Assignment 02: Histogram

COSC 102 - Spring '17 (DRAFT)

In this assignment, you will develop further experience working with arrays (including two-dimensional arrays) in Java and you will implement a sorting algorithm adapting the code found in our BJP textbook. You will also have an opportunity to practice handling user input through the use of a `Scanner` object as well as overloading functions.

1 Requirements

For this assignment, you are required to implement three modes to display histograms as well as test cases to validate the correctness of your implementation. Below two functions are described but you are expected to design your solution with more functions. It is important that you **read this entire document before you start writing any code.**

All of your code should be written in a Java file `Histogram.java`. Your implementation should follow the style sheet distributed in class, including the header comments. In addition you are required to submit a plain file called `logBook.txt`, at the top of which you include your name and this assignment information. Its content is a summary of what you plan and did each time you sit to work on this assignment: your incremental design for each task and the functions you developed, revised and your gradual testing process. Think about how you would report to a supervisor the progress you made each day on your project.

You are not allowed to use any `Arrays` or `List` methods of the Java API. You have to manipulate the basic `int[]` structure—the container object that holds a fixed number of values of a single type. You are allowed to use the built-in `length` property to determine the size of any array. You may consider using an array of arrays (also known as *a multidimensional array*) by using two in our case sets of brackets. To do so, you have to consider and determine the data representation that is most effective in terms of performance. It isn't required to use a 2D array but it does streamline the design. (I recommend looking at BJP to get the 2D array syntax right).

2 Make histogram for a hard-coded `int[]`

Write a function called `makeHistogram` that takes a single argument of your choice to send the integer data. Given a one dimensional array of integers `makeHistogram` role is to **print** out a chart representing for each distinct value in the input array its frequency as a bar of asterisks (*). Each asterisk represents an occurrence alongside of the integer value. Think about the quiz results I sent you.

For example for the hard-coded declaration

```
int[] arr = {8, 4, 6, 6, 8, 4, 2, 6};
```

the resulting histogram produced by `makeHistogram` is as follows

```
Histogram Results
=================
8: **
4: **
6: ***
2: *
```

The format is each unique integer—as they appear in sequence in the array—followed by a colon and the bar with one asterisk for each occurrence of that integer in the array.

Implement multiple tests to validate your function and don't erase them as you are graded on the evidence of your thorough testing.

3 Make histogram for interactive input

Write a function called `makeHistogram` which should take no arguments. This function will perform the same task as the previous function, though since it accepts no arguments it will prompt the user for the values. **Think about code reuse for the overall functionality of this alternate mode...**

This `makeHistogram` function first needs to prompt the user for the total number of integers they wish to enter, then prompt the user once for each of these values. To do this, you have to use a `Scanner` object as seen in mastermind (see `MMind.java` on Moodle) and described in BJP. For now, you **do not** need to account for the user entering a non-numerical input but you **have to handle properly a negative value for the first prompt**.

Below is an example of the user and program interactions to communicate the data and the resulting output:

```
How many integer(s) do you wish to enter?:  6
Enter value #1:  17
Enter value #2:  2
Enter value #3:  17
Enter value #4:  3
Enter value #5:  3
Enter value #6:  17

  Histogram Results
  =================
 17: ***
  2: *
  3: **
```

4 Sorted histogram

Add a boolean class variable `SORTED`, which default initialization is `false` when you submit your code. This class variable isn't a constant! The programmer, you and I, may change its value to `true` so that the histogram output shown on page 1 instead is

```
Histogram Results
=================
2: *
4: **
6: ***
8: **
```

To create this functionality you have to understand and use the Selection Sort algorithm presented in BJP Chapter 13 (p. 861–864 Fourth Ed, p. 803 Second Ed). Be careful how you integrate the code of the textbook, it is essential to understand it and know your different context in terms of data representation for example.

Once the output is sorted when `SORTED` is set to `true`, put it back to `false` and add a user prompt in the second `makeHistogram` function to similarly permit a sorted output. Both `makeHistogram` functions output should be dependent of the `SORTED` class variable. In the interactive mode the variable value is overwritten by user input.

5 Testing

As you implement these functions, it is important that you frequently **save, compile and test** your code! In addition to implementing the two functions outlined above, you must include a `main` method which contains the tests you used to verify your implementation.

You must include enough tests for each function so that you feel satisfied that you have covered all of your edge cases. Additionally, you may not reuse any of the test cases provided in this document.

6 Challenge – Alternate Format [OPTIONAL]

As an optional challenge for extra credit, write an additional function to output histograms in a vertical orientation: the asterisks are stacked vertically, and the unique numbers are listed horizontally.

Below are the alternate output given an array `int[] arr = {4, 6, 6, 8, 4, 2, 6, 8, 8, 8, 8, 8, 1}` with `SORTED = true`

```
Histogram Results
=================
        *
        *
        *
      * *
    * * *
* * * * *
1 2 4 6 8
```

and with `SORTED = false`

```
Histogram Results
=================
    *
    *
    *
  * *
* * *
* * * * *
4 6 8 2 1
```

Hint: consider an ASCII-like image representation. I used a 2D array: first replicating our horizontal bar graph and then rotating it through indices manipulation. It is a challenge. Don't feel bad if you haven't the time to try it.

7 Submit

Your submission for this assignment is due **Monday, February 13th** at **10:00PM**. Upload your `Histogram.java` and `logBook.txt` files to the appropriate link on our course's Moodle page. Be sure to only upload your `.java` file – **do not submit the `.class` file!**