GIFT School of Engineering and Applied Sciences

Spring 2022

CS-240: Object-oriented Programming – Lab

Lab-1 Manual

Methods and Arrays - Review

v1.4
5/28/2022

Task #1: Writing void Methods

In this task, you are being asked to write void methods in Java.

Write a void method called **getInput()** that takes input using a **Scanner** object, the **name** and **age** from a user. This method calls another method called **printInput()** that takes two arguments, one **String** (for the user name) and another **int** (for the user age) and prints both values using appropriate messages.

You may create method headers as:

```
public static void getInput()
```

and

```
public static void printInput(String name, int age)
```

NOTE: You need to call the **printInput()** method from the **getInput()** and pass appropriate values. The **main()** method will only call the **getInput()** method.

Also, perform input validation on the **age** argument, so that the method should only be called when the **age** is at least **10** and less than **70.**

1. Create a program called **InputMethodLab1.java.**
2. Use a Scanner object for the input.
3. Correctly call methods and display appropriate messages.

Task #2: Writing void Methods

In this task, you are being asked to write void methods in Java.

Write a method called **wordsInfo()** that accepts one **String** argument, and prints how many characters are there in that argument, as well as the number of vowels.

You may use the following header for this method:

```
public static void wordsInfo(String word)
```

For this exercise, assume that **a e i o u** are vowels. For example, if the method is called with an argument **"Harry"**, the method prints:

```
wordsInfo("Harry");
```

//method prints

5 characters.

1 vowel.

1. Create a program called **CharactersLab1.java**
2. Create appropriate variables and assign values using a **Scanner** object.
3. Correctly display appropriate messages.

Task #3: Writing void Methods

In this task, you are being asked to write void methods in Java.

Write a method called **printTable()** that accepts three positive integer arguments, and then prints the table of the number from the starting number to the ending number.

You may use the following header for this method:

```
public static void printTable(int number, int start, int end)
```

NOTE: Perform input validation so that all numbers must be greater than 0 and the "start" number should be less than "end" number.

1. Create a program called **TablesLab1.java.**
2. Create appropriate variables and assign values using a Scanner object.
3. Correctly display appropriate messages.

Task #4: Writing Value-returning Methods

In this task, you are being asked to write value-returning methods in Java.

Write a method called **charAtPosition()** that accepts two arguments: a **String** object, and a positive integer. It then returns the character located at position given by the integer argument.

You may use the following header for this method:

```
public static char charAtPosition(String word, int position)
```

For example, if the method is called with the arguments, **"GIFT University"** and **5**, the method returns the character **'U'** and a sample output should be:

```
Character at position 5 is: U
```

NOTE: A **String** starts from character position **0.** Also, perform input validation so that the **position** must be greater than or equal to **0.**

HINT: Use the **charAt(index)** method of the **String** to find the character in a **String.**

1. Create a program called **CharacterPositionLab1.java.**
2. Correctly call the method with appropriate arguments.
3. Correctly display appropriate messages.

Task #5: Writing Value-returning and void Methods

In this task, you are being asked to write value-returning and void methods in Java.

Your job is to write a program which will ask the user for rectangle's length and width, calculates the area of rectangle and display all information of the rectangle with the appropriate messages. Your program should have the following methods:

Value-returning methods:

1. Write a method called **getLength()** that asks the user to enter the length of rectangle and return that value as a double.

   You may use the following header for this method:

   ```
   public static double getLength()
   ```

2. Write a method called **getWidth()** that asks the user to enter the width of rectangle and return that value as a double.

   You may use the following header for this method:

   ```
   public static double getWidth()
   ```

3. Write a method called **getArea()** that takes the length and width and return the area of the rectangle.

   You may use the following header for this method:

   ```
   public static double getArea(double width, double
   length)
   ```

void methods:

4. Write a method called **dispaly()** that takes the length, width and area of the rectangle, and then print them with appropriate messages.

   You may use the following header for this method:

   ```
   public static void display(double width, double
   length, double area)
   ```

NOTE: Perform input validation so that length and width must be greater than **0.**

1. Create a program called **RectangleLab1.java.**
2. Create appropriate variables and assign values using a Scanner object.
3. Correctly display appropriate messages.

Task #6: Using Arrays and Methods

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

1. Write a method called **fillArray()** which will take an integer array as input and the fills the array with the user entered numbers.

   You may use the following header for this method:

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

2. Write another method called **printSumAverage()** which takes an integer array as input and prints the sum and average of the array elements.

   You may use the following header for this method:

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

Take a number from the user as the size of the array, call the method **fillArray()**, then call the method **printSumAverage()**.

NOTE: Perform input validation so that the size must be greater than 0.

1. Create a program called **ArraySumAverageLab1.java.**
2. Create appropriate variables and assign values using a **Scanner** object.
3. Correctly display appropriate messages.

