Kommt **Clean Code** in Studium und Ausbildung zu kurz?

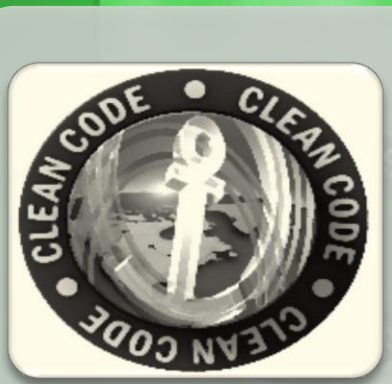
Björn Kimminich

https://twitter.com/bkimminich
https://linkedin.com/in/bkimminich
http://gplus.to/bkimminich
https://www.xing.com/profile/Bjoern_Kimminich

[C.C.D.]
CLEAN CODE DAYS
10. bis 12.09.2013
[DRESDEN]

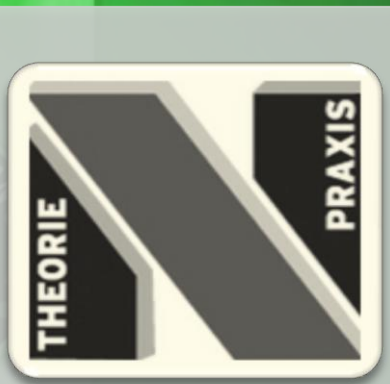
KCCiSuAzk 1.15 (06.09.2013)

Björn Kimminich



2007+

Software Architekt & Security Officer bei **Kuehne+Nagel** im Corporate Web Development



2011+

Nebenamtlicher Dozent für Java & Agile Software Entwicklung an der FH **Nordakademie**

2012+

Committer im **OWASP** Zed Attack Proxy Projekt und Leiter des zaproxy-test Unterprojekts

Woher kennen Sie das…?

- **Sorgfalt** wird als Teil des ***Arbeitsverhaltens*** bewertet…

  - … zusammen mit **Selbständigkeit** und **Leistungsbereitschaft**

… aus der **Grundschule!**

Inhaltsübersicht

Clean Code im Informatik-Studium

Sind Clean Coder unter Ihren Bewerbern?

Clean Code-Fortbildung im Unternehmen

Clean Code im Informatik-Studium

Lernschwierigkeitsgrad und Praxisrelevanz

Lernschwierigkeitsgrad

- Erste Programmiersprache
- Neue Sprachen/Technologien
- Professionelle Softwareentwicklung

Praxisrelevanz

- Der Aufwand würde sich also lohnen!

Clean Code ist (fast) unabhängig von der Programmiersprache

- Der Aufwand würde sich also **erst recht** lohnen!

Quelle: http://www.hanselman.com/blog/JavaScriptIsAssemblyLanguageForTheWebSemanticMarkupIsDeadCleanVsMachinecodedHTML.aspx

Clean Code im Curriculum (Uni Hamburg)

Aufbau des Studiengangs

Im 6-semestrigen Bachelorstudiengang SSE erlernen Sie im ersten Studienabschnitt die notwendigen Informatik-Grundlagen; danach stehen praktische Lehrveranstaltungen und die von Ihnen gesetzten Schwerpunkte im Vordergrund.

Wesentliche Studienziele sind

- Eine solide Basis in Softwareentwicklung und Softwaretechnik
- Beherrschung der Grundlagen der Informatik
- Method. Kompetenz, soziale u. kommunikative Fähigkeiten
- Erfahrung mit Software-Projekten

WS 1	Software-entwicklung I	Informatik im Kontext		Rechnerstrukturen	Mathematik für Studierende der Informatik
SS 1	Software-entwicklung II	Meth.-komp.	Pro-seminar	Formale Grundlagen der Informatik I	
WS 2	Software-entwicklungs-praktikum	Grundlagen von Datenbanken	Projekt-managem.	*Wahlpflicht Informatik*	*Wahlpflicht Informatik*
	Softwaretechnik	*Wahlpflicht Informatik*		*Wahl Anwendungsgebiet*	*Projekt*
SS 2	*Wahlpflicht Informatik*	*Seminar*		*Industriepraktikum*	
WS 3	*Wahl Anwendungsgebiet*	*Wahl Anwendungsgebiet*		Bachelorarbeit	

kursiv = Platzhalter, die primär Umfänge verdeutlichen sollen, keine Semesterzuordnungen
nicht kursiv = Pflichtveranstaltungen

praktikum

Softwaretechnik

Wahlpflicht Informatik

Quelle: http://www.min.uni-hamburg.de/bachelorstudiengaenge_/bsc_software-system-entwicklung.pdf

