Pseudo-code

String input from class constructor

global int total punctuation

global int zero count

**Method: void analyzeText**

**Stage 1:**

IF input contains one or more a-z characters

run text analysis methods

print "output ended"

ELSE

print "Empty or invalid input"

END-IF

**Stage 2:**

IF input contains one or more a-z characters

//run text analysis methods

run method letterFrequency

run method printWordAnalysis

run method printSentenceAnalysis

print "output ended"

ELSE

print "Empty or invalid input"

END-IF

**Stage 3:**

//IF input contains one or more a-z characters (run method inputValid)

IF run method inputValid = true

//run text analysis methods

run method letterFrequency

run method printWordAnalysis

run method printSentenceAnalysis

print "output ended"

ELSE

print "Empty or invalid input"

END-IF

**Method: boolean inputValid**

**Stage 1:**

trim outer spaces from input string

IF input is empty or null

return false

ELSE

count a-z letters within input

total up a-z letter counts

reset used global ints

IF total equals 0

return false

ELSE

return true

END-IF

END-IF

**Stage 2:**

trim outer spaces from input string

IF input is empty or null

return false

ELSE

//count a-z letters within input

new count array = run method countChars

//total up a-z letter counts

FOR each int in count array

total plus current int in array

END-FOR

reset used global ints

IF total equals 0

return false

ELSE

return true

END-IF

END-IF

**Stage 3:**

trim outer spaces from input string

IF input is empty or null

return false

ELSE

//count a-z letters within input

new count array = run method countChars

//total up a-z letter counts

FOR each int in count array

total plus current int in array

END-FOR

reset used global ints

IF total equals 0

return false

ELSE

return true

END-IF

END-IF

**Method: void letterFrequency**

**Stage 1:**

count a-z letters within input

convert counts to frequencies

print frequencies

print most used letter

print letter frequency bar chart

reset used global ints

**Stage 2:**

//count a-z letters within input

array letter count = run method countChars

//convert counts to frequencies

array frequencies = run method countToFreq

//print frequencies

run method printFreqs

//print most used letter

find most used char

print most used char if not blank

//print letter frequency bar chart

run method charBarChart

reset used global ints

**Stage 3:**

//count a-z letters within input

array letter count = run method countChars

//convert counts to frequencies

//array frequencies = run method countToFreq

array frequencies = run method countToFreq(array letter count)

//print frequencies

//run method printFreqs

run method printFreqs(array letter count, array frequencies)

//print most used letter

//find most used char

run method mostUsedChar(array letter count)

//print most used char if not blank

IF returned most used char not blank

print most used char

END-IF

//print letter frequency bar chart

run method charBarChart

reset used global ints

**Method: int[] countChars**

**Stage 1:**

FOR each character in input

IF current character is letter

add one to this letters count

ELSE-IF current character is punctuation

add one to punctuation count

END-IF

END-FOR

RETURN array count

**Stage 2:**

//FOR each character in input

make input only lowercase

convert input to character array

FOR each character in array

IF current character is letter

//add one to this letters count

add one to this letter position in letter count array

ELSE-IF current character is punctuation

add one to punctuation count

END-IF

END-FOR

RETURN array count

**Stage 3:**

//FOR each character in input

make input only lowercase

convert input to character array

initialise new 26 length int array letter count (a counter for each letter)

FOR each character in array

IF current character is letter

//add one to this letters count

add one to this letter position in letter count array

ELSE-IF current character is punctuation

add one to punctuation count

END-IF

END-FOR

RETURN array count

**Method: double[] countToFreq (int[])**

**Stage 1:**

find total valid characters

FOR each letter count above 0

letter frequency = letter count / total valid characters

END-FOR

RETURN frequencies

**Stage 2:**

//find total valid characters

FOR each letter count

add current letter count to total valid characters

END-FOR

//FOR each letter count above 0

FOR each letter count

IF current letter count > 0

letter frequency = letter count / total valid characters

END-IF

END-FOR

RETURN frequencies

**Stage 3:**

//find total valid characters

FOR each letter count

add current letter count to total valid characters

END-FOR

//FOR each letter count above 0

FOR each letter count

IF current letter count > 0

letter frequency = letter count / total valid characters

IF frequency < 0.01 AND frequency > 0

round letter frequency to 3 decimal places

ELSE

round letter frequency to 2 decimal places

END-IF

END-IF

END-FOR

RETURN frequencies

**Method: void printFreqs (int[], double[])**

**Stage 1:**

FOR each letter in alphabet where letter count > 0

print frequency AND count

END-FOR

print amount of omitted letters (zeroCount)

print amount of punctuation characters (totalPunctuation)

