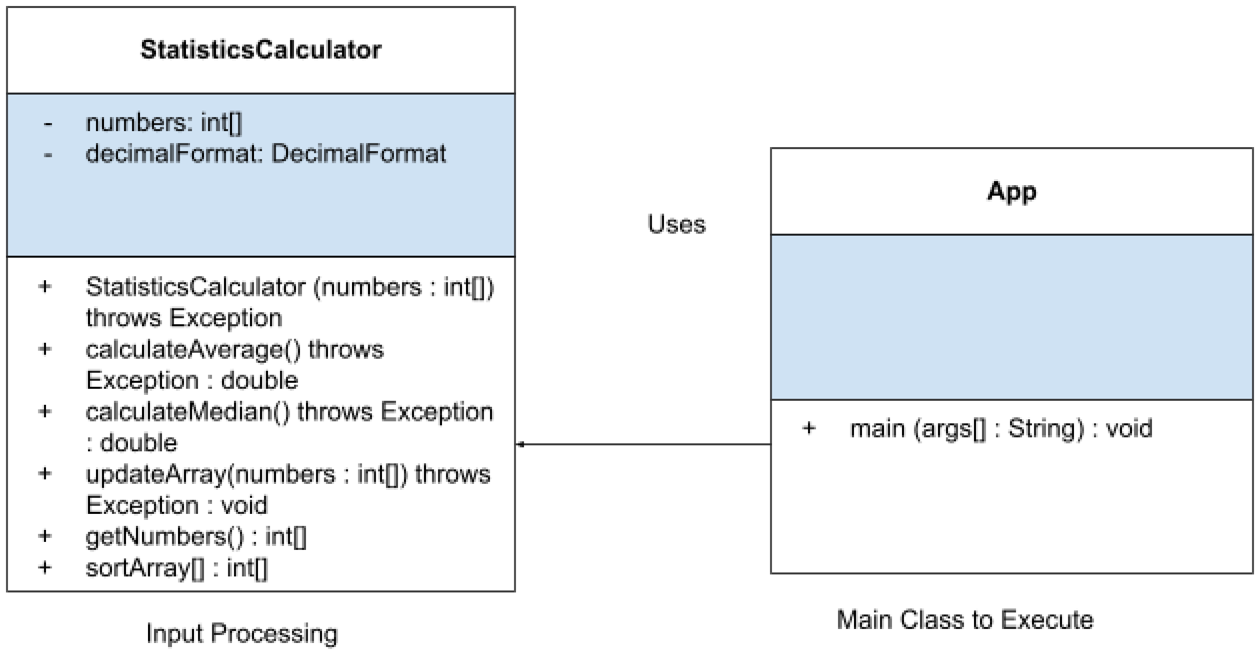
Assignment04-Q1:


StatisticsCalculator
- numbers: int[]
- decimalFormat: DecimalFormat
+ StatisticsCalculator (numbers : int[]) throws Exception
+ calculateAverage() throws Exception : double
+ calculateMedian() throws Exception : double
+ updateArray(numbers : int[]) throws Exception : void
+ getNumbers() : int[]
+ sortArray[] : int[]
Input Processing
Uses
App
+ main (args[] : String) : void
Main Class to Execute

App Class:
Handles user input and interacts with the StatisticsCalculator class, no instance variables

StatisticsCalculator Class:
Holds a private int array numbers so it is not accessible from outside, and a decimal format to format the output. The constructor StatisticsCalculator then takes an input int array from App and assigns it to numbers

sortArray() is a private helper method that is used within calculateMedian(). It returns the sorted int array since calculating median needs a sorted array

calculateAverage() gets the average by going through each number in the array, summing them and dividing them by the total items

calculateMedian() gets the median of the array (that has been sorted by sortArray()) by getting the middle number of the set. If it is odd in length then it is the exact middle, if it is even then average the 2 middle numbers to get it.

updateArray() takes a new input array and updates the array numbers within the class

getNumbers() returns the array to the user without editing/exposing the actual array whenever the function is called

These methods also throw exceptions if the given array is empty/has no elements, since the operations cannot be performed if there is nothing given

Assignment04-Q6:

In 3, 4, and 5, the divide-and-conquer principle was applied as follows:

In StringCleaner, The problem of string preprocessing was divided into three smaller tasks: removing punctuation, converting to lowercase, and removing extra spaces. Each task was implemented as a separate method (removePunctuation, convertToLowercase, removeExtraSpaces), making the code modular and easier to manage, before being combined into one function.

In StringAnalyzer, string analysis was divided into three smaller tasks: word frequency count, finding the longest word, and palindrome check. This immediately cleans the string using StringCleaner, and having the code modular and easier to extend, like StringCleaner

In TextAnalyzer,advanced string analysis was divided into additional tasks: calculating average word length and printing words in alphabetical order, while using helper methods to further distribute tasks (i.e removeDuplicates) . It also extends StringAnalyzer so it can use the methods inherited from it

Assignment04-Q7:

In 3, 4, and 5, the open-closed principle was applied as follows:

StringCleaner is designed to be self-contained and modular, with each task implemented as a separate method. This makes it easy to extend the class with new preprocessing tasks without modifying the existing methods.

StringAnalyzer was built with the open-closed principle in mind since it is a base for string analysis. It has its own methods but can easily be expanded upon (add more methods) without changing the ones that already work

This is displayed using TextAnalyzer, since it extends the StringAnalyzer class and inherits its methods, which it uses in the new methods it creates. It does not edit the code in StringAnalzyer, but builds upon it explicitly and can be built upon further

The benefits of applying open-closed is it allows for easy extensibility, since new functions can be added without breaking existing ones, and it makes it easier to maintain since we aren't making many changes. It also allows us to reuse code, further simplifying the programming for future testing/debugging.