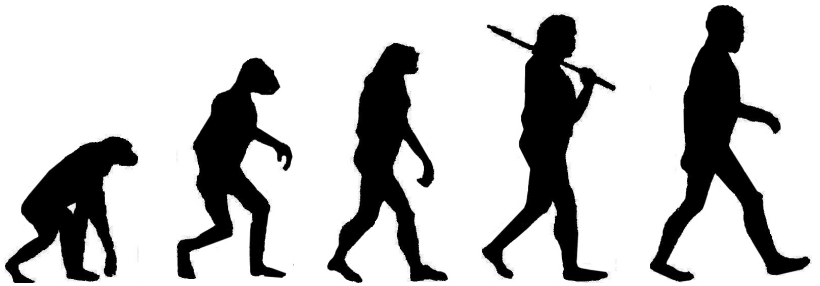
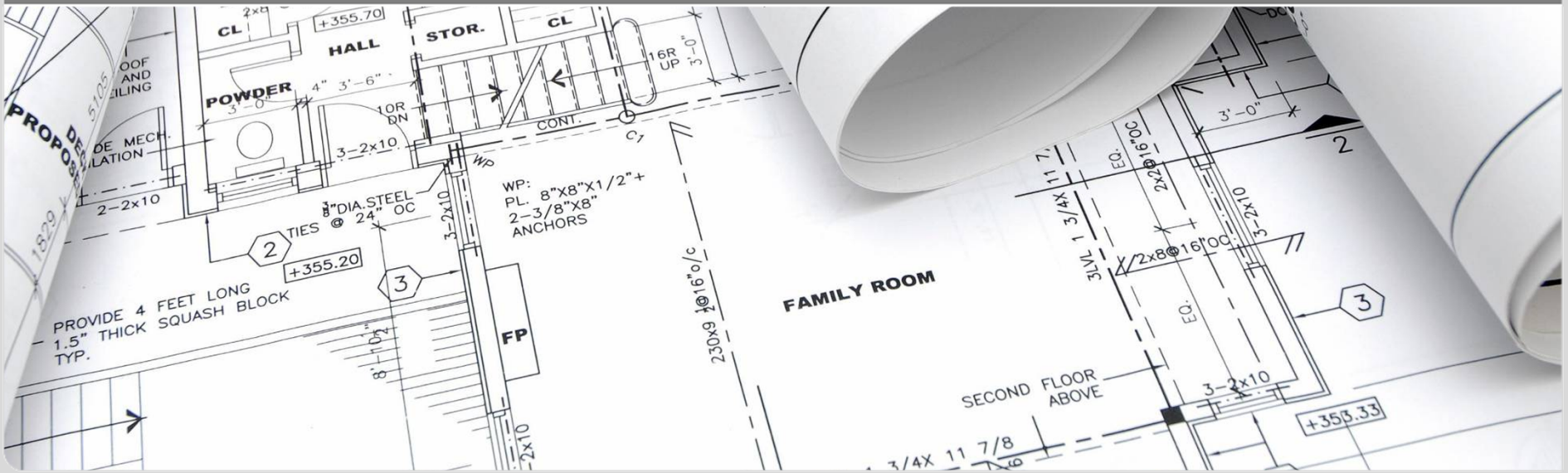
Contracts im Software Engineering

Karlsruher Entwicklertag 2012
Karlsruhe, 10. Mai 2012
Ben Romberg | Hagen Buchwald



Institut AIFB – KOMPLEXITÄTSMANAGEMENT



andrena OBJECTS
Experts in agile software engineering

Programmausschnitt: Agile Day (10. Mai 2012)

Uhrzeit	Main Track (Agile Reengineering) (Raum Baden)	Agilität und Qualität - Tests / Contracts / Clean Code (Raum Fidelitas)	Agile Entwicklungsprozesse (Raum Kraichgau)	Agile Teams / Collaboration (Raum Rebland)
09:00 - 09:15	**Begrüßung**			
09:30 - 10:15	**Test by Contract** Jonas Bergström	**Und doch - es lohnt sich: Agile Software Engineering** Simona Dumitru (SAP) Ralf Pannemans (SAP) Fahd Al-Fatish (andrena objects ag)	**Agile Konzepte im Unternehmen verankern** Prof. Dr. Heinz Züllighoven (C1 WPS) Jörn Koch (C1 WPS)	**Produkt-Innovationen gefällig? Frag deine Mitarbeiter!** Victoria Schiffer (XING)
10:15 - 10:45	**Kaffeepause**			
10:45 - 11:30	**Continuous Architecture Management: Erkennen und Verhindern von Struktureller Erosion** Ingmar Kellner (hello2morrow)	**Design by Contract - What's Next?** Ben Romberg (andrena objects ag) Hagen Buchwald (andrena objects ag)	**Spinning als Kern für das agile Vorgehen** Ralf Westphal (Freiberufler)	**Gestern Entwickler, heute ScrumMaster - Welche Skills braucht ein ScrumMaster wirklich?** Dr. Jürgen Hoffmann (GoAGile) Heiko Stapf (Cybermanufaktur)
11:45 - 12:30	**Refactoring von Legacy Code - Ein Praxisbericht** Andreas Leidig (msg Gillardon) Nicole Rauch (msg Gillardon)	**Asynchronous Pair Programming** Elmar Jürgens (cqse)	**Softskills für Entwickler** Christoph Mathis (improuv) Pierluigi Pugliese (improuv)	**Einer für alle, alle für einen** Markus Gärtner (it-agile GmbH) Meike Mertsch (it-agile GmbH)
12:30 - 13:30	**Mittagspause**			

Quelle: http://www.entwicklertag.de/agile-day

andrena OBJECTS
Experts in agile software engineering

Programmausschnitt: Agile Day (10. Mai 2012)

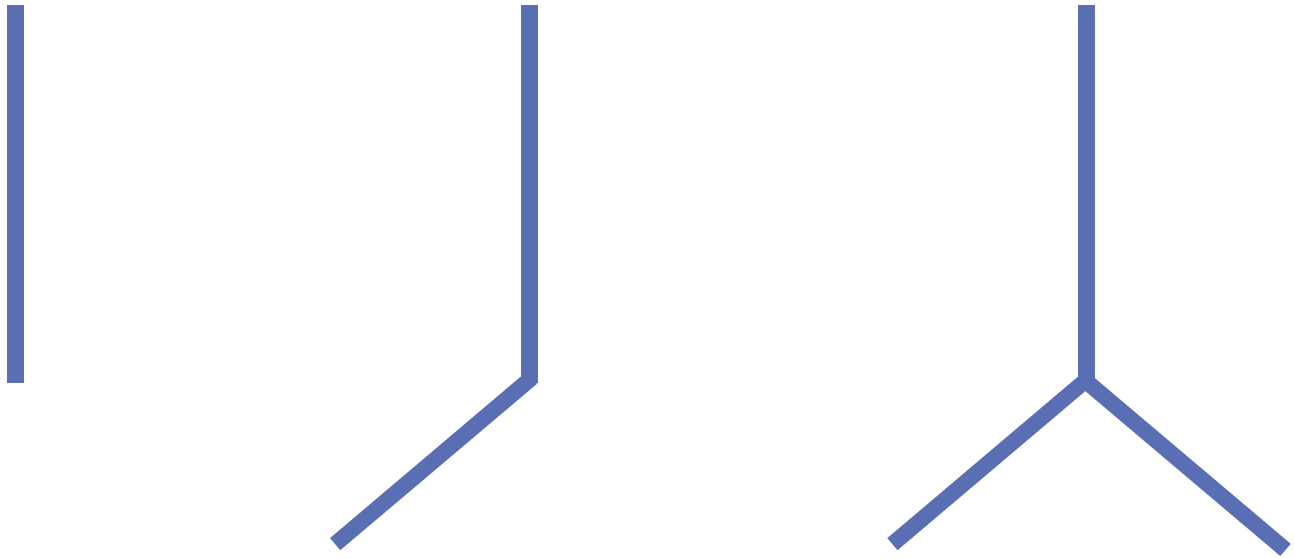
Uhrzeit	Main Track (Agile Reengineering) (Raum Baden)	Agilität und Qualität - Tests / Contracts / Clean Code (Raum Fidelitas)	Agile Entwicklungsprozesse (Raum Kraichgau)	Agile Teams / Collaboration (Raum Rebland)
09:00 - 09:15	**Begrüßung**			
09:30 - 10:15		**Und doch - es lohnt sich: Agile Software Engineering** Simona Dumitru (SAP) Ralf Pannemans (SAP) Fahd Al-Fatish (andrena objects ag)	**Agile Konzepte im Unternehmen verankern** Prof. Dr. Heinz Züllighoven (C1 WPS) Jörn Koch (C1 WPS)	**Produkt-Innovationen gefällig? Frag deine Mitarbeiter!** Victoria Schiffer (XING)
10:15 - 10:45	**Kaffeepause**			
10:45 - 11:30	**Continuous Architecture Management: Erkennen und Verhindern von Struktureller Erosion** Ingmar Kellner (hello2morrow)	**Contracts im Software Engineering** Ben Romberg (andrena objects ag) Hagen Buchwald (andrena objects ag)		**Gestern Entwickler, heute ScrumMaster - Welche Skills braucht ein ScrumMaster wirklich?** Dr. Jürgen Hoffmann (GoAGile) Heiko Stapf (Cybermanufaktur)
11:45 - 12:30	**Refactoring von Legacy Code - Ein Praxisbericht** Andreas Leidig (msg Gillardon) Nicole Rauch (msg Gillardon)		**Softskills für Entwickler** Christoph Mathis (improuv) Pierluigi Pugliese (improuv)	**Einer für alle, alle für einen** Markus Gärtner (it-agile GmbH) Meike Mertsch (it-agile GmbH)
12:30 - 13:30	**Mittagspause**			

Quelle: http://www.entwicklertag.de/agile-day

Gibt es Schutzengel in Java?
Und wenn ja: Wie sehen Sie aus?

Warum sind diese Symbole so wichtig für die Software-Qualität?

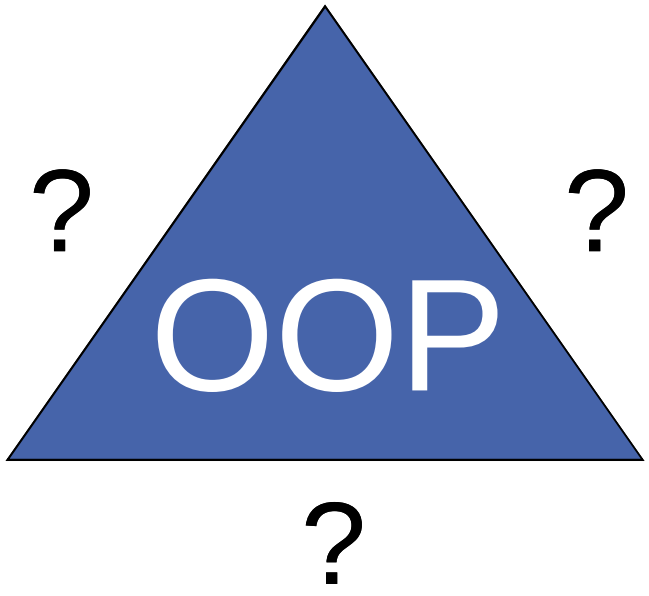
andrena OBJECTS
Experts in agile software engineering



Was haben Pokerkarten und Dominosteine mit Softwareverträgen zu tun?

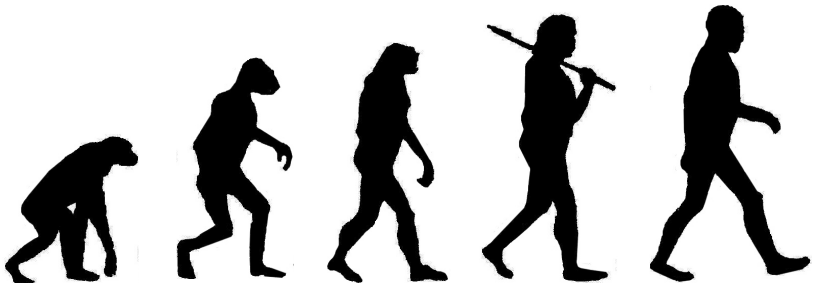
andrena OBJECTS
Experts in agile software engineering

Und was genau ist eigentlich Objektorientierte Programmierung?



