Testen von Java-Code mit JUnit

Apcon Workplace Solutions
Member of itelligence

- ➔Motivation
- ➔JUnit-Testklassen
- ➔JUnit-Testfälle
- ➔Struktur eines Testfalls

Henning Wolf
APCON Workplace Solutions GmbH
wolf@jwam.de

Motivation: Werkzeugunterstützung für Tests

Apcon Workplace Solutions
Member of itelligence

- Wir erinnern uns:
  - ➔ Testen ist schwierig
  - ➔ Testen ist aufwändig
  - ➔ Tests sollten wiederholt werden und damit wiederholbar sein
- Häufig werden aufgrund mangelnder Disziplin Tests nur teilweise oder nur gelegentlich durchgeführt.
- Selbst wenn Tests automatisiert durchgeführt werden: Wer reagiert in welcher Weise auf die Ausgaben der Testläufe?
- Ein Werkzeug kann uns viele der administrativen Aufgaben beim Testen abnehmen.
- Idealerweise ist ein Testwerkzeug in die Entwicklungsumgebung eingebunden.

JUnit

Apcon Workplace Solutions
Member of itelligence

- Das bekannteste Werkzeug zur Unterstützung von Modultests für Java ist *JUnit*.
- JUnit ist selbst in Java geschrieben.
- JUnit stellt einen Rahmen zur Verfügung, wie Testklassen geschrieben werden sollten.
- Testklassen, die nach den Konventionen von JUnit entwickelt wurden, lassen sich sehr leicht automatisch ausführen.
  - ➔ Dies ist die Voraussetzung für die automatische Wiederholung von Tests, die wir anstreben.
- Es existieren Einbindungen von JUnit in die beiden bekanntesten Entwicklungsumgebungen für Java:
  - ➔ *Visual Age für Java* und *JBuilder*

Wo kommt JUnit her?

Apcon Workplace Solutions
Member of itelligence

- Bestandteil von eXtreme Programming (XP)
  - ➔ Pair Programming
  - ➔ Collective Code Ownership, Truck Factor
  - ➔ Small and Simple, Quick Design
  - ➔ Refactoring (Martin Fowler)
  - ➔ 40-hour Week
- XP und JUnit wurden von Kent Beck und Erich Gamma entwickelt.

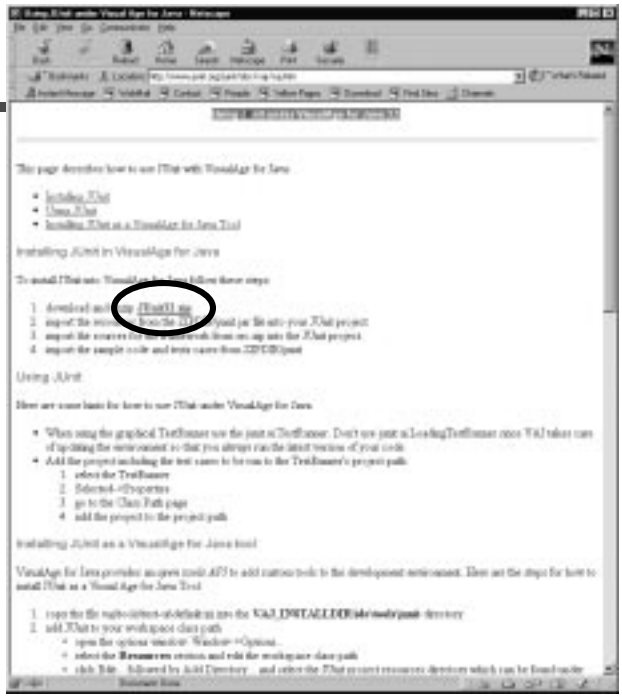
Entwicklungsprojekte mit JUnit

Apcon Workplace Solutions
Member of itelligence

- Wenn JUnit in die Entwicklungsumgebung eingebunden ist, können Testläufe als selbstverständlicherTeil des Entwicklungsprozesses angesehen werden.
- Dies ist das Ziel der Philosophie, die hinter JUnit steht ("test infected"). Tests sollten selbstverständlicher Bestandteil der täglichen Arbeit sein.
- In Entwicklungsprojekten bei Apcon WPS verwenden wir einen selbstentwickelten Integrationsklienten, der eine Integration nur zulässt, wenn alle JUnit-Testfälle für ein Projekt fehlerfrei durchlaufen.

Wo bekommt man JUnit her?

