# Purpose

The purpose of this exercise if to get experience with Visual Studio and features of C# discussed to date. The context is working with strings and translating English language text into Pig Latin.

# Specifications

## Create a Static Class (Tools)

The **Tools** class should be **Static**: all its methods and properties should be **static**. While you may include as many methods as you think helpful, it should include at least those specified here.

1. **Tokenize (String strIn, String strDelims)**. This method should parse **strIn** and return a **List<String>** containing all the “**words**” and **non-whitespace delimiters** (punctuation and so forth) from **strDelims**. Whitespace delimiters are used and they are important in finding the “words”, but they need not be retained in the returned **List <String>**. For example, if **strIn** = **“Angie, Tom, and Will were at the party.”** and if **strDelims = “ ,.?!/;:”** (note that the first character in **strDelims** is a space), then the returned **List<String>** should contain the following:

**Angie**

**,**

**Tom**

**,**

**and**

**Will**

**were**

**at**

**the**

**party**

**.**

1. **Format (List <String>)**. This method should format the strings in the list for display as one or more “sentences” that may wrap from one line to the next if it stretches more than 80 columns. In this case, it should break between words and/or punctuation as shown in this paragraph. Add whitespace (blank spaces, new line characters, etc.) as needed to improve the appearance of the result. This method should return the resulting formatted string.

## Create a PigLatin Class

This class is designed to translate from English to Pig Latin.

1. One of the constructor overloads for this class accepts a string parameter containing the English text to be translated. This parameter should be saved as a **property** named **Original**. It should then use **Tools.Tokenize** to extract the “words” and punctuation from **Original**, and pass the resulting **List<String>** into the **Translate** method.
2. **Translate (List <String> tokens)** inputs the **List<String>** from **Tools.Tokenize** and converts each “**word**” in the list to its **Pig** **Latin** equivalent. This method returns a **List<String>** (saved in the **Results property**) containing the **Pig** **Latin** **translations** of the contents of the **tokens** **List**. A partial list of the results of the example above follows

**Angielay**

**,**

**omtay**

**,**

**andlay**

**. . .**

1. A rough description of the process for translating an **English** **word** into **Pig** **Latin** follows.
2. A ***word*** is translated into ***pig*** ***latin*** as follows:
   1. If a “word” has no vowels, do not try to “translate” it. Leave it as is.
   2. Any combination of non-delimiters containing at least one vowel is considered to be a “word” for the sake of this discussion.
   3. If the word begins with a vowel, append “**lay**” to the end of the word.
   4. If the word begins with any other character, take all characters before the first vowel, remove them from the front of the word, append them to the end of the word, and append “**ay**” after that.
   5. Words such as ***cry****,* ***try****,* ***my****,* ***crypt***, and ***gym*** show that the letter *“****y****”* must be considered as a vowel for this translation’s purpose since it is the only vowel in some legitimate words. While it is permissible to try to determine when ***y*** is not a vowel (for example, “***y***” is not considered a vowel in the word “***you***”), that is not required. That is, it is permissible to assume ***‘y’*** is always a vowel, though you are not required to do so.
   6. Examples of the results of some translations include:
      1. **try** -> ***ytray***
      2. **apple** -> ***applelay***
      3. **string** -> ***ingstray***
      4. **egyptian** -> ***egyptianlay***
3. **ToString ( )** – this overload of **Object**.**ToString** will use the **Tools**.**Format** method to format the **List** returned by **Translate** into a complete formatted string of **Pig** **Latin** text that is ready to display.

## Create an Exercise1Driver Class

The driver class should be a menu-driven class that allows the user to input a string from a text file which may be quite a few lines long, tokenize the string, display the tokens, translate the tokens into Pig Latin, and display the result using ToString from the PigLatin class to format the results. The user should be able input, translate, and display strings from other text files, but it is not necessary for the program to work with more than one text file at a time.

The driver class should be broken into as many methods as necessary so that no method is overly long or complicated. The Main method should primarily call on other driver class methods to do the majority of its work.

# Deliverables

Once you have completed, documented, and sufficiently tested this project, zip the entire solution (after deleting the Obj and Bin folders), and name the zipped file as indicated in the Course Facts document. Your zipped document MUST include the entire solution (except for Obj and Bin) including all of the folders, files, and directory structure of you solution. Submit to your instructor and your main TA (Tomas Hill).