Orientation in Objects

TDLCR

Test Driven Legacy Code Refactoring

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Java Forum Nord Hannover, 20.10.2016

Version: 1.0

Abstract

Orientation in Objects

Bestandsanwendungen müssen gewartet und ggf. weiterentwickelt werden, bergen aber meist viele Defekte. Als Entwickler fürchten wir uns zudem, mehr Schaden anzurichten, weil das Verständnis für den Legacy Code fehlt. Refactoring kann zum Verstehen beitragen, endet aber aufgrund der typischerweise fehlenden automatisierten Tests in einem Blindflug.

Mit testgetriebener Entwicklung scheint es eine Allzweckwaffe für gutes Design und eine geringe Fehlerrate zu geben. Aber TDD und Legacy Code scheinen sich eigentlich auszuschließen. Anhand von Live Coding schauen wir, wie die testgetriebene Entwicklung trotzdem helfen kann, den Code ohne allzu große Bauchschmerzen anzupassen. Schöner Nebeneffekt wird das Entstehen eines automatisierten Testbetts sein, welches zukünftige Refactorings leichter machen wird.

Über mich

Orientation in Objects

Falk Sippach (@sippsack)

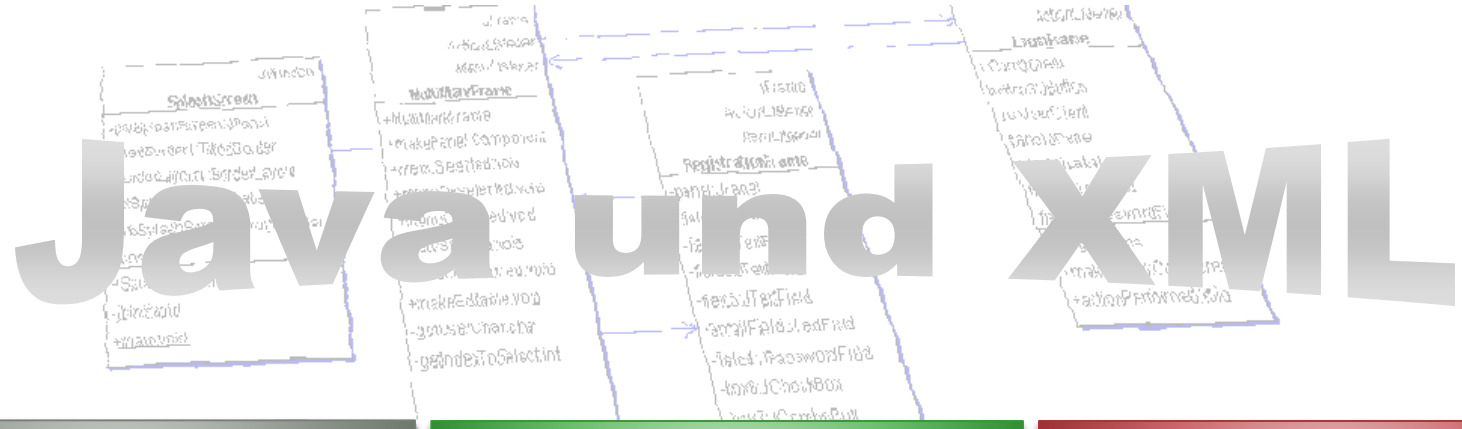
Trainer, Berater, Entwickler

Co-Organisator

Schwerpunkte
Architektur
Agile Softwareentwicklung
Codequalität

Java, XML und Open Source seit 1998

Orientation in Objects



) Software Factory)

- **Schlüsselfertige Realisierung von Java Software**
- **Individualsoftware**
- **Pilot- und Migrationsprojekte**
- **Sanierung von Software**
- **Software Wartung**

) Object Rangers)

- **Unterstützung laufender Java Projekte**
- **Perfect Match**
- **Rent-a-team**
- **Coaching on the project**
- **Inhouse Outsourcing**

) Competence Center)

- **Schulungen**, **Coaching**, **Weiterbildungsberatung**, **Train & Solve-Programme**
- **Methoden**, **Standards** und **Tools** für die Entwicklung von **offenen, unternehmens-weiten Systemen**

Orientation in Objects

TD**LC**R

Legacy Code

Foto von smpcas, CC0 Public Domain Lizenz, https://pixabay.com/de/pula-kroatien-amphitheater-erbe-827909/


Greenfield, Agile, TDD, …


// TODO: refactor this

Was ist Legacy Code?

