Rheinische Friedrich-Wilhelms-Universität Bonn, Institut für Informatik III

Vorlesung Aspektorientierte Softwareentwicklung (AOSD)
– Sommersemester 2008 –

Dr. Günter Kniesel, Daniel Speicher

Übungsblatt 2
Abgabe: 20.05.2008 23:59:59 per SVN
(Tutorien Freitag 23.05. + Montag 26.05.)

Hinweis: Der Code auf den sich die einzelnen Aufgaben beziehen ist im SVN zu finden unter https://svn.iai.uni-bonn.de/repos/IAI_Software/se/aosd2008ss/aufgaben/trunk/B02A...

Aufgabe 1 *(Semantik von Pointcuts und Patterns) (4 Punkte)*

Das Ziel dieser Aufgabe ist die Verwendung der primitiven Pointcuts

- `staticinitialization`
- `preinitialization`,
- `initialization`

und der Pointcuts

- `call(*..Dad+.new(..))`
- `execution(*..Dad+.new(..))`

in dem zur Verfügung gestellten Code (…/B02A1_InitializationExploration) zu erläutern. Geben Sie an, in welcher Reihenfolge die Codezeilen ausgeführt werden und wo sich die Static Join Point Shadows der durch die Pointcuts erfassten Join Points finden.

Aufgabe 2 *(Pointcuts und Vererbung)* (4 Punkte)

Der Begleitcode zu dieser Aufgabe (…/B02A2_PointcutExploration) überprüft, wann ein call() oder ein execution() Pointcut greift. Welche Kombinationen wurden erzeugt? Fassen Sie ihre Erkenntnisse systematisch zusammen. Wie lässt sich möglichst kurz und präzise beschreiben, wann die Pointcuts greifen?

Aufgabe 3 (6 Punkte)

Im SVN steht unter …/B02A03_BankCreditcard eine neue Variante des Programms *BankAccount* bereit. Die Konten sollen um die Festlegung des Überziehungsrahmens in Abhängigkeit von der verwendeten Kreditkarte erweitert werden. Die folgenden beiden Aufgaben sind durch Aspekte zu lösen:

a) Den Konten ein *Feld* hinzufügen in dem steht welche Kreditkarte der Kunde besitzt (*DinersClub* oder *Visa* in *Standard* oder *Gold*). Die Kreditkarte legt fest um wie viel € das Konto überzogen werden darf.

b) Die debit-Methode soll nun bei ihrem Aufruf überprüfen welche Kreditkarte das Konto hat und dann entscheiden um wie viel € das Konto überzogen werden darf.

Erweitern Sie die Klasse *Test* so, dass dort entsprechend alle Fälle vorkommen. ***Bitte beachten Sie, dass in den Java-Klassen (außer Test) nichts direkt verändert werden darf.***

LogicAJ

Für die nächsten Aufgaben benötigen Sie LogicAJ. Eine Anleitung zur Installation von LogicAJ finden Sie unter http://sewiki.iai.uni-bonn.de/research/logicaj/installation.

Bitte beachten Sie, dass man für LogicAJ und AspectJ zwei getrennte Eclipse-Instanzen braucht, da man sie NICHT nebeneinander betreiben kann!!! (AspectJ ersetzt den Java-Compiler von Eclipse durch seinen eigenen, der leider nicht die gleichen Schnittstellen bietet, so dass das auf den Eclipse-Compiler aufbauende LogicAJ nicht funktionieren kann.)

Aufgabe 4 *(Contracts) (4 Punkte)*

Schreiben Sie ein „declare warning“ Konstrukt, das folgende Contracts überprüft:

a) Alle Methoden und Felder beginnen mit einem Kleinbuchstaben

b) Alle Klassen beginnen mit einem Großbuchstaben

Hinweis: Verwenden Sie die Predicate Pointcuts von LogicAJ zur Manipulation von String Literalen und Listen.

Aufgabe 5 *(LogicAJ vs. AspectJ) (7 Punkte)*

Die Vorlesung beschreibt auf Folie 3-31den Eclipse Internal Package Access Contract, stellt aber keine Lösung vor.

a) Schreiben Sie einen „declare warning“ Aspekt in ApectJ der diesen Contract überprüft.

b) Schreiben Sie einen „declare warning“ Aspekt in LogicAJ der ebenfalls diesen Contract überprüft.

c) Tun die beiden Aspekte *genau* das Gleiche? Warum?

Überpüfen Sie mit ihren Aspekten das im SVN (…/B02A05_InternalPackageContract) zur Verfügung gestellte Projekt.