**Stage 2:**

//FOR each letter in alphabet where letter count > 0

FOR each letter in alphabet

IF letter count > 0

print frequency AND count

END-IF

END-FOR

print amount of omitted letters (zeroCount)

print amount of punctuation characters (totalPunctuation)

**Method: char mostUsedChar (int[])**

**Stage 1:**

Loop through letter count array and find most used char

**Stage 2:**

//Loop through letter count array and find most used char

FOR each char in letter count

IF current letter count > max letter count

biggest letter count = current letter count

most used letter = current letter

END-IF

END-FOR

RETURN most used char

**Method: void charBarChart (double[])**

**Stage 1:**

find the frequency with the most digits

print headings: "Character" AND "Frequency" AND "' \*' = 1%"

print separator line("===="etc.)

FOR each frequency > 0

print relevant letter

print frequency

print bar chart line("\*\*\*\*"etc.)

END-FOR

**Stage 2:**

//find the frequency with the most digits

FOR each frequency

IF digits in current frequency > digits in previous max digits

max digits = current digits

END-IF

IF current frequency > max frequency

max frequency = current frequency

END-IF

END-FOR

print headings: "Character" AND "Frequency" AND "' \*' = 1%"

//print separator line("===="etc.)

FOR 0 to max frequency \* 100 + 30

print '='

END-FOR

FOR each frequency > 0

print relevant letter

print frequency

print bar chart line("\*\*\*\*"etc.)

END-FOR

**Stage 3:**

//find the frequency with the most digits

FOR each frequency

IF digits in current frequency > digits in max digits

max digits = current digits

END-IF

IF current frequency > max frequency

max frequency = current frequency

END-IF

END-FOR

//print headings: "Character" AND "Frequency" AND "' \*' = 1%"

print " Character |"

run method spaces(1, max digits)

print " Frequency | '\*' = 1%"

//print separator line("===="etc.)

FOR 0 to max frequency \* 100 + 30

print '='

END-FOR

FOR each frequency > 0

print relevant letter with padding on left side

print "|" with padding on right side

run method spaces(frequency, max digits)

print frequency with padding on right side

print "|" with padding on right side

//print bar chart line("\*\*\*\*"etc.)

IF frequency > 0.01

FOR 0 to frequency \* 100

print '\*'

END-FOR

END-IF

END-FOR

**Method: void spaces (double, double)**

**Stage 1:**

number of spaces = digits in input 2 - digits in input 1

FOR number of spaces

print ' '

END-FOR

**Method: void printWordAnalysis ()**

**Stage 1:**

run method wordLengths

run method sortIntArray(array lengths)

run method removeDuplicates(array sortedLengths)

run method countOccurences(array lengthsNoDuplicates, array sortedLengths)

run method modeLength(array wordLengthOccurences, array lengthsNoDuplicates)

print "Mean word length:"

print mean word length

print "Median word length: "

print median word length

print "Mode word length"

print mode word length

run method sortArrBViaArrA(array lengthsNoDuplicates, array wordLengthOccurances)

run method toBarChart(sortedWordLengths, "Word Length", 4)

**Stage 2:**

//run method wordLengths

array lengths = run method wordLengths

//run method sortIntArray(array lengths)

array sortedLengths = run method sortIntArray(array lengths)

//run method removeDuplicates(array sortedLengths)

array lengthsNoDuplicates = run method removeDuplicates(array sortedLengths)

//run method countOccurences(array lengthsNoDuplicates, array sortedLengths)

array wordLengthOccurences = run method countOccurences(array lengthNoDuplicates, array sortedLengths)

//run method modeLength(array wordLengthOccurences, array lengthsNoDuplicates)

mode word length = run method modeLength(array wordLengthOccurences, array lengthsNoDuplicates)

print "Mean word length:"

//print mean word length

run method meanLength(array lengths)

print mean word length

print "Median word length: "

//print median word length

run method medianLength(array sortedLengths)

print median word length

print "Mode word length"

print mode word length

multi-array sortedWordLengths = run method sortArrBViaArrA(array lengthsNoDuplicates, array wordLengthOccurances)

run method toBarChart(multi-array sortedWordLengths, "Word Length", 4)