Alter Code, geerbter Code, Code den niemand mehr versteht …

"Some crap made by someone else"

Code entstanden unter Zeitdruck und ohne automatisierte Tests

Technische Schulden

Warum Legacy Code anfassen?

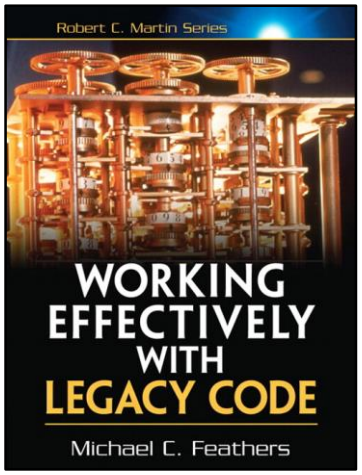
- Verstehen
- Erweitern
- Bugfixing
- Optimierung

Orientation in Objects

J. B. Rainsberger

> “Legacy code is **valuable code** that we feel **afraid to change**.

Foto von PublicDomainPictures, CC0 Public Domain Lizenz, https://pixabay.com/de/menschen-abdeckung-schrei-314481/



Orientation in Objects

Legacy Code Dilemma

“ Code **without tests** is **bad code**.

Michael Feathers

refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu ve n könnte man Code refactoren, um C Tests, Tests würden helfen, Code zu ve n könnte man Code refactoren, um C sts, Tests würden helfen, Code zu ve nte man Code refactoren, um Code zu refactoren bräuchte man Tests, **Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests,** Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu verstehen, um Code zu verstehen könnte man Code refactoren, um Code zu refactoren bräuchte man Tests, Tests würden helfen, Code zu

um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, **um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben**, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refactoren, um zu refactoren könnte man Tests schreiben, um Tests zu schreiben muß der Code testbar sein, um testbaren Code zu erhalten müßte man refact

Die Katze beißt sich in den Schwanz!

