the **first value** of the array, that is the value at **index 0 of the array**.
- You would then try to find the smaller or the larger value (as required) from the array, and replace the value stored in the variable with the new-found value.
- You would continue to do this until you reach the end of the array. At that point, the variable would have the smallest or the largest (as per the argument) value in the array.

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display appropriate messages based on the return value.

```
public static int getSLIndex(int[] array, char typeOfValue)
     {

          if(typeOfValue == 'S')
          {
               int size = array.length;
               int sIndex = 0;
               for(int i = 1; i < size; i++)
               {
                    if(array[i] < array[sIndex])
                    {
                         sIndex = i;
                    }
               }
               return sIndex;
          }
          else if(typeOfValue == 'L')
          {
               int size = array.length;
               int lIndex = 0;
               for(int i = 1; i < size; i++)
               {
                    if(array[i] > array[lIndex])
                    {
                         lIndex = i;
                    }
               }
               return lIndex;
          }
          else
          {
               return -1;
          }
```

```
}//getSLindex
```

## Task #5: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Write a method that prints the count of even and odd integers in an array.
You may use the following header for this method:

```
public static void countEvenOddIntegers(int[] array)
```

The method should print the count of both even and odd integers.

**NOTE:**

- **Write methods and the main() method in separate files.**

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display appropriate messages.

```
public static void countEvenOddIntegers(int[] array){

        int evenCount = 0;
        int size = array.length;

        for (int i = 0; i < size; ++i){
            if (array[i] % 2 == 0){
                ++evenCount;
            }//if
        }//for

        System.out.println("There are " + evenCount + " even
numbers and " + (size - evenCount) + " odd numbers in the
array.");
```

```
}//countEvenOddIntegers()
```

## Task #6: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Write a method that copies an array to another given array in reverse order. Make sure that the lengths of the two arrays are compatible before copying arrays.

You may use the following header for this method:

```
public static void copyReverseArrays(int[] source, int[] target)
```

For example, suppose that the source array has 10 values: **10 9 8 7 6 5 4 3 2 1**

When this method is called with a target array of length 10, the target array would contain values: **1 2 3 4 5 6 7 8 9 10**

**NOTE:**

- **Write methods and the main() method in separate files.**

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display appropriate messages.

```
public static void copyReverseArray(int[] source, int[] target)
     {
          if(source.length == target.length){
               int size = source.length;
               for(int i = 0; i < size; i++)
               {
                    target[size-1-i] = source[i];
               }
          }else{
               System.out.println("The size of source and target
array is not equal");
```

```
        }
    }
```

## Task #7: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Assume that the array is of **int** type. Write the following methods:

i. A method called **printElementsAtEvenIndex** that accepts an array of type **int** and prints every element at even index.
ii. A method called **printEvenElements** that accepts an array of type **int** and prints every element which is even.
iii. A method called **reversePrint** that accepts an array of type **int** and prints the array in reverse order.

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display appropriate messages.

```
public static void printElementsAtEvenIndex(int[] array)
    {
        int size = array.length;
        for(int i = 0; i < size; i = i + 2)
        {
            System.out.print(array[i] + "   ");
        }//for
        System.out.println();
    }//printElementsAtEvenIndex()

    public static void printEvenElements(int[] array)
    {
        int size = array.length;
```

```
        for(int i = 0; i < size; i = i + 2)
        {
            if(array[i] % 2 == 0)
                System.out.print(array[i] + "   ");
        }//for
        System.out.println();
    }//printElementsAtEvenIndex()

    public static void reversePrint(int[] array)
    {
        int size = array.length;
        for(int i = size - 1; i >= 0; --i)
        {
            System.out.print(array[i] + "   ");
        }//for
        System.out.println();
    }//reversePrint()
```

## Task #8: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

a) Write a method to sort the input integer array using the **sort** method of the **Arrays** class.
You may use the following header for this method:

```
public static void sortIArray(int[] array)
```

b) Write a method to sort the input String array using the **sort** method of the **Arrays** class.
You may use the following header for this method:

```
public static void sortSArray(String[] array)
```

c) Write a method to sort the input real numbers array using the **sort** method of the **Arrays** class.
You may use the following header for this method:

```
public static void sortDArray(double[] array)
```

**NOTE:**

- **Write methods and the main() method in separate files.**

**HINTS:**

- To use the class **Arrays,** you have to use the import statement as follows:

```
import java.util.Arrays;
```

- To sort an array, just use the **sort** method of the class **Arrays.** For example, suppose we have an array of integers called **array.** To sort this array, call the **Arrays.sort()** method as follows:

```
Arrays.sort(array);
```

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display array values before and after sorting.

## Task #9: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Write a method that accepts a **String** type array named words, a **char** type argument named character and prints all the strings from words array that have the given character at any location.

```
public static void findStringsByCharacter(String[] words,
char character)
```

**NOTE:**

- **Write methods and the main() method in separate files.**

**HINTS:**

- For each word in words array, check if the word has the character at any location.

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display array values before and after sorting.

```
public static void findStringsByCharacter(String[] array, char
character)
    {
        for(int i = 0; i < array.length; ++i)
        {
            String word = array[i];
            int strLength = word.length();
            for(int j = 0; j < strLength; j++)
            {
                char c = word.charAt(j);
                if(c == character){
                    System.out.println(word);
```

```
                    break;
                }//if
            }//inner-for
        }//outer-for
    }//findStringsByCharacter()
```

## Task #10: Using Arrays and Methods

In this task, you are being asked to write methods that manipulate arrays in Java.

Write a method that prints all the valid names from a **String** type array named **names**. A name is valid if:

i. It contains letters only.
ii. It contains at least 3 letters and at most 32 letters.
iii. The first letter must be an upper case letter

```
public static void printValidNames(String[] names)
```

**NOTE:**

- **Write methods and the main() method in separate files.**

1. Add this method to the **ArrayMethodsLab13.java** class. Do not create another class.
2. Use a **Scanner** object for the arguments input and for input values in the array.
3. Call the method with appropriate arguments in the **RunArrayMethodsLab13.java** file.
4. Display array values before and after sorting.

```
public static void printValidNames(String[] names)
    {
        //Names array length
        int namesLength = names.length;
        for(int i = 0; i < namesLength; ++i)
        {
            String name = names[i];
            int strLength = name.length();
            char firstCharacter = name.charAt(0);
            int lettersCount = 0;
            boolean isValid = true;
            if(Character.isUpperCase(firstCharacter))
```

```
                {

                        lettersCount++;
                        for(int j = 1; j<strLength && isValid; ++j)
                        {
                              char c = name.charAt(j);
                              if(Character.isLetter(c))
                              {
                                    lettersCount++;
                              }
                              else{
                                    isValid = false;

                              }
                        }//inner-for
                        if(lettersCount >= 3 && lettersCount <= 32
&& isValid)
                        {
                              System.out.println(name);
                        }

                }
            }//outer-for

      }//printValidNames
```