Teststufen & -Arten

Aufgabe "Average", Modul 450

Die folgenden Aufgaben beziehen sich auf die Beispielanwendung Average, die in C# geschrieben ist. Der Code könnte wesentlich eleganter geschrieben sein, was für das nächste Thema Code-Verbesserungen relevant sein wird. Das Programm verarbeitet Textdateien, welche auf jeder Zeile eine natürliche Zahl enthalten, beispielsweise:

```
3
5
7
9
1
3
```

Aus diesen Zahlen werden die Statistiken Mittelwert (arithmetisches Mittel), Median und Modus berechnet (siehe Erklärungen).

Aufgabe 0: Setup und Programm ausführen

Du kannst das Verzeichnis bzw. die Solution mit Visual Studio Code oder mit Visual Studio öffnen. Stelle sicher, dass .NET 7.0 (siehe Download-Seite) bei dir installiert ist. Führe das Programm wie in der Datei `README.md` beschrieben aus, indem du für die Beispiel-Datei (`data/numbers.txt`) alle drei Statistiken (mean, median, mode) berechnest.

Aufgabe 1: Manuelle Tests

Führe die folgenden manuellen Tests aus, wozu du neue Testdaten erstellen musst:

1. Berechnung von mean.
    1. Mit mehreren gleichen Zahlen.
    2. Mit unterschiedlichen Zahlen.
2. Berechnung von median.
    1. Mit einer geraden Anzahl von Zahlen.
    2. Mit einer ungeraden Anzahl von Zahlen.
3. Berechnung von mode.
    1. Mit einem eindeutigen Ergebnis (eine Zahl kommt am häufigsten vor).
    2. Mit einem mehrdeutigen Ergebnis (mehrere Zahlen kommen am häufigsten vor).

Dokumentiere deine Erwartungen und Ergebnisse in der Datei `aufgabe-1.txt`. Lege deine Testdaten in Dateien mit Namen der Form `test-x-y.txt` ab, wobei `x` für die übergeordnete und `y` für die untergeordnete Aufgabe steht, z.B. `test-3-2.txt` für die Berechnung des mode mit mehrdeutigem Ergebnis.

Aufgabe 2: Unittests

Schreibe Unittests für die Klasse Statistics. (Das Testprojekt `Average.Test` ist bereits vorbereitet und enthält einen leeren Testfall `FileAccessTest.cs`) Schreibe für jede (statische) Methode von Statistics die folgenden Tests:

1. Mean
    1. Mean von einer leeren Liste.
    2. Mean von einer Liste mit einem einzigen Element.
    3. Mean von einer Liste mit mehreren Elementen.
2. Median
    1. Median von einer leeren Liste.
    2. Median von einer Liste mit einem einzigen Element.
    3. Median von einer Liste mit mehreren Elementen (gerade Anzahl).
    4. Median von einer Liste mit mehreren Elementen (ungerade Anzahl).
3. Mode
    1. Mode von einer leeren Liste.
    2. Mode von einer Liste mit einem einzigen Element.
    3. Mode von einer Liste, bei der jede Zahl nur einmal auftaucht.
    4. Mode von einer Liste, in der eine Zahl häufiger auftaucht als alle anderen.

Tipp: Du benötigst die Methoden `Assert.Equal()` und `Assert.Throws()` von xUnit.

Aufgabe 3: Integrationstest

Die Klasse Average verwendet die nun getesteten Methoden aus der Klasse `Statistics` sowie die Klasse `FileAccess`, um die Statistiken auf Basis einer Datei zu berechnen. Für die folgenden drei (statischen) Methoden von Average sollst du Integrationstests im Projekt `Average.Test` schreiben:

1. ComputeMeanOfFile: Mean
2. ComputeMedianOfFile: Median
3. ComputeModeOfFile: Mode

Da diese Methoden einen Dateipfad als Parameter erwarten, musst du zuerst (temporäre) Dateien mit den Testdaten erstellen, um die genannten Methoden testen zu können. Der folgende Code sollte dir dabei als Vorlage behilflich sein:

```
// Arrange
string tempFilePath = Path.Combine(Path.GetTempPath(), "numbers.txt");
StreamWriter writer = new System.IO.StreamWriter(tempFilePath);
writer.WriteLine("1");
writer.WriteLine("2");
writer.WriteLine("3");
writer.Close();
// Act
// TODO
// Assert
// TODO
// Cleanup
File.Delete(tempFilePath);
```

Da du die eigentlichen Berechnungen schon in der vorherigen Aufgabe ausführlich getestet hast, genügt ein Testfall pro Methode. Füge die Testfälle dem Repository hinzu.

Aufgabe 4: Systemtests (optional)

Beschreibe in der Datei `aufgabe-4.txt`, wie sich die manuellen Tests aus Aufgabe 1 automatisieren lassen. Welche Schritte müssen hierzu automatisiert werden? ....