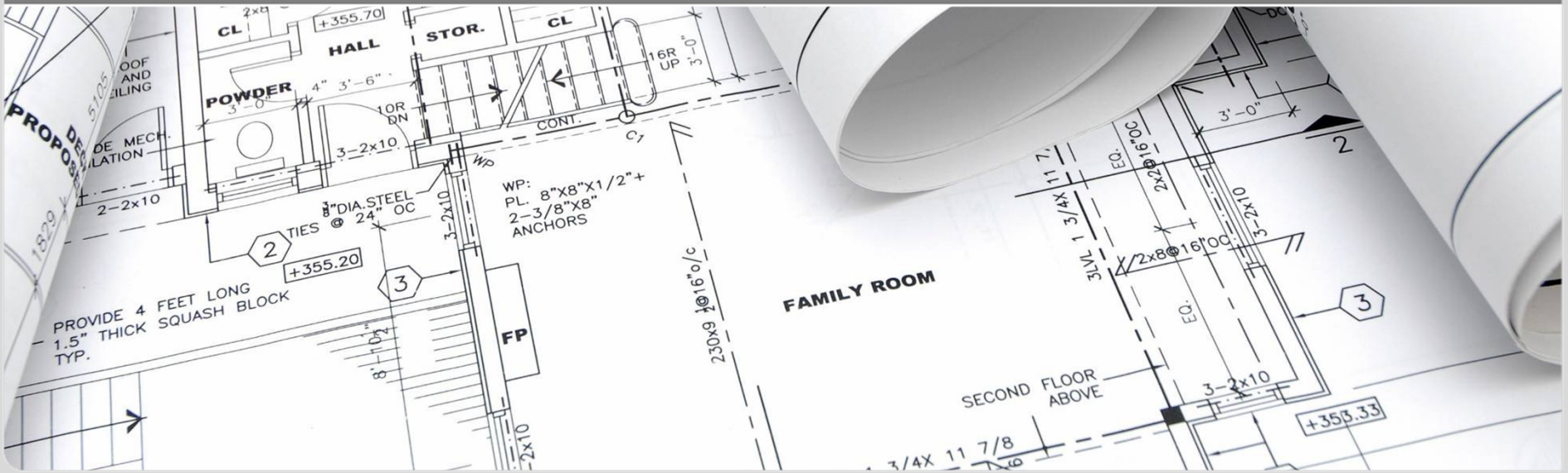
Karlsruher Institut für Technologie

Design by Contract und die Professionalisierung des Software-Engineerings



Karlsruher Entwicklertag 2011
Hagen Buchwald

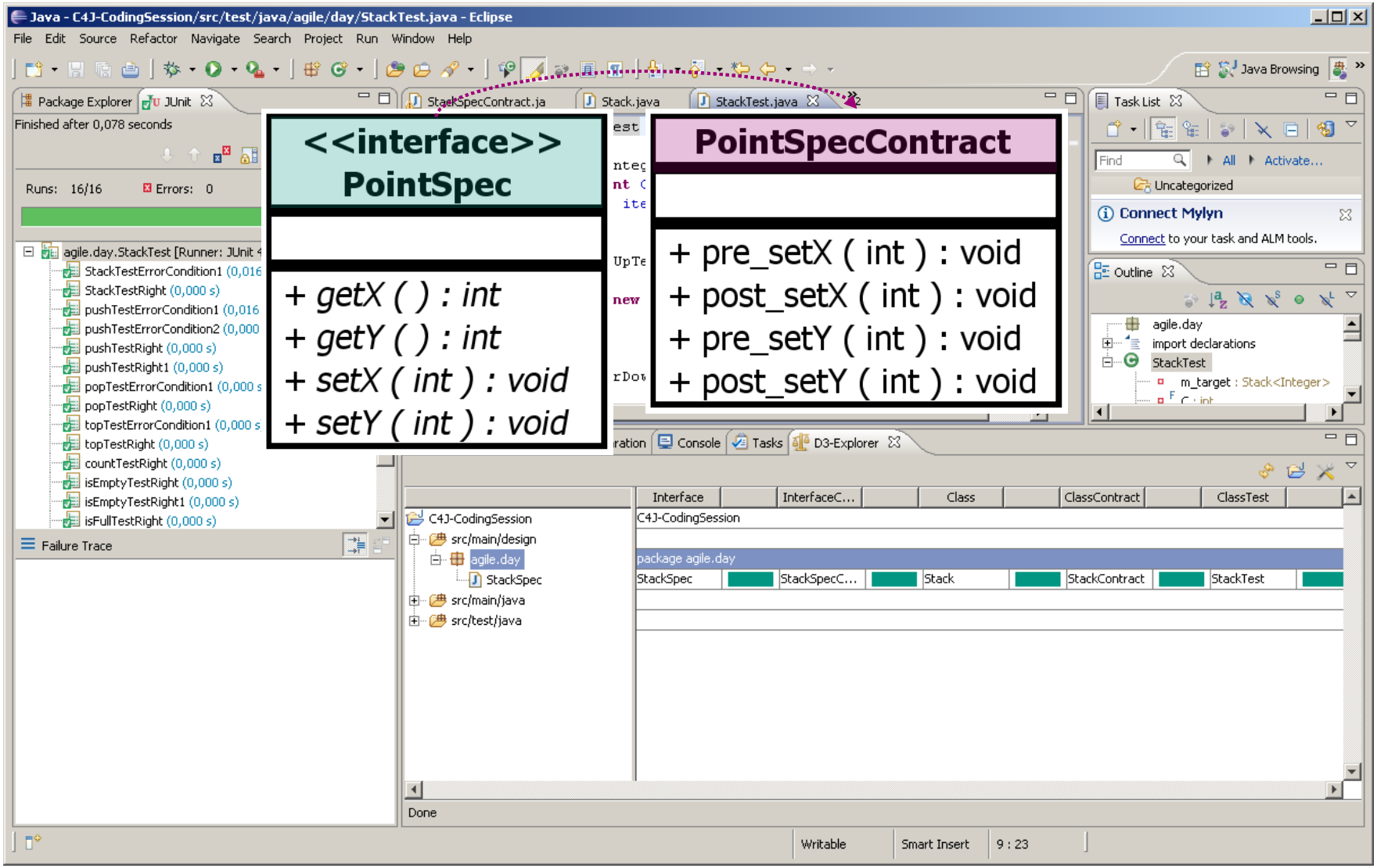
INSTITUT AIFB – KOMPLEXITÄTSMANAGEMENT



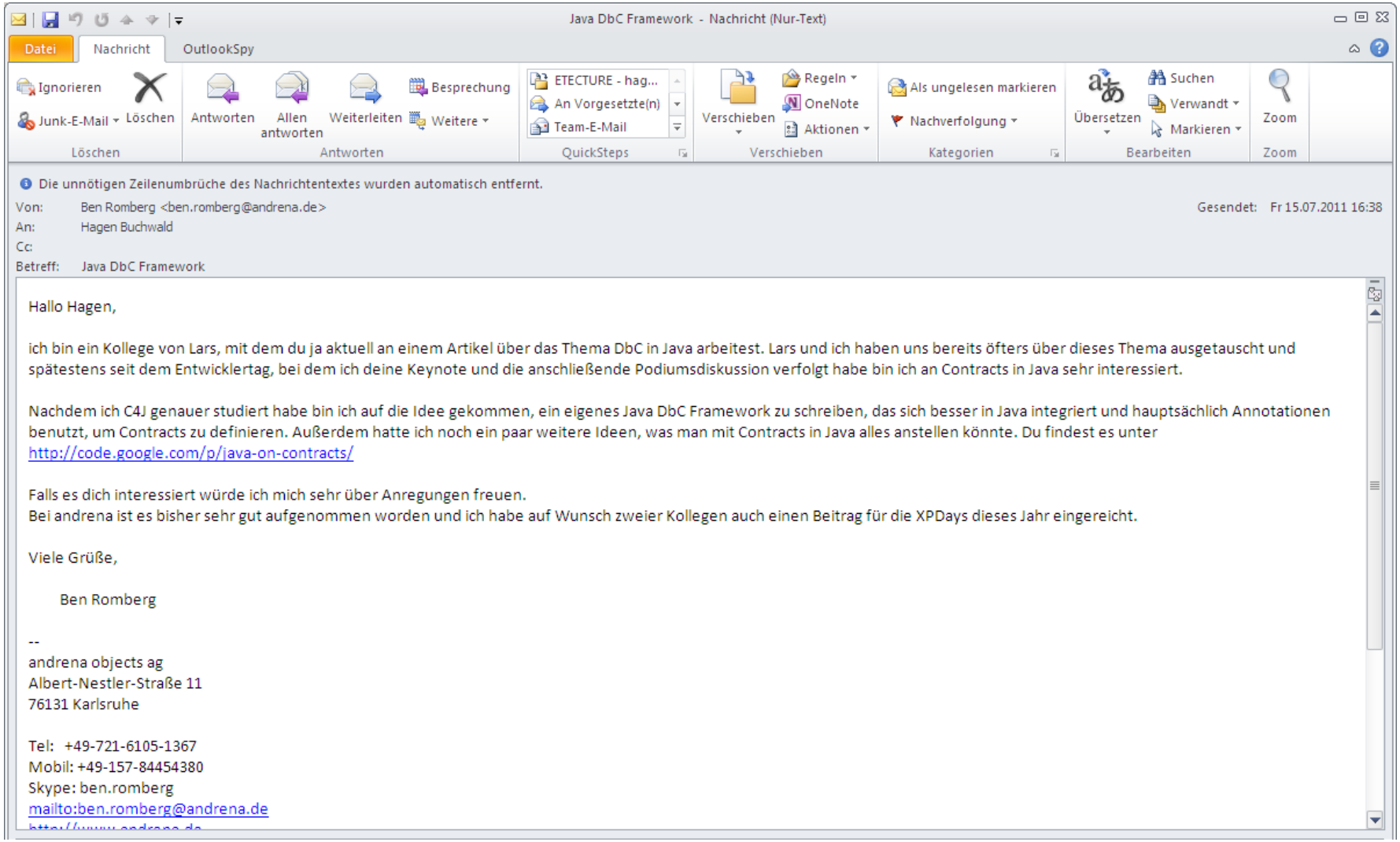
Fragen an Hagen Buchwald auf dem Karlsruher Entwicklertag 2011

1. Zementiert Design by Contract (DbC) nicht den Produktivcode, so dass ggf. Refactorings unterbleiben?
2. Unterstützen Entwicklungsumgebungen wenigstens ein halb-automatisches Refactoring von Vertragsklassen?
3. Welche Werkzeugunterstützung ist generell nötig, um effizient Design by Contract anzuwenden?
4. Kann man – als überzeugter TDD-Verfechter – Tests für TDD guten Gewissens aus Vertragsklassen ableiten?
5. Ist bei nicht-trivialen Beispielen der Vertrag wirklich einfacher als die Implementierung selbst? Oder läuft man Gefahr, mit den Vertragsklassen zu nahe an der Implementierung zu sein?
6. Verletzen Vertragsklassen nicht das DRY-Prinzip der Objektorientierung: Don‘t Repeat Yourself?
7. Können Vertragsklassen auch iterativ und inkrementell entstehen, gemäß des agilen Prinzips?

(Q2) Unterstützen Entwicklungsumgebungen das halb-automatische Refactoring von Vertragsklassen?



Mail von Ben Romberg (andrena objects) im Anschluss an den Entwicklertag 2011 an Hagen Buchwald (KIT).



Java DbC Framework - Nachricht (Nur-Text)

Datei | Nachricht | OutlookSpy

Ignorieren | Junk-E-Mail | Löschen | Antworten | Allen antworten | Weiterleiten | Besprechung | Weitere | ETECTURE - hag... | An Vorgesetzte(n) | Team-E-Mail | Verschieben | Regeln | OneNote | Aktionen | Als ungelesen markieren | Nachverfolgung | Übersetzen | Suchen | Verwandt | Markieren | Zoom

Löschen | Antworten | QuickSteps | Verschieben | Kategorien | Bearbeiten | Zoom

Die unnötigen Zeilenumbrüche des Nachrichtentextes wurden automatisch entfernt.

Von: Ben Romberg <ben.romberg@andrena.de>
An: Hagen Buchwald
Cc:
Betreff: Java DbC Framework

Gesendet: Fr 15.07.2011 16:38

Hallo Hagen,

ich bin ein Kollege von Lars, mit dem du ja aktuell an einem Artikel über das Thema DbC in Java arbeitest. Lars und ich haben uns bereits öfters über dieses Thema ausgetauscht und spätestens seit dem Entwicklertag, bei dem ich deine Keynote und die anschließende Podiumsdiskussion verfolgt habe bin ich an Contracts in Java sehr interessiert.

Nachdem ich C4J genauer studiert habe bin ich auf die Idee gekommen, ein eigenes Java DbC Framework zu schreiben, das sich besser in Java integriert und hauptsächlich Annotationen benutzt, um Contracts zu definieren. Außerdem hatte ich noch ein paar weitere Ideen, was man mit Contracts in Java alles anstellen könnte. Du findest es unter
http://code.google.com/p/java-on-contracts/

Falls es dich interessiert würde ich mich sehr über Anregungen freuen.
Bei andrena ist es bisher sehr gut aufgenommen worden und ich habe auf Wunsch zweier Kollegen auch einen Beitrag für die XPDays dieses Jahr eingereicht.

Viele Grüße,

Ben Romberg

--
andrena objects ag
Albert-Nestler-Straße 11
76131 Karlsruhe

Tel: +49-721-6105-1367
Mobil: +49-157-84454380
Skype: ben.romberg
mailto:ben.romberg@andrena.de

Contracts im Software Engineering
Agenda

andrena OBJECTS
Experts in agile software engineering

1. Was versteht man konkret unter Objektorientierter Programmierung?
2. Ist Java eine vollständig objektorientierte Programmiersprache?
3. Wie können in Java Abstrakte Datentypen modelliert werden?
4. Was sind Softwareverträge – und wie funktionieren sie in Java?
5. Was verändert sich, wenn wir Softwareverträge in Java nutzen?
6. Wie passen Softwareverträge und TDD zusammen?
7. Fragen & Antworten

Was ist Objektorientierte Programmierung?

Definition nach Alan Kay (2003):

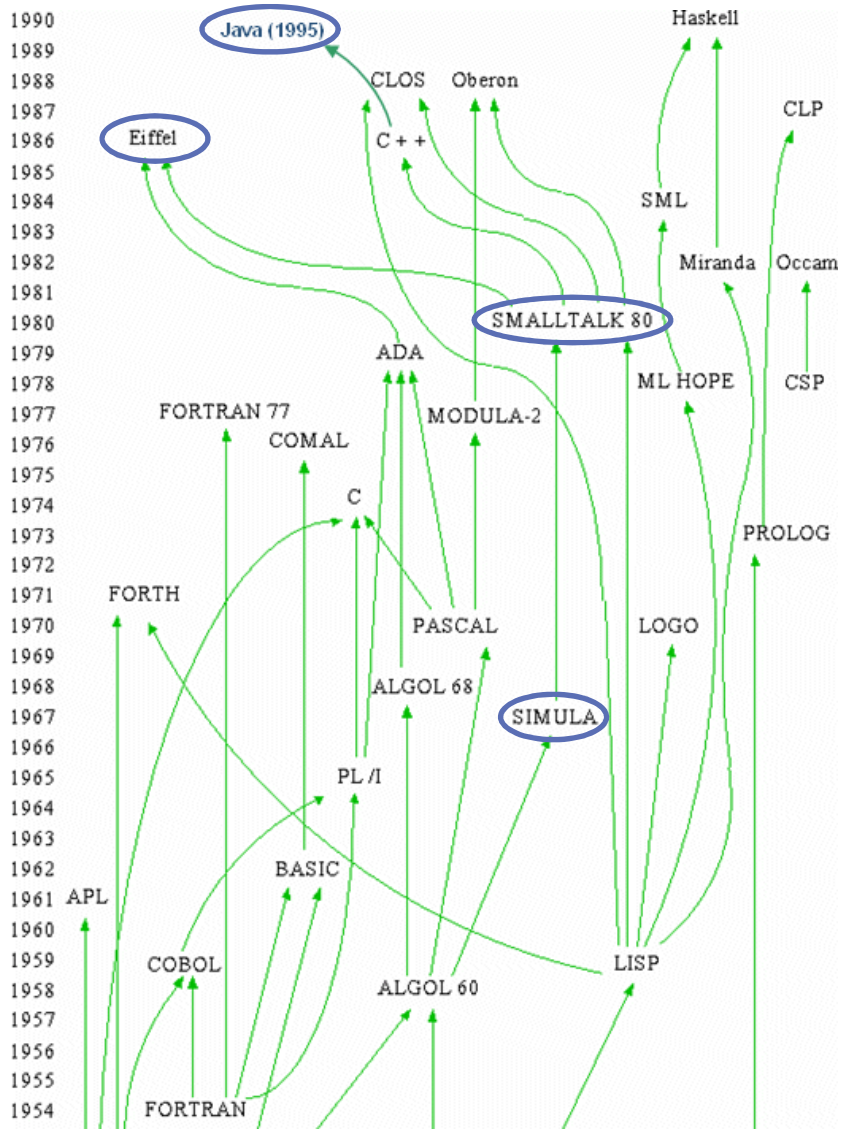
“OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late-binding of all things. It can be done in Smalltalk and in LISP. There are possibly other systems in which this is possible, but I’m not aware of them. ”

“One of the things I should have mentioned is that there were two main paths that were catalysed by Simula.

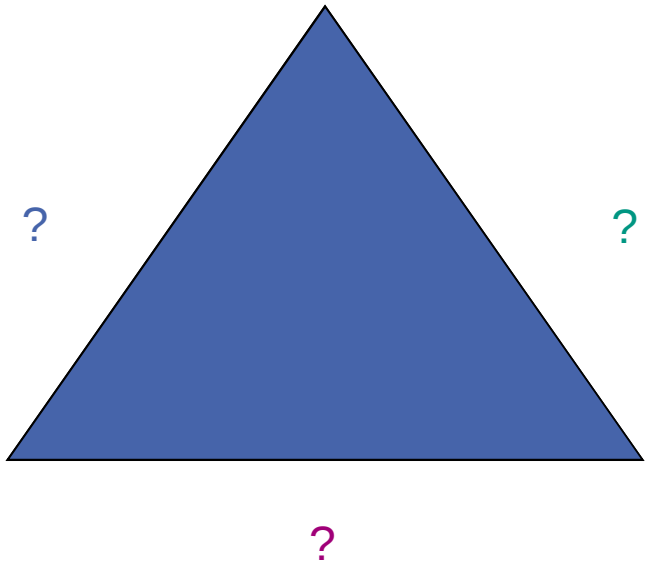
The early one was the bio/net non-data-procedure route that I took.