Modul „Softwaretechnik“ (Uni Hamburg)

	sie auch den Bezug zum Qualitätsbegriff für Software herstellen.
Inhalt	- Fortgeschrittene Entwurfs- und Modellierungsmethoden - Grundlagen der Softwarearchitektur - Gestaltung interaktiver Systeme - Klassische Vorgehensmodelle und agile Methoden - Requirements Engineering - Tests zur Qualitätssicherung - Qualitätsmodelle für Software, Softwaremetriken - Werkzeuge der Software-System-Entwicklung: - zur Analyse von SW-Architekturen zur Unterstützung des Entwicklungsprozesses
Lehrveranstaltungen und Lehrformen	Vorlesung Softwaretechnik — 4 SWS
	Übungen zu Softwaretechnik — 2 SWS

Angestrebte Lernergebnisse	Die Teilnehmer haben ein Verständnis für die Herausforderungen, die bei der Entwicklung großer Software-Systeme auftreten, und kennen Konzepte und Methoden der Softwaretechnik, um diesen Herausforderungen zu begegnen. Dies schließt Kenntnisse über die Architektur größerer Software-Systeme und über Vorgehensmodelle zu deren systematischer Entwicklung im Team ein. Die Teilnehmer besitzen Grundkenntnisse einer iterativ, zyklischen Vorgehensweise sowie der Gestaltung interaktiver Systeme und können diese in den Zusammenhang von softwaretechnischen Aktivitäten wie Kontextanalyse, Anforderungsermittlung und Anwendungsmodellierung einbetten. Dabei können sie auch den Bezug zum Qualitätsbegriff für Software herstellen.

Quelle: http://www.informatik.uni-hamburg.de/Info/Studium/Modulhandbuch_8.0.pdf

Clean Code im Curriculum (FH Wedel)

VORLESUNGSÜBERSICHT
Bachelor-Studiengang Informatik (B_Inf11.0)

Beginn WS

Semester	Gruppe	Lehrveranstaltung	SWS V	+	Ü	ECTS	Summe ECTS / Sem.
1	Informatik	Grundlagen der Funktionalen Programmierung	2	+	3	5	
		Informationstechnik	4	+	0	5	
		Programmstrukturen 1	3	+	1	5	
	Mathematik / Naturw. Grundlagen	Analysis	3	+	1	4	
		Diskrete Mathematik	5	+	3	7	
	Fächerübergreifende Grundlagen	Digitaltechnik 1	2	+	0	2	
		Prakt. Digitaltechnik	0	+	1	2	30
2	Informatik	Programmstrukturen 2	4	+	2	6	
		Rechnerstrukturen	2	+	0	2	
		Workshop Assembler	0	+	4	4	
	Mathematik / Naturw. Grundlagen	Aufgabe OR	0	+	1	1	
		Automaten und Formale Sprachen	3	+	1	4	
		Lineare Algebra	2	+	2	4	
		Operations Research	3	+	0	3	
		Statistik 1	3	+	1	4	
	Fächerübergreifende Grundlagen	Digitaltechnik 2	2	+	0	2	30
3	Informatik	Algorithmen und Datenstrukturen in C	4	+	2	8	
		Anwendungsentwicklung in ERP-Systemen	2	+	2	4	
		Datenbanken	4	+	2	6	
		Methoden der Softwaretechnik	2	+	0	2	
		Programmier-Praktikum	0	+	0	2	
		UNIX	2	+	2	4	
	Fächerübergreifende Grundlagen	Allgemeine Betri					
4	Informatik	Datenbankmana					
		Echtzeitsysteme					
		Geschäftsproze					
		Interface-Techno					
		Objektorientierte					
		Rechnernetze					
		Software-Design					
		Softwaretechnik					
5	Informatik	Anwendungen d					
		Betriebssysteme					
		Compilerbau					
		Prakt. Echtzeitsysteme	0	+	2	2	
		Prakt. Rechnernetze	0	+	2	2	
		(AS) Prozessmodellierung und Anwendung	1	+	3	4	
		(AS) Softwarequalität	2	+	0	2	
		(AS) Systemanalyse	2	+	0	2	
	Fächerübergreifende Grundlagen	Assistenz	0	+	2	4	
		Communication Skills	0	+	2	2	
		Medienrecht	2	+	0	2	
		Projektmanagement	2	+	0	2	30
6	Informatik	Seminar	0				
		Softwareprojekt	0				
		(AS) IT-Sicherheit	2				
		(AS) Systemkonzepte im E-Commerce	1				
		(AS) Workshop Webservices	0				
	Fächerübergreifende Grundlagen	Datenschutz	3	+	0	2	
		(AS) Grundlagen der Computergrafik	2	+	2	4	
		(AS) Technologie der Mediengestaltung	2	+	2	4	
	Auslandssemester = 22 ECTS -Punkte (ersetzt (AS))						30
7	Externe Praxisphasen / Bachelor-Thesis	Bachelor-Thesis	0	+	0	12	
		Betriebspraktikum (mind. 12 Wochen)	0	+	0	17	
		Mündliche Abschlussprüfung	0	+	0	1	30

