

# AOP Day 07

eine Veranstaltung der  
**SIG AspectJ**

organisiert vom

- AOP-Komitee
  - Oliver Böhm
  - Darko Palic
  - Ludger Solbach
- und der guten Fee
  - Bori Gerhardt

*powered by*

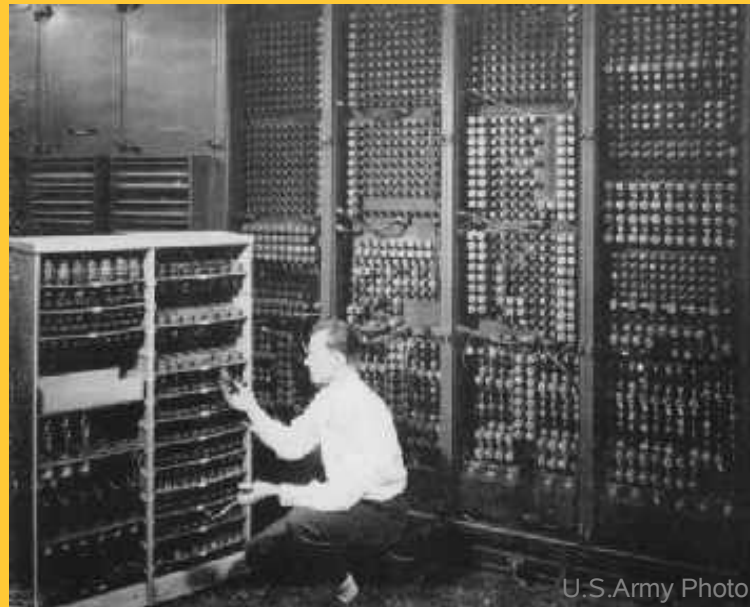


- Dez. 2005: AspectJ Winter Camp
- 2006:
  - 1 Veranstaltung der SIG AspectJ mit Arno Schmidmeier
  - Nachfragen nach weiterem Event
- Sept. 2006: Anfrage von Jonas Bonier (Terrakotta)
  - Idee eines AOP-Days / Vorschlag
  - Jonas Bonier war begeistert, Zusage

