

## Übungen zu „Prinzipien von Programmiersprachen“ Blatt 8

**Aufgabe 24.** Erweitere die *counter* Klasse um eine Methode  $accesses : \{\} \rightarrow Nat$ , die protokolliert, wie oft *get* aufgerufen wurde.

```
function new-counter{ } : Counter =  
  let  
    val x = ref 0  
  in  
    { get = fun{ } => !x,  
      inc = fun{ } => x := !x + 1 }  
  end
```

Wie ändert sich die Klassendefinition, wenn nicht die Anzahl der *get* Aufrufe pro Objekt, sondern die Gesamtzahl aller *get* Aufrufe protokolliert werden soll?

**Aufgabe 25.** In der Vorlesung (§5.3) haben wir verschiedene Ansätze kennengelernt, objektorientierte Konzepte zu simulieren. Wie werden in diesen Ansätzen die simulierten Objekte jeweils zur Laufzeit repräsentiert? Verwende für die Überlegungen die auf Blatt 2 eingeführte *Umgebungssemantik*. Welche Teile der Repräsentation werden von allen Objekten geteilt, welche Teile sind spezifisch für ein einzelnes Objekt? Wie verhält sich die Repräsentation zu realen Implementierungen objektorientierter Sprachen?

**Aufgabe 26.** Ein weiteres Merkmal objektorientierter Sprachen ist die *Objektidentität*: die Operation *same* ( $e_1, e_2$ ) ergibt *true*, wenn beide Argumente zum gleichen Objekt evaluieren (das vom gleichen **new** Aufruf erzeugt wurde) und *false* sonst. Wie kann die Simulation von Objekten erweitert werden, so dass sie die Objektidentität unterstützt?

**Aufgabe 27.** Viele Datenstrukturen können wahlweise mit Variantentypen oder mit Klassen implementiert werden. Die folgenden Codeschnipsel implementieren einfache, arithmetischer Ausdrücke. Einmal mit Variantentypen:

```
data Expr = Nat Nat | Add{ expr1 : Expr, expr2 : Expr }  
function value (expr : Expr) : Nat =  
  case expr of Nat n      => n  
             | Add node => value (node.expr1) + value (node.expr2)  
  end
```

und einmal mit Klassen:

```

type Expr =
  object
    method value : Nat
  end
function nat (n : Nat) : Expr =
  new class
    method value : Nat = n
  end
function add (expr1 : Expr, expr2 : Expr) : Expr =
  new class
    method value : Nat = expr1.value + expr2.value
  end

```

Diskutiere die relativen Vor- und Nachteile beider Implementierungen. Wie gut sind die Implementierungen erweiterbar:

1. Wie einfach lässt sich die Syntax der Ausdrücke erweitern (z.B. um eine Multiplikationsoperation)?
2. Wie einfach lässt sich neue Funktionalität implementieren (z.B. ein Pretty Printer)?

Was muss jeweils geändert werden? Ist für die Änderung der Quellcode des Programms nötig oder reicht eine Beschreibung der Schnittstellen?