The other one, which came a little later as an objects of study was abstract data types, and this got much more play.”

Quelle: http://www.purl.org/stefan_ram/pub/doc_kay_oop_en

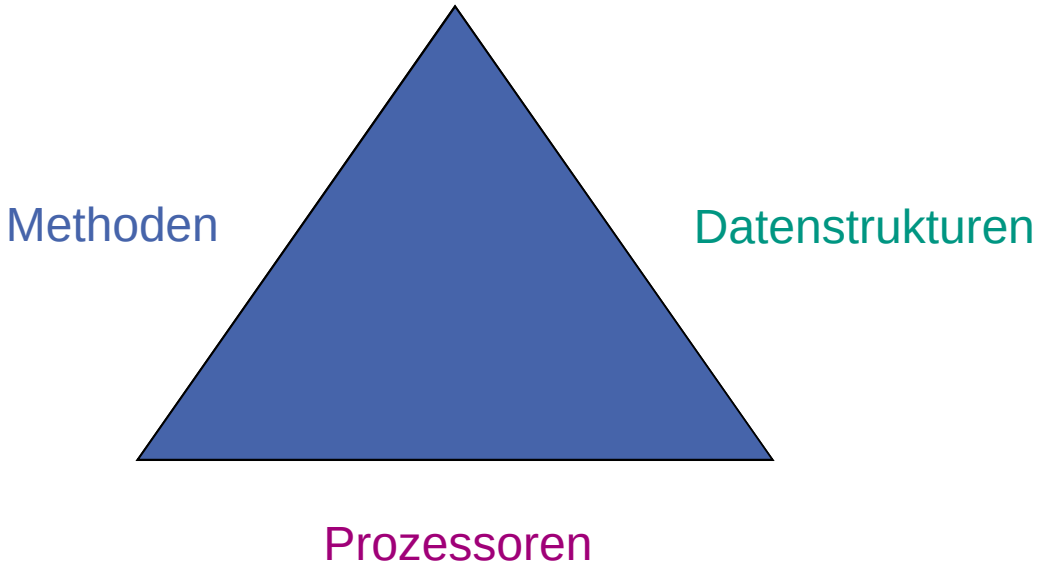


Welches sind die drei Kernelemente eines Softwaresystems?



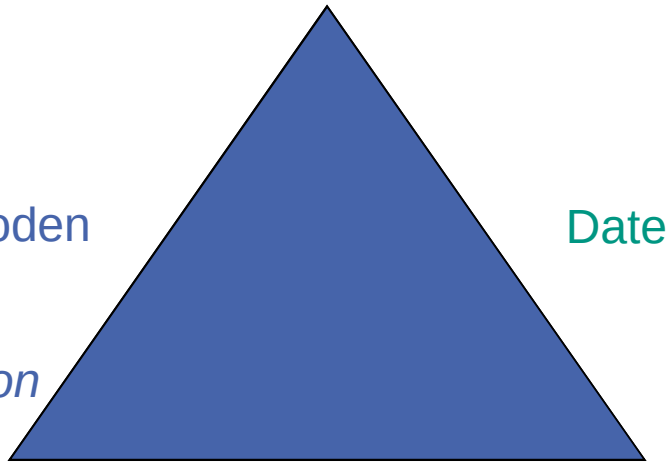
Die drei Kernelemente eines Softwaresystems sind: Methoden, Datenstrukturen und Prozessoren.

Ein Softwaresystem besteht aus Prozessoren,
die Methoden auf Datenstrukturen ausführen.



Objekt-Orientierte Programmierung (OOP). Definition nach Bertrand Meyer.

Ein Softwaresystem besteht aus Prozessoren,
die Methoden auf Datenstrukturen ausführen.



„Prädikat“ ï Methoden

AM

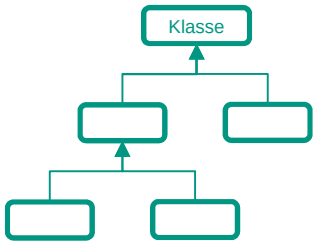
„Was ist die Hauptfunktion des Systems?“

Datenstrukturen ð Objekt

ADT

„Auf was arbeitet das System?“

Prozessoren



OOP ist die Erstellung von Softwaresystemen
als strukturierte Sammlung von Implementierungen
Abstrakter Datentypen (ADTs).

Quelle: Bertrand Meyer, Object-Oriented Software Construction, 1988

Contracts im Software Engineering
Agenda

andrena OBJECTS
Experts in agile software engineering

1. Was versteht man konkret unter Objektorientierter Programmierung?
2. Ist Java eine vollständig objektorientierte Programmiersprache?
3. Wie können in Java Abstrakte Datentypen modelliert werden?
4. Was sind Softwareverträge – und wie funktionieren sie in Java?
5. Was verändert sich, wenn wir Softwareverträge in Java nutzen?
6. Wie passen Softwareverträge und TDD zusammen?
7. Fragen & Antworten

Die 7 Stufen zur Objektorientierung am Beispiel `Eiffel`

Karlsruher Institut für Technologie

Ü

Stufe 7

Mehrfaches und wiederholtes Erben

Man kann Klassen deklarieren, die Erben von mehr als einer Klasse sind
und mehr als einmal erben.

Ü

Stufe 6

Polymorphismus und dynamisches Binden

Programm-Elemente dürfen sich auf Objekte aus mehr als einer Klasse beziehen,
und Operationen dürfen unterschiedliche Realisierungen in unterschiedlichen Klassen haben.

Ü

Stufe 5

Vererbung

Eine Klasse kann als Einschränkung oder Erweiterung einer anderen definiert werden.

Ü

Stufe 4

Klassen

Jeder nicht-einfache Typ ist ein Modul, und jeder Modul höherer Stufe ist ein Typ.

Ü

Stufe 3

Automatische Speicherplatzverwaltung

Unbenutzte Objekte sollen ohne Programmiereingriff
vom unterliegenden Sprachsystem freigegeben werden.

Ü

Stufe 2

Datenabstraktion

Objekte müssen als Implementierung abstrakter Datentypen (ADTs) beschrieben werden.

Ü

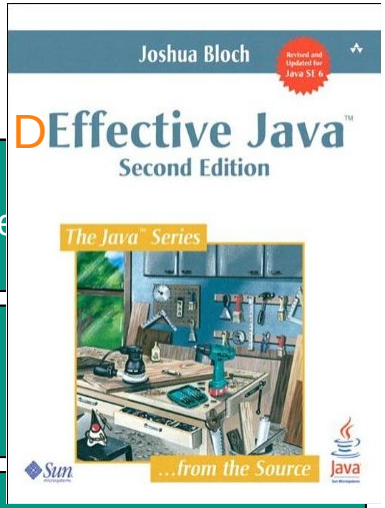
Stufe 1

Objektbasierte, modulare Struktur

Systeme werden auf Grundlage ihrer Datenstruktur modularisiert.

Quelle: Bertrand Meyer, Object-Oriented Software Construction, 1988

Die 7 Stufen zur Objektorientierung am Beispiel Java



Stufe 7 — Mehrfaches und wiederholtes Erben
Man kann Klassen deklarieren, die Erben von mehr als einer Klasse
und mehr als einmal erben.

Stufe 6 — Polymorphismus und dynamisches Binden
Programm-Elemente dürfen sich auf Objekte aus mehr als einer Klasse
und Operationen dürfen unterschiedliche Realisierungen in unterschiedlichen

Stufe 5 — Vererbung
Eine Klasse kann als Einschränkung oder Erweiterung einer anderen definiert werden.

Stufe 4 — Klassen
Jeder nicht-einfache Typ ist ein Modul, und jeder Modul höherer Stufe ist ein Typ.

Stufe 3 — Automatische Speicherplatzverwaltung
Unbenutzte Objekte sollen ohne Programmiereingriff
vom unterliegenden Sprachsystem freigegeben werden.

Stufe 2 — Datenabstraktion
Objekte müssen als Implementierung abstrakter Datentypen (ADTs) beschrieben werden.

Stufe 1 — Objektbasierte, modulare Struktur
Systeme werden auf Grundlage ihrer Datenstruktur modularisiert.

Quelle: Bertrand Meyer, Object-Oriented Software Construction, 1986

Bertrand Meyer (1988): Spezifikation von Software durch zur Laufzeit überprüfbare Verträge.

- IT-Systeme werden immer mächtiger, jedoch damit einhergehend auch immer komplexer – und damit steigt die Wahrscheinlichkeit von Fehlern!
- Dieses Komplexitätsproblem war einer der entscheidenden Treiber der Objektorientierung.
- Bertrand Meyer wies 1988 mit seinem Buch „Objektorientierte Software-Entwicklung“ einen Ausweg aus dieser Komplexitätsfalle.

Quelle: Bertrand Meyer, Object-Oriented Software Construction, 1988

Contracts im Software Engineering
Agenda

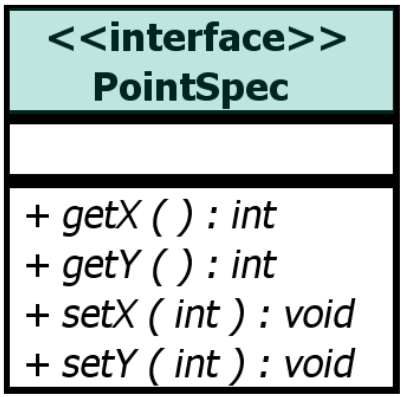
andrena OBJECTS
Experts in agile software engineering

1. Was versteht man konkret unter Objektorientierter Programmierung?
2. Ist Java eine vollständig objektorientierte Programmiersprache?
3. Wie können in Java Abstrakte Datentypen modelliert werden?
4. Was sind Softwareverträge – und wie funktionieren sie in Java?
5. Was verändert sich, wenn wir Softwareverträge in Java nutzen?
6. Wie passen Softwareverträge und TDD zusammen?
7. Fragen & Antworten

Die Klasse `Point` beschrieben in Form eines Abstrakten Datentyps (ADT).

Karlsruher Institut für Technologie

```
TYPE
    PointSpec

FUNCTIONS
    getX() : int
    getY() : int
    setX(int x) : void
    setY(int y) : void


PRE-CONDITIONS
    setX(int x) : x >= 0
    setY(int y) : y >= 0

AXIOMS (POST-CONDITIONS)
    setX(int x) : getX() == x
    setY(int y) : getY() == y
```



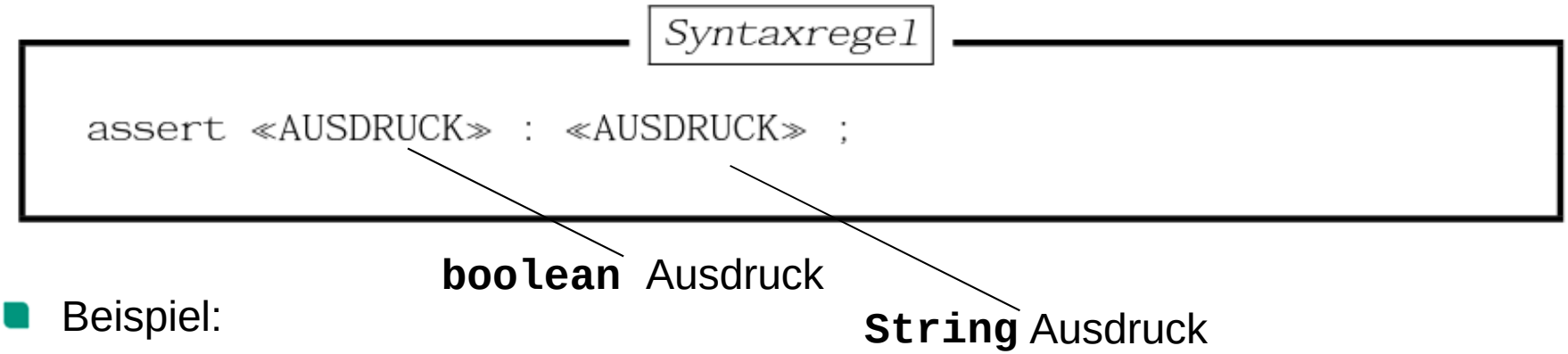
Stufe 2

Datenabstraktion

Objekte müssen als Implementierung abstrakter Datentypen (ADTs) beschrieben werden.

Assertions in Java can *only* be applied to *implemented* methods, not to *abstract* methods or *abstract* classes.

- Syntax von Assertions in Java:



- Beispiel:

```
public double inverse(double x)
{
    assert x != 0 : "x_not_0";
    double result = 0;
    result = 1.0 / x;
    return result;
}
```

Assertions in Java can not be applied on an abstract level and can not be inherited to a subclass!

Syntax rule taken from: Ratz, Seese: Grundkurs Programmieren in Java, 2010

Java ist gemäß der Definition von B. Meyer *keine* vollständig objektorientierte Programmiersprache!

Karlsruher Institut für Technologie

Stufe 7
Mehrfaches und wiederholtes Erben
Man kann Klassen deklarieren, die Erben von mehr als einer Klasse sind
und mehr als einmal erben.

Û

Stufe 6
Polymorphismus und dynamisches Binden
Programm-Elemente dürfen sich auf Objekte aus mehr als einer Klasse beziehen,
und Operationen dürfen unterschiedliche Realisierungen in unterschiedlichen Klassen haben.

Ü

Stufe 5
Vererbung
Eine Klasse kann als Einschränkung oder Erweiterung einer anderen definiert werden.

ü

Stufe 4
Klassen
Jeder nicht-einfache Typ ist ein Modul, und jeder Modul höherer Stufe ist ein Typ.

Ü

Stufe 3
Automatische Speicherplatzverwaltung
Unbenutzte Objekte sollen ohne Programmiereingriff
vom unterliegenden Sprachsystem freigegeben werden.

ü

Stufe 2
Datenabstraktion
Objekte müssen als Implementierung abstrakter Datentypen (ADTs) beschrieben werden.

Û

Stufe 1
Objektbasierte, modulare Struktur
Systeme werden auf Grundlage ihrer Datenstruktur modularisiert.

Ü

Quelle: Bertrand Meyer, Object-Oriented Software Construction, 1988

Contracts im Software Engineering
Agenda

andrena OBJECTS
Experts in agile software engineering

1. Was versteht man konkret unter Objektorientierter Programmierung?
2. Ist Java eine vollständig objektorientierte Programmiersprache?
3. Wie können in Java Abstrakte Datentypen modelliert werden?
4. Was sind Softwareverträge – und wie funktionieren sie in Java?
5. Was verändert sich, wenn wir Softwareverträge in Java nutzen?
6. Wie passen Softwareverträge und TDD zusammen?
7. Fragen & Antworten

Was tun Kinder zuerst, wenn sie miteinander spielen wollen?

Sie legen die Spielregeln fest!

Was machen Unternehmer, wenn sie kooperieren möchten?

