



Universität Karlsruhe (TH)

Lehrstuhl für Programmierparadigmen

Fortgeschr. Objektorientierung SS 2009 <http://pp.info.uni-karlsruhe.de/>
Dozent: Prof. Dr.-Ing. G. Snelting snelting@ipd.info.uni-karlsruhe.de
Übungsleiter: Andreas Lochbihler lochbihl@ipd.info.uni-karlsruhe.de
Dennis Giffhorn giffhorn@ipd.info.uni-karlsruhe.de

Blatt 9

Ausgabe: 19.06.2009

Besprechung: 24.06.2009

1. Callbacks in C++

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

2. Innere Klassen von inneren Klassen

Betrachten Sie das folgende Java-Programm:

```
1 class A {
2     int i;
3     static int j;
4
5     class B {
6         int k;
7         static int l;
8         void b() {
9             ++i; ++j;
10        }
11        class C {
12            void c() {
13                ++i; ++j;
14                ++k; ++l;
15            }
16        }
17        static class D {
18            void d() {
19                ++i; ++j;
20                ++k; ++l;
21            }
22        }
23    }
24 }
25 class E {
26     int i;
27     static int j;
28 }
```

```

29     static class F {
30         int k;
31         static int l;
32         void f() {
33             ++i; ++j;
34         }
35         class G {
36             void g() {
37                 ++i; ++j;
38                 ++k; ++l;
39             }
40         }
41         static class H {
42             void h() {
43                 ++i; ++j;
44                 ++k; ++l;
45             }
46         }
47     }
48     void e() {
49         ++i; ++j;
50         ++F.k; ++F.l;
51         F f=new F();
52         ++f.k; ++f.l;
53     }
54     static void e1() {
55         ++i; ++j;
56         ++F.k; ++F.l;
57         F f=new F();
58         ++f.k; ++f.l;
59     }
60 }

```

- (a) Welche der Klassendefinitionen sind illegal und warum?
- (b) Streichen sie die illegalen Klassen. Welche der verbleibenden Member-Zugriffe sind illegal und warum?

3. Zugriff auf äußere Members aus anonymen Klassen heraus

Betrachten Sie das folgende Java-Programm:

```
1 class Bar {
2
3     int y;
4
5     interface Foo {
6         void bar();
7     }
8
9     static Foo foobar(int x) {
10
11         Foo b = new Foo() {
12             int z = y;
13             public void bar() {
14                 z = x;
15             }
16             public String toString() {
17                 return Integer.toString(z);
18             }
19         };
20
21         x = 7;
22         return b;
23     }
24
25     void test() {
26         Foo f = foobar(3);
27         f.bar();
28         System.out.println(f);
29     }
30
31     public static void main(String[] args) {
32         new Bar().test();
33     }
34 }
```

- (a) Dieses Programm kompiliert nicht! Welche Fehler treten an welchen Stellen aus welchem Grund auf?
- (b) Korrigieren Sie die bemängelten Stellen des Programms, bis es compiliert. Was gibt das Programm dann aus? Warum?

4. Vererbungsbeziehungen bei inneren Klassen

Betrachten Sie folgende Klassenhierarchie:

```
1 class Bar {
2
3     final int x = 42;
4
5     interface I {
6         final int X = x;
7         void bar();
8     }
9
10    abstract class A implements I { }
11
12    static class B extends A {
13        public void bar() { }
14    }
15
16    static class C implements I {
17        public void bar() { }
18    }
19 }
20
21 class Foo {
22     Bar b;
23
24     class D implements Bar.I {
25         public void bar() {}
26     }
27
28     void test() {
29         Bar.I i;
30         i = new Bar.B();
31         i = new Bar.C();
32         i = new Bar.A() {
33             public void bar() {}
34         };
35         i = new D();
36     }
37
38 }
```

An welchen Zeilen meldet der Compiler Fehler? Wie muss man den Code anpassen, damit das Programm compiliert?