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Prakt. Rechnernetze	0	+	2	2
(AS) Prozessmodellierung und Anwendung	1	+	3	4
(AS) Softwarequalität	2	+	0	2
(AS) Systemanalyse	2	+	0	2

Auslandssemester = 22 ECTS -Punkte (ersetzt (AS))

Externe Praxisphasen / Bachelor-Thesis

Quelle: http://www.fh-wedel.de/fileadmin/bereiche/pruefungsamt/uebersichten/Bachelor_7_3_Beginn_WS/B_Inf11.0_Beginn_WS.pdf

Modul „Softwarequalität“ (FH Wedel)

1.2.16.2.2 Softwarequalität (Teil 12a)

Lehrveranstaltung	Softwarequalität
Dozent(en)	[redacted]
Hörtermin	5
Art der Lehrveranstaltung	Wahl (durch Wahlblock)
Lehrform / SWS	Vorlesung: 2 SWS
ECTS	2
Lehr- und Medienform(en)	Tafel, Beamerpräsentation, Overheadfolien, Handout, Softwaredemonstration

fhwedel
UNIVERSITY OF APPLIED SCIENCES

Inhalt

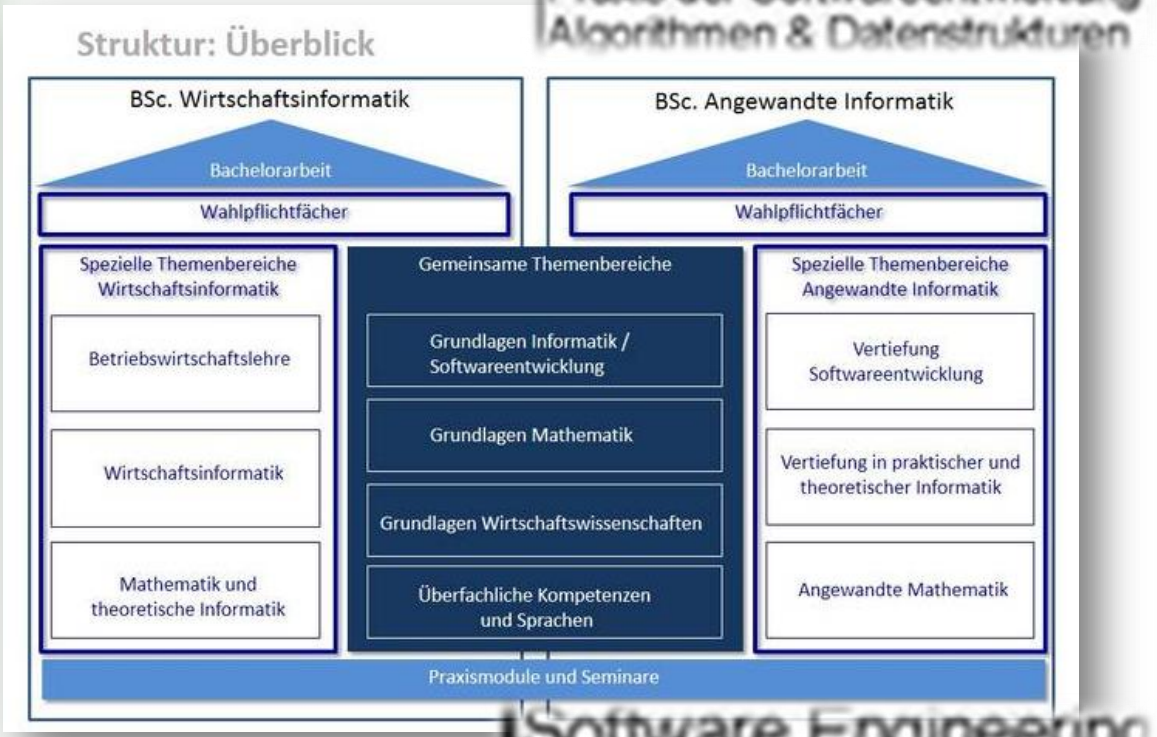
- Einführung und Motivation
  - Definition des Begriffs „Software-Qualität“
  - Bedeutung der Software-Qualität
- Merkmale der Software-Qualität
- Software-Maße und -Metriken
- Modelle der Software-Qualität
- Einschränkungen der Software-Qualität und ihre Gründe
- Software-Qualitätsmanagement
  - Aufgabenbereiche
  - Grundlegende Prinzipien
- Maßnahmen der Software-Qualitätssicherung
  - Konstruktive Maßnahmen
    - Prozessbezogene Maßnahmen

88

    - Produktbezogene Maßnahmen
  - Analytische Maßnahmen
    - Statische Prüftechniken
    - Dynamische Prüftechniken
- Usability als benutzerzentriertes Qualitätsmerkmal
  - Psychologische Rahmenbedingungen
  - Arbeitswissenschaftliche Grundlagen
  - Gestaltungstheorien und -modelle
