**Assignment04-Q1:**

| **StatisticsCalculator** |
| --- |
| -data: int[] |
| +StatisticsCalculator(data: int[])  throws IllegalArgumentException<br>+calculateAverage(): double  throws ArithmeticException<br>+calculateMedian(): double  throws ArithmeticException<br>+updateArray(newData: int[]): void  throws IllegalArgumentException<br>+getArray() : int[] |

The StatisticsCalculator class is performs statistical operations, including calculating the average and median of an array of integers. This class handles invalid inputs through exception handling and provides meaningful error messages. It also contains methods to update and access its array of data.

The constructor for the class first validates the given int array "data" to ensure it is not null or empty, potentially throwing an IllegalArgumentException, then initializes the private int[[] data with a clone of the provided integer array.

The private int[] data stores the set of integers that the StatisticsCalculator operates on. It is private to prevent external modifications.

(+calculateAverage(): double): This public method calculates the average of the elements in "data", returning the average as a double formatted to two decimal places. If the array is empty, it throws an ArithmeticException.

(+calculateMedian(): double): This public method calculates the median of the elements in "data", returning the median as a double formatted to two decimal places. If the array is empty, it throws an ArithmeticException.

(+updateArray(newData: int[]): void): This method first validates that the "newData" array is not empty or null, potentially throwing an IllegalArgumentException, then sets the value of "data" to a clone of "newData" (cloning prevents external modifications).
(+getArray() : int[]): This method returns a copy of the "data" array, preventing direct modification from outside the class.

**Assignment04-Q6:**

The divide-and-conquer principle was applied in the StringCleaner, StringAnalyzer, and TextAnalyzer classes by dividing specific responsibilities among them. StringCleaner is responsible for preprocessing tasks like removing punctuation and extra spaces, which makes it reusable for

other classes needing clean strings. The StringAnalyzer class builds on this by focusing on analyzing cleaned string, with tasks like counting word frequency and palindrome checking. And the TextAnalyzer extends this further by adding specific text analysis features. While there is some overlap between the TextAnalyzer and StringAnalyzer, the TextAnalyzer adds new features, and so each class concentrates on its own layer of functionality. Each method in these classes also performs a distinct operation, ensuring that each function is handled independently. This all aligns with the divide-and-conquer principle by breaking down the code into isolated components that work together, benefiting the reusability and maintainability of the code.

**Assignment04-Q7:**

The open-closed principle is applied in questions 3, 4, and 5 by making each class open for extension but closed for modification. StringCleaner focuses on string preprocessing, with self-contained methods for tasks like removing punctuation and extra spaces. StringAnalyzer then reuses StringCleaner without altering its code, adding analysis functionalities like counting word frequency on top of the cleaned data. TextAnalyzer extends StringAnalyzer to add more advanced analysis features (like calculating average word length) without modifying the underlying logic of StringAnalyzer.

By using the open-closed principle, each class can get more functionality without changing existing code, which minimizes the risk of introducing bugs into the stable, working code, and so makes the system easier to maintain overall.