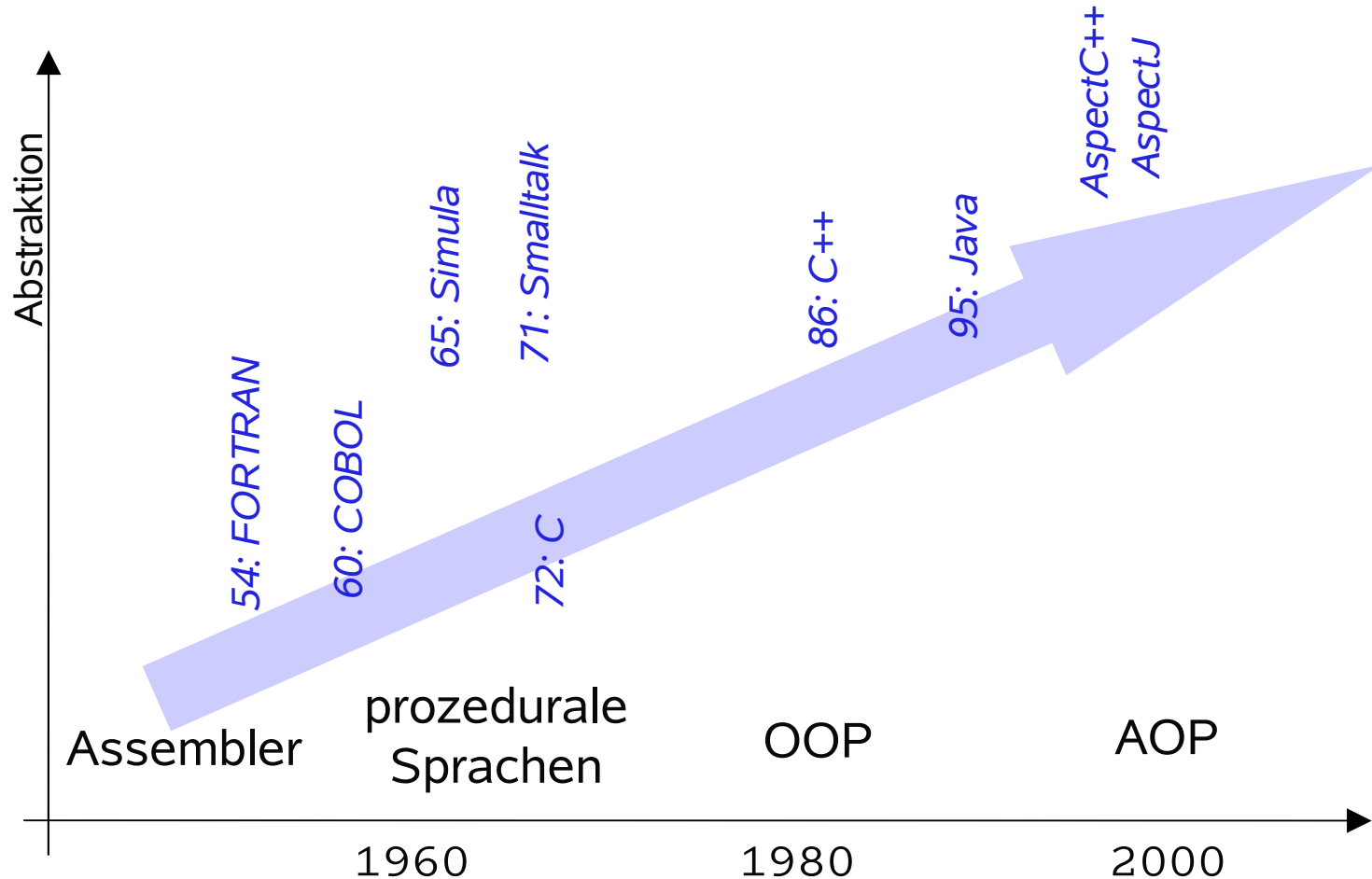
- Dez. 2006: Aufruf zum CfP
  - 1 Zusage (Jonas Bonier)
  - erste Anmeldungen
- 16. März: Abgabetermin für CfP
  - 3 Wochen vorher: Absage Jonas Bonier
  - 2 Wochen vorher: 1 CfP von Karsten Becker
  - 1 Woche vorher: Ausarbeitung Plan B
  - letzte Woche:
    - 6 Einreichungen
    - letzte Einreichung: 19 Sekunden vor Abgabeschluß (Jürgen Höller)
- 18. März: Programm online
- 22. März: Absage Alte Scheuer

09:00 - 09:30	Beginn, Begrüßung, Überblick
09:45 - 10:30	Enterprise AOP: Enterprise AOP: AspectJ Support in Spring 2.0/2.1 Jürgen Höller, Interface 21
10:45 - 11:30	Vergleich von Weaving Technologien in Aspekt Orientierten Programmiersprachen Karsten Becker, HAW Hamburg
11:45 - 13:00	Workshop: Hello AspectJ Oliver Böhm, agentes AG
13:00 - 13:45	Mittagspause
13:45 - 16:45	Workshop: Writing Reusable Aspect Libraries Arno Schmidmeier, AspectSoft
17:00 - 17:45	Aspekte im Projekt-Alltag Ludger Solbach, Deutscher Sparkassenverlag
18:00 - 18:45	Risiken und Nebenwirkung der AOP Markus Knauß, Universität Stuttgart
19:00 - 19:30	Ausblick und Verlosung
ab 20:00	Social Event: @After-Day-Party