  - Grundlegende Interaktionstechniken
    - Kommandosprachen
    - Menütechnik
    - Formulartechnik
    - Direkte Manipulation
  - Usability im Entwicklungsprozess

Quelle: http://www.fh-wedel.de/fileadmin/bereiche/pruefungsamt/modulhandbuecher/Bachelor_Informatik_11_0.pdf

Clean Code im Curriculum (FH Nordakademie)



Praxis der Softwareentwicklung
Algorithmen & Datenstrukturen

Software Engineering
Softwarequalitat
Gestaltung von Inform

Modulübersicht Angewandte Informatik Bachelor of Science (gültig ab A14)
Technische und formale Grundlagen
Technische Grundlagen der Informatik 1
Technische Grundlagen der Informatik 2
Formale Grundlagen der Informatik
Diskrete Mathematik 1
Diskrete Mathematik 2
Angewandte Mathematik
Angewandte Informatik
Programmierung 1
Programmierung 2
Praxis der Softwareentwicklung
Algorithmen & Datenstrukturen
Usability Engineering
Software Engineering
Softwarequalität
Gestaltung von Informationssystemen
Datenbanksysteme
Internet Anwendungsarchitekturen
IT-Sicherheit
Programmierparadigmen
Informatikseminar
Fachübergreifende Grundlagen
Informatik und Gesellschaft
IT-Organisation und Projektmanagement
Allgemeine Volkswirtschaftslehre
Allgemeine Betriebswirtschaftslehre
4 Wahlpflicht
Wahlpflichtfach 1
Wahlpflichtfach 2
5 Studium Generale
Englisch 1
Englisch 2
wissenschaftl. Arbeiten und Methoden
Seminare aus dem aktuellen Angebot
6 Abschlussarbeit
Bachelorthesis
7 Praxisanteile / Praktika
Praxismodule

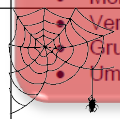
NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