Karlsruher Institut für Technologie

Kunde

Rechte

Pflichten

Anbieter

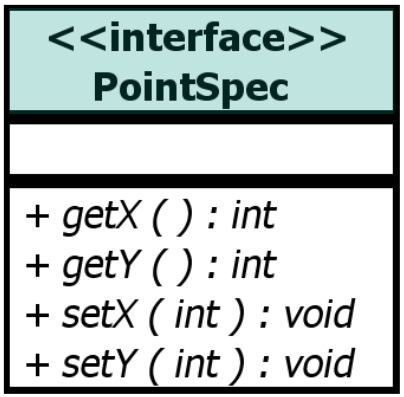
Rechte

Pflichten

Sie schließen einen Vertrag ab!

Die Klasse `Point` beschrieben in Form eines Abstrakten Datentyps (ADT).

```
TYPE
    PointSpec

FUNCTIONS
    getX() : int
    getY() : int
    setX(int x) : void
    setY(int y) : void


PRE-CONDITIONS
    setX(int x) : x >= 0
    setY(int y) : y >= 0

AXIOMS (POST-CONDITIONS)
    setX(int x) : getX() == x
    setY(int y) : getY() == y
```



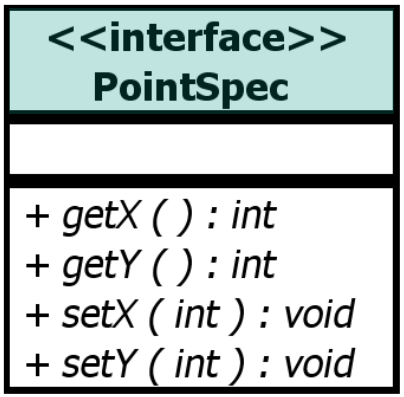
Stufe 2

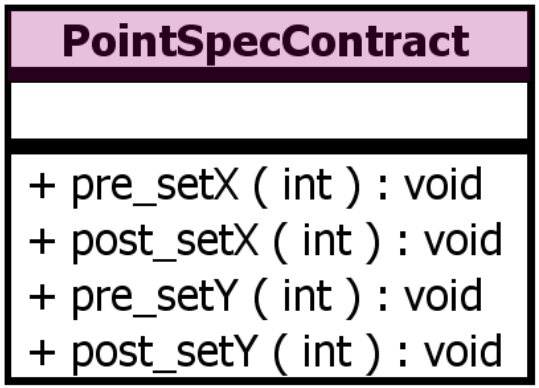
Datenabstraktion

Objekte müssen als Implementierung abstrakter Datentypen (ADTs) beschrieben werden.

Die Klasse `Point` beschrieben in Form eines Abstrakten Datentyps (ADT).

```
TYPE
    PointSpec

FUNCTIONS
    getX() : int
    getY() : int
    setX(int x) : void
    setY(int y) : void


PRE-CONDITIONS
    setX(int x) : x >= 0
    setY(int y) : y >= 0

AXIOMS (POST-CONDITIONS)
    setX(int x) : getX() == x
    setY(int y) : getY() == y
```



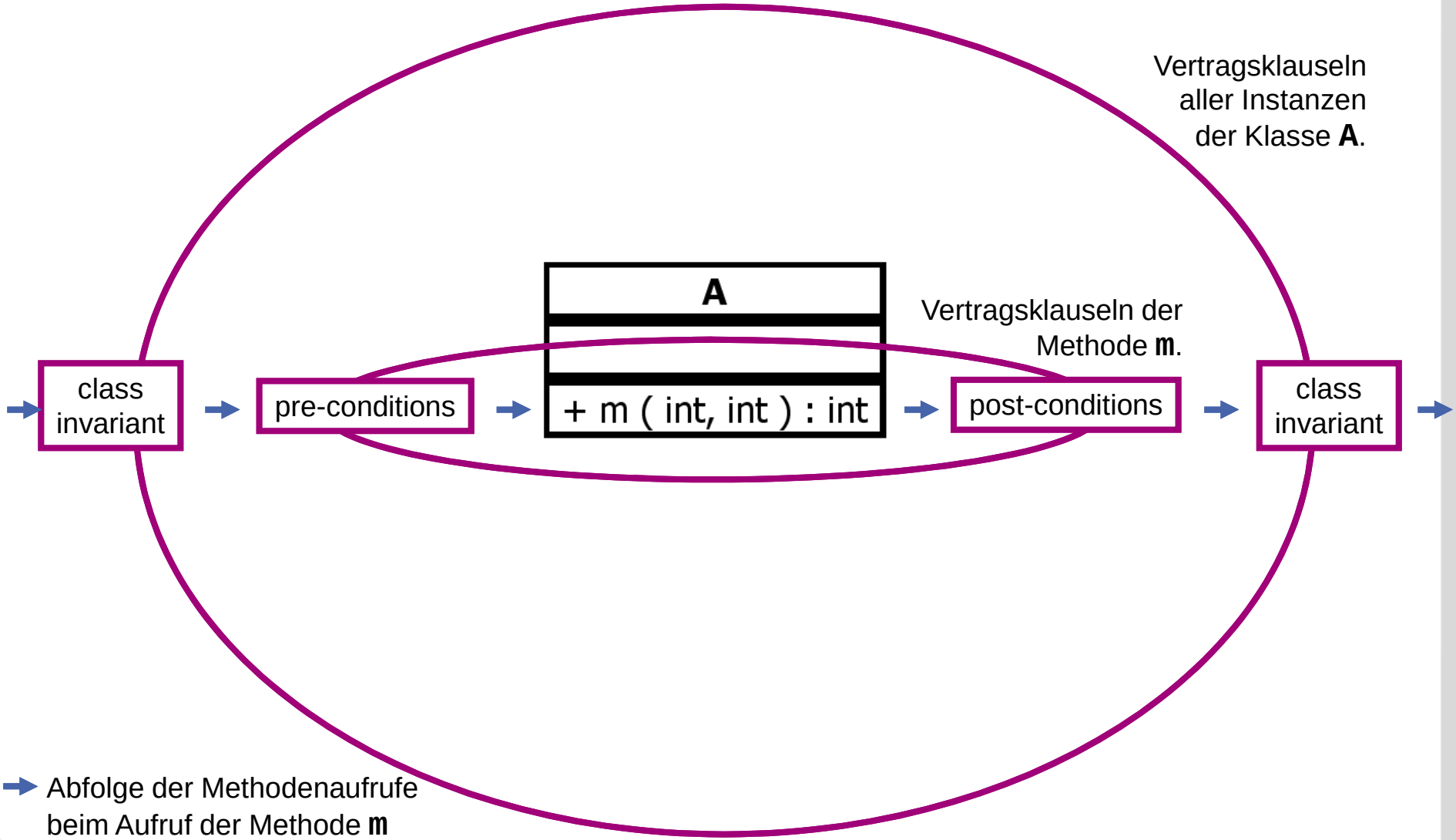


Stufe 2

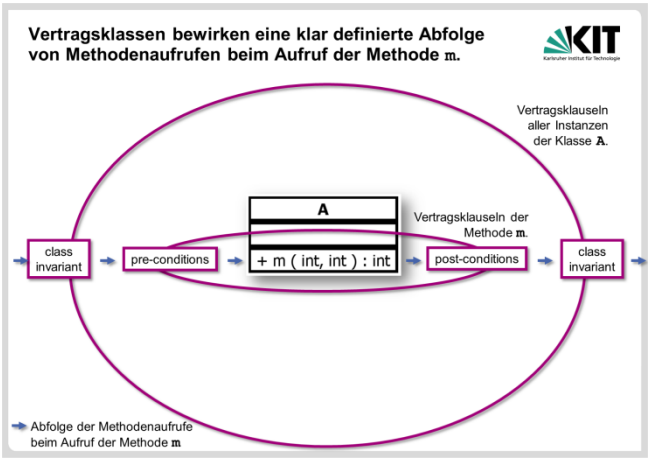
Datenabstraktion

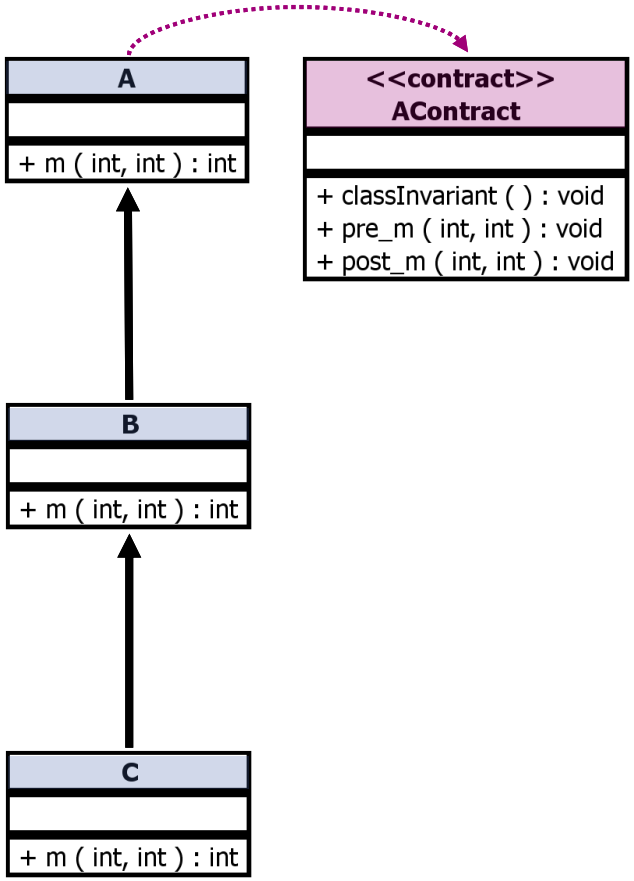
Objekte müssen als Implementierung abstrakter Datentypen (ADTs) beschrieben werden.

Vertragsklassen bewirken eine klar definierte Abfolge von Methodenaufrufen beim Aufruf der Methode `m`.

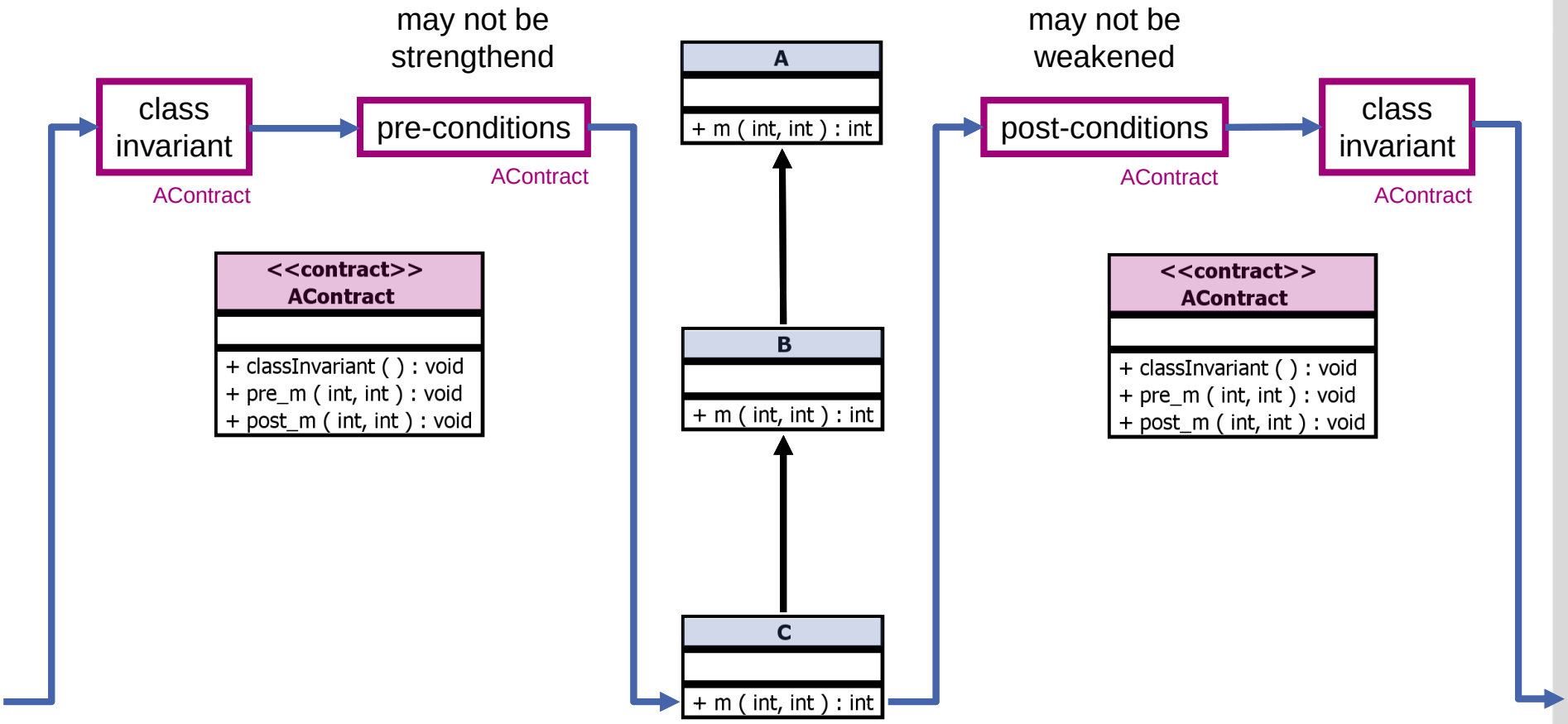


Verträge werden automatisch vererbt, sobald eine geschützte Klasse (bzw. Interface) beerbt wird.





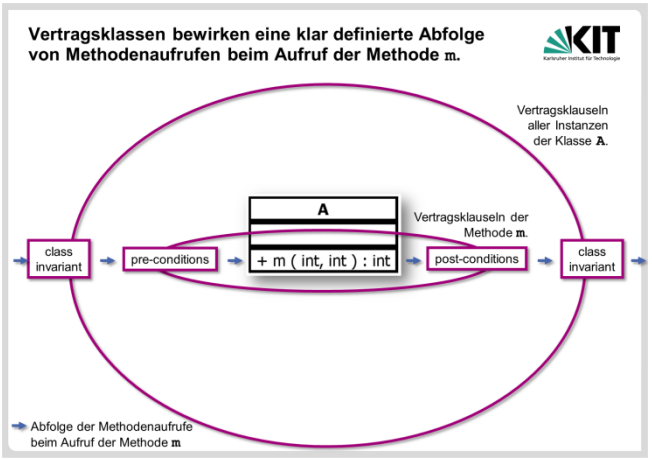
Beispiel der Vererbung beim Aufruf der Methode `c.m(1,2)` mit `c` Instanz von `C`.