Was muß bei VisualAge beachtet werden?



Die ersten Schritte

Apcon Workplace Solutions
Member of itelligence

- Um einen Modultest mit JUnit durchzuführen, müssen zwei Schritte vollzogen werden:
  - ➔ Für eine zu testende Klasse muss eine Testklasse erstellt werden, die in ihrem Format den Anforderungen von JUnit entspricht. Diese Testklasse enthält eine Reihe von Testfällen.
  - ➔ JUnit muss so gestartet werden, dass es die Testfälle dieser neu erstellten Testklasse ausführt.

Erstellen einer JUnit-Testklasse

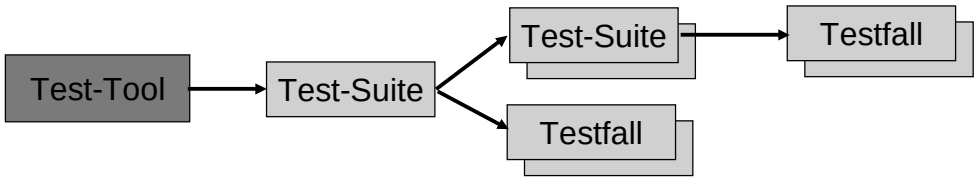
Apcon Workplace Solutions
Member of itelligence

- Damit eine Testklasse das richtige Format bekommt, beerbt sie in einem ersten Schritt die Klasse `TestCase`, die von JUnit zur Verfügung gestellt wird.
- Zwei Voraussetzungen müssen erfüllt sein, damit von der Klasse `TestCase` geerbt werden kann:
  - ➔ Der Entwicklungsumgebung müssen die Klassen von JUnit bekannt gemacht werden.
  - ➔ In der Klassendefinition der neuen Testklasse muss die Oberklasse `TestCase` *importiert* werden.

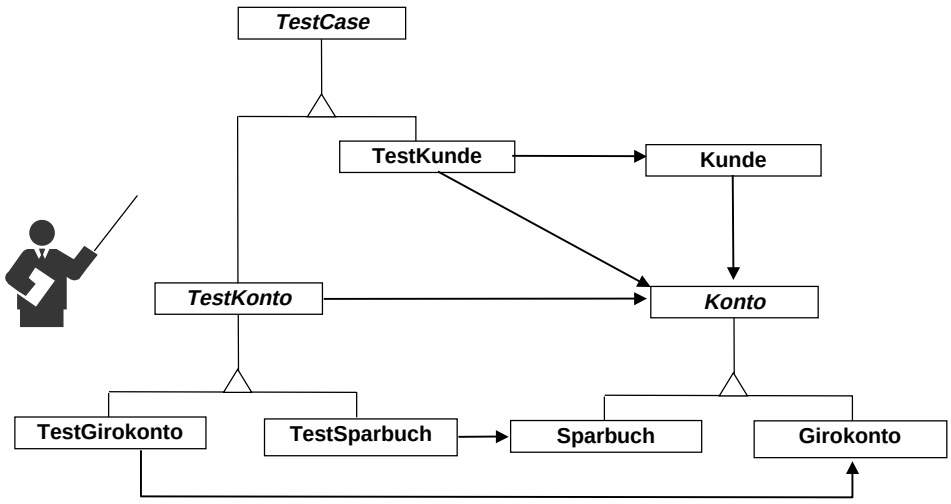
JUnit-Test Struktur

Apcon Workplace Solutions
Member of itelligence

- Testklassen werden hierarchisch zu Test-Suites gebündelt.
- Test-Suites können von einem Test-Tool ausgeführt werden.
- Das Test-Tool führt jede Operation der zu testenden Klasse mindestens einmal aus.
- Ergebnisse werden protokolliert.



Beispiel



Struktur einer JUnit-Testklasse

```
import junit.framework.TestCase;

/**
 * Kommentar für die Testklasse.
 */
public class TestSparbuch extends TestCase
{
  public TestSparbuch (String name) {
      super(name);
  }
  ...
}
```

Mit einer import-Klausel wird eine Klasse aus einem anderen Package innerhalb einer Klassendefinition "benutzbar".

Eine JUnit-Testklasse erbt von `TestCase`.

Dieses Format ist für den Konstruktor einer JUnit-Testklasse vorgeschrieben.

