Unit Testing

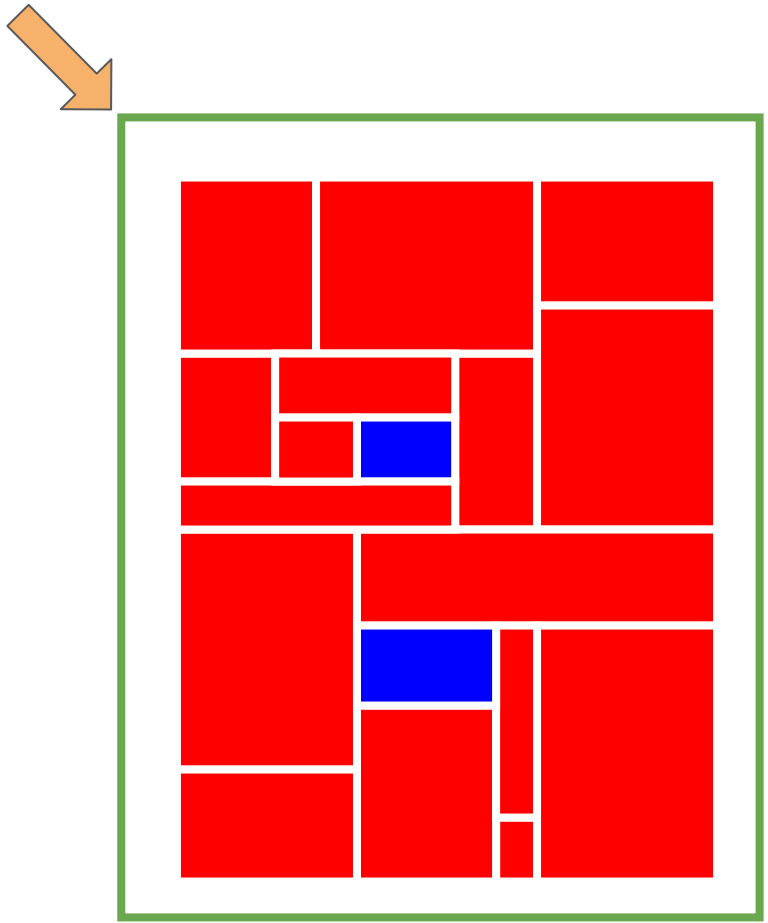
am Beispiel von C#

Inhaltsverzeichnis

Was sind Unit-Tests?

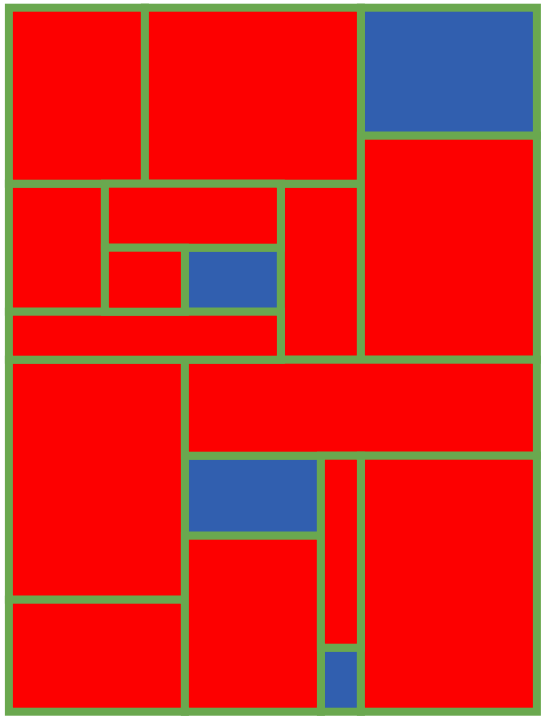
Traditionelles Testen

- Test des Systems als Ganzes
- Einzelne Komponenten werden selten getestet
- Fehler werden seltener erkannt
- Fehler im Gesamtsystem schwieriger zu beheben



Unit Testing

- Jeder Teil der Sofware wird einzeln getestet
- Einzelne Komponenten werden durch automatische Tests öfter getestet
- Fehler werden schneller erkannt
- Fehler im Modul einfacher zu beheben



Wie soll ich vorgehen?

Tests vor/während Implementierung

- auch TDD (Test Driven Development)

- Grey-Box Tests
- Entwicklung in Mikroiterationen
- Vorteile:
  - gute Code Coverage
- Nachteile:
  - Zeitbedarf