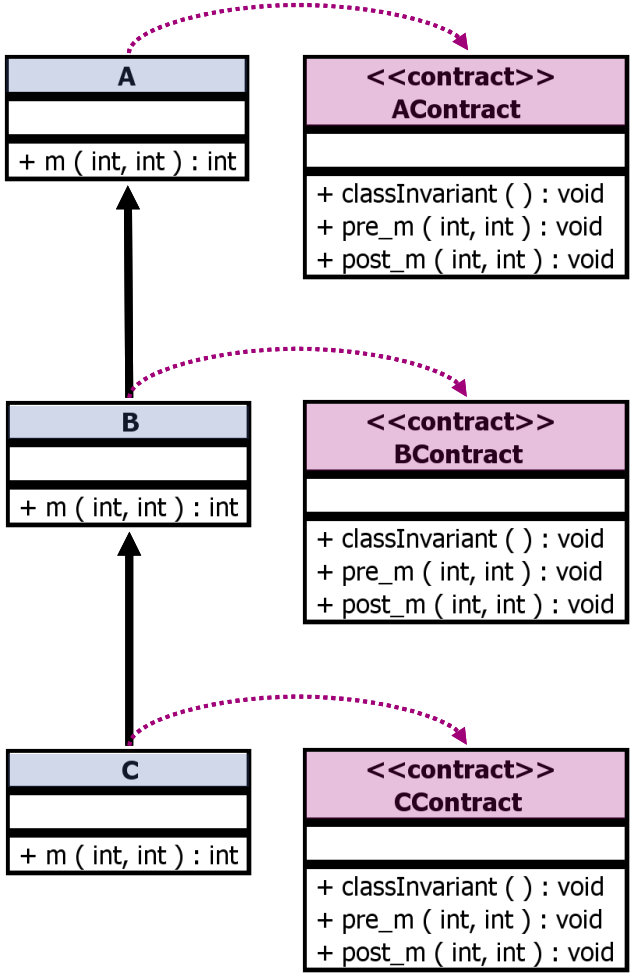


→ Abfolge der Methodenaufrufe beim Aufruf von `c.m(1,2)`

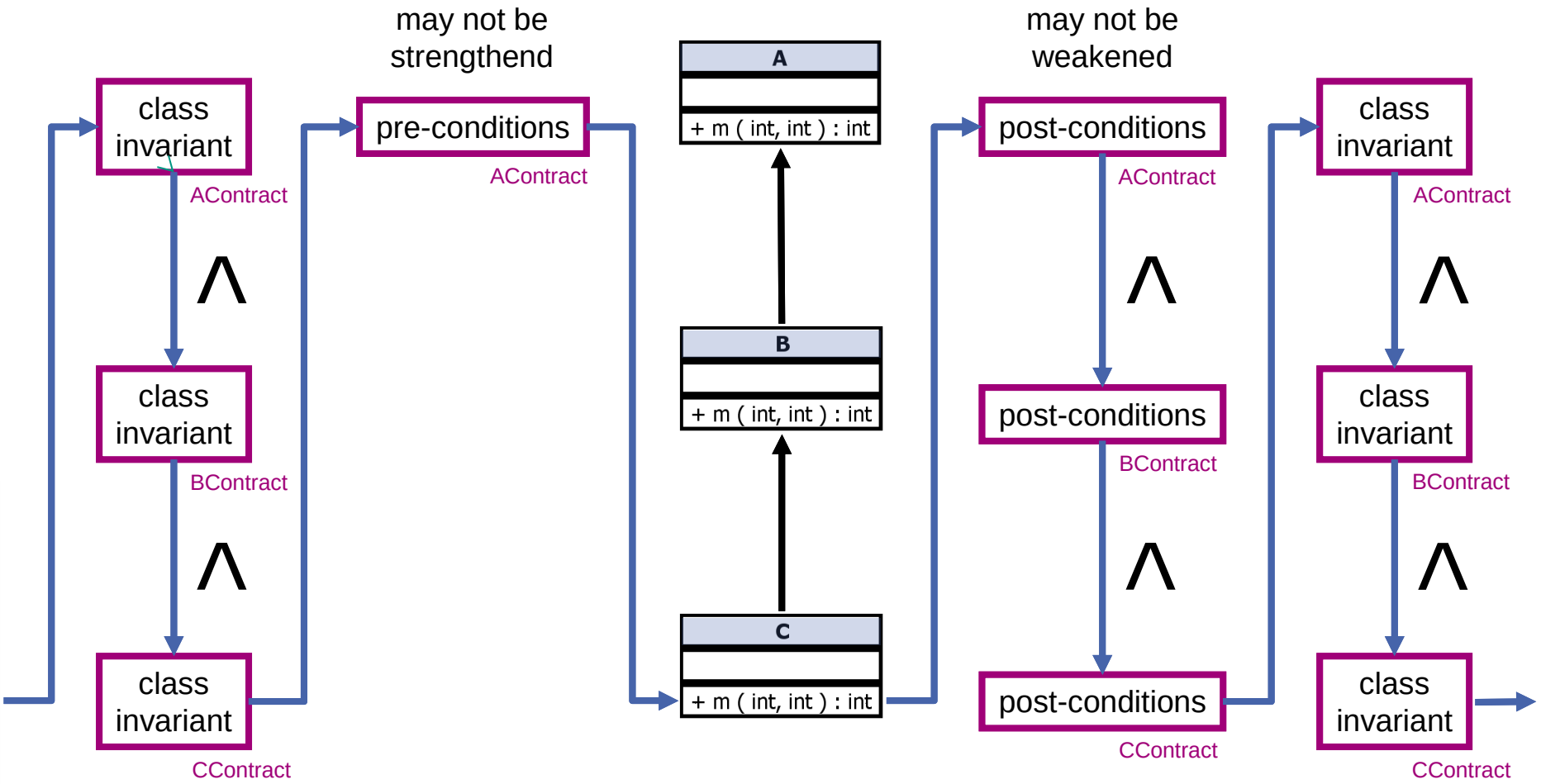
Hinweis: idealisierte Darstellung: Tatsächlich wird aus Optimierungsgründen in C4J auf das Durchlaufen der Klasseninvariante vor dem Prüfen der Pre-Conditions verzichtet.

Beispiel der LSP-Umsetzung in C4J beim Aufruf der überschriebenen Methode `c.m(1,2)` mit `c` Instanz von `C`.





Beispiel der LSP-Umsetzung in C4J beim Aufruf der überschriebenen Methode `c.m(1,2)` mit `c` Instanz von `C`.



→ Abfolge der Methodenaufrufe beim Aufruf von `c.m(1,2)`

Hinweis: idealisierte Darstellung: Tatsächlich wird aus Optimierungsgründen in C4J auf das Durchlaufen der Klasseninvariante vor dem Prüfen der Pre-Conditions verzichtet.
Das Zeichen ∧ bedeutet die UND-Verknüpfung LSP-konformer Zusicherungen.

Wer ist hier anders als die anderen?

`int`

`boolean`

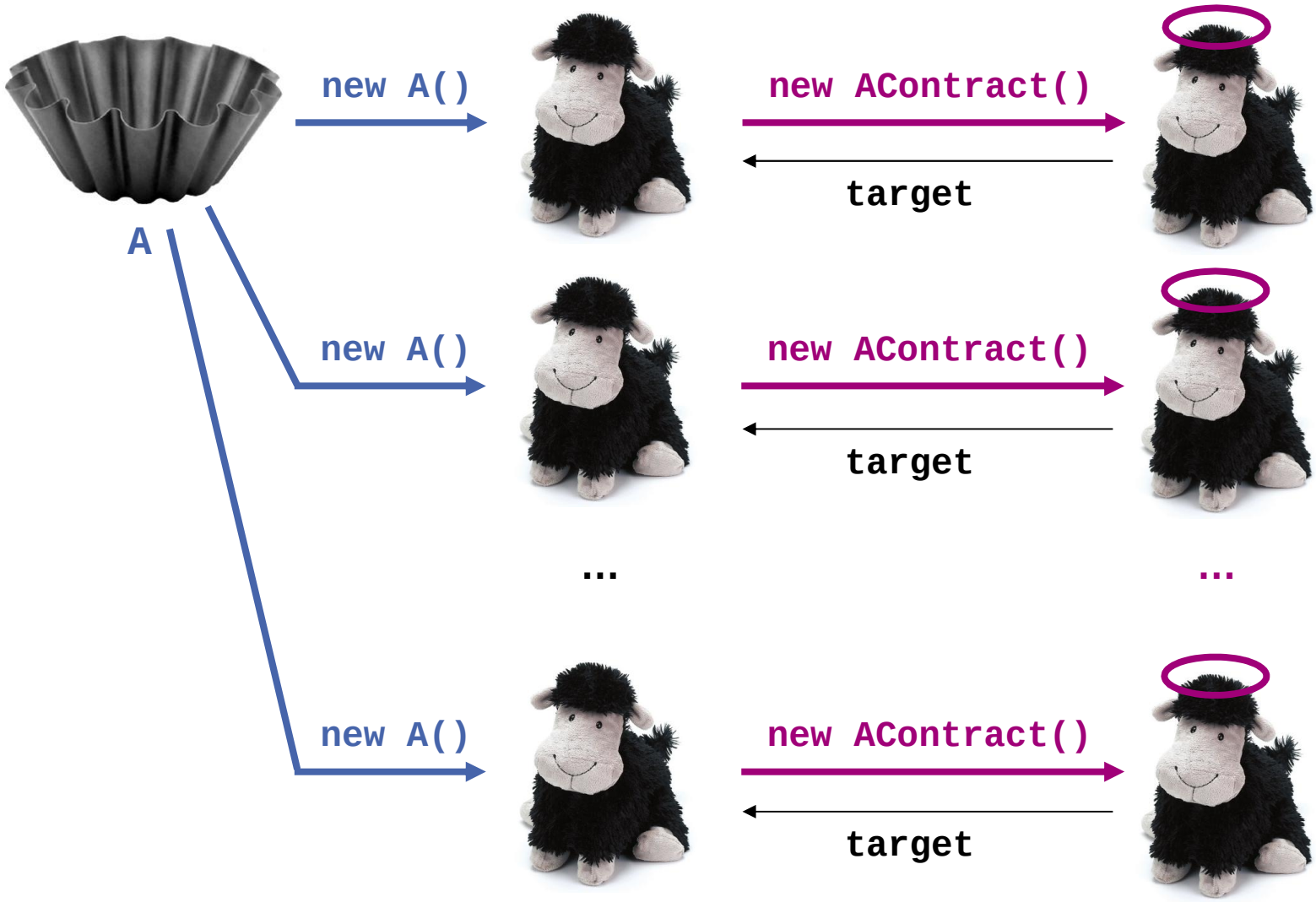
`String`

`char`

`double`

Gibt es Schutzengel in Java? Ja! C4J generiert für jedes Objekt sein persönliches Schutz-Objekt!

Für jede Klasse, zu der eine Vertragsklasse gehört, wird beim Instanziieren ein Objekt und ein Schutz-Objekt erzeugt!



Contracts for Java: C4J 4.0
Die Syntax

andrena OBJECTS
Experts in agile software engineering

C4J 4.0 Syntax	Erläuterung
`pre()`	Vorbedingung einer Methode (precondition).
`post()`	Nachbedingung einer Methode (postcondition).
`result()`	Rückgabewert der Methode (nur in postconditions verfügbar).
`old()`	Abfragen des Zustands eines Objekts vor der Methodendurchführung (nur in postconditions verfügbar).
`@Target`	Annotation zum Markieren der Referenz auf das zu schützende Objekt.
`@ClassInvariant`	Klasseninvariante (class invariant).
`@Pure`	Annotation zum Markieren einer seiteneffektfreie Methode (Abfrage), so dass die Klasseninvariante nicht überprüft werden muss.
`@ContractReference` oder `@Contract`	Annotation zum entkoppelten Verweis auf die schützende Vertragsklasse (oder zu schützende Klasse).

Contracts for Java: C4J 4.0
Die Syntax anhand von Beispielen

andrena OBJECTS
Experts in agile software engineering

C4J 4.0 Syntax	Beispiel

`pre()`
`post()`
	`@Override`
	`public void setX ( int x ) {`
	`  if (pre()) {`
	`  assert x >= 0 : "x >= 0";`
	`  }`
	`  if (post()) {`
	`  assert target.getX() == x : "x set";`
	`  }`
	`}`

`result()`	`public int getX () {`
	`  if (pre()) {`
	`  // no pre-condition identified yet`
	`  }`
	`  if (post()) {`
	`  int result = result();`
	`  assert result >= 0 : "result >= 0";`
	`  }`
	`}`

Contracts for Java: C4J 4.0
Die Syntax anhand von Beispielen

andrena OBJECTS
Experts in agile software engineering

C4J 4.0 Syntax	Beispiel
`@Target`	`@Target` `private PointSpec target;`

`@ClassInvariant`	`@ClassInvariant`
	`public void classInvariant() {`
	`  assert target.getX() >= 0 : "x >= 0";`
	`}`

`@Pure`	`@Pure` `public int getX();`
`@ContractReference`	`@ContractReference (PointSpecContract.class)` `public interface PointSpec`
`@Contract`	`@Contract` `public class PointContract`

Der Ursprung von C4J: C4J 2.7.5 – Contracts for Java

C4J

Design By Contract for Java

Navigation

Overview
Ease of use
Powerful
Class invariant example
Pre condition example
Post condition example
Interface example
Target members
Inheritance of contracts
Under the hood
Running
Todo
Downloads
About

News

2011-04-02 - C4J 2.7.5

Overview

Contracts for Java (C4J) is a *Design By Contract* (DBC, see Wikipedia DBC definition and Eiffel DBC docs) framework for Java 1.5 and later. The primary goal for C4J is ease of use. DBC is about design and quality, aspects of programming that a lot programmers don't spend enough time and energy on.
Therefore a DBC framework must be simple and painless to use. At the same time the framework must be powerful.
C4J is simple *and* powerful.

I am not going to try to convince any readers that DBC indeed is a very powerful design and quality assurance technique, so if you are not already convinced of that, please follow the links above and you may be convinced to try this tool out! These are my two favorites though:

- To be able to define meaningful contracts you are forced to split lengthy methods in to small, well defined, methods with a single responsibility. This single "side effect" of DBC has very positive effects on the code quality, maintainability, and readability.
- Your tests are verified against the *real* usage of your application, not against some test cases that may not even be real use cases.
- If you are dealing with legacy code that you are afraid of refactoring, contracts are perfect to add to existing code with no risk involved.

Source: Jonas Bergström: http://c4j.sourceforge.net/

C4J 2.7.5 ist sehr einfach anwendbar, da die Softwareverträge vollständig in Java formuliert werden.

Karlsruher Institut für Technologie

Ease of use

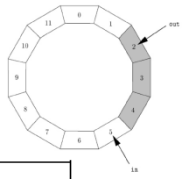
Adding class invariants, pre conditions, and post conditions to your regular Java objects is easy with C4J. Contracts for a class (the target) are implemented in a separate class (the contract). The contract is linked to the target at compile time using an annotation, and the actual contract verification code is tied to the target using instrumentation during class loading at run time.

There are several reasons for defining contracts in plain Java and in a separate class:

- You don't have to learn any new language constructs.
- You will get help from your IDE when implementing the contract (as opposed to implementing contracts in comment blocks).
- It is possible to put break points in your contract if you want to debug them.
- Separating the contract code from the application makes it possible to remove the risk of accidentally deploying the contracts together with your application.
- You get to keep your beautiful code nice and clean without all the ugly assertions.

Source: Jonas Bergström: http://c4j.sourceforge.net/

Keep your Code nice and clean!
Gegenbeispiel: ModernJass.



