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 6 Ausgabe: 9.6.2008 Besprechung: 11.6.2008

1. **Refactoring**

   Sie haben das Refactoring-Pattern "Replace Conditional with Polymorphism" kennengelernt.

   (a) Wenden Sie es auf das folgende Beispiel an:

```
1         class Employee ...
2         ...
3         int payAmount() {
4           switch(type) {
5             case ENGINEER:
6             return salary;
7             case SALESMAN:
8             return salary+commission;
9             case MANAGER:
10            return salary+bonus;
11          }
12        }
13        ...
```

   (b) Wo liegen die Probleme nach der Anwendung des Patterns?

   (c) Welche Möglichkeiten sehen Sie, dieses Problem elegant zu lösen?

2. **Iteratoren und komplexe Datenstrukturen**

   Gegeben sei folgende Klasse:

```
1       class Tree<E> {
2         Tree<E> left, right;
3         E data;
4         TreeIterator<E> iterator();
5       }
```

   (a) Skizzieren Sie den Code für eine Implementation eines entsprechenden Iterators.

   (b) Wie müsste die Implementierung für andere Reihenfolgen, in der die Elemente durchlaufen werden (Pre-, Post- oder In-Order), geändert werden?

3. **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.