## von Assembler bis AspectJ



# Die Geschichte der Programmiersprachen



SIE	H, T
AUS	W, I, E
ASS	EM
BLE	R
IS	ES
ABE	R
NET	

- **viele Dialekte**
  - Prozessor-abhängig
  - HW-abhängig
- **schnell**
- **nur von Spezialisten zu verstehen**

**Was man nicht in  
Assembler machen kann,  
muss man löten!**

# prozedurale Sprachen

## Zwetschgenkuchen

### ▪ Bsp: Backanleitung

Mürbeteig: 250gr. Mehl, 1/2 Backpulver  
80gr. Fett, 80gr. Zucker  
1 Vanillezucker, 1 Prise Salz, 2 Eier  
Belag: 1kg Zwetschgen, 2 Eßl. Hagelzucker

Aus den Zutaten einen Mürbeteig herstellen und  
1 Stunde kaltstellen.

Zwetschgen entsteinern und einschneiden.

Rundes Backblech beschmieren und mit Teig  
auslegen. Den Teigboden mehrmals mit einer  
Gabel einstechen und mit Semmelbrösel bestreuen.

Zwetschgen rosolenförmig legen.

Im Backofen 20-30 Minuten backen bei 180-200°C.

Nach dem Backen mit Hagelzucker bestreuen.

PASCAL  
C COBOL  
FORTRAN

**kuchen = backen(butter,  
mehl, zucker, eier, ...)**

# OO-Sprachen

Ruby

Smalltalk

SIMULA



C++

Python

Java

Eiffel



```
kuchen = new Kuchen(butter, mehl, eier);  
kuchen.backe();
```

Kuchen
Zutaten Größe
add backe

# Aspektorientierte Sprachen



- = OOPS, um Aspekte angereichert, z.B.
- Logging-Aspekt
  - Security-Aspekt
  - Transaktion-Aspekt
  - ...

AspectS

AspectJ

AspectC++

# Ein kleiner Ausflug mit AOP



# Beispiel: Bankkonto

## fachliche Concerns

- **Einzahlung**
- **Auszahlung**
- **Überweisung**
- **Bonitätsprüfung**
- ...

## nichtfachliche Concerns

- **Logging**
- **Performance**
- **Authorisierung**
- **Sicherheit**
- ...



# Ein Fallbeispiel...



```
public class Konto {  
    private double kontostand = 0.0;  
  
    public double abfragen() {  
        return kontostand;  
    }  
  
    public void einzahlen(double betrag) {  
        kontostand = kontostand + betrag;  
    }  
  
    public void abheben(double betrag) {  
        kontostand = kontostand - betrag;  
    }  
  
    public void ueberweisen(double betrag, Konto anderesKonto) {  
        this.abheben(betrag);  
        anderesKonto.einzahlen(betrag);  
    }  
}
```

Konto
Kontostand
abfragen einzahlen abheben ueberweisen

## ...mit Logging

```
public class Konto {  
  
    private static Logger log = Logger.getLogger(Konto.class)  
    private double kontostand = 0.0;  
  
    ...  
  
    public void einzahlen(double betrag) {  
        kontostand = kontostand + betrag;  
        log.info("neuer Kontostand: " + kontostand);  
    }  
  
    public void abheben(double betrag) {  
        kontostand = kontostand - betrag;  
        log.info("neuer Kontostand: " + kontostand);  
    }  
  
    ...  
  
}
```

**Achtung!**  
**Codeverschmutzung!**

neue  
Anforderung:

alle Konto-  
Bewegungen  
müssen  
protokolliert  
werden!

