Histogram - Grade Distribution

1 Introduction

This assignment will have you working with 1-D arrays, console/file IO, and command-line args.

2 Problem Description

You are a Georgia Tech professor teaching a class. It is the end of the semester and you wish to see how your students performed, so you write a Java program that will create a histogram of the grade distribution. You want this histogram program to be able to give you a very detailed view or a very broad view of the grade distribution. To accomplish this, it asks the user how many bins the grades should be split into.

3 Solution Description

Put your code in a file named `GradeHistogram.java`.

We have provided you with a CSV file that has a list of students and their grades. A CSV file is just a text file with data partiioned by commas and (in this case) newlines.
These grades are not sorted but they are bound between 0 and 100 (inclusive). For example, the file may look like:

```
Glenn Hollingsworth, 91
Chris Simpkins, 100
Thomas Shields, 89
Bob, 55
Alice, 95
Eve, 87
```

Use command-line arguments to inform your program of the name of the grades file - see the example below for how to pass the file name in while running the program.

Using an array, you must count the frequency of each grade value in the file and print it to the standard output as a horizontal histogram. For every grade in a particular range, print a pair of brackets

You may only read through the file once.

You must also label the range of each histogram bar and allow the user to indicate what size interval they would like the histogram to be made with. You must allow the user to specify this size interval in one of two ways:

1. Firstly, the size interval may be specified as an additional command line arg, e.g `java GradeHistogram grades.csv 5`
2. If the second command line arg is not present, your program must ask the user for the bucket size.

Running the program should look like this:

> **Note:** `$` is the command prompt on Unix. On Windows it will look like `C:>`

```
$ java GradeHistogram grades.csv
Grades loaded!
What bucket size would you like?
>>> 10

100 - 91 | [][][][][][][][][][][][]
 90 - 81 | [][][][][][][][][][][][][][][][][][][][][][][][][][]
 80 - 71 | [][][][][][][][][][][][][][]
 70 - 61 | [][][][][][][][][][][][][][][][][]
 60 - 51 | [][][][][][][]
 50 - 41 | [][][][]
 40 - 31 | [][][][][][]
 30 - 21 | [][]
 20 - 11 |
 10 -  1 | []
  0 -  0 | []

$ java GradeHistogram grades.csv 5
Grades loaded!

100 - 96 | [][][][][][]
 95 - 91 | [][][][][][]
 90 - 86 | [][][][][][][][][][][][][][][][][]
 85 - 81 | [][][][][][][][][]
 80 - 76 | [][][][][][][][][][][][]
 75 - 71 | [][]
 70 - 66 | [][][][][][][][][][]
 65 - 61 | [][][][][][][]
 60 - 56 | [][][][][]
 55 - 51 | [][]
 50 - 46 | [][][]
 45 - 41 | []
 40 - 36 | [][][]
 35 - 31 | [][][]
 30 - 26 | []
 25 - 21 | []
 20 - 16 |
 15 - 11 |
 10 -  6 | []
  5 -  1 |
  0 -  0 | []
```

> **Note:** The pipe characters should be aligned and your program must not exclude any subrange between 0 and 100.

4 Tips

- You may assume that you always get valid input.
- You may assume the text file has valid numbers.
- 101 is a prime number.
- An array is a fixed size data structure; you need to know ahead of time how big it needs to be. How do we do this?
- You can give interpretations to the indices and contents of an array to arrive at creative solutions to problems. Code smart, not hard.
- Creating a `Scanner` object with a file will throw a checked exception. Don't worry about what this means — for now, just append `throws Exception` to the end of the `main` method signature wherein the file is opened.

5 Checkstyle

You must run checkstyle on your submission. The checkstyle cap for this assignment is **15** points. Review the Style Guide and download the Checkstyle jar. Run Checkstyle on your code like so:

```
$ java -jar checkstyle-6.2.1.jar *.java
Audit done. Errors (potential points off):
0
```

The message above means there were no Checkstyle errors. If you had any errors, they would show up above this message, and the number at the end would be the points we would take off. The Java source files we provide contain no Checkstyle errors. For this assignment, there will be a maximum of **15** points lost due to Checkstyle errors (1 point per error). In future homeworks we will be increasing this cap, so get into the habit of fixing these style errors early!

6 Turn-in Procedure

Submit `GradeHistogram.java` and nothing else to T-Square. Do not submit any compiled bytecode (`.class` files) or the Checkstyle jar file. When you're ready, double-check that you have submitted and not just saved a draft.

Please remember to run your code through Checkstyle!

Verify the Success of Your Submission to T-Square
Practice safe submission! Verify that your HW files were truly submitted correctly, the upload was successful, and that the files compile and run. It is solely your responsibility to turn in your homework and practice this safe submission safeguard.

1. After uploading the files to T-Square you should receive an email from T-Square listing the names of the files that were uploaded and received. If you do not get the confirmation email almost immediately, something is wrong with your HW submission and/or your email. Even receiving the email does not guarantee that you turned in exactly what you intended.

2. After submitting the files to T-Square, return to the Assignment menu option and this homework. It should show the submitted files.

3. Download copies of your submitted files from the T-Square Assignment page placing them in a new folder.

4. Recompile and test those exact files.

5. This helps guard against a few things.

   (a) It helps insure that you turn in the correct files.

   (b) It helps you realize if you omit a file or files. [1] (If you do discover that you omitted a file, submit all of your files again, not just the missing one.)

   (c) Helps find last minute causes of files not compiling and/or running.

[1] Missing files will not be given any credit, and non-compiling homework solutions will receive few to zero points. Also recall that late homework will not be accepted regardless of excuse. Treat the due date with respect. The real due date is midnight. Do not wait until the last minute!