Assignment 1
unfay ithway igPay atinLay
EE422C - The University of Texas at Austin – Spring 2025

Assigned: January 14th 2025
Due: 11:59pm, January 27th, 2025

Purpose – A compiler is a language translator, and so is the solution to this problem.
Pig Latin is a twist of English for people who don't want others to know what they're communicating about. Your task is to write a program that translates English phrases into their Pig Latin equivalent. This will also give you the opportunity to learn more about manipulating String objects.

Complete and submit your Java program (.java file) using Gradescope by 11:59pm on Monday, January 27th. Late submissions turned in within 24 hours of the deadline will be docked 10%, and those received after 24 hours will receive a 0.

Tips on accepting help from me or the TA. If you come to us for help we will first ask to see your current version of the design – show it to us no matter how rough it is. If we tell you that it or some part of it won't work, then make us explain why it won't to your satisfaction. We will try to help you formulate your solution; whatever that might be!! Remember that there are many different designs that can solve this problem – not just one – **and one that works is all that is required**.

Assignment Development Requirements

Design**,** write and run a Java program that will convert a sequence of given English language phrases into Pig Latin, according to the translation rules of Pig Latin given below.

A. Input: Using the given Translator class (supplied to you), the input to the program will be one English language phrase at a time to be translated into Pig Latin. The input lines for translation will come from a file, whose filename is to be specified in the command line argument. The translator class will handle reading the file, one line at a time. The file may contain multiple lines, but the template code we provide handles this.
B. Output: for each input phrase you are to output (to the screen) the original phrase and the resultant Pig Latin equivalent phrase, with appropriate identifying labels on each (see below for example).
C. **The given Translator class gives you a skeleton program to use. What you need to do is to fill in the *translate()* method to implement your solution.** Put in your code where it says so in the comments. You may add anything you need to the given code, but do not delete anything from it.
D. A sample set of test phrases to be processed by your program will be provided. You are strongly encouraged to create your own test phrases too, as we will test your code with phrases other than just those we give you.
E. **The input and output phrases must be represented as Strings.**
F. Test your program well to make sure it works correctly. Use the given input data set when it becomes available. Remember that your program will be graded on system closely resembling the ECE Linux servers, so be sure it works there.

Sample output: An example output from your program for the single input phrase "Fun with pig latin!" must look **exactly** like this on the screen:

```
The phrase that was input is: Fun with pig latin!
The Pig Latin version is: unFay ithway igpay atinlay!
```

Notes: The autograder is very picky! You must have the exact wording above. Each colon as shown above must be followed by a single space. The punctuation at the end must the same as the original sentence. So, if there is no ‘.’, ‘!’, or ‘?’ at the end, your answer should not have one either. Do not add extra output. Consider the above a design specification you have been given to meet.

Pig Latin Translation Rules for English *Words*:

Here are the basic rules for translating English words into Pig Latin: You may assume that all words are either correct English words or that they fall under Rule 6.

Rule 1. For words that begin with a single consonant, take the consonant off the front of the word and add it to the end of the word. Then add ay after the consonant. Here are some examples:
cat => atcay
dog => ogday
simply => implysay
noise => oisenay

Rule 2. For words that begin with double or multiple consonants take the group of consonants off the front of the word and add them to the end, adding ay at the very end of the word. Here are some examples:
scratch => atchscray
thick => ickthay
flight => ightflay
grime => imegray

Rule 1 and Rule 2 cannot be applied simultaneously.

Rule 3. For words that begin with a vowel, just add yay at the end. The letter y is a consonant when found at the beginning of a word. For examples:
is => isyay
apple => appleyay
under => underyay
octopus => octopusyay

Rule 4. Hyphenated words are treated separately with “-” placed in between. For example:
well-documented => ellway-ocumentedday

Rule 5. Contracted words are treated as one word. Apply one of Rules 1-3 as applicable up to the ’ character if needed. For example:
y’all => ’allyay
don’t => on’tday

Rule 6. All other words and symbols not covered by rules 1-5 are not to be changed. Rule 6 is applied to non-words such as H1N1 or H*&N (i.e. not punctuation marks). It also applies to non-alphabetic words, or words with non-alphabetic characters embedded in them. When punctuation is included inside of a word, you can do whatever you want, except crashing. Punctuation symbols are defined as: , " (...) : ; ' . ! ?
Note: punctuation symbols can also serve as word delimiters along with white space.

For example:
: => :
897 => 897

Additional tips and considerations:

- Coding standards - Starting with this assignment you must follow the class coding standards (see them on the class Canvas page). These standards ensure your program is readable by others. Failure to follow these standards shall result in deductions on your assignment.
- You must provide the results of running Javadoc on your submission via Canvas upload. Zip up your directory and upload that to Canvas under the assignment. Please be sure you follow the minimum requirements for documenting your code, but more is always better when it comes to documentation!
- We may be using your program again in future assignments, so design it for further adaptations and document it well. Also comment your program so that its logic could be explained to your grandparents (for example).
- Copying all or part of someone else's program is considered plagiarism, which will be dealt with according to the policy in the course syllabus.
- When in doubt, ask a TA for help, as opposed to assuming stuff

What to Submit

Access Gradescope from Canvas and upload a .zip of only the file *Translator.java* file we provided you that you have completed to convert from English to Pig Latin. Gradescope will run your implementation on 4 or 5 test phrases. Keep in mind that we will test a large number of other phrases as well, so make sure your code is robust enough to handle situations that may not be covered by the sample phrases we give you and the results shown. Also upload the contents of your Javadoc-generated "man page" for your solution to Canvas by the deadline.