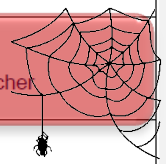
Quelle: http://www.nordakademie.de/de/bachelorstudiengaenge/angewandte-informatik

Modul „Programmierung 2“ (FH Nordakademie) 2011/2012

Curriculum „Programmierung 2“ (I09+I10)

Vorlesung	3. Semester	4. Semester
1	• Vorstellungsrunde • Organisatorisches • Erstes Java Programm • Elementare Konstrukte • Primitive Datentypen • Codegestaltung	• Organisatorisches • Clean Code • Agile Softwareentwicklung • Test First • Techniken agiler Programmierung
2	• Klassen und Objekte • Attribute und Methoden • Eclipse • Concurrent Versioning System	• Code Retreat
3	• Arrays • Strings • Vererbung • Polymorphie	• Threads und Prozesse • Nebenläufigkeit • Synchronisation • Monitore
4	• Liskov’sches Substitutionsprinzip • Abstrakte Klassen • Schnittstellen aka Interfaces • Pakete	• Verteilung von Anwendungen • Grundlagen RMI • Umsetzung Chat mit RMI
5	• Object • Wrapper • Collections • Iteratoren	• Graphische Oberflächen • Einführung in SWT • Widgets • Layout Manager • Events
6	• Autoboxing und Unboxing • Variable Argumentenanzahl • Typisierte Klassen / Generics • Aufzählungstypen	• GUI Builder • Jigloo • Entwurf graphischer Anwendungen
7	• Fortsetzung Übung 6	• Malen mit SWT • Mouse Events
8	• Ausnahmesituationen • Streams und File I/O	• Anforderungen für die Prüfung • Übungsaufgabe zur Prüfungsvorbereitung
9	• Fortsetzung Übung 7	• Fortsetzung Übungsaufgabe zur Prüfungsvorbereitung





NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

„Praxis der Softwareentwicklung“ (FH Nordakademie) 2013

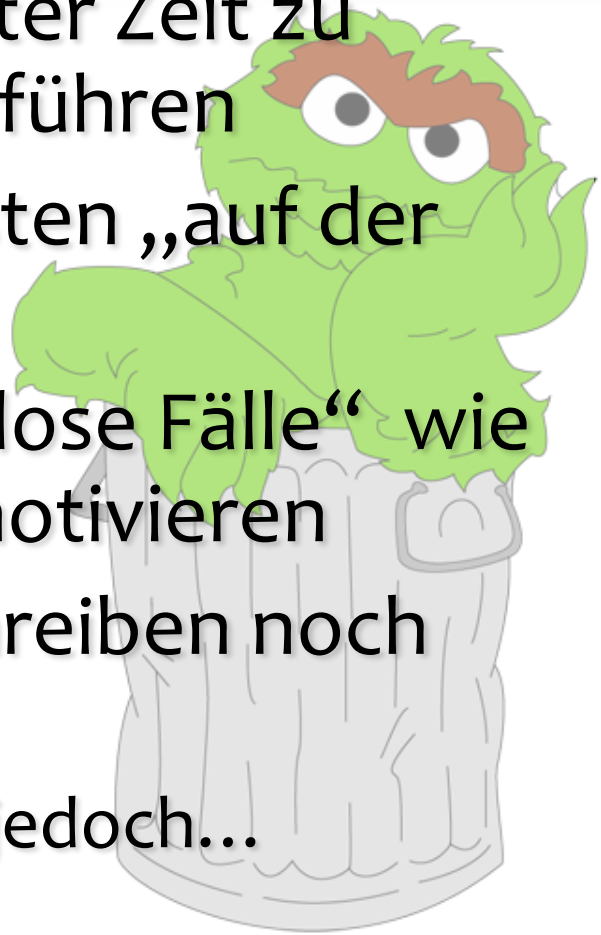
Curriculum „Praxis der Softwareentwicklung“ (ab I11)