Task #7: Using Arrays and Methods

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

1. Write a method called **fillArray()** which will take an integer array as input and the fills the array with the user entered numbers.

   You may use the following header for this method:

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

2. Write a method called **absoluteArray()** which will take an integer array as input and takes the absolute of all the array elements.

   You may use the following header for this method:

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

Take a number from the user as the size of the array, call the method **fillArray()**, then call the method **absoluteArray**().

3. Write a method called **printArray()** which will take an integer array as input and prints all its elements.

   You may use the following header for this method:

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

NOTE: Perform input validation so that the array size must be greater than 0.

1. Create a program called **ArrayAbsoluteLab1.java.**
2. Create appropriate variables and assign values using a **Scanner** object.
3. Correctly display appropriate messages.

Task #8: Using Arrays and Methods

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

1. Write a method that swaps the elements of the array using the following steps:
    a. Declare a temporary variable and copy the value at **0th** index in it.
    b. Now, you need to start copying array elements from their current index to the previous index (which comes one place before their current position). That is, the value placed at index 1 will be copied to index 0, the value at index 2 will be copied to index 1, and so on. The last value placed at index n-1 will be copied to index n-2.

       HINTS:

       - The final value in an array is always at index **n – 1**, where **n** is the size of the array.
       - To do this copying, a *for* loop would be easier to write. You will need to write code to copy the value of (i + 1)'th index to the i'th index.

    c. Finally, replace the value of **n-1'th** index with the value of temporary variable you declared in step (a).

    You may use the following header for this method:

    ```
    public static int[] arraySwapValues(int[] array)
    ```

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

    The method will return an array which would contain the values: **2 3 4 5 6 7 8 9 10 1**

2. Write a method called **printArray()** which will take an integer array as input and prints all its elements.

    You may use the following header for this method:

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

NOTE: Declare and initialize the array without taking input from user.

Print the array before and after calling the **arraySwapValues(int[] array)** method.

1. Create a program called **ArraySwapValuesLab1.java.**
2. Create appropriate variables and assign values using a **Scanner** object.
3. Correctly display appropriate messages.

Task #9: Using Arrays and Methods

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

1. Write a method that returns the reverse of a given integer array.

   You may use the following header for this method:

   ```
   public static int[] reverseArray(int[] array)
   ```

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

   The method will return an array which would contain the values: **1 2 3 4 5 6 7 8 9 10**

2. Write a method called **printArray()** which will take an integer array as input and prints all its elements.

   You may use the following header for this method:

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

NOTE: Declare and initialize the array without taking input from user.

Print the array before and after calling the **reverseArray (int[] array)** method.

1. Create a program called **ArrayReverseLab1.java.**
2. Create appropriate variables and assign values using a **Scanner** object.
3. Correctly display appropriate messages.

Task #10: Using Arrays and Methods

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

1. Write a method that returns the smallest value from a given integer array.

   You may use the following header for this method:

   ```
   public static int smallest(int[] array)
   ```

2. Write another method that returns the largest value from a given integer array.

   You may use the following header for this method:

   ```
   public static int largest(int[] array)
   ```

3. Write a third method that returns the middle value from a given integer array if the array length is *odd*, or returns the average of the two middle values if the array length is *even*.

   You may use the following header for this method:

   ```
   public static double middle(int[] array)
   ```

NOTE: Declare and initialize the array without taking the input from user.

1. Create a program called **ArrayValuesLab1.java.**
2. Create appropriate variables and assign values using a **Scanner** object.
3. Correctly display appropriate messages.

Task #1: Post Lab Task

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

1. Write a method that swaps the value of the array in such a way that all negative values come before the positive values in the array. The method should then return the array.

   You may use the following header for this method:

   ```
   public static int[] negativeBeforePositive(int[] array)
   ```

   For example, suppose that the array has 10 values: **25, 12, 0, -7, -4, 26, -2, 18, -9, 11**

   The method will return an array which would contain the values: **-7, -4, -2, -9, 25, 12, 0, 26, 18, 11**

   HINTS:

   - Create a temporary array equal to the size of the array passed as argument.
   - Now, write a loop that scans the argument array from start till end. It finds and copies all negative values to the temporary array, and then does the same for the positive values, including zero.
   - No ordering of values is required.

2. Write a method called **printArray()** which will take an integer array as input and prints all its elements.

   You may use the following header for this method:

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

NOTE: Declare and initialize the array without taking the input from user.

Treat the 0 as a value greater then negative numbers, but lesser than positive numbers.

Print the array before and after calling the **negativeBeforePositive(int[] array)** method.

1. Create a program called **ArrayNegativeBeforePositiveCP.java.**
2. Create appropriate variables and assign values using a **Scanner** object.
3. Correctly display appropriate messages.