Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
CAU Kiel, Informatik und Prakt. Mathematik
Prof. Dr. M. Hanus, Dr. Frank Huch

24118 Kiel
Christian-Albrechts-Platz 4
Tel.: 0431 / 880-7277

7. Übung „Prinzipien von Programmiersprachen“

Bearbeitung bis zum 14. Dezember 2004

Aufgabe 1

Wir betrachten Unterklassenbildung, wie sie in der Vorlesung definiert wurde (mit Ko- und Kontravarianz). Gegeben seien folgende Klassendefinitionen:

```
class A {
   τ3 c1;
   public τ1 m (τ2 x) {
     ...
   }
}
```

```
class C {
   public void m() {
      ...
   }
}
```

```
class B extends A {
   τ3' c1;
   public τ1' m (τ2' x) {
     ...
   }
}
```

```
class D extends C {
   public void m() {
      ...
   }
}
```

Geben Sie alle möglichen Kombinationen für $\tau_1, \tau_2, \tau_3, \tau_1', \tau_2', \tau_3' \in \{\texttt{C}, \texttt{D}\}$ an, die laut Vorlesung erlaubt sind.

Welche Kombinationen für die Typen sind in Java erlaubt? Bei welchen handelt es sich in Java wirklich um Vererbung?

Aufgabe 2

Definieren Sie eine abstrakte Klasse `SearchTree`. Ein Objekt dieser Klasse soll einen geordneten binären Baum repräsentieren. Die Klasse soll die beiden abstrakten Methoden `boolean contains(int x)` und `SearchTree insert(int x)` besitzen. Die Objekte, welche in dem Baum gespeichert werden sollen müssen vergleichbar sein (`CompLess`), sprich eine Methode `boolean less (CompLess other)` implementieren.
Konkretisieren Sie den Suchbaum mittels zweier Klassen `Node` und `EmptySearchTree`. Ein Objekt der Klasse `Node` besitzt drei Attribute: Eine Objekt, welches `CompLess` implementiert und die Attribute `left` und `right` vom Typ `SearchTree`. Ein Objekt der Klasse `EmptySearchTree` besitzt kein Attribut.
Geben Sie für Zahlen eine Implementierung für `CompLess` an. Programmieren Sie eine Schleife, in der nach und nach beliebige Zahlen in einen Suchbaum eingefügt werden bzw. nachgeschlagen werden können, welches z.B. folgende Interaktion ermöglicht:

```
> insert 3
> lookup 2
2 is not in tree.
> insert 2
> lookup 2
2 is in tree.
```

wobei die Texte hinter den `>`-Zeichen Benutzereingaben sein sollen.

Aufgabe 3
In dieser Aufgabe sollen Sie sich mit dem Hugs-System vertraut machen.

a) Verwenden Sie hierzu zunächst das in der Vorlesung vorgestellte Programm zur Fakultätsberechnung.

   Untersuchen Sie wieviele Schritte zur Berechnung von `fac 10` notwendig sind. Die Ausgabe der Schrittzahl kann in Hugs mit `:set +s` eingeschaltet werden.

b) Implementieren Sie die Fibonacci-Funktion in Haskell.

$$\begin{aligned} fib(0) &= 0 \\ fib(1) &= 1 \\ fib(n) &= fib(n-1) + fib(n-2) \quad \forall n > 1 \end{aligned}$$

   Bestimmen Sie anhand der benötigten Reduktionsschritte die Zeitkomplexität Ihrer Implementierung (liegt in $O(??)$). Die einfachste Methode ist eine graphische Auswertung.

   Können Sie Ihre Funktion entscheidend optimieren?

Wichtige Interpreter-Kommandos des Hugs-Systems:

- `:l` *filename*

  Löschen aller vorhandenen Funktionsdefinitionen (außer Prelude) und Laden des Programms *filename*.

- `:r`

  Wiederholt die letzte Ladeoperation, somit werden alle Funktionsdefinitionen aktualisiert.

- `:q`

  Verlassen des Hugs-Systems.

- `:?`

  Kommandoliste wird angezeigt.

- *Ausdruck* (z.B. `fac 3`)

  Start einer Funktionsberechnung.