Page Composition

The aim of the C# program was to read in a string of words from an 'xml' file, and create a page of text which outputs on a 'txt' file. The input followed set conditions, to subsequently be processed, creating the desired output file. The initial conditions was firstly to only consist of lower case a-z characters. Secondly, each word must contain a vowel and words larger than 3 characters must contain at least 2. Finally, the vowels must be in alphabetical order; any words that do not meet these conditions are discarded by the program. The process the input goes through depends on the format set in the 'xml' file. There are 5 different formats; which process the text differently and further vary depending on the given parameters.

The first and most important format is 'Fill', as all formats follow the same process only to include their own process to comply with their conditions and rules. 'Fill' takes in the text and 'Wrap' parameter. The wrap is an integer value that is above zero, and represents the limit of characters that are on a single line. The program will go through the given string of words, and add up the length of each word until the length surpasses the value of the wrap. In which case the current word is amended to have a "\n" new line feed before printing the word again. The process is repeated until there are no input words left, and the processed text is printed to the output 'txt' file.

The second format is 'FillSoft'; this format takes in a second integer value, 'WrapSoft'. This parameter value is less or equal to the 'Wrap' value, and restricts a line containing 2 or more words having a length greater than itself.

The third format is 'FillAdjust'; using the 'Wrap' parameter this format has an extra method at the end of a line, which is triggered should the line consist of 2 or more words. The method adds an additional 'Space' character after each existing space on the line based on the difference between the length of the line and the 'Wrap' value. The output text will have every line equally spaced, in conditions that have an unequal amount of spaces result in the word with the most amount of vowels containing the extra spaces.

The fourth format is 'LineMoment'; using another integer parameter 'ColumnMoment'. This format goes through a lengthy formula, to calculate value of characters in a word depending on their position in the line, their position in the alphabet and the value given in 'ColumnMoment'. The line will then be formatted by repositioning the words based on the sum of character values.

The final format is 'FillSet'; unlike 'Fill' the order of the words on the page do not correspond to the order of the words in the input. The output order will produce the least amount of lines on a page, by going through a length method.