Orientation in Objects

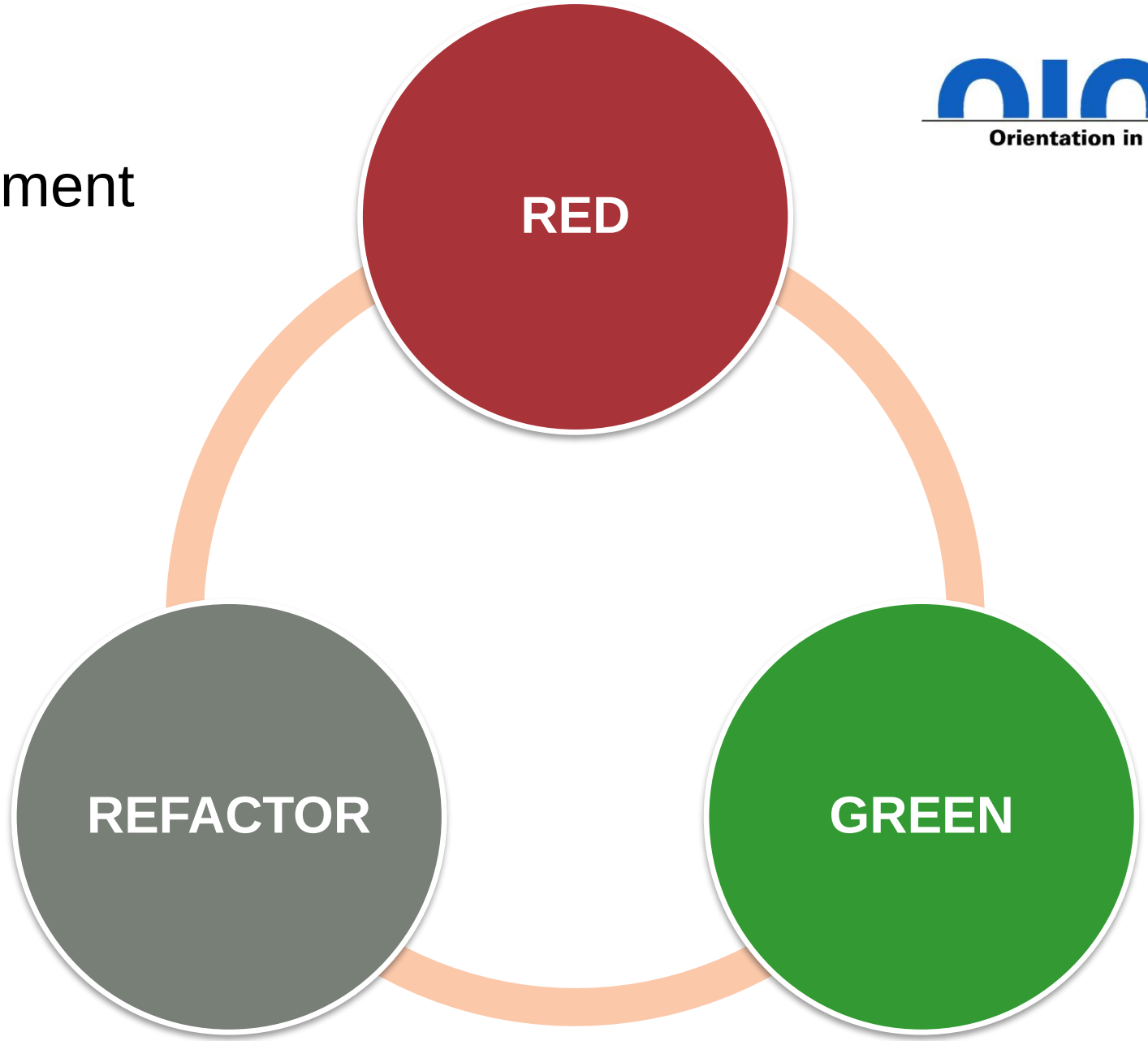
TDLCR

Test Driven Refactoring

TDD hilft beim Verbessern des Codes

Orientation in Objects

Test
Driven
Development



Orientation in Objects

"Gesetze" von TDD

Orientation in Objects

- Schreibe einen fehlschlagenden Test, bevor du Code für das Produktivsystem verfasst.
- Schreibe nur so viel Code für den Test, damit er kompiliert (Rot)
- Schreibe nur so viel Code, um den Test zu bestehen (Grün)
- Refactor: Duplikation entfernen

Wo kann TDD bei Legacy Code helfen?

- ~~Verstehen~~
- Erweitern
- ~~Bugfixing~~
- ~~Optimierung~~

Orientation in Objects

TDD vs. Legacy Code

Test vor dem Code vs. Code schon da

Orientation in Objects

TDLCR

Test Driven Legacy Code Refactoring

Erweitern Strategie 1:

Edit and Pray!

Foto von Myriams-Fotos: https://pixabay.com/en/rosary-faith-pray-folded-hands-1211064/ (CC0 Public Domain Lizenz)

Erweitern Strategie 2:

Canary in the coal mine

Orientation in Objects

1. **Sicherheitsnetz**
2. **Neue Funktionalität mit TDD**
3. **Testen des Bestands-Codes**

Foto von bella67: https://pixabay.com/de/spinnennetz-mit-wasserperlen-netz-921039/ (CC0 Public Domain Lizenz)

Golden Master

gegenwärtiges Verhalten dokumentieren und erhalten

Foto von istara: https://pixabay.com/de/gold-bar-goldbarren-reich-geld-296115/ (CC0 Public Domain Lizenz)

```
# suppose that our legacy code is this program called 'game'
$ game > GOLDEN_MASTER

# after some changes we can check to see if behaviour has changed
$ game > OUT-01
$ diff GOLDEN_MASTER OUT-01
# GOLDEN_MASTER and OUT-01 are the same

# after some other changes we check again and...
$ game > OUT-02
$ diff GOLDEN_MASTER OUT-02
# GOLDEN_MASTER and OUT-02 are different -> behaviour changed
```

Golden Master

(aka characterization tests)

Konsolenausgaben abprüfen

Orientation in Objects

```
@Before
public void init() {
    originalSysOut = System.out;
    consoleStream = new ByteArrayOutputStream();
    PrintStream printStream = new PrintStream(consoleStream);
    System.setOut(printStream);
}

@Test
public void testSimpleOutput() {
    System.out.println("Hallo Publikum!");
    System.out.print("Hallo Falk!");
    assertEquals("Hallo Publikum!\r\nHallo Falk!", consoleStream.toString());
}

@After
public void teardown() {
    System.setOut(originalSysOut);
}
```

