Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen

Fortgeschr. Objektorientierung SS 2008		http://pp.info.uni-karlsruhe.de/
Dozent:	Prof. Dr.-Ing. G. Snelting	snelting@ipd.info.uni-karlsruhe.de
Übungsleiter:	Daniel Wasserrab	wasserra@ipd.info.uni-karlsruhe.de
	Andreas Lochbihler	lochbihl@ipd.info.uni-karlsruhe.de

Übungsblatt 7 Ausgabe: 16.6.2008 Besprechung: 18.6.2008

1. **Multimethoden**

   Gegeben sei das folgende Programmfragment

```
1  class A { }
2  class B extends A {
3     void f(A x) { }
4     void f(B x) { }
5  }
6  class C extends B {
7     void f(A x) { }
8  }
9  class D extends B {
10    void f(B x) { }
11 }
12 ...
13    A a=new A();
14    A a_b=new B();
15    B b=new B();
16    B b_c=new C();
17    B b_d=new D();
18    C c=new C();
19    D d=new D();
20
21    b_c.f(a);
22    b_c.f(a_b);
23    b_c.f(b);
24    b_d.f(a);
25    b_d.f(a_b);
26    b_d.f(b);
27    c.f(a);
28    c.f(a_b);
29    c.f(b);
30    d.f(a);
31    d.f(a_b);
32    d.f(b);
```

   (a) Nennen Sie Beispiele, in denen Multimethoden von Vorteil wären.

   (b) Welche Methoden werden in normalem Java und welche bei Verwendung von Multi-Methoden aufgerufen?

(c) Gibt es Aufrufe, die jeweils in einer Variante eindeutig, aber in der anderen mehrdeutig sind?

(d) Machen Sie Vorschläge, wie man zur Laufzeit mit Mehrdeutigkeiten umgehen könnte.

2. **Callbacks in C++**

(a) Die in der Vorlesung vorgestellten Callbacks benutzen immer genau 2 Parameter. Kann man das System so erweitern, dass eine beliebige Anzahl von Parameter möglich ist? Was muss man dafür tun?

(b) Wie könnte man diesen Mechanismus analog in Java implementieren? Welche Einschränkungen bzw. Probleme können dabei auftreten?

3. **Aspekt-Orientierte Programmierung**

Folgendes ist Teil einer Bibliotheks-Verwaltung:

```
1       class User {
2         String name;
3         int year_of_birth;
4       }
5       class Book {
6         String title;
7         String category;
8         User lend_to;
9       }
10      class Library {
11        void borrow(User u, Book b) {
12          if(b.lend_to!=null)
13            throw new Error("user "+b.lend_to+
14                            " has to return the book first");
15          b.lend_to=u;
16        }
17        void giveback(User u, Book b) {
18          if(b.lend_to!=u)
19            throw new ConfusedLibraryException();
20          b.lend_to=null;
21        }
22      }
```

Erstellen Sie mit den Möglichkeiten von AspectJ, die Sie aus der Vorlesung kennen, für jeden dieser Punkte einen Aspekt:

(a) Die Klasse `User` enthält ab sofort eine Liste aller Bücher, die der Benutzer ausgeliehen hat.

(b) Keinem Benutzer ist es möglich, mehr als 3 Bücher gleichzeitig ausgeliehen zu haben.

(c) Kindern unter 12 ist es nicht mehr möglich, Bücher der Kategorie “Adult” auszuleihen.

4. **Visitor Pattern und Multimethoden**

   Sie haben eine Hierarchie mit folgenden Java-Klassen:

```
1  abstract class Expression { }
2
3  public class IntExpression extends Expression {
4      int value;
5  }
6
7  public class AddExpression extends Expression {
8      Expression e1, e2;
9  }
10
11 public class MultExpression extends Expression {
12     Expression e1, e2;
13 }
```

   Nun möchten Sie nachträglich eine *prettyPrint*()-Methode implementieren, welche eine Expression korrekt formatiert ausgibt, ohne sie direkt in die Klassen zu implementieren. Welche Probleme entstehen dabei? Lösen Sie dieses Problem einmal durch Anwendung des *Visitor Patterns* und einmal mit *Multimethoden*. Welche Vor- und Nachteile haben beide Ansätze?