Test Doubles

Aufgabe "Test Doubles", Modul 450

Die folgenden Aufgaben beziehen sich auf die Beispielanwendung **AverageTestDoubles**, welche eine modifizierte Version der in C# geschriebenen Beispielanwendung Average der letzten beiden Übungen ist.

1. Vorbereitung: Refactoring

Die Klasse `Average` verwendet die Klasse `FileAccess` zum Einlesen der Zahlen. Diese Abhängigkeit soll für die Unittests durch verschiedenartige Test `Doubles` ersetzt werden. Um dies zu vereinfachen, soll die Referenz auf die konkrete Klasse `FileAccess` durch eine Referenz auf ein entsprechendes Interface (`ToBeNamed`) ersetzt werden:

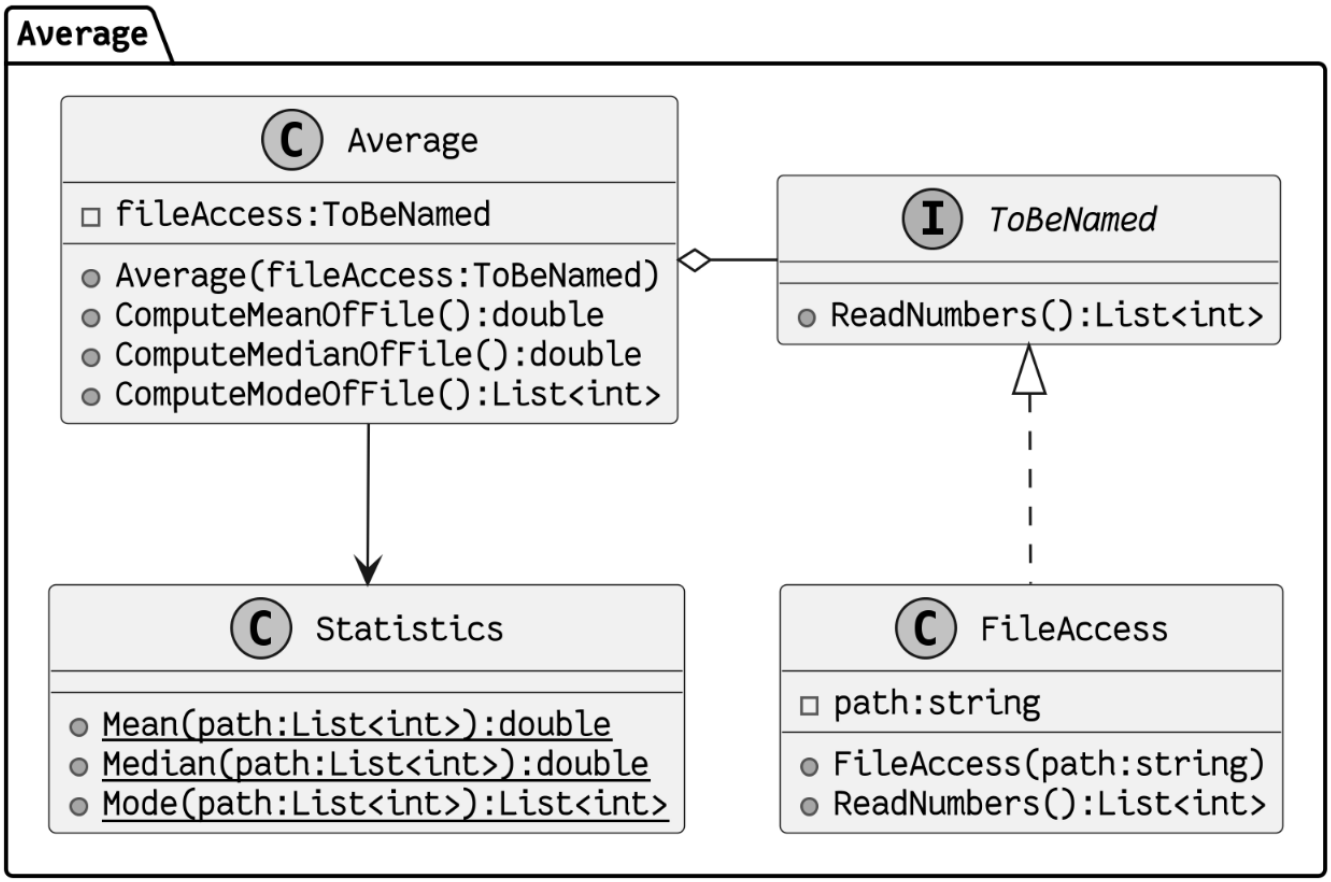

Abbildung 1: Entkopplung von `Average` und `FileAccess` über das (noch zu benennende) Interface

