CS 1400 - Lab 8

Maximum Points: 10 pts.

Lab Topics:

- Use of the array and classes
- Sorting a group of numbers and shifting the array

Use the following Coding Guidelines:

1) Download the template file Lab8.java from Blackboard and fill-in-the-blanks to create your Java program.
2) Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
3) Keep identifiers to a reasonably short length.
4) Use uppercase for constants. Use upper camel case for classes. Use lower camel case for all other identifiers (variables, methods, objects).
5) Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches, and loops. Be consistent with the number of spaces or tabs that you use to indent.
6) Use white space to make your program more readable.
7) Use comments to explain how the parts of your program work.

Problem Description

For this lab, you will create a basic array of numbers, fill in the elements of the array by prompting the user for input, sort the elements of array and display the sorted elements to the user.

Step 1: Create a class called Lab8. Make sure your file is called Lab8.java

At the beginning of each programming assignment you must have a comment block with the following information:

```
/*------------------------------------------------------------
// AUTHOR: your name.
// FILENAME: title of the source file.
// SPECIFICATION: your own description of the program.
// FOR: CS 1400- Lab #8
// TIME SPENT: how long it took you to complete the lab.
//----------------------------------------------------------- */
```

Step 2: Import libraries and declare variables you need

After the documentation block, you will need to import Scanner from java.util.

At the top of your main function, you can declare variables you need. You might need at least three variables, one for array size, one for swap values, and one for Scanner.

```
import java.util.Scanner;

        public class Lab8
        {
        public static void main(String[] args)
        {
        // ============================================================
        // Declare some variables. You might need
        //
        // - an integer for the array size,
        // - a int for the exchange element, and
        // - a scanner object for requesting input from System.in
        //
        // -->
        }
        }
```

Step 3: Request array size from the user and declare the array

Next, use the scanner object you have to request from the user the number of elements the array will have. Remember that you should always print a message indicating what is to be input before requesting information (by Scanner's .next functions) from the user.

```
        // ============================================================
        // Request array size from the user
        //
        // - Print this message "How many elements in the array?"
        // -->
        // - Request an integer from the user using the Scanner object and save it
        //-->
```

Step 4: Declare a new array and fill in the array by user input

Next, use the integer value previously requested from the user to declare a new array of a length specified by the user. This is very similar to the way we have declared Scanner, Person, Student, and Employee objects in the past. Make sure to use an integer array.

```
        // Declare a int array by the size you received above
        // -->
```

Afterward, implement a for loop to iterate available indices and assign each cell one user input.

The variable numOfElements in the following snippet is a placeholder. You can assign it the proper value or define your own loop condition.

```
        // ============================================================
        // Fill in the Array

            int numOfElements = -1;
            for (int i = 0; i < numOfElements; i++)
            {
            // Display the message: "Please enter the next value:"
            // -->
            // Request the next element (int) from the user,
            // save it to the ith element of the array
            // -->
            }

```

Note that this loop starts at 0 and its final iteration occurs when i is equal to the number of elements minus 1 (array.length – 1). This is important because Arrays in Java are *zero-indexed*, which means that the first element in the array is stored at the 0th position in the array. Inside the for loop, display a message to the user, request the next element, and store that value in the appropriate position of the array.

Step 5: Sort the array elements

Use two for loops to compare each element of array to all elements that are after this element in the array. If any element indexed by i below is greater than any element indexed by j then exchanges them. Please read the snippet for details.

The algorithm here is a modification of Selection Sort.

```
        // ============================================================
        // Sort the array's elements in increasing order
        //
        // Here we will use Selection Sort like algorithm.
```

```

            // The first for loop iterates all elements as element_i
            for (int i = 0; i < numOfElements; i++)
            {
            // The second for loop finds the right position of element_i
            for (int j = i + 1; j < numOfElements; j++)
            {
                // Compare ith value and jth value,
                //
                // - If array[i] >= array[j], swap these two values
                //
                // For example, let array[i] = 10, array[j] = 20, to swap
                // array[i] and array[j] means array[i] will become 20
                // and array[j] will have 10.
                //
                // To swap the values in two position, you would need an
                // extra variable to temporarily hold the value. For example,
                //
                //      temp = array[i];
                //      array[i] = array[j];
                //      array[j] = temp;
                //
                // -->
            }
        }

```

Step 6: Remove the minimum value

In this step shift all the elements one step to the left to remove the minimum element in the array. For this part, use a for loop and in each step, you need to put the value of element i+1 in the element i and put the zero in the last element of array.

```
        // ============================================================
        // Remove the minimum in the sorted array
        //
        // As our array is sorted in the increasing order, to remove the
        // minimum, we just remove the first element in the array. It is like
        // shifting the array to the left by

        // Because we are doing shifting, in each iteration, we use two
        // values, one at index i, the other at index i+1.
        // To prevent us from going over the boundary of array, the following
        // offset variable should be set as the correct value.
        int offset = -1;
        //In this loop, we move the element at I + 1 to the position i

        for (int i = 0; i < numOfElements - offset; i++)
        {
        // Move array[i + 1] to array[i]
        }
        // The last element will be set as zero. Remember the greatest index
        // should be array.length - 1.
        // -->
```

Step 7: Search for an element and remove it

In this step, ask the user for a value to search. You can use a for loop to find whether the target exists in the array. If so, remove it from the array by shifting.

You can design the logic by yourself. The following template shows one possible solution for your reference.

```java
        // ============================================================
// Search for an element and remove it

            // Ask the user which element to remove
            System.out.println("Enter the value to search and remove:");
            // Use your Scanner to get a value for search
            // int valueToRemove =

                // To search, we can iterate all values, record the index of
                target (t),
                // and then shift to the left values from t to the end.
                boolean isFound = false;

                for (int i = 0; i < numOfElements; i++)
                {
                // if ith element == valueToRemove,
                //     Set a flag isFound
                //
                // if isFound,
                //     if i + 1 is available
                //         move element i + 1 to index i

                //     if i + 1 is not available
                //         set element i as zero
                }

                if (isFound)
                {
                System.out.println("Search element found");
                }
                else
                {
                System.out.println("Search element NOT found");
                }
```

Step 8: Output

Please follow the Sample Output to see the format of the output. You are supposed to print the array after each operation (Step 5 sorting, Step 6 shifting, Step 7 search and removing).

```java
    // ============================================================
    // Display the final array

    System.out.println("\nThe final array");

    for (int i = 0; i < numOfElements; i++)
    {
        // Print ith element, do NOT include line break
        // -->
    }
    // Print a line break
    // -->
```

Sample Output

Below is an example of what your output should roughly look like when this lab is completed. Text in RED represents user input.

Sample Output 1

```
How many elements in the array?
5
Please enter the next value:
3
Please enter the next value:
6
Please enter the next value:
2
Please enter the next value:
8
Please enter the next value:
4
The array after sorting
2, 3, 4, 6, 8

The array with the minimum removed
3, 4, 6, 8, 0

Enter the value to search and remove:
6
Search element found

The final array
3, 4, 8, 0, 0
```

Sample Output 2

```
How many elements in the array?
3
Please enter the next value:
13
Please enter the next value:
1
Please enter the next value:

4
The array after sorting
1, 4, 13

The array with the minimum removed
4, 13, 0

Enter the value to search and remove:
13
Search element found

The final array 4, 0, 0
```