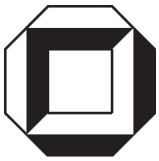

Universität Karlsruhe (TH)

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 11	Ausgabe: 02.07.2009	Besprechung: 08.07.2009

1. **Cardelli-$\leq$**

   (a) Geben Sie Cardelli-Typen für die folgenden Funktionen an:

```
1  int S(int x) { return x+1; }
2  int SD(double x) { return x+1; }
3  int twice(int f(int), int y) { return f(f(y)); }
4  int twice_S(int y) { return twice(S, y); }
5  int twice_SD(int y) { return twice(SD, y); }
```

   (b) Sind alle Aufrufe im Cardelli-Typsystem erlaubt? In C++?

   (c) Leiten Sie den Typ für den Ausdruck $(\lambda x.\ \lambda y.\ y(x))(42)$ durch Rückwärtsanwendung der Regeln her. Notieren Sie in jedem Schritt die Typannahmen. Dies ist die Menge der jeweils aufgetretenen Teil-Ausdrücke und ihrer Typen.

2. **Typkonvertierungen bei Methodenredefinition**

Betrachten Sie das folgende C++-Programm:

```
1  #include <iostream>
2  using namespace std;
3  class A {
4  public:  virtual void foobar() {
5    cout << "A::foobar called" << endl;
6  }
7  };
8  class B : public A {
9  public:  virtual void foobar() {
10   cout << "B::foobar called" << endl;
11 }
12 };
13 class RA {
14 private: A* a;
15 public:  virtual A* foo() { return a; }
16   virtual void bar(B* b) {
17     cout << "RA::bar called" << endl;
18   }
19 };
20 class RB : public RA {
21 private: B* b;
22 public:  virtual B* foo() { return b; }
23   virtual void bar(A* a) {
24     cout << "RB::bar called" << endl;
25   }
26 };
27 main() {
28   A* a; B* b; RA* ra; RB* rb; RA *rap;
29
30   ra->foo()->foobar(); ra->bar(a); ra->bar(b);
31   rb->foo()->foobar(); rb->bar(a); rb->bar(b);
32   rap=rb;
33   rap->foo()->foobar(); rap->bar(a); rap->bar(b);
34 }
```

(a) Entsprechen die Redefinitionen von *foo* und *bar* den Bedingungen des Cardelli-Typsystems? Geben Sie diese Bedingungen an!

(b) Was sagt ein C++-Compiler zu dem Programm? Welche der Aufrufe in `main()` sind legal, welche sind illegal? Welche Funktionen werden zur Laufzeit wirklich aufgerufen? Welche Gemeinsamkeiten und Unterschiede bestehen zum Cardelli-Typsystem?

(c) Kann man das Beispiel 1:1 nach Java übertragen? Bietet Java in Bezug auf das Cardelli-Typsystem mehr oder weniger Möglichkeiten als C++?

3. **Parameter- und Rückgabetypen bei Methodenredefinitionen**

(a) In C++ (und Java) sind Methodenredefinitionen mit kleineren Rückgabetypen erlaubt. Betrachten Sie folgende Klassenhierarchie:

```
1 class Top {};
2 class Left : public Top {};
3 class Right : public Top {
4 public:
5   virtual Top* f() { return new Top(); } };
6 class Bottom : public Left, public Right {
7 public:
8   virtual Bottom* f() { return new Bottom(); }
9 };
```

Welche Ausgabe liefert der Compiler? Wie erklären Sie Sich das? Was bedeutet das für das C++ Typsystem?

(b) Es gibt Sprachen, welche bei Methodenredefinitionen größere Parametertypen erlauben. In C++ (und Java) ist das jedoch nicht erlaubt. Betrachten Sie folgende Klassenhierarchie:

```
1  class Left {};
2  class Right {};
3  class Bottom : public Left, public Right {};
4
5  class A {
6  public: virtual void f(Bottom b) { ... }
7  };
8  class B : public A {
9  public: virtual void f(Right r) { ... }
10 };
```

Nehmen wir an, Redefinitionen mit größeren Parametertypen wären in C++ erlaubt. Welche Probleme könnten in diesem Programm auftreten? Zeigen Sie das an einem unsicheren Methodenaufruf.