—Assignment #3—

Make sure to review the submission requirements on pg. 4 of the Lab #1 document.

Follow the Style Guide as given in the Resources section of our conneX course page.

Learning Outcomes

When you have completed this assignment, you should understand:

- How to use procedural decomposition to break down a problem into methods.
- How to use `while` loops to process variable length input.
- How to use `Scanner` to get a line of `String` data type user input.
- The use of `String` object methods.
- The use of `if/else` statements.
- How to develop your own strategy to test code for correctness.

You will write a program to translate a user input message either from English to Pig Latin, or vice versa. Call your program `PigLatin.java` and compile and run it frequently as you work through each piece to build it.

Pig Latin takes the first letter of a word and places it at the end of the word together with the letters "ay" added. For example, the word "play" becomes "laypay" in Pig Latin. This translation process needs to be done on each individual word in a message that the user will enter if they choose to translate from English to Pig Latin. Otherwise, the message the user enters should already be in Pig Latin, and the translation reverts the message back to English.

The first sample run shows the program translating from English to Pig Latin:

```
--------------------------------
Welcome to Pig Latin translator!
Enter 1 to translate to Pig Latin,
enter 2 to translate back to English: 1
................................

What is your message? a rose by any other name would smell as sweet

Your message translated to Pig Latin is:
aay oseray ybay nyaay theroay amenay ouldway mellsay saay weetsay
--------------------------------
```

The second sample run shows the program translating from Pig Latin to English:

```
--------------------------------
Welcome to Pig Latin translator!
Enter 1 to translate to Pig Latin,
enter 2 to translate back to English: 2
................................

What is your message? aay oseray ybay nyaay theroay amenay ouldway mellsay saay weetsay

Your message translated to English is:
a rose by any other name would smell as sweet
--------------------------------
```

The user might enter a number other than 1 or 2. Your program should reprompt the user until they enter the number 1 or 2. For example:

```
--------------------------------
Welcome to Pig Latin translater!
Enter 1 to translate to Pig Latin,
enter 2 to transalte back to English: 3
Please enter 1 or 2: 4
Please enter 1 or 2: 5
Please enter 1 or 2: 10
Please enter 1 or 2: 1
................................

What is your message? why is it not possible to enter more numbers

Your message translated to Pig Latin is:
hyway siay tiay otnay ossiblepay otay ntereay oremay umbersnay
--------------------------------
```

You don't have to worry about the user entering punctuation, or whether the user enters capital letters or not. Stick to user input that avoids these as you test, and only have words separated by exactly one space, no spaces at the beginning or end of the input.

Your translated message should never have an extra space at the end!

You should have two methods called `toPigLatin` and `toEnglish`, but you also need to create your own methods to help break up your code to be more efficient, easier to read, and especially to make it easier for you to test and self-correct. There are two other methods that you can create which make both `toPigLatin` and `toEnglish` much easier to write and test. Try to work in a similar fashion to the instructions in Assignment #2 to work your way up to a solution. A small hint: you should use `while` loops instead of `for` loops in this program. If you are ever tempted to copy and paste your code, that is a huge red flag that you need to write a method!

Your output should look exactly like the sample output given, except for the user input and translated message, of course, which vary by use.

There are two important caveats to note:

(1) Java has a small issue with the `Scanner` methods `nextInt` and `nextLine` when called in this order. The `Scanner` does not parse the newline character when the user presses "enter" after they give an integer, so you have to call `nextLine` once to parse the newline character, and then call it again to get the user's message.

(2) You will need the following combined relational expressions in your code:

```
i < message.length() && message.charAt(i) != ' '
```

where `i` is an index used to access characters in the user's input `message`. You could use different variable names.
The important thing to note is that the relational expressions must be written in the order given, because Java computes the left expression first, and if it is `false` it does not try to compute the right expression. If you tried to write the expressions in the opposite order, then your program would have a runtime error when it tries to access a character in the message that is out of range.

If you decide to write a method to test parts of your code, you could leave that extra code in your submission, but make sure to get rid of any calls to it before you submit. If you have an adequate testing method, then it can earn you up to 5 buffer marks (doesn't contribute to a mark above 30).

Grading

Marks will be allocated out of 30 points:

- (0 points) You do not submit any program file.
- (5 points) Your program does not compile. The marker cannot fix the complier error easily, or if they can, the output is severely mismatched with the assignment requirements.
- (10 points) Your program does not compile, but the marker can fix the error easily.
- (15 points) Your program compiles, but only outputs a few of the assignment requirements.
- (20 points) Your program compiles, outputs most of the assignment requirements, and is missing documentation comments or has formatting issues.
- (25 points) Your program compiles, outputs everything required, but is missing documentation comments or has formatting issues.
- (30 points) Your program compiles, outputs everything required, and has proper documentation and formatting.