# ...mit AspectJ

```
public aspect LogAspect {  
  
    private static Logger log = Logger.getLogger(LogAspect.class);  
  
    after(double neu) : set(double Konto.kontostand) && args(neu) {  
        log.info("neuer Kontostand: " + neu);  
    }  
  
}
```

Konto
Kontostand
abfragen einzahlen abheben ueberweisen

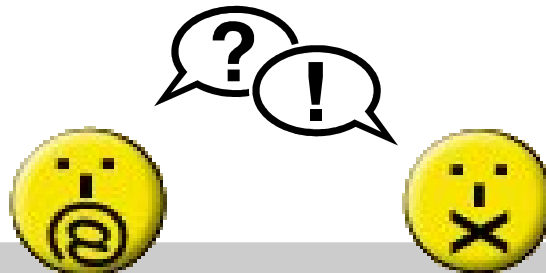


*Separation of Concerns*

LogAspect

# Do you speak AspectJ?

- **Joinpoint**
  - Eingriffspunkte im Programm, an denen Code erweitert oder modifiziert werden soll
- **Pointcut**
  - eine Auswahl von Joinpoints
- **Advice**
  - der Code für den Pointcut
- **Introduction**
  - Erweiterung anderer Klassen, Interfaces, Aspekte um zusätzliche Funktionalität
- **Aspect**
  - Konstrukt, in dem das obige abgelegt wird

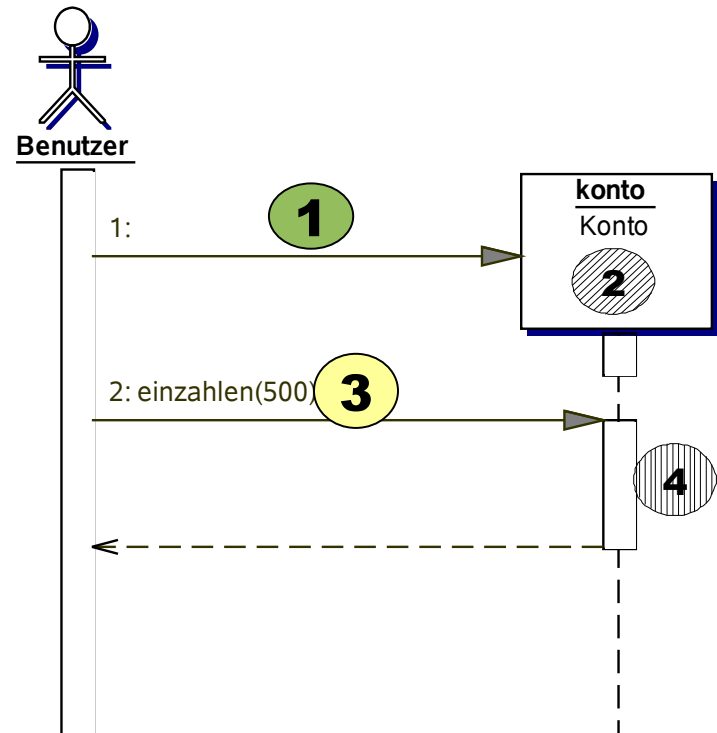


# Joinpoints können sein

- Aufrufen einer Methode
- Ausführen einer Methode
- Zugriff auf eine Variable
- Behandeln einer Exception
- Initialisierung einer Klasse
- Initialisierung eines Objekts