Was mache ich bei GUI-Anwendungen?


```
public void account(int minuten, int stunde, int minute) {
    System.out.println(String.format("Berechne Gespräch mit
    boolean mondschein = false;
    double preis = 0;
    this.
    // Mon
    if (st
        mo
    // Ges
```

gebuehr : double - Kunde
tarif : Tarif - Kunde
account(int minuten, int stunde, int minute) : void - Kunde
berechnePreis(int minuten, double d) : double - Kunde

3

```
public static void main(String... args) throws Exception {
  WebDriver driver = new FirefoxDriver();
  driver.get("http://www.retest.de");
  while (true) {
    List<WebElement> links = driver.findElements(By.tagName("a"));
    links.get(random.nextInt(links.size())).click();
    Thread.sleep(500);
    List<WebElement> fields =
        driver.findElements(By.xpath("//input[@type='text']"));
    WebElement field = fields.get(random.nextInt(fields.size()));
    field.sendKeys(randomString());
    Thread.sleep(500);
  }
}
```

https://entwicklertag.de/frankfurt/2016/sites/entwicklertag.de.frankfurt.2016/files/slides/Bei%20uns%20testen%20lauter%20Affen_0.pdf

2

Neue Features ausliefern …

Foto von PublicDomainPictures: https://pixabay.com/en/adorable-baby-basket-beautiful-boy-21259/ (CC0 Public Domain Lizenz)

Aufhören Legacy Code zu schreiben!

Orientation in Objects

- sonst wird die Codebasis nur schlimmer und man entfernt sich immer mehr davon, jemals Tests hinzuzufügen

- **Keine neuen Features mehr ohne Unit-Testing!**

- Sprout Method + Wrap Method als Hilfen

Sprout Method

Orientation in Objects

```
public int berechne(int a, int b) {
   int c = a + b;
   // weitere wichtige Aufgaben

   // neues Verhalten
   return verdoppeleResult(c);
}

protected int verdoppeleResult(int result) {...}
```

testgetrieben entwickeln

neues Feature in eigene Methode

Sprout Method und TDD (1)

Orientation in Objects

```
// nicht kompilierenden Test schreiben

@Test
public void testVerdoppeleResult() {
  assertEquals(2, rechner.verdoppeleResult(1);
}

// Kompilerfehler

protected int verdoppeleResult(int result) {
  return 0;
}

// Test ist rot
```

Sprout Method und TDD (2)

Orientation in Objects

```
// einfachste Lösung, damit Test grün wird

protected int verdoppeleResult(int result) {
  return 2;
}

// nichts zu refactoren
// von vorn beginnen, weiteren Test schreiben usw.

@Test
public void testVerdoppeleResult() {
  assertEquals(2, rechner.verdoppeleResult(1);
  assertEquals(4, rechner.verdoppeleResult(2);
}
```

Tipps TDD

Orientation in Objects

- kleine Schritte
- Workflow (Red – Green - Refactor) einhalten
- Ideen für weitere Testfälle in eine Liste schreiben und nach und nach abarbeiten (kleine Schritte!)

