New concepts needed to complete this assignment include: Static Methods. No methods in this assignment require a recursive solution.

What Are Histograms?

Histograms are a convenient way to visualize data distributions. A histogram is displayed by plotting the frequency (or number of occurrences) along the horizontal axis, and intervals of some observed variable along the vertical axis. One interesting application of a histogram is to track the distribution of the use of letters in the alphabet in everyday writing. For instance, let’s say I was interested in studying the distribution of letters used by Shakespeare in his comedy, *As You Like It*. I input the following line into my histogram generator program:

All the world’s a stage, And all the men and women merely players; They have their exits and their entrances, And one man in his time plays many parts, His acts being seven ages.

Figure 1: Example Histogram
Figure 1 shows the histogram of the frequency of letters occurring in this quote. The observed variables, which are the letters in the alphabet run along the vertical axis. The frequency of those letters appearing in the given quote is plotted against the horizontal axis.

The resulting histogram may or may not be surprising. The most frequently used letters in this quote are: e, a, n, s, t, r, etc. Apply this histogram over the entire collection of English language, and you will notice a similar pattern. Studying text statistics using histograms established the foundation for technologies like Morse code, Braille, and file compression (like zip). Indeed, knowing that a, e, and s are some of the most frequently-used letters, designers of Braille and Morse codes assign those letters with the simplest representation, because people will encounter them often. Conversely, letters that are least frequently used (e.g., q, x, z, etc.) are given the longest representations.

**Program Requirements (95pts)**

- Go to the course page, and copy over the two files for this homework: Alphabet.java and Histogram.java. Read over the code template to get an understanding.

- Open Alphabet.java first. The first new item that might jump out at you is this line:

```
public static final int SIZE = 26;
```

The `final` keyword means that this variable’s value cannot be changed. In other words, this line defines a constant, called Alphabet.SIZE to be 26. In Java, constants can be referenced by ClassName.constantName. In this case, Alphabet.SIZE will always return 26.

- Just below this constant, two static methods are waiting to be implemented by you:
  - `public static int getPositionByLetter(char letter)`: This case-insensitive method returns the position of a letter in the alphabet, where 'a' returns 0, 'b' returns 1, etc.
  - `public static char getLetterByPosition(int position)`: This method returns the letter, given its position in the alphabet where 0 returns 'a', 1 returns 'b', etc.

- I would implement and test these methods before moving on.

- Now open Histogram.java. Your program must input a String of arbitrary length using Scanner’s `nextLine()` method. (Important: while Java does not have an upper limit on input length, BlueJ does impose a character limit of 254).

- The histogram can be represented with an integer array of size Alphabet.SIZE, where each element in the array holds the count of a letter in the input String. The 0th element in the histogram array holds the count of 'a,' 1st element holds the count of 'b,' and so on. The array in Figure 2 represents a histogram with 6 a’s, 3 d’s, and 5 e’s.

![Histogram as an Integer Array](image)
- You must populate this histogram array by counting each letter in the input String, and updating this array appropriately.

- To do this, you need to have a way to map each character in the alphabet to an index in this array, and vice versa. Hmm.. how might those methods you implemented in the Alphabet class help?

- You are encouraged implement as many methods as you see fit for completing this program. Indeed, expert programmers often only use a few lines inside main() to call the appropriate methods. However, you must implement the following:
  
  - public static int countLetters(String str, char letter): Returns the occurrences of a letter in the given string. This method ignores case.
  
  - public static void populateHistogram(String str, int[] hist): Populates the given histogram, given the String and an array representing the histogram. Suppose the string str is ‘puget sound,’ Figure 3.

Before calling populateHistogram() String str = "puget sound";

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

After calling populateHistogram() String str = "puget sound";

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 3: Contents of Hist Before and after call

- public static void printHorizontal(int[] hist): Given an array, representing a histogram of letters, print out the histogram horizontally. This method assumes that hist[0] holds the occurrences of 'a', hist[1] holds occurrences of 'e', and so on. For each vowel, print the count of occurrences, then print an asterisk * per occurrence.