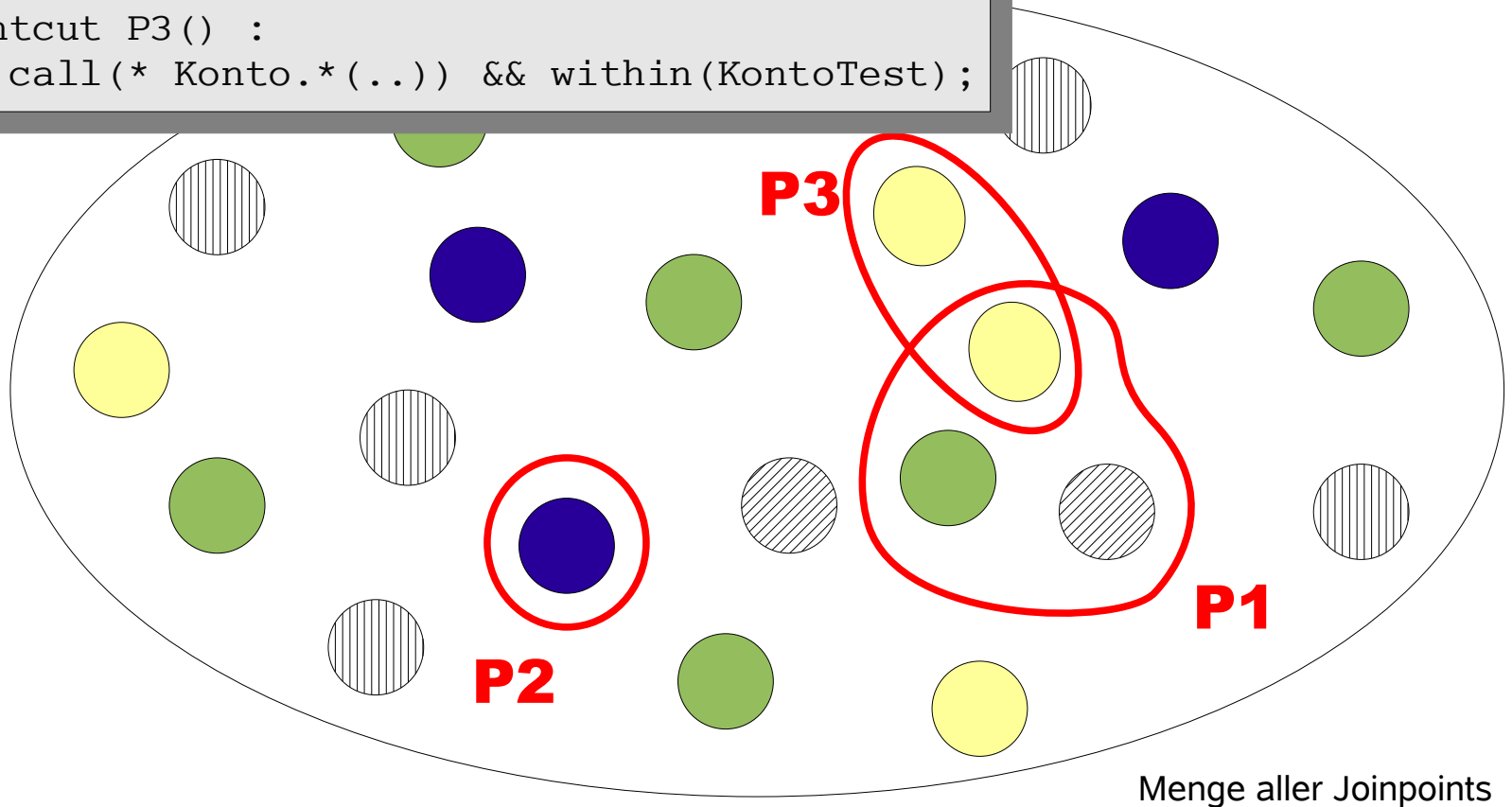
```
Konto konto = new Konto();  
konto.einzahlen(500.0);
```

- (1) Konstruktor aufrufen
- (2) Objekt initialisieren
- (3) Methode aufrufen
- (4) Methode ausführen



# Pointcuts

```
pointcut P2() : set (double Konto.kontostand)
pointcut P3() :
  call(* Konto.*(..)) && within(KontoTest);
```



