**Technische Universität München**
**Fakultät für Informatik**
**D. Gmach, M. Rickert,**
**Dr. M. Schneider**

**WS 2005/2006**
**Lösungsvorschläge zu Blatt 10**
**13. Januar 2006**

# Übungen zu Einführung in die Informatik I

## Aufgabe 41 Arithmetische Ausdrücke (Lösungsvorschlag)

a) `simple_expr` definiert einfache arithmetische Ausdrücke, d. h. nur Konstanten (`Const of int`) sind erlaubt.

```
# type simple_expr =
   | Const of int
   | Add   of simple_expr * simple_expr
   | Sub   of simple_expr * simple_expr
   | Mul   of simple_expr * simple_expr
   | Div   of simple_expr * simple_expr;;
type simple_expr =
    Const of int
  | Add of simple_expr * simple_expr
  | Sub of simple_expr * simple_expr
  | Mul of simple_expr * simple_expr
  | Div of simple_expr * simple_expr
```

Die Funktion `simple_eval e` gibt den int Wert des Ausdrucks `e` zurück.

```
# let rec simple_eval = function
   Add (e1,e2) -> (simple_eval e1) + (simple_eval e2) |
   Sub (e1,e2) -> (simple_eval e1) - (simple_eval e2) |
   Mul (e1,e2) -> (simple_eval e1) * (simple_eval e2) |
   Div (e1,e2) -> (simple_eval e1) / (simple_eval e2) |
   Const i     -> i;;
val simple_eval : simple_expr -> int = <fun>
```

Beispiel für einen einfachen Ausdruck:

```
# let simple_exp =
   Add (
      Const 5,
      Mul (
         Const 2,
         Const 3
      )
   );;
val simple_exp : simple_expr = Add (Const 5, Mul (Const 2, Const 3))
# simple_eval simple_exp;;
- : int = 11
```

b) `variable` definiert eine feste und endliche Menge an Variablen

```
# type variable = X | Y | Z;;
type variable = X | Y | Z
# type expr =
   | Const of int           (** constant expressions *)
   | Var   of variable      (** variables *)
   | Add   of expr * expr
   | Sub   of expr * expr
   | Mul   of expr * expr
   | Div   of expr * expr;;
type expr =
    Const of int
  | Var of variable
  | Add of expr * expr
  | Sub of expr * expr
  | Mul of expr * expr
  | Div of expr * expr
```

`eval r e` gibt den Wert des Ausdrucks `e` unter Berücksichtigung der Variablenbelegung `r` zurück.

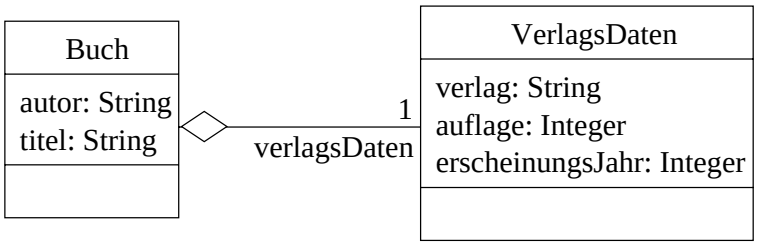
```
# let rec eval rho = function
    Add (e1,e2) -> (eval rho e1) + (eval rho e2) |
    Sub (e1,e2) -> (eval rho e1) - (eval rho e2) |
    Mul (e1,e2) -> (eval rho e1) * (eval rho e2) |
    Div (e1,e2) -> (eval rho e1) / (eval rho e2) |
    Const i     -> i |
    Var v       -> rho v;;
val eval : (variable -> int) -> expr -> int = <fun>

# let exp =
   Add (
      Var X,
      Mul (
         Var Y,
         Var Z
      )
   );;

val exp : expr = Add (Var X, Mul (Var Y, Var Z))

# let rho_1 = function X -> 5 | Y -> 2 | Z -> 3;;
val rho_1 : variable -> int = <fun>
# let rho_2 = function X -> 3 | Y -> 2 | Z -> 5;;
val rho_2 : variable -> int = <fun>
# eval rho_1 exp;;
- : int = 11
# eval rho_2 exp;;
- : int = 13
```

## Aufgabe 42 Quicksort in Java (Lösungsvorschlag)

```
public class Quicksort {

    public static int iter = 0;

    public static void printArray(int[] a) {
//      for (int i = 0; i < a.length; i++) { // Java 1.4
//          System.out.print(a[i] + ", ");
//      }
        for (int i: a) {  // Java 1.5
            System.out.print(i + ", ");
        }
        System.out.println();
    }

    public static void swap(int[] a, int i, int j) {
        int temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }

    public static void sort(int[] a, int first, int last) {
        if (first < last ) {
            int pivot = a[last]; //take last element as pivot element
            int storeIndex = first;

            // using in-place partition
            for (int i = first; i < last; i++) {
                if (a[i] <= pivot) {
                    swap(a, storeIndex, i);
                    storeIndex++;
                }
            }
            swap(a, last, storeIndex); // Move pivot element to its final place

            sort(a, first, storeIndex-1);
            sort(a, storeIndex+1, last);
        }
    }

    public static void sort(int[] a) {
        sort(a, 0, a.length-1);
    }

    public static void main(String args[]) {

        int[] array = {5,2,6,34,11,87,0,3,99,4};

        printArray(array);
        sort(array);
        printArray(array);
    }

}
```