Karlsruher Institut für Technologie

```
@ModelDefinitions({
  @Model( name = "mTheBuffer", type = Object[].class),
  @Model( name = "mStoreIndex", type = Integer.class),
  @Model( name = "mExtractIndex", type = Integer.class)})
@InvariantDefinitions({
  @Invariant("size() >= 0"),
  @Invariant("(0 <= mStoreIndex - mExtractIndex) &&" +
      " (mStoreIndex - mExtractIndex <= mTheBuffer.length)") })
public interface BufferSpec<T> {

  @Pure
  @Post("@Result == mTheBuffer.length")
  public int size();

  @Pure
  @Post("@Result == (mStoreIndex == mExtractIndex)")
   public boolean empty();

  @Post("@Result == (mStoreIndex - mExtractIndex == mTheBuffer.length)")
  @Pure public boolean full();

  @Also({
    @SpecCase(
      pre = "o != null && !full()",
      post = "@Old(mStoreIndex) == mStoreIndex - 1"),
    @SpecCase(
      pre="o == null", signals = NullPointerException.class ) })
  public void add(@Name("o") T o);

  @Pre("!empty()")
  public T getNext();

  @SpecCase(
    pre="!empty()", post = "@Old(mExtractIndex) == mExtractIndex - 1")
  public T remove();

  @Pure
  @Post("@Result == @Exists(Object tmp : mTheBuffer;
    tmp != null && o.equals(tmp))")
  public boolean contains(@NonNull @Name("o") T o);

  @Post("@Result != null")
  public BufferSpec<T> copy();
}
```

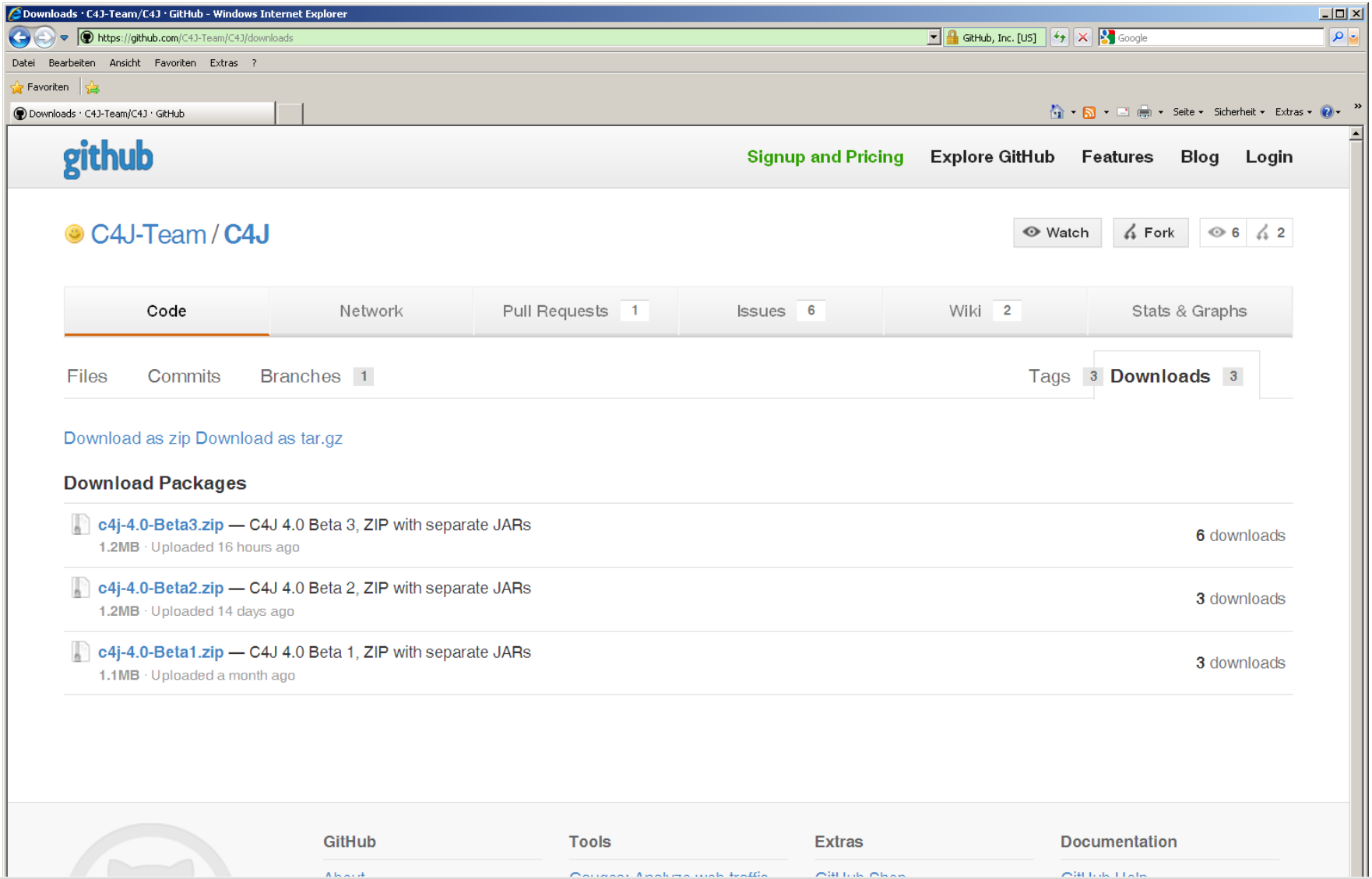
Listing 7.1: Declaration and specification of the ring buffer interface.

```
public interface BufferSpec<T> {

  public int size();

  public boolean empty();

  public boolean full();

  public void add(T o);

  public T getNext();

  public T remove();

  public boolean constains(T o);

  public BufferSpec<T> copy();
}
```


Source: Johannes Rieken, Modern Jass – Design by Contract for Java, Master Thesis 2007

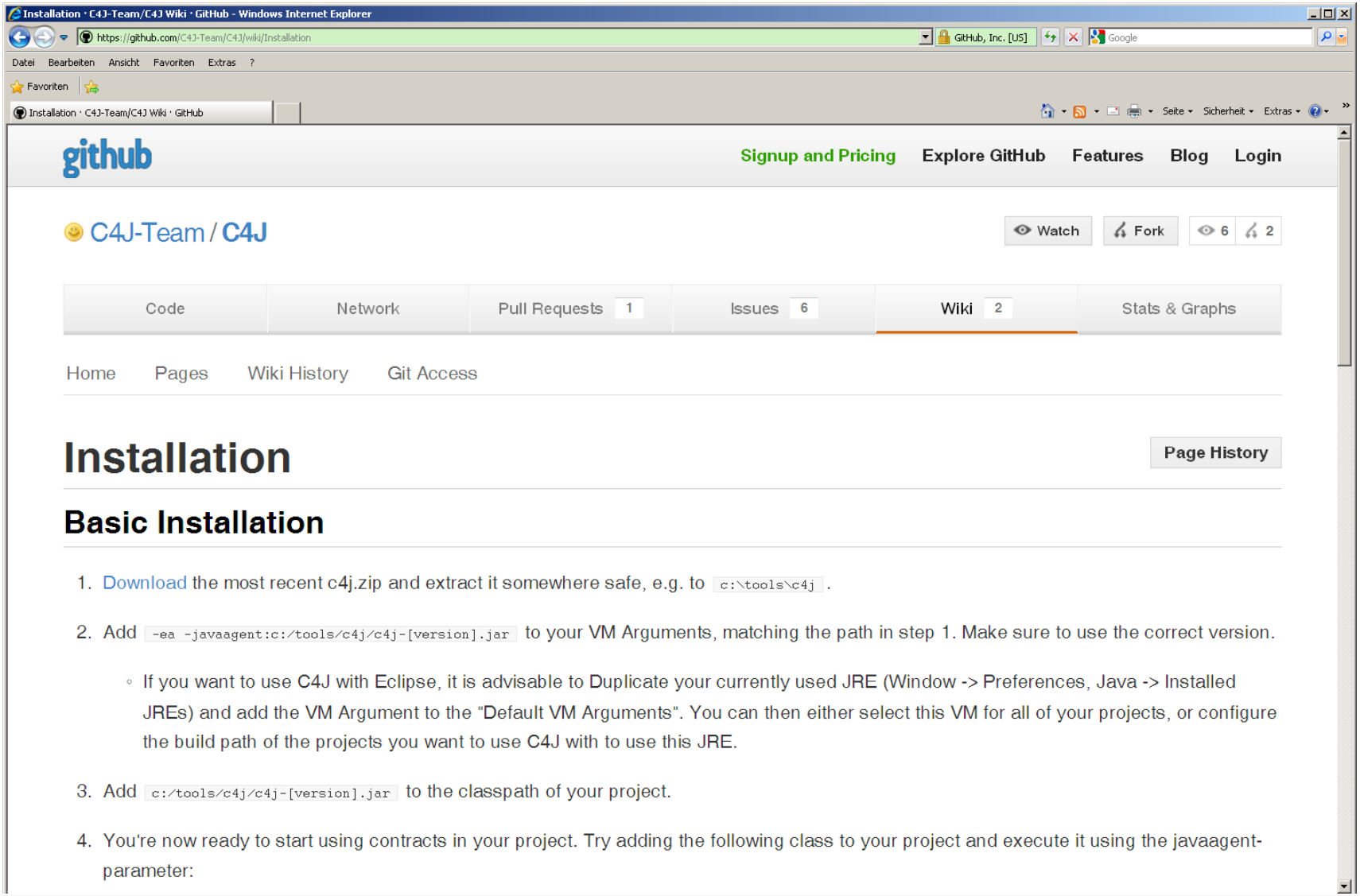
C4J 4.0 – das agile C4J

andrena OBJECTS
Experts in agile software engineering



C4J 4.0 – das agile C4J

andrena OBJECTS
Experts in agile software engineering



Verbesserungen von C4J 4.0 gegenüber C4J 2.7.5

andrena OBJECTS
Experts in agile software engineering

- C4J 2.7.5
  - Verträge werden in Java formuliert, d.h. es muss keine zusätzliche Sprache erlernt werden und die volle Sprachmächtigkeit von Java steht zur Formulierung der Verträge zur Verfügung.
  - Verträge stehen in eigenen Klassen, sind also konsequent von der Implementierung getrennt, so dass der Produktivcode fast vollständig frei von Zusatzkonstrukten gehalten werden kann.
  - Verträge sind normale Java-Klassen und können daher in der gewohnten IDE erstellt und nachvollzogen werden.
  - Verträge können auch für Interfaces definiert werden.
  - Vererbung wird gemäß LSP (Liskovsches Substitutionsprinzip) vollständig unterstützt.
- C4J 4.0 bringt darüber hinaus folgende Verbesserungen:
  - agil: Vertragsklassen sind vollständig refaktorierbar.
  - non-intrusive: Der Produktivcode kann - im non-intrusive Modus – vollständig frei von Zusatzkonstrukten gehalten werden (ideal für Legacy Code!)
  - old(), unchanged() und zur Laufzeit überwachte @Pure Eigenschaft von Methoden*
  - robust: Parametrisierbar, ob AssertionErrors zum Systemabbruch oder lediglich Logging der Vertragsverletzung führen.

* : in C4J 4.0 noch experimenteller Status

Contracts im Software Engineering
Agenda

andrena OBJECTS
Experts in agile software engineering

1. Was versteht man konkret unter Objektorientierter Programmierung?
2. Ist Java eine vollständig objektorientierte Programmiersprache?
3. Wie können in Java Abstrakte Datentypen modelliert werden?
4. Was sind Softwareverträge – und wie funktionieren sie in Java?
5. Was verändert sich, wenn wir Softwareverträge in Java nutzen?
6. Wie passen Softwareverträge und TDD zusammen?
7. Fragen & Antworten

Contracts for Java am KIT – ein Experiment über 3 Jahre hinweg mit 600 Studenten pro Jahr.

Evolution des Software-Engineerings am Beispiel der Klasse `Point`.

Karlsruher Institut für Technologie

```
// Point
int x;
int y;
```

```
class Point
{
  int x;
  int y;
}
```

Point
- x : int
- y : int

+ «constructor» Point (int, int)
+ getX ( ) : int
+ getY ( ) : int
+ setX (int) : void
+ setY (int) : void

<<interface>> PointSpec

+ getX ( ) : int
+ getY ( ) : int
+ setX (int) : void
+ setY (int) : void

<<interface>> ObjectSpec

+ equals (Object) : boolean

PointSpecContract

+ pre_setX (int) : void
+ post_setX (int) : void
+ pre_setY (int) : void
+ post_setY (int) : void