1.1 Vorgehen

1. Denk dir einen passenden Namen für das Interface `ToBeNamed` aus. Tipp: Die Schnittstelle macht nichts anderes als eine Reihe von Zahlen zu liefern.
2. Erstelle das Interface im `Average`-Projekt.
3. Passe die Deklaration der Klasse `FileAccess` so an, dass sie das neue Interface implementiert.
4. Passe die Klasse `Average` so an, dass du keine Referenzen mehr auf `FileAccess` hast, sondern nur noch auf das neue Interface. (Funktioniert das Demoprogramm bzw. die Klasse Program anschließend noch? Teste das!) Damit ist die Abhängigkeit `FileAccess` von der Klasse `Average` entkoppelt. Die Klasse Average kann nun mit einem Test Double, welches den Dateizugriff simuliert, als Unittest getestet werden.

2. Unittests mit Test Doubles

Da `Average` nur über eine einzige Abhängigkeit verfügt, die in jedem Fall aufgerufen wird, ist ein Test mit einem *Dummy* nicht sinnvoll. Die folgenden Test Doubles können aber zum Testen von Average verwendet werden:

5. Fake
6. Stub
7. Mock
8. Spy

In den folgenden Aufgaben sollen entsprechende Unittests mit Test Doubles entwickelt werden. Du kannst dabei eine beliebige Methode der Klasse `Average` (`ComputeMeanOfFile`, `ComputeMedianOfFile`, `ComputeModeOfFile`) testen. Die Test Doubles sollen das in der vorherigen Aufgabe definierte Interface implementieren.

2.1 Fake

Entwickle einen Fake, der nicht wie `FileAccess` auf das Dateisystem zugreift, sondern in einem *Dictionary* Pseudo-Dateipfade zu einer Zahlenreihe zuordnet, beispielsweise so:

Pfad (Key)	**Zahlen (Value)**
"/path/to/an/empty/file"	[]
"/path/to/some/other/file"	[1, 2, 3, 4, 5]
"C:\test-data\third-file.txt"	[7, 34, 2]

Es soll möglich sein, dem Fake-Objekt neue “Dateien” hinzuzufügen. (Die Testdaten können so im *Arrange*-Teil des Unittests definiert und hinzugefügt werden.)

Überlege dir, in welches Projekt diese Klasse gehört: `Average` oder `Average.Test`? Schreibe anschließend mindestens einen Unittest mit dieser Fake-Implementierung.

2.2 Stub

Entwickle einen *Stub*, der immer die gleichen hart-kodierten Werte zurückliefert. Entwickle einen weiteren Stub, welcher per Konstruktor eine Liste von Werten entgegennimmt, die beim Aufruf zurückgeliefert werden.

Schreibe anschließend zwei Testfälle für die gleiche Methode mit den gleichen Zahlen. Verwende für die beiden Testfälle je einen anderen Stub. Welche der beiden Implementierungen macht den Test besser lesbar? Ist der zweite, flexiblere Stub wirklich noch ein Stub, oder schon ein Fake?

2.3 Mock

Entwickle einen *Mock*, der wie der Stub in der vorherigen Aufgabe immer die gleichen Werte zurückliefert. Der Mock soll über einen internen Zähler verfügen, der bei null startet, und bei jedem Aufruf um eins erhöht wird.

Schreibe anschließend einen Testfall, der diese Mock-Implementierung verwendet. Teste dabei auch per Assertion auf Basis des Zählers, ob der Mock tatsächlich aufgerufen worden ist.

2.4 Spy

Schreibe einen *Spy*, der einen Wrapper um die Klasse `FileAccess` herum bildet. Da hier die originale Implementierung zum Einsatz kommt, ist das Aufsetzen einer Testumgebung (Zahlen in einer temporären Datei) wieder nötig. Der Spy soll sich mittels Zähler merken, wie oft `FileAccess.ReadNumbers()` aufgerufen worden ist. Außerdem sollen die zurückgelieferten Werte in einer Liste protokolliert werden.

Schreibe anschließend einen Testfall, der diesen Spy verwendet. Teste dabei auch per Assertion, ob erstens `FileAccess.ReadNumbers()` genau einmal aufgerufen worden ist, und zweitens ob die in der Datei definierten Zahlen als Rückgabewert protokolliert worden sind.