For example, if your histogram array holds the content in the lower part of Figure 3 after you’ve populated it, then the following is printed to the BlueJ terminal:

a (0):
b (0):
c (0):
d (1): *
e (1): *
f (0):
g (1): *
h (0):
i (0):
j (0):
k (0):
public static void printVertical(int[] hist). This method will input a histogram array and print it out vertically. The level of difficulty of this method is substantially harder than printHorizontal(). Spend time carefully designing it, and don’t give up! Here’s an example:
Start Coding Like a Pro: Test Suites

- Prospective CS majors and minors take note! Test suites are a necessity as you move on to more advanced courses and projects, so get used to it now!

- As you saw in the lab, sometimes it’s more intuitive (or necessary) to test out your methods by writing some “dummy code” to assure they’re working as expected before moving on.

- It may not seem like it, but you will save yourself a tremendous amount of time by testing out each method as you implement them. It is generally much more time consuming to dig for bugs later, when you don’t even know which method is causing the error!

- For example, assuming that you have implemented `countLetters()`, you can test it out by placing the following code inside `public static void main(String[] args):

```java
public Histogram
{
    /** main method */
    public static void main(String[] args)
    {

        /** Test Suite for countLetters() */
        // should be 2
        System.out.println(countLetters("David", 'd'));

        // should still be 2
        System.out.println(countLetters("David", 'D'));

        // should be 0
        System.out.println(countLetters("David and Goliath", 'z'));

        // should be 3
        System.out.println(countLetters("David and Goliath", 'a'));
    }

    // rest of static methods omitted
}
```

- Again, coming up with good test cases to test methods’ correctness is an art-form. For each method you implement, be creative and think up cases that might either break it or elicit informative outputs. Trivial cases like empty-string and 0 always serve as good test cases.
Sample Program Interaction

Enter any string:
Add to that the magnetic pull of Mount Rainier, the ocean, the rain forests and it won’t take
long to create a four-year agenda not just of outings, but also of research experiences
in the region.

To print a horizontal histogram, enter 1. Otherwise, enter 2 to print a vertical histogram: 1

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>15</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>5</td>
</tr>
<tr>
<td>d</td>
<td>4</td>
</tr>
<tr>
<td>e</td>
<td>20</td>
</tr>
<tr>
<td>f</td>
<td>5</td>
</tr>
<tr>
<td>g</td>
<td>5</td>
</tr>
<tr>
<td>h</td>
<td>6</td>
</tr>
<tr>
<td>i</td>
<td>9</td>
</tr>
<tr>
<td>j</td>
<td>1</td>
</tr>
<tr>
<td>k</td>
<td>1</td>
</tr>
<tr>
<td>l</td>
<td>4</td>
</tr>
<tr>
<td>m</td>
<td>2</td>
</tr>
<tr>
<td>n</td>
<td>14</td>
</tr>
<tr>
<td>o</td>
<td>15</td>
</tr>
<tr>
<td>p</td>
<td>2</td>
</tr>
<tr>
<td>q</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>11</td>
</tr>
<tr>
<td>s</td>
<td>7</td>
</tr>
<tr>
<td>t</td>
<td>19</td>
</tr>
<tr>
<td>u</td>
<td>6</td>
</tr>
<tr>
<td>v</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>1</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
</tr>
</tbody>
</table>

Submission and Style Standards (5pts)

□ Be sure to specify your names and roles in the README.txt file.

□ Ensure that your program’s description, your name, and date are specified in the comment block at
the top of the file containing the main() method. You can use the block that is automatically provided
by BlueJ. For instance,

```java
/**
 * This program generates a histogram...
 * @author Histo Graham
 * @version 08/24/2014
 */
```
☐ Effective commenting where appropriate.

☐ Proper and consistent indenting.

☐ When necessary, a **README.txt** file with specific instructions for your program.

☐ Please zip up submit your source files (that is, only files with a `.java` extension), and submit the zipped file on Moodle.