Academic Honesty Rules for This Assignment

This assignment is individual work. You may:

- Use the textbook and class videos, including code in the textbook and videos.
- Use the documentation at docs.oracle.com/en/java/ as a resource. Do not borrow code from the documentation.
- Ask me questions.

Discussing this assignment with any other person (in person or on the Internet), searching for answers on the Internet, or using sources other than the ones listed above is a violation of the class academic honesty policy. Please ask if you have any questions about this.

Specifications

Problem Description

You are going to write a program that obtains birth weight data in grams from the keyboard, in a loop, until the user enters a sentinel value of -1.

The program will categorize the data as follows:

- 0-999 grams: Extremely Low Birth Weight
- 1000-1499 grams: Very Low Birth Weight
- 1500-2499 grams: Low Birth Weight
- 2500 and above: Normal Birth Weight

Input: The user will enter weights until they enter the sentinel value of -1. You may assume the user will enter integer data at the prompt, but you must validate that values entered are positive.

Output: After the user has entered the sentinel value, the program will display a summary table composed of category, average weight within the category, number of births within the category, and percentage of overall births within the category.

The program will validate the weights as they are entered. The system will respond to any non-positive value that is not -1 with an error message. Weights that are invalid will be discarded.

Interface: This is a console program. Do not use JOptionPane.

Your program should use the following prompt:

- `Please enter the next birth weight, -1 when done:`

Your program should use the following error message for invalid input:

- `That is an invalid weight.`

If no valid data is entered, the system should display:

- `No weights were entered.`

If any valid data is entered, the system should display an output table. A sample output table looks like this:

```
Category            Average             Number              % of Total
Extremely LBW       520.00              1                   1.92
Very LBW            0.00                0                   0.00
Low Birth Weight    2164.00             3                   5.77
Normal Weight       3531.33             48                  92.31
```

Design specifications: Use only techniques that are covered in the first three lessons. Use console input and output.

Do not, under any circumstances, use `System.exit` or `return`. Use `break` only in `switch` statements. Do not submit code that includes these constructs (except for `break` in `switch` statements). Submit whatever you can write that does not include these constructs.

Your program must use a sentinel-controlled loop to obtain input from the user.

The program should total all counts for all categories and should count the number of weights entered for each category. When all input has been entered, it should compute percentages by dividing category count by count for all categories, and then multiplying the result by 100. It should compute average by dividing category total by category count.

Clarifications: Include in your submission everything you've clarified with the professor.

Assumptions: Include in your submission anything you assumed but didn't clarify. Include a justification.

Testing

Included with this assignment is a tab-delimited table of live birth data taken from the following website: http://www.nber.org/data/vital-statistics-natality-data.html

You may use a subset of this data for your test cases, or you may use it as a guide for what a natural range of birthweights in a population looks like and invent your own test cases.

Test Case 1

Purpose

Test a case with all valid inputs.

Input

Data:

2183	3686	3459	3515	3912	3345	3941	5358	3260	1899	3657	4905	3062
3912	520	3005	3232	3487	2920	3430	2410	4536	3600	3033	3459	3686
3062	3033	3345	3657	2835	3430	2948	3771	3600	3941	3147	3459	4054
2977	3430	3742	3260	4111	3572	3062	3090	3856	3714	3430	2807	3771

Expected output

Computations:

ELBW: Values: 520; Average: 520; Percentage of total: 1 / 52 * 100 = 1.92%

VLBW: Values: -; Average: 0; Percentage of total: 0

LBW: Values: 2183, 1899, 2410; Average: 2164; Percentage of total: 3 / 52 * 100 = 5.77%

NBW: Values: (all the rest); Average: 3531.333; Percentage of total: 48 / 52 * 100 = 92.31%

Output**:**

```
Category            Average             Number              % of Total
Extremely LBW       520.00              1                   1.92
Very LBW            0.00                0                   0.00
Low Birth Weight    2164.00             3                   5.77
Normal Weight       3531.33             48                  92.31
```

Your program running

Paste your program executing in the testing document here. Show a complete execution: both inputs and output.

Conclusions

Describe whether your program's output matches expectations.

Test Case 2

Purpose

Invalid weights entered

Input

Data:

2183 3686 0 3515 3345 5358 3260 -2 3657 3062

Expected output

Computations:

ELBW: Values: -; Average: -; Percentage of total: 0

VLBW: Values: -; Average: 0; Percentage of total: 0

LBW: Values: 2183; Average: 2183; Percentage of total: 1 / 8 * 100 = 12.5%

NBW: Values: (all the rest); Average: 3697.571; Percentage of total: 7 /8 * 100 = 87.5%

Output*:*

```
Please enter the next birth weight, -1 when done: 2183
Please enter the next birth weight, -1 when done: 3686
Please enter the next birth weight, -1 when done: 0
That is an invalid weight.
Please enter the next birth weight, -1 when done: 3515
Please enter the next birth weight, -1 when done: 3345
Please enter the next birth weight, -1 when done: 5358
Please enter the next birth weight, -1 when done: 3260
```

```
Please enter the next birth weight, -1 when done: -2
That is an invalid weight.
Please enter the next birth weight, -1 when done: 3657
Please enter the next birth weight, -1 when done: 3062
Please enter the next birth weight, -1 when done: -1
Category            Average             Number              % of Total
Extremely LBW       0.00                0                   0.00
Very LBW            0.00                0                   0.00
Low Birth Weight    2183.00             1                   12.50
Normal Weight       3697.57             7                   87.50
```

Your program running

Paste your program executing in the testing document here. Show a complete execution: both inputs and output.

Conclusions

Describe whether your program's output matches expectations.

Test Case 3

Purpose

No weights entered

Input

Data:

-1

Expected output

```
No weights were entered.
```

Your program running

Paste your program executing in the testing document here. Show a complete execution: both inputs and output.

Conclusions

Describe whether your program's output matches expectations.

Turn in

1. Your java program (the .java file only). Make sure that it meets the requirements of the rubric.

2. A word-processed document containing the following:

   Section 1: Clarifications and assumptions

   If you clarified specifications or made assumptions that were not clarified with the professor prior to submission, include them here with justifications.

Section 2: Testing

Copy / paste your program running on all test cases and include any additional test cases you used to thoroughly test your program.

Zip the java file and the word-processed document into a folder named JavaProgram2_your initials, for example, JavaProgram2_msb.

Upload the zipped folder to Blackboard.

Rubric

An exceptional-quality assignment will meet the following standards:

- Meeting functional and design specifications, 70 points
  The Java program works and meets all of the specifications, with no additional unspecified functionality. The programmer has used programming techniques from the first three lessons only. If the program misses specifications or does not function correctly, errors are acknowledged with a thorough and reflective analysis in the testing section (points will be removed for missed specifications).
- Communicating with identifiers and white space, 10 points
  The program makes appropriate use of variables. Variables and constants are named according to convention and are named for understandability and purpose. White space, both vertical and horizontal, is correctly used for readability and meets programming conventions.
- Communicating through documentation, 10 points
  The Java program contains comments including the programmer's name and date. There are block comments (as many as necessary) for each distinct block of code which accurately describe what the block is accomplishing by relating the code to the problem being solved. Javadoc is included and meets the javadoc standards.
- Assumptions and Testing, 10 points
  Testing is thorough. If there are errors, they are described in the testing section. If there are questions, they are answered thoughtfully in the testing section. All assumptions are made explicit. The assignment will be reduced by a minimum of 10 points if the programmer claims a non-working program meets specifications.