Wrap Method

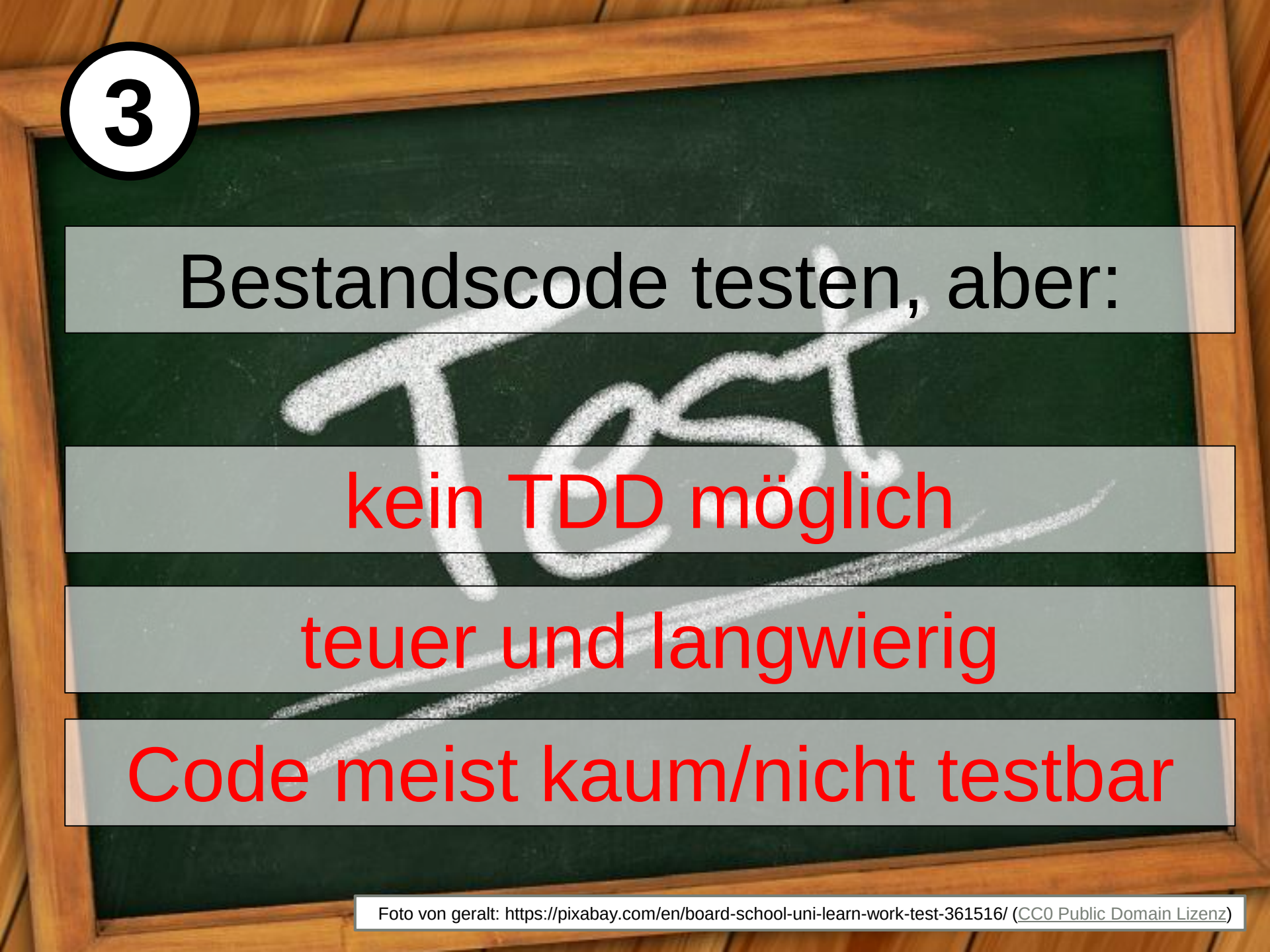
Orientation in Objects

- ähnlich zu Sprout Method, neues Verhalten aber vor oder am Ende der zu ändernden Methode

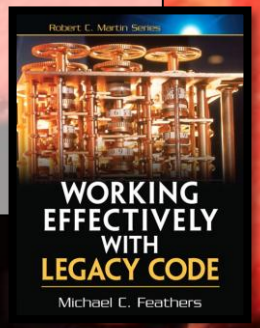
```
public int berechne(int a, int b) {
   logResult(c);
   return berechnePrivate(a, b);
}

private int berechnePrivate(int a, int b) {
   int c = a + b;
   // weitere wichtige Aufgaben
   return c;
}

protected void logResult(int result) {...}
```

Live-Coding

Foto von StockSnap: https://pixabay.com/de/codierung-gesch%C3%A4ft-arbeiten-macbook-699318/ (CC0 Public Domain Lizenz)



1. Identify what to change
2. Identify what to test
3. **Break dependencies**
4. Write the tests
5. Modify and refactoring



Orientation in Objects

Seam (Nahtstelle)

Ein Seam ist eine Stelle, an der man das Verhalten editieren kann, ohne direkt an dieser Stelle zu ändern.

