andrena OBJECTS

Experts in agile software engineering

Record – Refactor - Replay

Stefan Mandel

Java-Code den man nicht hinterlassen sollte …

- Hat zu wenige (automatisierte) Tests
- Hat unkontrollierte Seiteneffekte
  - Auf globale Variablen
  - Und Argumente
- Verzichtet auf Objektorientierung
  - Wenig Struktur (Patterns)
  - Generische Typen (Maps)
  - Objekte kapseln nur Daten, nicht Funktionen
- Und behandelt Ausnahmen uneinheitlich oder nicht

© 2017 andrena objects ag

Aber was, wenn man mir solchen Code hinterlässt …

Tests schreiben? Refactoring?

- Grundsätzlich schon
  - Aber der Code ist schwer testbar
  - Aber der Code ist ungetested und damit schwer refactorbar
  - Aber oft ist nicht einmal genügend Domänenwissen da, um zu beurteilen, welcher Code fehlerhaft und welcher korrekt ist
- Und Charakterisierungstests
  - Sind stumpfsinnig und langweilig
  - Und erfordern viel Zeitaufwand und Sorgfalt

© 2017 andrena objects ag

Charakterisierungstests ?

Stumpfsinnig, Langweilig, Zeitaufwändig?

- Sowas machen inzwischen doch Maschinen …
- Prinzip:
  - Record: Zeichne zu jeder Funktion
    - Den Zustand vor der Ausführung
    - Den Zustand nach der Ausführung
  - Replay: Führe Tests folgender Form aus:
    - Stelle den Zustand vor der Ausführung her
    - Vergleiche das Ergebnis mit dem aufgezeichneten Zustand nach der Ausführung

© 2017 andrena objects ag

Jetzt zu Tante Emmas Online-Shop

Hier hat ein Legacy Coder Arbeit geleistet

- Im Projekt finden wir ein Verzeichnis src/hints
  - Mit den Tasks, die wir behandeln
    - Als reveal-md: reveal-md tasks.md --css tasks.css
    - Als PDF: tasks.pdf
    - Als Text: task1.md – task5.md
- Wir beginnen mit **task1**
  - Wir starten die Anwendung
  - Sie ist erreichbar unter http://localhost:8080
  - Wir machen uns ein wenig mit der Anwendung vertraut
  - Ein Blick in den Source-Code kann nicht schaden

© 2017 andrena objects ag

Wir installieren Testrecorder

10 min

- **task2**
- Dazu fügen wir testrecorder-with-dependencies.jar ins dem Klassenpfad hinzu
- Wir benötigen eine Konfiguration, d.h. eine Klasse
  `AgentConfig extends DefaultTestRecorderAgentConfig`
- Und laden die Anwendung mit der erstellten `AgentConfig`
- Wir annotieren `PriceCalculator.computePrice` mit `@Recorded`
- Wir starten die Anwendung und beobachten die generierten Tests
- Wir starten die generierten Tests

© 2017 andrena objects ag

Wir installieren Testrecorder

- Testrecorder zeichnet folgende Zustände auf:
  - Argumente (vor und nach dem Aufruf)
  - Ergebnisse (nach dem Aufruf)
  - This/Instanzvariablen (vor und nach dem Aufruf)
  - *Exceptions (nach dem Aufruf)*

© 2017 andrena objects ag

Globale Abhängigkeiten Eliminieren

10 min

- **task3**
- Testrecorder braucht manchmal noch ein paar zusätzliche Informationen
  - Z.B. welche globalen/statischen Variablen aufgezeichnet werden sollen
- Finde die kritischen Globalen Variablen
- Annotiere Sie mit `@Global` (Oder erweitere die Konfiguration)
- Starte die Anwendung erneut und führe ein paar Aktionen durch
- Betrachte die Tests

© 2017 andrena objects ag

Globale Abhängigkeiten Eliminieren

- Testrecorder zeichnet folgende Zustände auf:
  - …
  - Globalen/Statische Variablen (vor und nach dem Aufruf)

© 2017 andrena objects ag

Definition von Input Abhängigkeiten

10 min

- **task3**
- Testrecorder arbeitet grundsätzlich funktional
  - Eingabezustände werden als Globale Variablen, Instanzvariablen oder Argumente aufgezeichnet
  - Ausgabezustände werden als Globale Variablen, Instanzvariablen, Ergebnisse, Argumente und Ausnahmen aufgezeichnet
  - Input/Output muss gekennzeichnet werden
- Finde die Methoden, die zufällige Objekte (Input) liefern
- Annotiere Sie mit `@Input` (oder erweitere die Konfiguration)
- Starte die Anwendung erneut und führe ein paar Aktionen durch
- Betrachte die Tests

© 2017 andrena objects ag

Definition von Input Abhängigkeiten

- Testrecorder zeichnet folgende Zustände auf:
  - …
  - Input (während dem Aufruf)
    - Zustandsänderungen die vom Benutzer/Zufall/Zeit induziert werden
  - *Output (während dem Aufruf)*
    - *Zustandsänderungen an fremden Systemen, die nicht mehr vollumfänglich in der JVM abgeprüft werden können*

© 2017 andrena objects ag

Beheben von Fehlern

15 min

- Irgendwas stimmt mit den Produkten nicht
  - Manche Ereignisse verändern Produkte
- Zunächst korrigieren wir die Tests die solche Änderungen abtesten
- Dann korrigieren wir den Code
- Und starten die Tests
- Und korrigieren ggf. noch Tests die wir übersehen haben
  - Bzw. finden Strategien dafür

© 2017 andrena objects ag

Beheben von Fehlern

© 2017 andrena objects ag

Weitere Informationen …

http://testrecorder.amygdalum.net/

- Dokumentation:
  - Anleitung zur Konfiguration
  - Anleitung für die Aufzeichnung (mit und ohne Agent)
  - Beispiele
- Funktionsumfang
  - Ausprobieren
  - Komplexe Objekte (z.B. reflect, native, concurrent) funktionieren noch nicht
  - Komplexe Situationen (z.B. recording von overridden Methoden) funktionieren noch nicht
- Tests
  - > 90% Testüberdeckung
  - Bugs werden über automatisierte Tests reproduziert und gefixt
- API ist noch nicht stabil
  - Jede neue Version ist eine Major Version
- Contribution und Feedback willkommen (https://github.com/almondtools/testrecorder)

© 2017 andrena objects ag

Vielen Dank!

© 2017 andrena objects ag