Vorlesung	3. Semester	4. Semester
1	• Vorstellungsrunde • Organisatorisches • Erstes Java Programm • Elementare Konstrukte • Primitive Datentypen • Codegestaltung	• Organisatorisches • Agile Softwareentwicklung • Techniken agiler Programmierung
2	• Klassen und Objekte • Attribute und Methoden • Eclipse • Concurrent Versioning System	• Clean Code: Motivation • Clean Code: Names
3	• Arrays • Strings • Vererbung • Polymorphie	• Clean Code: Functions
4	• Liskov’sches Substitutionsprinzip • Abstrakte Klassen • Schnittstellen aka Interfaces • Pakete	• Clean Code: Comments • Clean Code: Formatting
5	• Object • Wrapper • Collections (Java5) • Iteratoren (Java5) • Typisierte Klassen	• Testen • Unit-/Integrations-/Akzeptanz-Tests • JUnit4 •
6	• Fortsetzung Übung 5	• TDD • Code Kata (Bowling Game)
7	• Autoboxing und Unboxing • Variable Argumentenanzahl • Aufzählungstypen • Ausnahmesituationen • Streams und File I/O	• BDD Teststil (given-when-then) • Mocking (Mockito) • Matchers (Hamcrest)
8	• Graphische Oberflächen • Einführung in SWT • Widgets • Layout Manager • Events • Entwurf graphischer Anwendungen	• Code Retreat
9	• Malen mit SWT • Mouse Events • Cliffhanger: Video „Clean Code“ Episode 1	• Anforderungen für die Prüfung • Übungsaufgabe zur Prüfungsvorbereitung

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Manche mögen‘s … schmutzig!

- „Dirty Hacking“ kann in kürzester Zeit zu tollen und sichtbaren Erfolgen führen
- Studenten wollen sich am liebsten „auf der grünen Wiese“ austoben
- Der Dozent möchte „Hoffungslose Fälle“ wie auch „Naturtalente“ nicht demotivieren
- „Spaghetti-Code“ tut beim Schreiben noch nicht sonderlich weh…
  - …ein Partywochenende später, jedoch…



Mein Code von letzter Woche…

Wie kann ich nun die Studenten zum sauber(er)en Programmieren motivieren?!

Mit den richtigen Übungsaufgaben!

Quelle: http://www.bikerszene.de/biker/stummelhund/medien/image/affe_schaut_erstaunt-444228-1.html#img

Greenfield Projekt + Pair Programming

Tic Tac Toe – Part 1

- Create a two-player hotseat „Tic Tac Toe“
  - Implement the GUI in SWT or any other Java Rich Client API you prefer
- Choose meaningful and adequate names for
  - ... packages
  - ... classes
  - ... methods
  - ... variables
- Only Pair Programming!

Quelle: http://www.slideshare.net/BjrnKimminich/agile-software-development-in-practice

Brownfield Projekt + Veränderung im Team

Tic Tac Toe – Part 2

- Add a single player vs. computer game mode to your Tic Tac Toe
- The bot should play smart but doesn't have to be an AI

- Only Pair Programming but not the same pairs as last time!
  - Means: One of each previous pair will work with unknown code!

Quelle: http://www.slideshare.net/BjrnKimminich/agile-software-development-in-practice

Legacy Code + Boy Scout Rule

Tic Tac Toe – Part 3

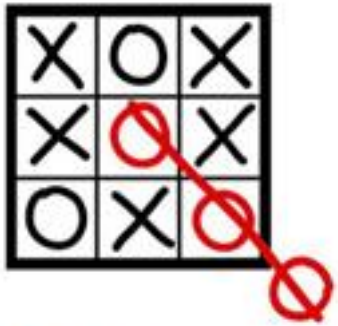
- Same students as last time should pair up…
  - … but work on the code of another pair!!!
- Cleanup time!
  - Rename!
  - Refactor!
  - Reformat!
  - Remove redundancy!
- Present your results to the original code owners

Quelle: http://www.slideshare.net/BjrnKimminich/agile-software-development-in-practice

Nachträgliches Unit-Testing

Testing Tic Tac Toe

- Create unit tests for various classes of your Tic Tac Toe implementation



- Concentrate on testing logic like
  - Game rules (e.g. taken fields cannot be chosen)
  - AI (e.g. should always pick center field if empty)
- Why is it harder/impossible to test the UI?

Quelle: http://www.slideshare.net/BjrnKimminich/agile-software-development-in-practice

Sind Clean Coder unter Ihren Bewerbern?

