**Pseudocode**

**Methods may use any of the following variables from the Object:**

*input* – user input as String

*inputAsArray –* user input as character array

*inputForDisplay* – user input as String formatted for display in the console

*numberOfSpaces –* number of spaces in the input

*charsIncludingSpaces –* total number of characters in the input

*charsExcludingSpaces –* number of characters excluding spaces in the input

*numberOfWords –* number of words in the input

*characterFrequency –* number of character occurrences in the input as an array

*highestCount –* highest value contained in *characterFrequency*

*numberOfRecognisedCharacters –* total number of characters that matched the static *characterArray*

*numberOfUnrecognisedCharacters –* total number of characters that did not match the static *characterArray*

*relativeFrequency –* the frequency of character occurrences in relation to the number of characters (excluding spaces) in the input as a String array rounded to a max of 3 decimal places

*wordLengths* – the list of each words length (letters only)

*wordLengthDisplay* – the wordLengths list formatted for display in the console

*wordLengthFrequencies* – the frequency of words of a specific length (1, 2, 3, 4, 5, 6, 7, 8+)

*wordLengthFrequenciesDisplay* – the *wordLengthFrequencies list formatted for display in the console*

*characterArray –* a static array of the characters that will be recognised and analysed by the program

**Analysis Methods - pseudocode**

**NumberOfSpaces** *takes in***inputAsArray**

\**Method to count the number of spaces in the input\**

Initialise “count” to zero

For each character in the input, do:

If char is a space, do:

Add one to “count”

End loop

Return “count”

**NumberOfWords** *takes in* **numberOfSpaces**

*\*Method to calculate the number of words in the input\**

If “numberOfSpaces” is equal to zero, do:

Return zero

Else, do:

Return numberOfSpaces + 1

**CountCharacters** *takes in* **inputAsArray** *and* **a character**

*\*Method to count the number of character occurrences for a given character\**

Initialise “count” to zero

For each character in the input, do:

If char is same as given character, do:

Add one to “count”

End loop

Return “count”

**CharsIncludingSpaces** *takes in* **inputAsArray**

*\*Method returns total number of characters in input\**

Return length of “inputAsArray”

**CharsExcludingSpaces** *takes in* **charsIncludingSpaces** *and* **numberOfSpaces**

*\*Method returns total number of characters excluding spaces\**

Return “charsIncludingSpaces” minus “numberOfSpaces”

**CharacterFrequency** *takes in* **inputAsArray** *and* **characterArray**

*\*Method returns an array of the character occurrences for a static array of characters – uses the above character occurrences method\**

Initialise “array” as new int array of size fifty-one

For each character in the alphabet list, do:

Set the current “array” value to the return value of Count characters method

End loop

Return “array”

**HighestCount** *takes in* **characterFrequency**

*\*Method to calculate the highest value in Character frequency array – used for bar chart\**

Initialise “count” to zero

For each value in character frequency, do:

If the current character frequency value is greater than “count”, do:

“count” equals current character frequency value

End loop

Return “count”

}

**NumberOfRecognisedCharacters** *takes in* **characterFrequency**

*\*Method to add the values in Character frequency array\**

Initialise “count” to zero

For each value in character frequency, do:

Add current value of character frequency to “count”

End loop

Return “count”

**NumberOfUnrecognisedCharacters** *takes in* **numberOfRecognisedCharacters** *and* **charsExcludingSpaces**

*\*Method to subtract the total values of Character frequency from total number of characters without spaces\**

Return “charsExcludingSpaces” minus “numberOfRecognisedCharacters”

**RelativeFrequency** *takes in* **characterFrequency** *and* **charsExcludingSpaces**

*\*Method returns the frequency of character occurrences in relation to the number of characters (excluding spaces) in the input as a String array rounded to a max of 3 decimal places\**

Initialise “relativeFrequency” as new double array of size fifty-one

Initialise “doubleCharacterFrequency” as new double array of size fifty-one

Initialise “formattedRelativeFrequency” as new string array of size fifty-one

For each value in “characterFrequency”, do:

Set current “doubleCharacterFrequency” value to current “characterFrequency” value

End loop

For each value in “doubleCharacterFrequency”, do:

If “charsExcludingSpaces” is greater than zero, do:

Set current “relativeFrequency” value to current “doubleCharacterFrequency” divided by “charsExcludingSpaces”

Else, do:

Set current “relativeFrequency” value to zero

End loop

Initiate new “decimal format” to three decimal places

Set rounding mode to ceiling

For each value in “doubleCharacterFrequency”, do:

If “charsExcludingSpaces” is greater than zero, do:

Set current “formattedRelativeFrequency” to decimal formatted current “relativeFrequency” value

Else, do:

Set current “formattedRelativeFrequency” to zero

End loop

Return “formattedRelativeFrequency”

**WordLengths** *takes in* **inputAsArray** *and* **characterArray**

*\*Method returns an ArrayList of the lengths of each word in the string. Method only counts letters of the alphabet\**

Initialise “wordLengths” as a new ArrayList of type Integer

Initialise “count” to zero

For each character in “inputAsArray”, do:

If character is not a space, do:

For each letter of alphabet, do:

If letter is equal to current character, do:

Add one to “count”

End loop

If character is a space **AND** “count” is not zero, do:

Add “count” to “wordLengths”

Reset “count” to zero

End loop

Add “count” for final word to “wordLengths”

Return “wordLengths”