Aufbrechen stark gekoppelter Abhängigkeiten

aka Subclass and Extract
aka Extract and Override Call
aka … (weitere verschiedene Varianten)

Subclass and Override

Orientation in Objects

- es gibt protected Methoden für die externen Dependencies
- in einer Subklasse mit leerem oder speziellem Inhalt überschreiben
- Verwenden der Subklasse in den Tests

```
public void saveOrder(int orderId) {
  Order order = orderRepository.getOrderById(orderId);
  getOrderChanges();
  saveOrderToFile(order);
}

protected void saveOrderToFile(Order order) {
  // File IO
}
```

Subclass and Override (2)

Orientation in Objects

```
public class OrderServiceForTests extends OrderService {
  @Override
  protected void saveOrderToFile(Order order) {

    // nichts tun (keine IO, keine externen Dependenies)

    // oder order-Parameter extern überprüfbar machen
  }
}
```

Extract and Override Call

Orientation in Objects

- zu überschreibenden Code zunächst herauslösen (Extract Method)
- dann wie Subclass and Override

```
public void sendOrderConfirmation(int orderId) {
  Order order =
      orderRepository.getOrderById(orderId);

  Mail email = new MailMessage(defaultSender,
      subject, ...));

  smtpClient.send(email);
}
```

Extract and Override Call

Orientation in Objects

- zu überschreibenden Code zunächst herauslösen (Extract Method)
- dann wie Subclass and Override

```
public void sendOrderConfirmation(int orderId) {
  Order order =
      orderRepository.getOrderById(orderId);

  Mail email = new MailMessage(defaultSender,
      subject, ...));

  smtpClient.send(email);
}
```

Extract Pure Method/Function

Orientation in Objects

- Codestellen isolieren
- Ziel: separat testen, Duplikation reduzieren
- Vorteil von reinen Methoden: seiteneffektfrei, keine Statusänderung
- Vorsicht: wir müssen Code ändern bevor Tests da sind
  - immer kleine Schritte, Tools/IDEs für das Refactoring verwenden
  - Extract Method ist typischerweise sicher

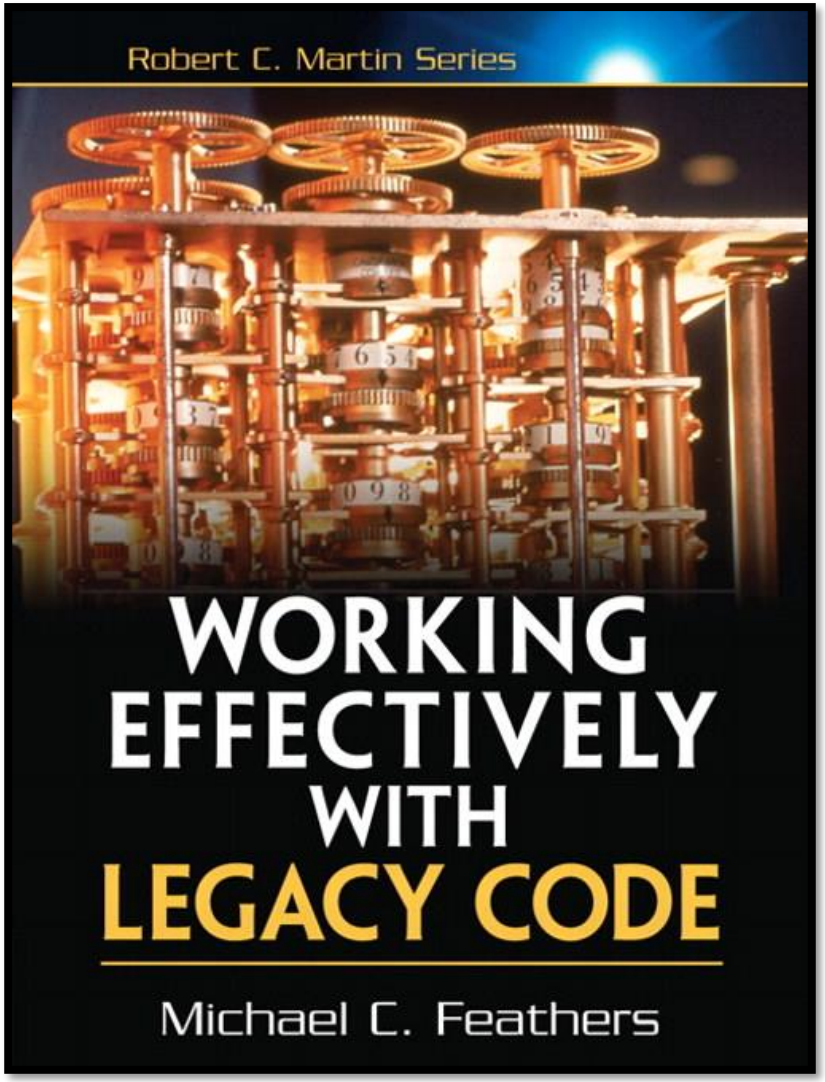
Extract Interface

Orientation in Objects

- Klasse extrahieren (Methode verschieben)
- Interface herausziehen
- Aufrufer ändern und das Interface verwenden
- leichter austauschbar gegen Dummy, Stub, Mock

