Assignment04-Q1:

StatisticsCalculator
-intArray: Integer Array -average: Double -median: Double -sum: Integer
+StatisticsCalculator(listOfInt: Integer Array) -calculateSum() -sortArray(listOfInt: Integer Array) +calculateAverage(): Double +calculateMedian(): Double +updateArray(newArray: Integer Array) +getArray(): Integer Array

The intArray, average, median and sum variables will all be private variables in the class. intArray can be accessed by using the getArray() method, while the rest are printed out by the program. The constructor method StatisticsCalculator will accept an Integer Array, and set that equal to the instance variable intArray. calculateSum will calculate the sum of all the items in the array. calculateAverage will calculate the average using the sum and the length of the array. calculateMedian will calculate the median using the sum and the length of the array. updateArray will allow the user to change the array. getArray will return the current array.

Assignment04-Q6:

I did apply the principle of divide-and-conquer. I did this by ensuring that each method had its own purpose within the larger class, such as the wordLength and getWordLength methods, which determined the average word length and returned the average word length, respectively. I did this because divide and conquer allows me to divide the code up into bite sized chunks, allowing me to tackle smaller portions of the code as opposed to the whole thing at once. This ensures that code can be written in a team of engineers as well, with every engineer being responsible for one class or method. This ensures that products are built and maintained quickly.

Assignment04-Q7:

I applied the open-closed principle by ensuring my classes were open for extension but closed for modification. In Question 4, I extended the class I designed in Question 3, but I did not modify the class designed in Question 3 in any way. I simply added new methods. Similarly, in Question 5, I extended the classes from Question 3 and Question 4, and added new methods, while still using some methods from the superclasses. The benefit of the open-closed principle is that it allows you to repurpose your code and easily add new features without rewriting everything. This ensures scalability and allows many engineers to repurpose work from other engineers.