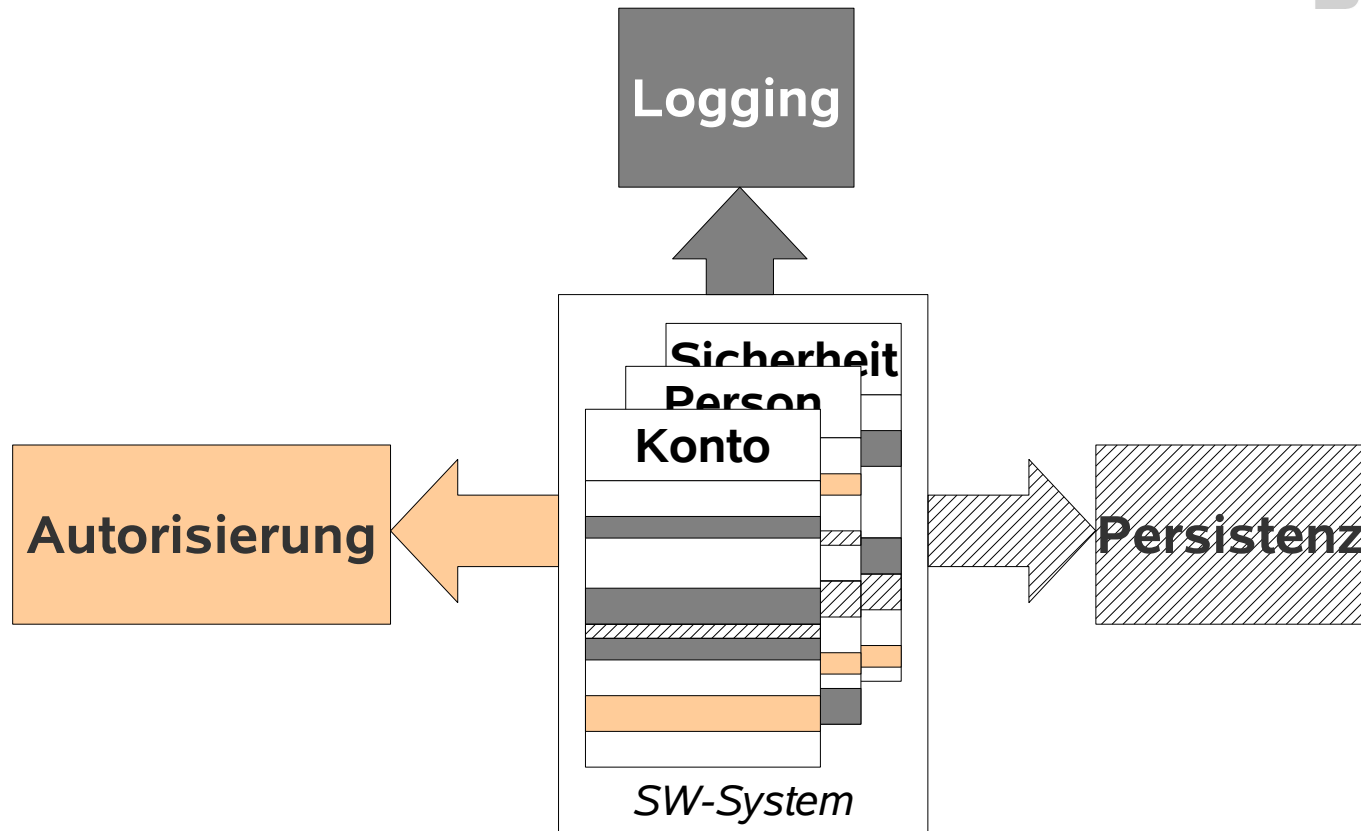
# Advice

- **Code, der eingewebt wird**
  - before()
  - after()
  - around()
- **Ähnlichkeit mit Methoden**

```
after() : P3() {  
    log.info("TEST: " + thisJoinPoint);  
}
```