Woran man Clean Coder im Bewerbungsgespräch erkennt

Code-Beispiele aus Bewerbungsgesprächen

```
Query query = createQuery(hql);
if (query.list() != null && !query.list().isEmpty()) {
    return (OrderReferenceBo) query.list().get(0);
}
return null;
```

1x `query.list()` in lokale Variable!

3x `query.list()` kann sehr „teuer“ sein!

```
Iterator<Integer> it = list.iterator();
while (it.hasNext()) {
    final Integer each = it.next();
    if (each.intValue() == 5) {
        list.remove(each);
    }
}
```

Listenmanipulation während Iteration!

`it.remove();`

Weitere Auffälligkeiten?

Methode auslagern:
`if (!isEmpty(result)){…}`

```
Query query = createQuery(hql);
if (query.list() != null && !query.list().isEmpty()) {
    return (OrderReferenceBo) query.list().get(0);
}
return null;
```

`return null` ist generell schlechter Stil

Warum Liste laden, wenn man immer nur 1 Element haben will?

```
Iterator<Integer> it = list.iterator();
while (it.hasNext()) {
    final Integer each = it.next();
    if (each.intValue() == 5) {
        list.remove(each);
    }
}
```

Nichtssagende Variablennamen:
`list`, (`each`, `it`)

Magic Number

Tipps

- Immer auch einen „Techniker“ dabei haben
- Codebeispiele mit einem konkreten technischen Problem sollten zusätzlich auch subtilere Probleme enthalten
  - schlechte Lesbarkeit
  - schlechte Variablen-, Methoden- & Parameternamen
  - sinnlose oder falsche Kommentare/Javadocs
- Lieber ein umfangreicheres Codebeispiel mit mehreren Problemen unterschiedlicher Natur als viele isolierte Codeschnipsel…
- … dann aber auch genug Zeit dafür einplanen!

Clean Code-Fortbildung im Unternehmen

Beispiel: Kühne+Nagel (AG & Co.) KG

- 2-tägige Schulung in Clean Code & TDD
  - Für alle „programmierenden“ Mitarbeiter (inkl. Externer)
    - Software Engineers
    - Software Architects
    - Quality Engineers

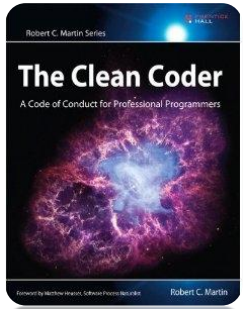
- Firmeninternes „Code Retreat“
- Persönliche Weiterbildung wird gefördert
  - Ein „Exploration Day“ pro Monat
  - Vorträge von Entwicklern für Entwickler („Java Jam“)
- Teilnahme an einschlägigen Konferenzen
  - W-JAX, SeaCon, C.C.D., Agile.ee, …

Clean Code Commitment bei K+N

Wir sind professionelle Software-Ingenieure

- Wir brechen keine Builds oder Testsuiten.
- Wir verwenden gute Namen.
- Unsere Methoden tun eine Sache gut.
- Unsere Methoden und Klassen sind klein.
- Wir halten uns an Konventionen, Muster und Praktiken.
- Wir haben Tests.
- Wir eliminieren Bugs, wo immer wir sie finden.
- Andere können lesen und verstehen, was wir schreiben.
- Wir denken an die Personen, die warten müssen, was wir hinterlassen.
- Wir handeln als Team.
- Wir sind @autoren von "Clean Code".

KUEHNE+NAGEL



Fazit

Und die Moral von der Geschicht‘…

- Clean Code ist nichts, was man erst nach X Jahren Berufserfahrung erlernen kann
- „Einsteiger“ erkennen oftmals den Mehrwert von Clean Code eher als „alte Hasen“
- Themen wie Clean Code, Agile Methoden sowie Software-Wartbarkeit sollten fester(er) Bestandteil von Vorlesungen & Prüfungen sein
- Unternehmen sollten mithelfen ihre Entwickler zu „Software Craftsmen“ zu machen
- **Clean Code lohnt sich. Immer.**

Vielen Dank…

… für Ihre Aufmerksamkeit!