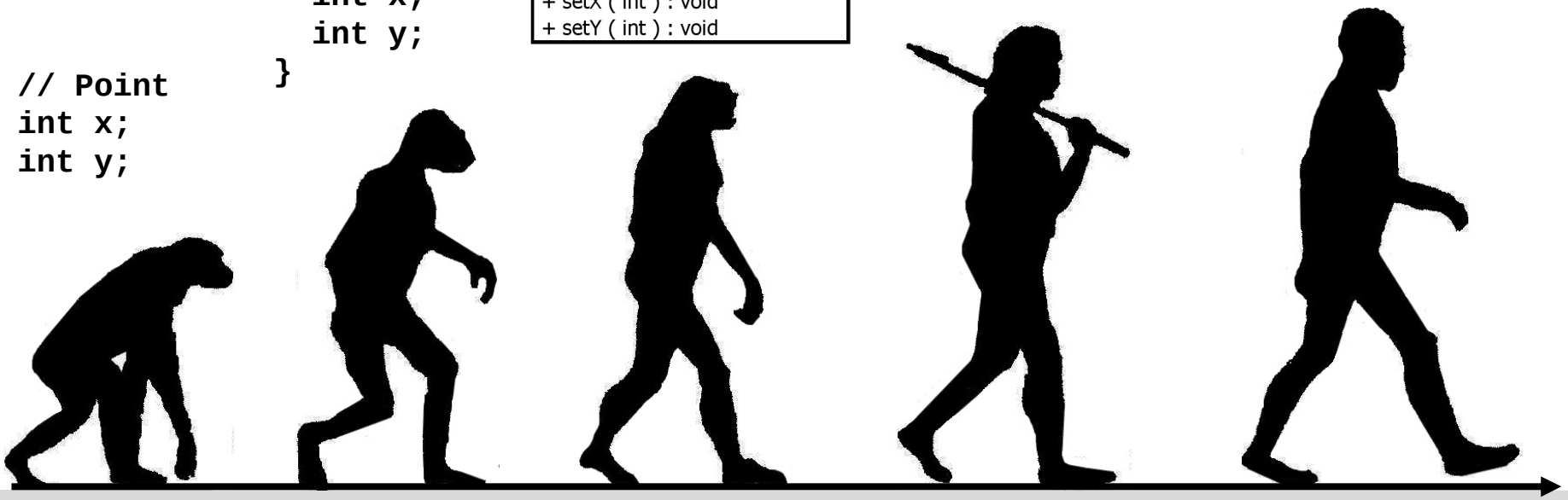
ObjectSpecContract

- z : Object

+ post_equals (Object) : void

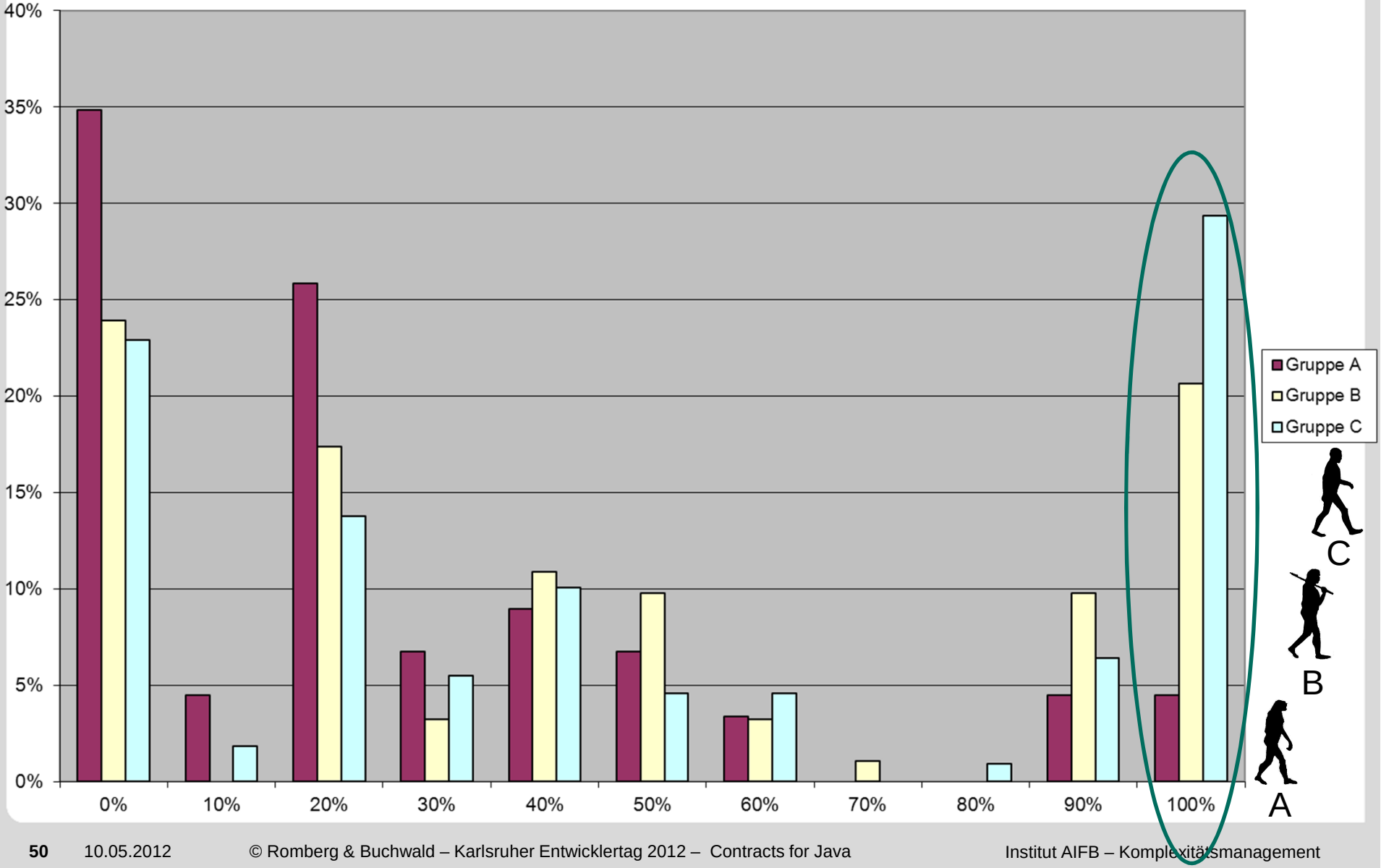
Point
- x : int
- y : int

+ «constructor» Point ( )
+ getX ( ) : int
+ getY ( ) : int
+ setX (int) : void
+ setY (int) : void
+ equals (Object) : boolean



homo structurlos | homo methodicus | homo modulus | homo abstractus | homo objectus

Die Vervollständigung von Java um Ebene 2 (ADTs) steigert die Produktivität der Softwareentwicklung signifikant!



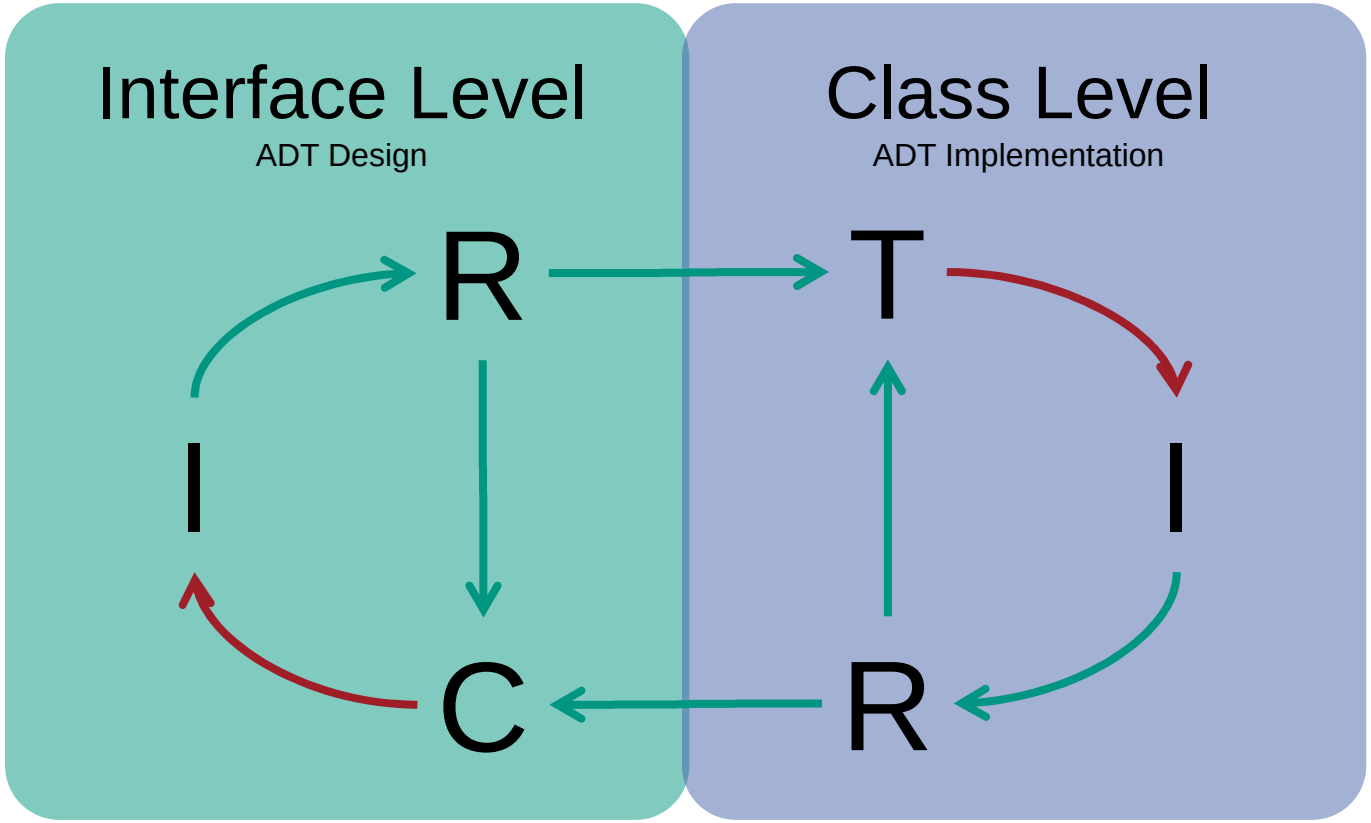
Contracts im Software Engineering
Agenda

andrena OBJECTS
Experts in agile software engineering

1. Was versteht man konkret unter Objektorientierter Programmierung?
2. Ist Java eine vollständig objektorientierte Programmiersprache?
3. Wie können in Java Abstrakte Datentypen modelliert werden?
4. Was sind Softwareverträge – und wie funktionieren sie in Java?
5. Was verändert sich, wenn wir Softwareverträge in Java nutzen?
6. Wie passen Softwareverträge und TDD zusammen?
7. Fragen & Antworten

OOP: Zusammenspiel von ADT Design auf der Interface-Ebene und ADT Implementierung auf der Klassen-Ebene

andrena OBJECTS
Experts in agile software engineering

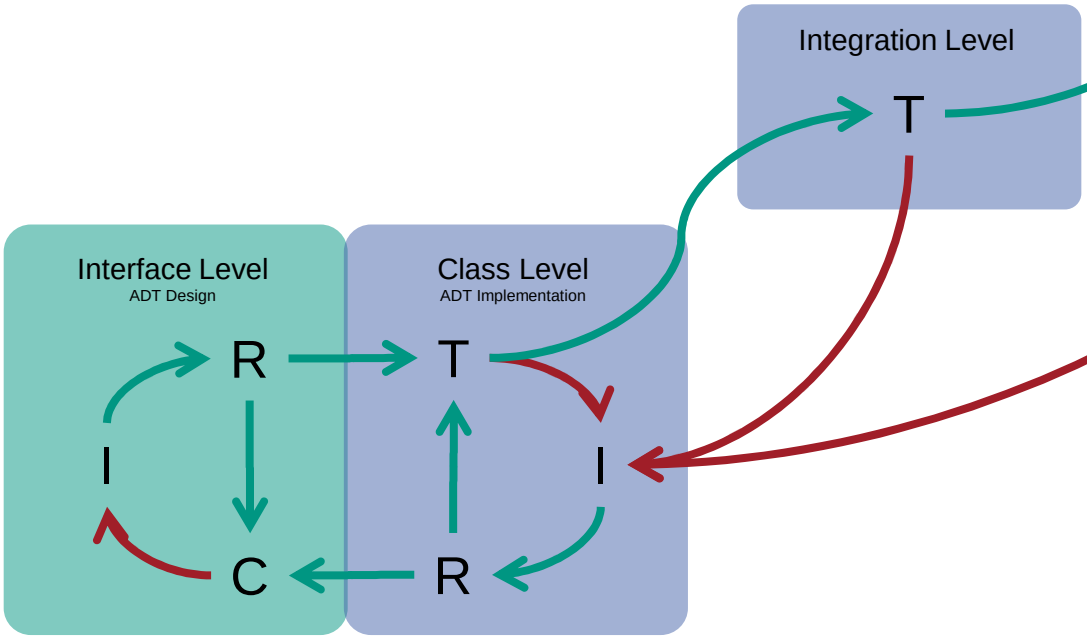


OOP ist die Erstellung von Softwaresystemen
als strukturierte Sammlung von Implementierungen
Abstrakter Datentypen (ADTs).

T : Test
I : Implement
R : Refactor
C : Contract

Zusammenspiel von Design by Contract, Test Driven Development und Test by Contract

andrena OBJECTS
Experts in agile software engineering

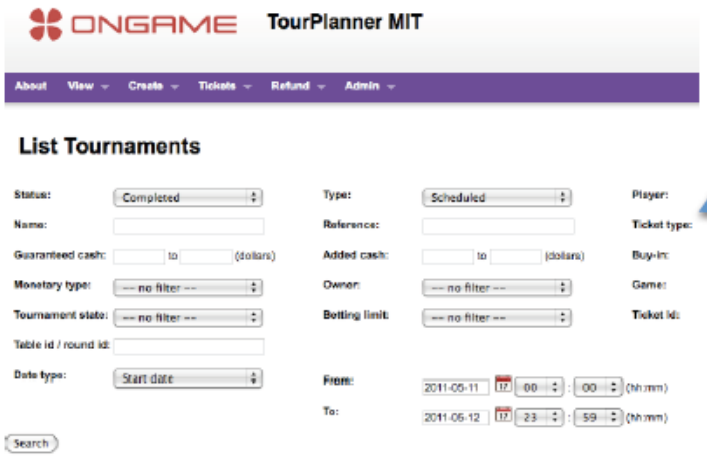


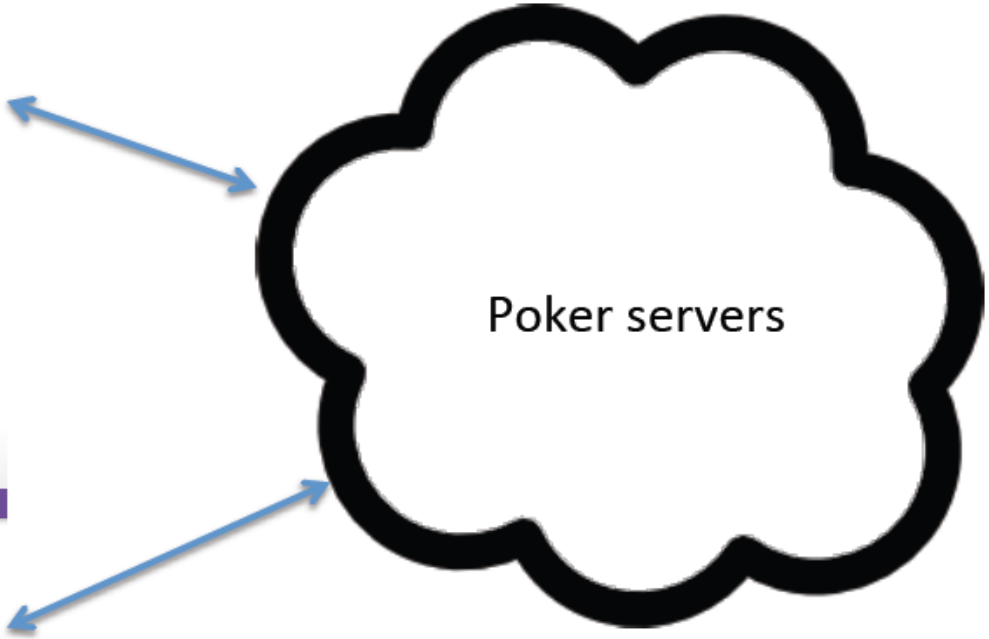
Was haben Pokerkarten und Dominosteine mit Softwareverträgen zu tun?

andrena OBJECTS
Experts in agile software engineering

Was hat Poker mit Softwareverträgen zu tun?