**WordLengthFrequencies** *takes in* **wordLengths**

\**Method returns an ArrayList of the frequency of words of specific lengths (1, 2, 3, 4, 5, 6, 7, and 8+ letters)\**

Initialise “wordLengthFrequencies” as a new ArrayList of type Integer

Initialise “count” to zero

For each increment from one to eight (inclusive), do:

For each value in “wordLengths”, do:

If increment is equal to eight, do:

If value is greater than or equal to increment, do:

Add one to “count”

Else

If value is equal to increment, do:

Add one to “count”

End loop

Add “count” to “wordLengthFrequencies”

Reset “count” to zero

End loop

Return “wordLengthFrequencies”

**Display Methods – pseudocode**

**InputForDisplay** *takes in* **inputAsArray**

*\*Method returns a string formatted with new lines so that the text will display well in the console\**

Initialise “inputForDisplay” as an empty string

Initialise “count” to zero

For each character in “inputAsArray”, do:

Add current “inputAsArray” character to “inputForDisplay”

If current “inputAsArray” character is a space, do:

Add one to “count”

If current “inputAsArray” character is a space **and** count is divisible by twenty, do:

Add a new line to “inputForDisplay”

End loop

Return “inputForDisplay”

**WordLengthDisplay** *takes in* **wordLengths**

*\*Method returns a string formatted to display the length of each word on a new line with the appropriate word number e.g. Word 1 is 3 letters long\**

Initialise “wordLengthDisplay” as an empty string

For each value in wordLengths, do:

Add “Word” plus loop increment plus “is” plus value plus “letters long.” To wordLengthDisplay

End loop

Return “wordLengthDisplay”

**WordLengthFrequenciesDisplay** *takes in* **wordLengthFrequencies**

*\*Method returns a string formatted to display the frequency of 1, 2, 3, 4, 5, 6, 7, or 8+ letter words\**

Initialise “wordLengthFrequenciesDisplay” as an empty string

For each increment between one and eight (inclusive), do:

If increment is less than eight, do:

Add “There are” plus value of “wordLengthFrequencies” for increment plus “words containing” plus increment plus “letters.” To wordLengthFrequenciesDisplay

Else, do:

Add “There are” plus value of “wordLengthFrequencies” for increment plus “words containing 8 or more letters.” To wordLengthFrequenciesDisplay

End loop

Return “wordLengthFrequenciesDisplay”

**LongestWordForDisplay** *takes in* **longestWord**

*\*Method returns a String to display the longest word/s\**

Initialise “longestWordForDisplay” as an empty string

Initialise “longestWordList” as an empty string

For each word in “longestWord”, do:

If the word is the 1st in the list, do:

Add the word to “longestWordList”

Else, do:

Add the word to “longestWordList” preceded by “, “

End loop

If there is only one word in “longestWord”, do:

Add “The longest word is “ + “longestWordList” to “longestWordForDisplay”

Else, do:

Add “The longest words are : “ + “longestWordList” to “longestWordForDisplay”

Add “ at “ + length of “longestWord”index(0) + “ letters long.”

Return “longestWordForDisplay”

**ToString \*\*\*\*Removed\*\*\*\***

*\*Method overload to display the object attributes well in the console\**

Return the following text:

New line

The text you are analysing is:

New line

“inputForDisplay”

New line

Number of words: “numberOfWords”

New line

Number of spaces: "numberOfSpaces”

New line

Total characters including spaces: "charsIncludingSpaces”

New line

Total characters excluding spaces: "charsExcludingSpaces”

New line

Total number of recognised characters: "numberOfRecognisedCharacters”

New line

Total number of unrecognised characters: "numberOfUnrecognisedCharacters"

New line

“wordLengthFrequenciesDisplay”

New line

**BasicAnalysisForDisplay** *takes in* **various attributes**

*\*Method puts various text and attributes into an ArrayList to be used for display\**

Initialise “basicAnalysisForDisplay” as an empty ArrayList of type String

Add each line of text as per ToString (above)

Return “basicAnalysisForDisplay”

**PrintBasicAnalysis**

*\*Method loops through each item in ArrayList and prints to console\**

For each item in “basicAnalysisForDisplay”, do:

Print item to console

End loop

**DisplayFrequencies** *takes in* **characterArray** *and* **characterFrequency** *and* **relativeFrequency**

*\*Method to display the characterArray, characterFrequency, and relativeFrequency arrays as a table\**

*Decided not to write pseudocode for this method, instead I have detailed the idea for display below and the code will be created using trial and error*

Split the arrays into two: A-Z in the top table, the remainder in the bottom table so that it fits in the console

Create reusable strings to use for the table outline

Look at formatting to set a static width for the columns (<http://www.java2s.com/Code/Java/Data-Type/Formatstringsintotable.htm>)

Use for loops to print out the arrays

**BarChartFrequencies** takes in **characterArray** *and* **characterFrequency** *and* **highestCount**

*\*Method to display the characterArray and characterFrequency as a bar chart\**

If “highestCount” is less than five, do:

Set “highestCount” to five

Print chart title

For each count in “highestCount”, do:

Print count and appropriate space (use if/else for spacing)

For each value in “characterFrequency”, do:

If value does not equal zero **and** value is greater than or equal to count, do:

Print ‘X’ and a y-axis divider

Else, do:

Print a space and a y-axis divider

End loop

End loop

Print the x-axis of the chart

Print a space and a y-axis divider

For each character in “characterArray”, do:

Print character

End loop