**Method: void printSentenceAnalysis()**

**Stage 1:**

run method sentenceLengths

run method sortIntArray(array lengths)

run method removeDuplicates(array sortedLengths)

run method countOccurences(array lengthsNoDuplicates, array sortedLengths)

run method modeLength(array sentenceLengthOccurences, array lengthsNoDuplicates)

print "Mean sentence length:"

print mean sentence length

print "Median sentence length: "

print median sentence length

print "Mode sentence length"

print mode sentence length

run method sortArrBViaArrA(array lengthsNoDuplicates, array sentenceLengthOccurences)

run method toBarChart(sortedSentenceLengths, "Sentence Length", 2)

**Stage 2:**

//run method sentenceLengths

array lengths = run method sentenceLengths

//run method sortIntArray(array lengths)

array sortedLengths = run method sortIntArray(array lengths)

//run method removeDuplicates(array sortedLengths)

array lengthsNoDuplicates = run method removeDuplicates(array sortedLengths)

//run method countOccurences(array lengthsNoDuplicates, array sortedLengths)

array sentenceLengthOccurences = run method countOccurences(array lengthNoDuplicates, array sortedLengths)

//run method modeLength(array sentenceLengthOccurences, array lengthsNoDuplicates)

mode sentence length = run method modeLength(array sentenceLengthOccurences, array lengthsNoDuplicates)

print "Mean sentence length:"

//print mean sentence length

run method meanLength(array lengths)

print mean sentence length

print "Median sentence length: "

//print median sentence length

run method medianLength(array sortedLengths)

print median sentence length

print "Mode sentence length"

print mode sentence length

multi-array sortedSentenceLengths = run method sortArrBViaArrA(array lengthsNoDuplicates, array sentenceLengthOccurences)

run method toBarChart(multi-array sortedSentenceLengths, "Sentence Length", 2)

**Method: int[] wordLengths()**

**Stage 1:**

trim outer spaces from input and replace line breaks with spaces

FOR each character

IF current char is letter or space

add char to final string

ELSE-IF current char is full stop

add space to final string

END-IF

END-FOR

remove all multi spaces (make sure only one space between each word)

FOR each word in final string

add word length to array word lengths

END-FOR

RETURN array word lengths

**Stage 2:**

//trim outer spaces from input and replace line breaks with spaces

trim outer spaces from input

replace line breaks with space

make input lowercase

//FOR each character

array input chars = input to char array

FOR each char in array input chars

IF current char is letter or space

add char to final string

ELSE-IF current char is full stop

add space to final string

END-IF

END-FOR

remove all multi spaces (make sure only one space between each word)

//FOR each word in final string

array words = final string split via ' '

array lengths = size of array words

FOR each word in array words

add word length to array word lengths

END-FOR

RETURN array word lengths

**Method: int[] sentenceLengths()**

**Stage 1:**

trim outer spaces from input and replace line breaks with spaces

FOR each character

IF current char is '!' or '?'

replace char with '.'

END-IF

END-FOR

remove all multi spaces (make sure only one space between each word)

FOR each sentence in final string

run method sentenceLeng(current sentence)

add sentence length to array sentence lengths

END-FOR

RETURN array sentence lengths

**Stage 2:**

//trim outer spaces from input and replace line breaks with spaces

trim outer spaces from input

replace line breaks with space

make input lowercase

//FOR each character

array input chars = input to char array

FOR each char in array input chars

IF current char is '!' or '?'

replace char with '.'

END-IF

END-FOR

remove all multi spaces (make sure only one space between each word)

//FOR each sentence in final string

array sentences = final string split via '.'

array lengths = size of array words

FOR each sentence in array sentences

run method sentenceLeng(current sentence)

add sentence length to array sentence lengths

END-FOR

RETURN array word lengths

**Method: int sentenceLeng(String)**

**Stage 1:**

FOR each character in input

IF current char is a letter OR is a space

add current char to final string

END-IF

END-FOR

remove all multi spaces from final string (only one space between each word)

array words = final string split via ' '

RETURN size of array words

**Stage 2:**

//FOR each character in input

convert input to lower-case

convert input to char array

FOR each character in char array

IF current char is a letter OR is a space

add current char to final string

END-IF

END-FOR

remove all multi spaces from final string (only one space between each word)

array words = final string split via ' '

RETURN size of array words

**Method: void toBarChart (int[][], String, int)**

**Stage 1:**

find total amount of words or sentences

find frequencies of each word or sentence

find the highest word or sentence length

find the frequency with the most digits and the highest frequency

print headings

print each row of table AND print bar-chart characters

**Stage 2:**

//find total amount of words or sentences

FOR each [] in multi-array input ([this][0] is length and [this][1] is count)

add [this][1] to total

END-FOR

//find frequencies of each word or sentence

array frequencies = length of multi-array input

FOR each [] in multi-array input ([this][0] is length and [this][1] is count)

frequency = [this][1] / total

END-FOR

//find the highest word or sentence length

FOR each [] in multi-array input ([this][0] is length and [this][1] is count)