andrena OBJECTS
Experts in agile software engineering

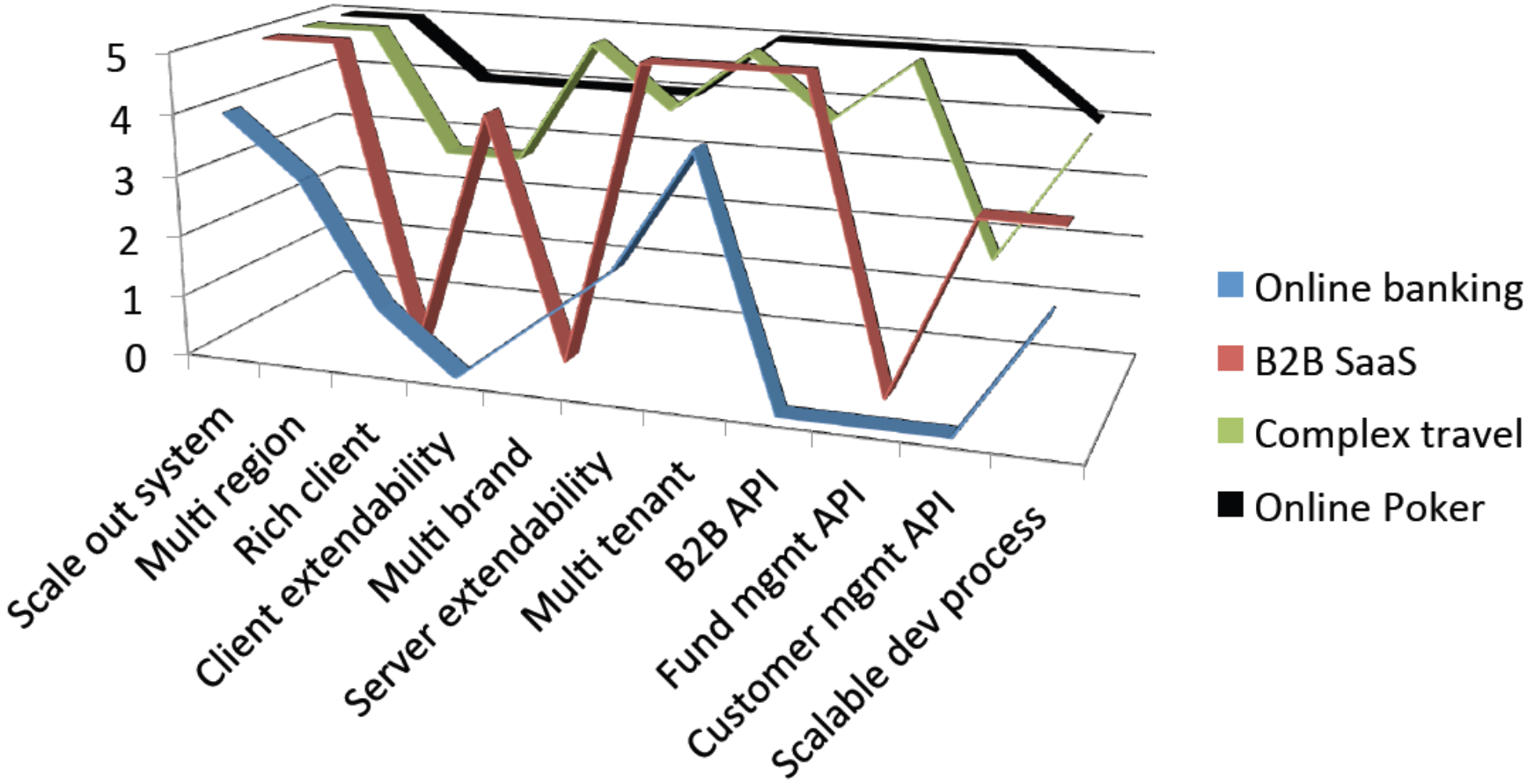




Source: Jonas Bergström: Vortrag auf dem Karlsruher Entwicklertag 2011

Die Anforderungen an ein white label online Poker-System sind sehr hoch!

andrena OBJECTS
Experts in agile software engineering



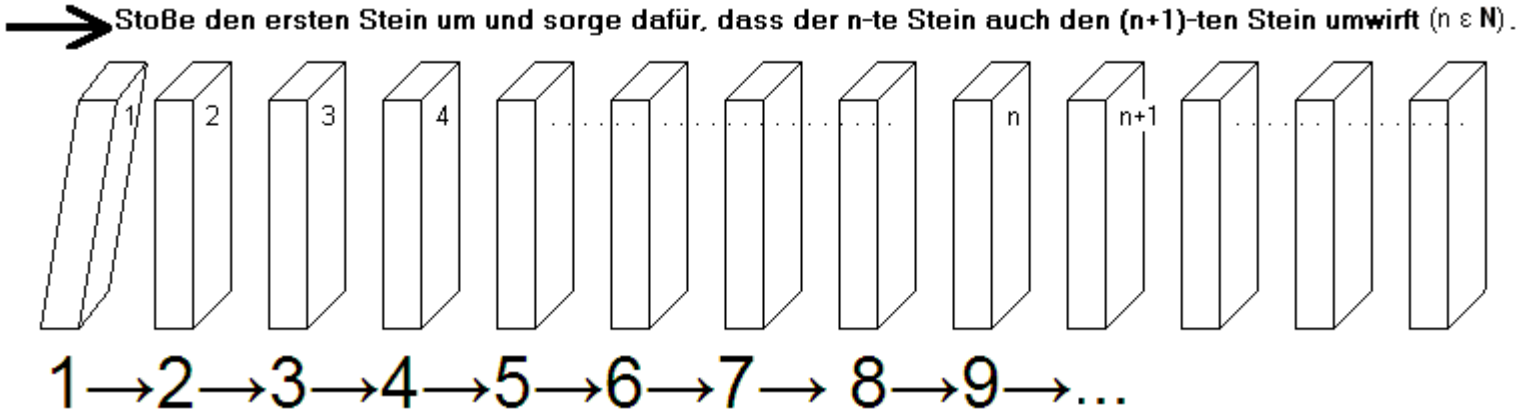
Source: Jonas Bergström: Vortrag auf dem Karlsruher Entwicklertag 2011

PokerBots simulieren Spieler und testen die Poker-Engine bei aktivierten Verträgen!

andrena OBJECTS
Experts in agile software engineering

Source: Jonas Bergström: Vortrag auf dem Karlsruher Entwicklertag 2011

Test by Contract simuliert das Prinzip der vollständigen Induktion für sehr große *n*, ohne jedoch vollständig zu sein.



Die vollständige Induktion erfasst durch den variablen Induktionsschritt beliebig viele Schritte, die man von 1 aus konkret durchführen kann. Das verdeutlicht die Grafik links.
Diese Methode ist mit dem Dominoeffekt vergleichbar: Wenn der erste Dominostein fällt und durch jeden fallenden Dominostein der nächste umgestoßen wird, so wird schließlich jeder Dominostein irgendwann umfallen.
Im Unterschied zum Domino, bei dem zwar beliebig viele, aber immer endlich viele Steine vorliegen, gibt es aber unendlich viele natürliche Zahlen, so dass keine beliebig lange konkrete Induktion alle Zahlen erreicht.
Nur über den variablen Induktionsschritt wird die Induktion *vollständig* und erreicht tatsächlich alle Zahlen.

Quelle: Joachim Mohr, Vollständige Induktion, Wikipedia

Wirkung des Einsatzes von Test by Contract mit C4J und PokerBots auf die Qualität der bwin PokerEngine.

andrena OBJECTS
Experts in agile software engineering

bwin PokerEngine
ohne C4J

Fehlermeldungen
pro Betriebsmonat

bwin PokerEngine
mit C4J

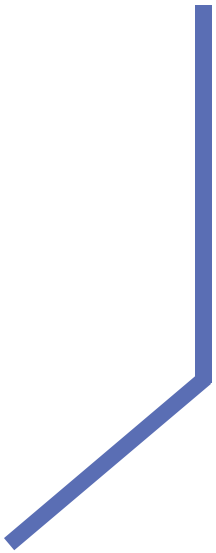
Fehlermeldungen nach
30 Betriebsmonaten

Warum sind diese Symbole so wichtig für die Software-Qualität?

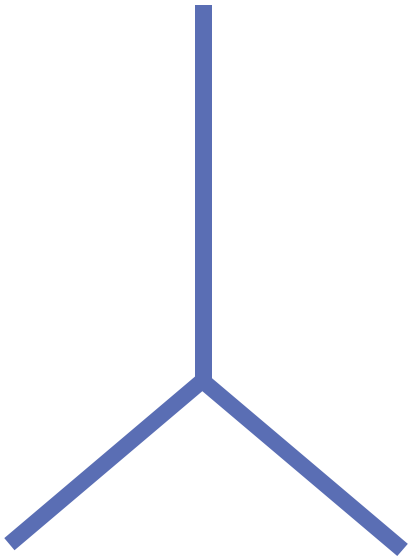
andrena OBJECTS
Experts in agile software engineering

Boundary Condition

Edge Condition



Corner Condition



Gibt es Schutzengel in Java? Ja! C4J generiert für jedes Objekt sein persönliches Schutz-Objekt!

Contracts im Software Engineering
Agenda

andrena OBJECTS
Experts in agile software engineering

1. Was versteht man konkret unter Objektorientierter Programmierung?
2. Ist Java eine vollständig objektorientierte Programmiersprache?
3. Wie können in Java Abstrakte Datentypen modelliert werden?
4. Was sind Softwareverträge – und wie funktionieren sie in Java?
5. Was verändert sich, wenn wir Softwareverträge in Java nutzen?
6. Wie passen Softwareverträge und TDD zusammen?
7. Fragen & Antworten

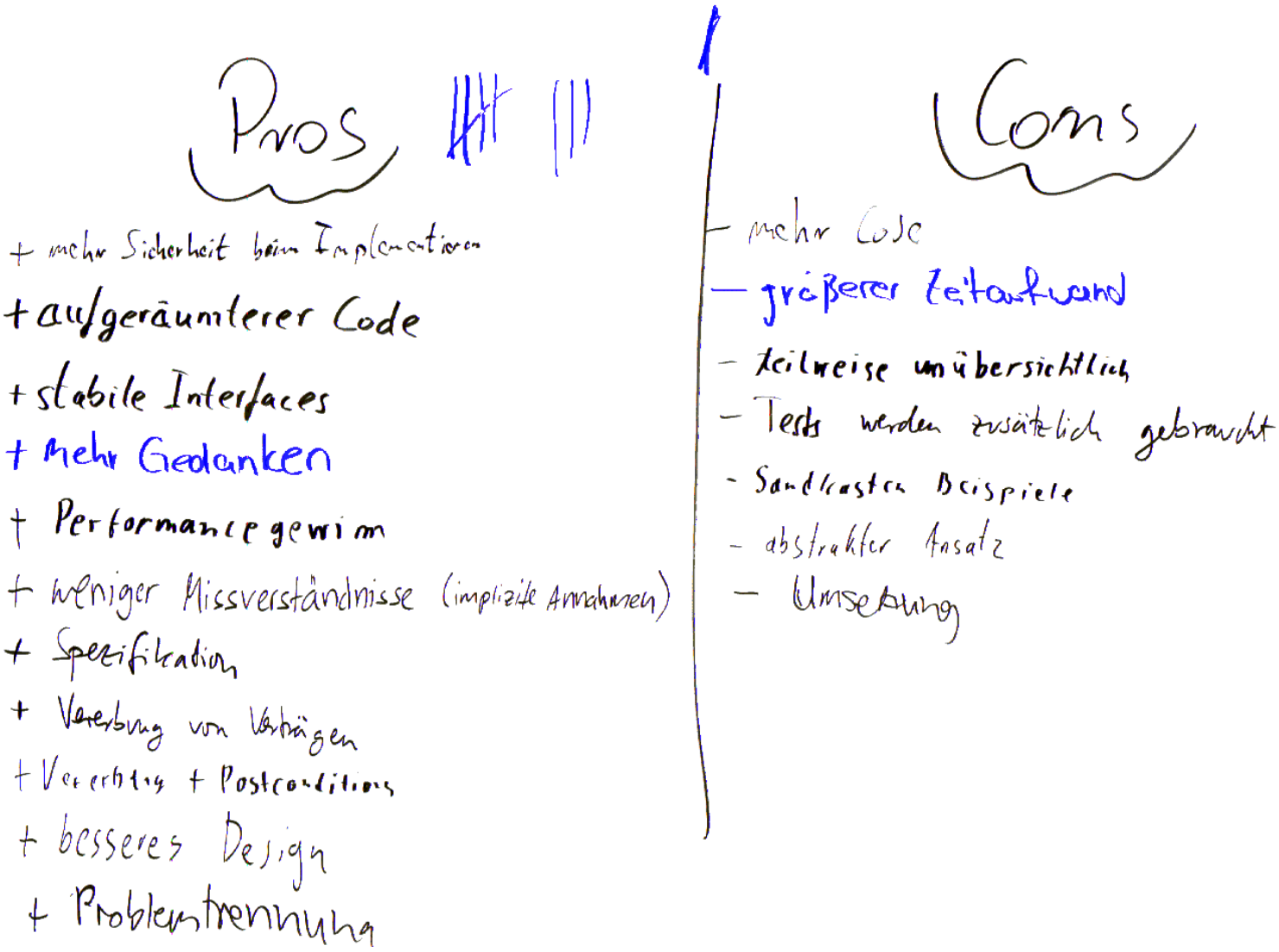
Fragen an das Publikum des C4J-Vortrags auf dem Karlsruher Entwicklertag 2012

Karlsruher Institut für Technologie

1. Zementiert Design by Contract (DbC) nicht den Produktivcode, so dass ggf. Refactorings unterbleiben? (Ja | Nein)
2. Unterstützen Entwicklungsumgebungen wenigstens ein halb-automatisches Refactoring von Vertragsklassen? (Ja | Nein)
3. Welche Werkzeugunterstützung ist generell nötig, um effizient Design by Contract anzuwenden? (Ja | Nein)
4. Kann man – als überzeugter TDD-Verfechter – Tests für TDD guten Gewissens aus Vertragsklassen ableiten? (Ja | Nein)
5. Ist bei nicht-trivialen Beispielen der Vertrag wirklich einfacher als die Implementierung selbst? (Ja | Nein)
6. Verletzen Vertragsklassen nicht das DRY-Prinzip der Objektorientierung: Don‘t Repeat Yourself? (Ja | Nein)
7. Können Vertragsklassen auch iterativ und inkrementell entstehen, gemäß des agilen Prinzips? (Ja | Nein)

Feedback der Teilnehmer des andrena-internen Contracts for Java Workshops vom 19.04.2012

andrena OBJECTS
Experts in agile software engineering



andrena OBJECTS
Experts in agile software engineering

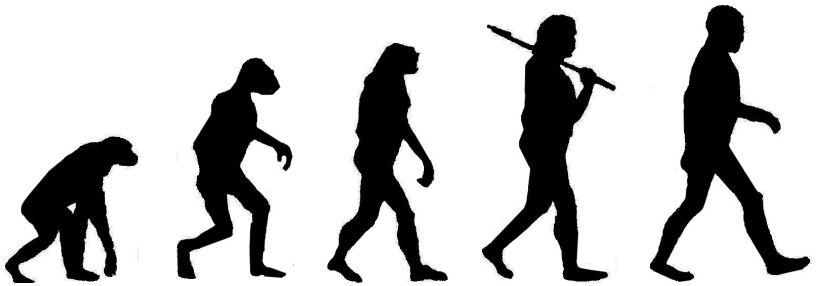
Programmausschnitt: Tutorial Day (11. Mai 2012)

Uhrzeit	Raum Berlin	Raum Paris	Albert-Nestler-Straße 9, 1. OG
09:00 - 09:15	**Wolfpack Programming** Helge Nowak (Cincom Systems)	**Design by Contract by Example Workshop** Ben Romberg (andrena objects ag) Yana Stoeva (andrena objects ag) Hagen Buchwald (andrena objects ag)	
09:30 - 10:15			
10:15 - 10:45			
10:45 - 11:30			
11:45 - 12:30			
12:30 - 13:30			
13:30 - 14:15			**Advanced ASE: TDD in der Realität** Marc Philipp (andrena objects ag) Andi Scharfstein (andrena objects ag) Leif Frenzel (andrena objects ag) David Burkhart (andrena objects ag)
14:30 - 15:15			
15:15 - 15:45			
15:45 - 16:30			
16:45 - 17:30			

Quelle: http://www.entwicklertag.de/tutorial-day

Contracts im Software Engineering

Karlsruher Entwicklertag 2012
Karlsruhe, 10. Mai 2012
Ben Romberg | Hagen Buchwald



Institut AIFB – KOMPLEXITÄTSMANAGEMENT