Viele weitere Refactorings für Legacy Code

Orientation in Objects



Weitere Tools/Hilfsmittel

Orientation in Objects

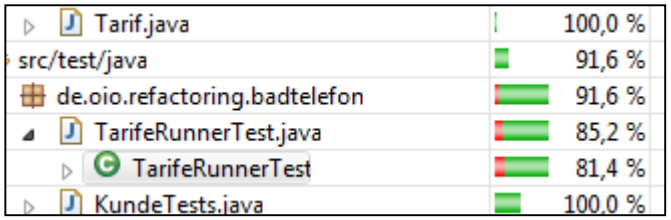
- private Methoden testen
  - nicht schön, aber praktisch
  - per Reflection oder mit Frameworks
  - z. B. private Pure- und Sprout-Methoden testen

- Mocking
  - Exctract and Override und noch mächtiger
  - Interaktion mit Umgebung testen
  - erwartete Parameter und Aufrufreihenfolge sicherstellen

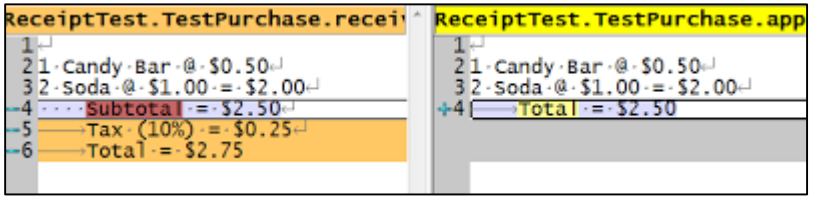
Tools

Orientation in Objects

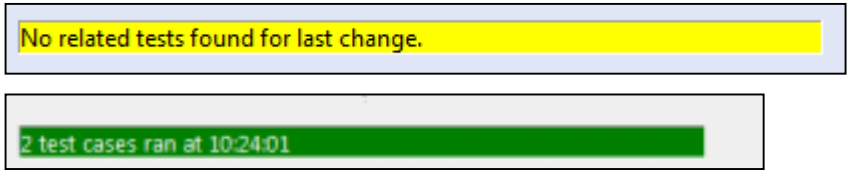
Code Coverage



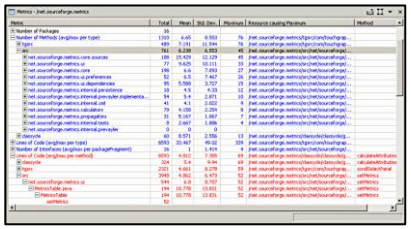
Approval Tests



Infinitest



Eclipse Metrics



Legacy Code Retreat

https://github.com/jbrains/trivia

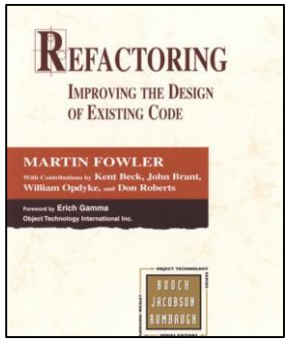
Links

Orientation in Objects

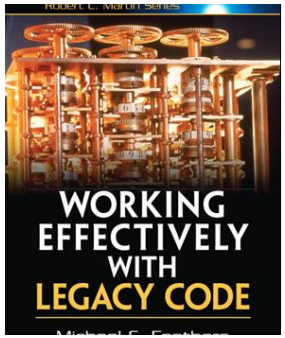
- Code-Beispiel der Live-Demo
  - https://github.com/sippsack/BadTelefon-Test-Driven-Legacy-Code-Refactoring
- anderes Code-Beispiel für Legacy Code
  - https://github.com/jbrains/trivia
- Blog: Techniken zu Legacy Code-Retreat
  - http://blog.adrianbolboaca.ro/2014/04/legacy-coderetreat/

Literaturhinweise

Orientation in Objects



- Refactoring
  - **Sprache: Englisch**
    Gebunden - 464 Seiten - Addison Wesely
    Erscheinungsdatum: 1. Juni 1999
    ISBN: 0201485672



- Working Effectively with Legacy Code
  - **Sprache: Englisch**
  - **Gebunden**

Orientation in Objects

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de