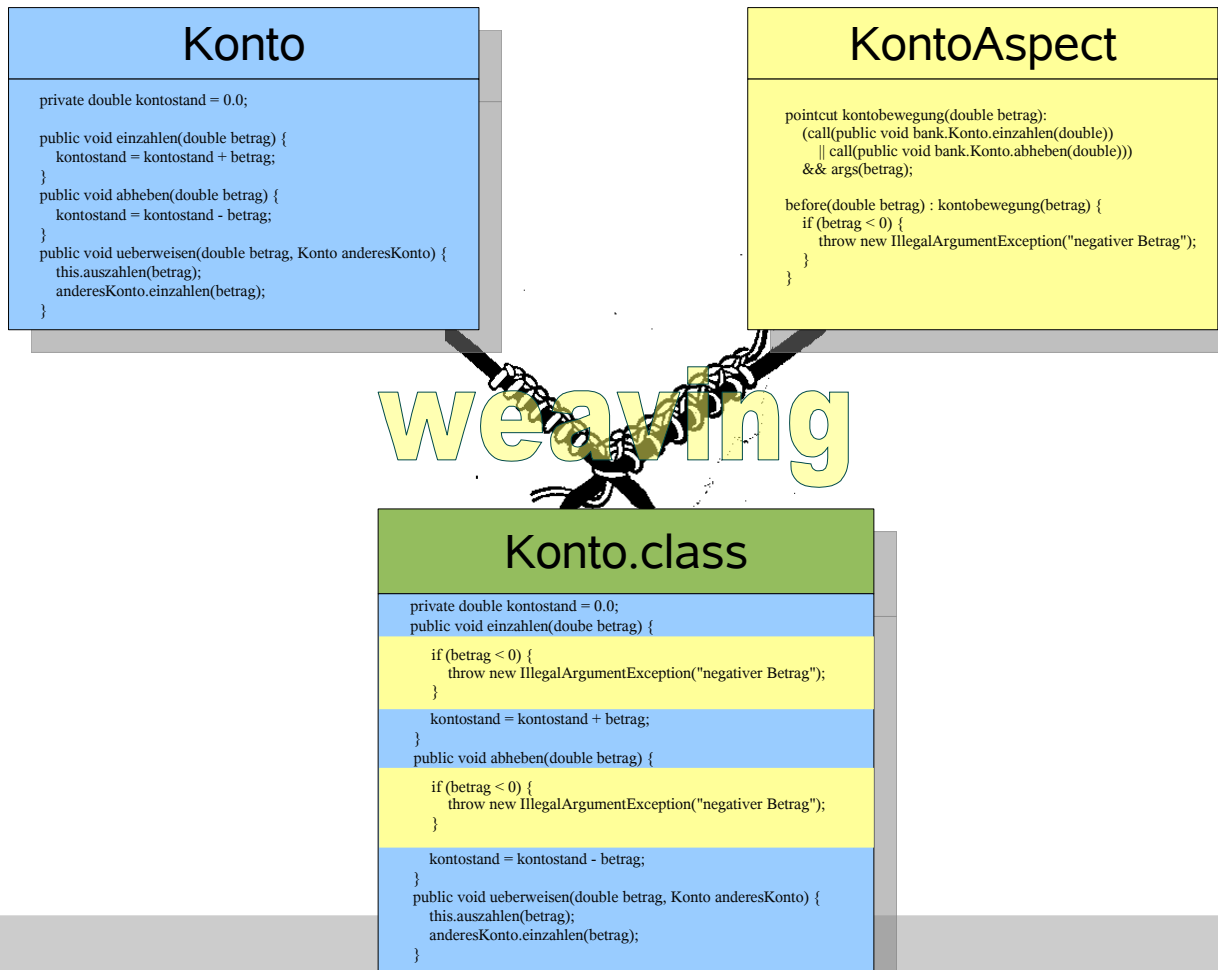
```
TEST: call(void bank.Konto.einzahlen(double))  
TEST: call(double bank.Konto.abfragen())  
...
```

# Do you think in AOP?



Das System als Menge von "Concerns"

# Der Webe-Vorgang (Weaving)



Viel Spaß auf dem AOP-Day '07  
wünscht  
das AOP-Komitee