Die Testfälle einer JUnit-Testklasse

Apcon Workplace Solutions
Member of itelligence

- Jede Prozedur in einer JUnit-Testklasse, die keine formalen Parameter definiert und deren Name mit "test" beginnt, wird von JUnit als ein Testfall angesehen.
- Für jeden Testfall (also jede test...-Prozedur) wird von JUnit für die Durchführung des Tests ein Exemplar der Testklasse erzeugt. Bei jedem Exemplar wird dann genau eine der test-Prozeduren aufgerufen.
- Damit für diesen Aufruf von einem definierten Zustand ausgegangen werden kann, kann in einer parameterlosen Prozedur `setUp` eine beliebige Initialisierung vorgenommen werden.
- Häufig werden in `setUp` ein oder mehrere Exemplare der Klasse erzeugt, die getestet werden soll.

Testfälle in der Klassendefinition

Apcon Workplace Solutions
Member of itelligence

```
public class TestSparbuch extends TestCase
{
      ...
  public void setUp()
  {
      _sparbuch = new Sparbuch();
  }

  public void testEinzahlen() { ... }
  public void testAuszahlen() { ... }
  ...
}
```

`setUp` ist eine Prozedur aus der Klasse `TestCase`, die in einer konkreten Testklasse redefiniert wird. `setUp` wird vor Ausführung eines Testfalls auf einem neu erzeugten Exemplar der Testklasse aufgerufen.

Zwei Testfälle

Aufbau eines Testfalls

- Im Körper einer Testfallprozedur werden üblicherweise Aufrufe an die zu testenden Exemplare durchgeführt.
- Die Ergebnisse dieser Aufrufe werden dann verglichen mit erwarteten Ergebnissen.
- Das meist boolesche Ergebnis eines solchen Vergleichs wird an eine der geerbten Prozeduren übergeben, die auf diesen Wert entsprechend reagiert.
- Weicht der erwartete Wert vom tatsächlichen ab, sorgt diese Prozedur dafür, dass eine entsprechende Fehlermeldung ausgegeben wird.

Ein Beispiel für einen Testfall

```
public class TestSparbuch extends TestCase
{
  ...
  public void testEinzahlen()
  {
    _sparbuch.einzahlen(500);
    double saldo = _sparbuch.gibSaldo();
    assert(saldo == 500);
      ...
  }
}
```

Die Referenz `_sparbuch` ist gültig, weil in einer Testfallprozedur davon ausgegangen werden kann, dass `setUp` unmittelbar vorher aufgerufen wurde.

`assert` ist eine der Prozeduren, die von `TestCase` geerbt wurde und mit deren Hilfe Fehlermeldungen generiert werden können.

Einige Prozeduren aus `TestCase`

Apcon Workplace Solutions
Member of itelligence

```
assert(boolean condition)
assert(String message, boolean condition)
assertEquals(int expected, int actual)
assertEquals(Object expected, Object actual)
...
assertEquals(String message, int expected, int actual)
...
assertNotNull(Object object)
assertNotNull(String message)
assertSame(Object expected, Object actual)
```

Das JUnit-Motto

Apcon Workplace Solutions
Member of itelligence

keep the bar green to keep the code clean...

www.junit.org

Zusammenfassung (1)

Apcon Workplace Solutions
Member of itelligence

- In einem großen Softwaresystem ist es häufig schwierig festzustellen, wo ein Problem seine Ursache hat.

  ➔ Fehlersuche ist komplex und langwierig!

- Kleine Änderungen haben unabsehbare Folgen, deshalb wird so wenig wie möglich geändert.

  ➔ Qualität nimmt schnell ab!

Zusammenfassung (2)

Apcon Workplace Solutions
Member of itelligence

Vorgehen mit JUnit:

- Testen und Programmieren im schnellen Wechsel.
- Jede neue Klasse und Operation sofort testen.
- Idealerweise die Testklasse vor der zu testenden Klasse schreiben.
- Am Ende eines Tages müssen alle Testfälle korrekt durchlaufen.
- Testfälle wie andere Klassen der Anwendung auch schreiben (Dokumentation, vernünftige Strukturierung etc.)
- Testen zu einer Grundhaltung machen (Test-Infected)

Testing Frameworks für andere Programmiersprachen

Apcon Workplace Solutions
Member of itelligence

- http://www.xprogramming.com/software.htm

- Ada
- C++, Visual C++
- Delphi
- Forte 4GL
- GemStone/S
- JavaScript
- Objective-C
- Oracle
- Perl
- PhpUnit
- Smalltalk
- Visual Basic