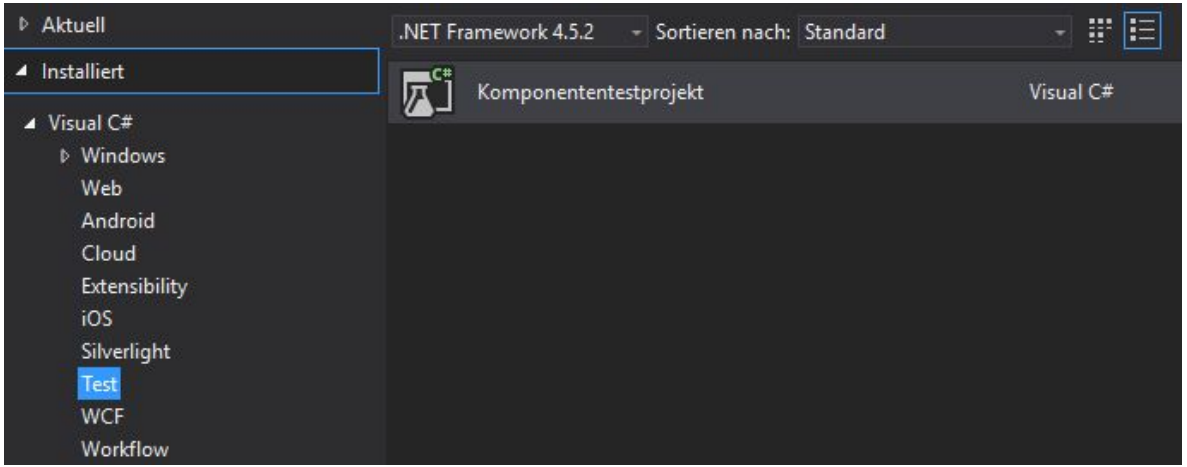
Tests nach Implementierung

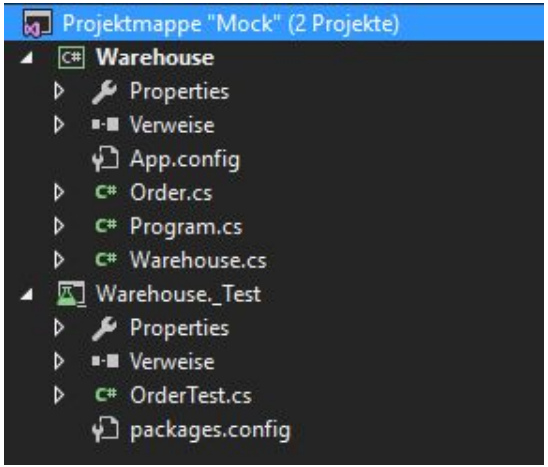
- Bsp. Wasserfallmodell

- White-Box-Tests
- Gesamte Tests nach Entwicklung
- Nachteile:
  - “Betriebsblindheit”
  - schlechte Code Coverage
  - oft schwer zu implementieren

Wie erstelle ich Unit-Tests?

Komponententestprojekt





Projektmappe > Hinzufügen > Neues Projekt

Visual C# > Test > Komponententestprojekt

UnitTestKlasse + Methoden

```
1   using System;
2   using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4   namespace StockManager._Test
5   {
6       [TestClass]
7       public class CalculatorTest
8       {
9           [ClassInitialize]
10          public static void ClassInitialize(TestContext context)
11          {
12
13          }
14
15          [ClassCleanup]
16          public static void ClassCleanup()
17          {
18
19          }
20
21          [TestInitialize]
22          public void TestInitialize()
23          {
24          }
25
26          [TestCleanup]
27          public void TestCleanup()
28          {
29
30          }
```

```
41          [TestMethod]
42          public void TestMethod()
43          {
44
45          }
```

Wie funktionieren Unit-Tests?

Aufbau von Unit-Tests

- Setup
  - Data
  - Expectations
- Excercise
- Verify
- Teardown

Prinzipien von Unit-Tests

- möglichst Isoliert
- Wiederholbar
- Automatisierbar
- Einfach zu Schreiben/Lesen

Tests-Doubles

Dummy

“Dummy” für ein Objekt, d.h. :

- Objekt wird als Platzhalter benutzt und kann herumgegeben werden
- Kann nicht benutzt werden
- besitzt meistens keine Implementierung

Bsp. :

- Rechnung

Stub

Platzhalter für ein Objekt, d.h. :

- Neuimplementierung einer Methode die bei unterschiedliche Kriterien immer gleich Ausgabe tätigt

Bsp. :

- Währungsrechner
- Festplattengröße

Fake

Vereinfachung für ein Objekt, d.h. :

- eigentlich Funktion ist durch neue Implementierung vereinfacht
- wird meistens öfter als einmal aufgerufen und später wieder benutzt

Bsp. :

- Datenbank

Verifikation

state Verification

- Verifikation ob die Methode richtig funktioniert, durch Analyse des SUT[1] und seiner mitwirkenden Klassen

[1]: System under Test

Mocking

behaviour Verification

- Verifikation ob die Methode richtig funktioniert, durch Analyse der aufgerufenen Methoden der mitwirkenden Klassen durch das SUT

[1]: System under Test

Mock

Kontrollstruktur für ein Objekt, d.h. :

- Objekt wird simuliert
- Vor dem Test werden bestimmte Verhalten programmiert
- Nach dem Test kann überprüft werden was mit dem Objekt passiert ist

Bsp. :

- Logging

Fragen?

Danke für eure Aufmerksamkeit!