IF[this][0] > max length

max length = [this][0]

END-IF

END-FOR

//find the frequency with the most digits and the highest frequency

FOR each frequency in frequencies

IF digits in frequency > digits in max frequency

max digits = current frequency

END-IF

IF frequency > max frequency

max frequency = current frequency

END-IF

END-FOR

//print headings

print input padding AND input heading AND '|'

print input padding AND "Frequency"

print '|' AND "'\*' = 1%"

//print each row of table AND print bar-chart characters

FOR each [] in multi-array input AND frequency in frequencies ([this][0] is length and [this][1] is count)

print input padding

run method Spaces([this][0])

print [this][0]

print input padding AND '|'

run method Spaces(frequency, max length)

print frequency

print input padding AND '|'

print bar-chart for frequency

END-FOR

**Stage 3:**

//find total amount of words or sentences

FOR each [] in multi-array input ([this][0] is length and [this][1] is count)

add [this][1] to total

END-FOR

//find frequencies of each word or sentence

array frequencies = length of multi-array input

FOR each [] in multi-array input ([this][0] is length and [this][1] is count)

frequency = [this][1] / total

END-FOR

//find the highest word or sentence length

FOR each [] in multi-array input ([this][0] is length and [this][1] is count)

IF[this][0] > max length

max length = [this][0]

END-IF

END-FOR

//find the frequency with the most digits and the highest frequency

FOR each frequency in frequencies

IF digits in frequency > digits in max frequency

max digits = current frequency

END-IF

IF frequency > max frequency

max frequency = current frequency

END-IF

END-FOR

//print headings

print input padding AND input heading AND '|'

print input padding AND "Frequency"

print '|' AND "'\*' = 1%"

//print each row of table AND print bar-chart characters

//FOR each [] in multi-array input AND frequency in frequencies ([this][0] is length and [this][1] is count)

FOR each [] in multi-array input with counter from 0 ([this][0] is length and [this][1] is count)

print input padding

run method Spaces([this][0])

print [this][0]

print input padding AND '|'

run method Spaces(frequencies[counter], max length)

//print frequency

print frequencies[counter]

print input padding AND '|'

//print bar-chart for frequency

FOR 0 to frequency\*100

print '\*'

END-FOR

END-FOR

**Method: int[] removeDuplicates(int[])**

**Stage 1:**

FOR each number in array input

add number to set

END-FOR

output = convert set to array

RETURN array output

**Stage 2:**

//FOR each number in array input

create new set

FOR each number in array input

add number to set

END-FOR

//output = convert set to array

array output = size of set

FOR each number in set

add current number to array output

RETURN array output

**Method: int[] countOccurences(int[], int[])**

**Stage 1:**

FOR each number in array input 1

count how many occurrences of current number in array input 2

END-FOR

RETURN array output

**Stage 2:**

array output = array input 1 size

FOR each number in array input 1

//count how many occurrences of current number in array input 2

FOR each number in array input 2 with counter

IF current array input 2 number = current array input 1 number

add 1 to array output[counter]

END-IF

END-FOR

END-FOR

RETURN array output

**Method: int[] sortIntArray(int[])**

**Stage 1:**

put array input into copy array output

DO

organize array into numerical order

WHILE has swapped at least once

RETURN output

**Stage 2:**

put array input into copy array output

DO

//organize array into numerical order

FOR 1 to size of array output

IF previous number > current number

swap numbers

END-IF

END-FOR

WHILE has swapped at least once

RETURN output

**Method: int[][] sortArrBViaArrA(int[], int[])**

**Stage 1:**

multi-array output = [size of array input1][2]

insert array input1 into multi-array input [][0]

insert array input2 into multi-array input [][1]

sort multi-array output via [][0]

RETURN output

**Stage 2:**

multi-array output = [size of array input1][2]

//insert array input1 into multi-array input [][0]

//insert array input2 into multi-array input [][1]

FOR 0 to size of array input 1 with counter

FOR 0 to size of array input 2

output[counter][1] = array input 2[counter]

END-FOR

output[counter][0] = array input 1[counter]

END-FOR

//sort multi-array output via [][0]

DO

//organize array into numerical order

FOR 1 to size of array output

IF previous number([this-1][0] > current number ([this][0])

swap numbers in both output[][0] and output[][1]

END-IF

END-FOR

WHILE has swapped at least once

RETURN output