TDD As If You Meant It

Ralf Westphal
info@ralfw.de

ralfw.blogspot.com
@ralfw
www.ralfw.de
www.clean-code-developer.de

TDD-Regeln

1. Red: Schreibe einen Test, der fehlschlägt
2. Green: Tue das einfachste Mögliche, um den Test erfolgreich zu machen
3. Refaktorisiere den Code nach Gusto
4. Goto 1.

TDDaiymi-Regeln

1. Red: Schreibe einen Test, der fehlschlägt
2. Green: Tue das einfachste Möglich *in der Testmethode*, um den Test erfolgreich zu machen
3. Refaktorisiere den Code wie folgt:
    - *Extrahiere eine Nicht-Test-Methode aus dem Testcode in die Testklasse*
    - *Bewege Code aus dem Test in eine schon existierende Methode*
    - *Erzeuge eine Nicht-Test-Klasse nur als Ziel für schon existierende Methoden*

- Goto 1.

Aufgabe „Word Wrap“

[…]

I turned to Jerry and asked: "Word Wrap?"

"Yeah, it's a simple problem to understand, but it's oddly difficult to solve. The basic premise is really simple. You write a class called Wrapper, that has a single static function named wrap that takes two arguments, a string, and a column number. The function returns the string, but with line breaks inserted at just the right places to make sure that no line is longer than the column number. You try to break lines at word boundaries."

I thought about this for a moment and then said: "You mean like a word processor, right? You break the line by replacing the last space in a line with a newline."

Jerry nodded. "Yeah that's the idea. Pretty simple huh?”

[…]

Aus Robert C. Martin „The Craftsman 62, The Dark Path“, http://thecleancoder.blogspot.de/2010/10/craftsman-62-dark-path.html

Testfälle

1. „zeile“, 5 -> „zeile“
2. „zeile1zeile2“, 6 -> „zeile1\nzeile2“
3. „zeile1zeile2zeile3“, 6 -> „zeile1\nzeile2\nzeile3“

4. „zeile1 zeile2“, 6 -> „zeile1\nzeile2“
5. „zeile1 zeile2“, 7 -> „zeile1\nzeile2“
6. „zeile1 zeile2“, 8 -> „zeile1\nzeile2“

Was ich gelernt habe

- Refaktorisierung ist nicht mehr optional, sondern muss vorgenommen werden
- Dopplungen in Testmethoden sorgen ganz natürlich dafür, dass Aspekte in Lösungsmethoden verpackt werden (DRY)
- Weitere Methoden entstehen nur bei Sensibilität für SRP

- Es ist wirklich wichtig, Lösungen zuerst in der Testmethode zu implementieren
  - Nur so werden Aspekte sichtbar, die sonst in Lösungsmethoden untergehen würden

Ressourcen

- Keith Braithwaite, TDD as if you meant it, http://cumulative-hypotheses.org/2011/08/30/tdd-as-if-you-meant-it/
  - Hier auch Links zu weiteren Betrachtungen zum Thema
- Keith Braithwaite, TDD as if you meant it (Video), http://www.infoq.com/presentations/TDD-as-if-You-Meant-It
- Gojko Adzic, Thought-provoking TDD exercise at the Software Craftsmanship conference, http://gojko.net/2009/02/27/thought-provoking-tdd-exercise-at-the-software-craftsmanship-conference/
- Gojko Adzic, TDD as if you meant it – revisited, http://gojko.net/2009/08/02/tdd-as-if-you-meant-it-revisited/

- Kata Word Wrap
  - Beschreibung: http://codingdojo.org/cgi-bin/wiki.pl?KataWordWrap
  - Robert C. Martins Musterlösung: http://thecleancoder.blogspot.de/2010/10/craftsman-62-dark-path.html
  - Peter Koflers Lösungssammlung: http://blog.code-cop.org/2011/08/word-wrap-kata-variants.html

Referent

- Ralf Westphal (www.ralfw.de) ist freiberuflicher Berater, Projektbegleiter, Autor und Trainer für Themen rund um .NET Softwarearchitektur. Er ist Autor von mehr als 450 Publikationen und Microsoft Most-Valued-Professional für Softwarearchitektur.
- Mit Stefan Lieser hat er die Initiative „Clean Code Developer“ für mehr Softwarequalität ins Leben gerufen (www.clean-code-developer.de) und betreibt das Beratungsnetzwerk www.clean-code-advisors.de.