## Aufgabe 43 Die Klasse Buch in Java und UML (Lösungsvorschlag)

a) Die Aufgabenstellung legt nahe, für die Verlagsdaten eine eigene Klasse einzuführen, die durch Aggregation mit der Klasse Buch in Beziehung steht. Ein Eintrag über die Verlagsdaten ist also Bestandteil einer Buchbeschreibung. Dies führt zu folgendem UML-Diagramm.

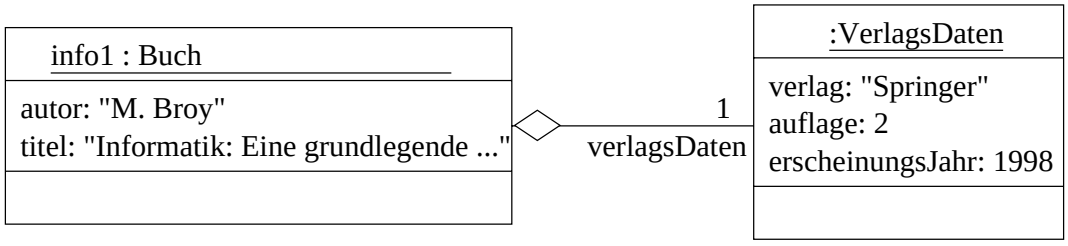



In Java wird das Buch dann mit der Definition der folgenden beiden Klassen realisiert.

```
public class Buch {
   private String autor;
   private String titel;
   private VerlagsDaten verlagsDaten;

   public Buch () {
      autor = "XXXX";
      titel = "XXXX";
      verlagsDaten = new VerlagsDaten();
   }

   public Buch (String a, String t, VerlagsDaten v) {
      autor = a;
      titel = t;
      verlagsDaten = v;
   }

   public Buch (String aut, String tit, String verl, int aufl, int
       jahr) {
      autor = aut;
      titel = tit;
      verlagsDaten = new VerlagsDaten(verl,aufl,jahr);
   }
}

public class VerlagsDaten {
   private String verlag;
   private int    auflage;
   private int    erscheinungsJahr;

   public VerlagsDaten () {
       verlag = "XXX";
       auflage = 0;
       erscheinungsJahr = 0;
   }

   public VerlagsDaten (String v, int a, int j) {
       verlag = v;
       auflage = a;
       erscheinungsJahr = j;
```

```
    }
}
```

b) Die Instanz für das Buch von M.Broy wird in UML wie untenstehend notiert.




In den Klassendfinitionen wurden jeweils mehrere Konstruktoren realisiert. Damit ist es möglich auf unterschiedliche Weise Objekte der Klassen zu erzeugen. Für die Bearbeitung der Teilaufgabe a) reicht jeweils die Angabe eines Konstruktors aus. Um jedoch die Möglichkeit zu haben, ein Objekt mit unterschiedlichen Daten zu instanziieren, sollte ein Konstruktor definiert werden, der die Übergabe der gewünschten Daten als Parameter erlaubt. Wird nur der Konstruktor ohne Parameter anegeben, müsste die Klasse zusätzlich noch die `set`-Operationen anbieten, um Belegung der einzelnen Attribute mit den Daten zu ermöglichen.

Nachfolgend zwei Möglichkeiten, das Objekt `info1` zu instanziieren.

```
Buch info1 =
     new Buch ("M. Broy",
               "Informatik: Eine grundlegende Einführung, Band 1.
                  Programmierung und Rechnerstrukturen",
               "Springer", 2, 1998);

Buch info1 =
     new Buch ("M. Broy",
               "Informatik: Eine grundlegende Einführung, Band 1.
                  Programmierung und Rechnerstrukturen",
               new VerlagsDaten ("Springer", 2, 1998));
```

## Aufgabe 44 Numerische Integration (Lösungsvorschlag)

a) Numerische Integration mittels Ocaml:

```
let equidistantpoints (x0, x1) n =
        let dist = abs_float((x0 -. x1)/. float_of_int(n)) in
        let rec points (x, last) =
                if (x > last) then last :: []
                else x :: points (x +. dist, last)
        in points (x0,x1);;

let integrate f (x0, xn, n) =
        let rec sum f list = match list with
                        |[]|(_ :: []) -> failwith "Zu wenig Punkte
                           "
                        |xi :: xip1 :: [] -> 0.5*.(xip1 -. xi)*.(f
                            xi +. f xip1)
```

```
                        |xi :: xip1 :: rest ->
                                0.5*.(xip1 -. xi)*.(f xi +. f xip1
                                   ) +. sum f (xip1 :: rest)
        and points = equidistantpoints (x0, xn) n
        in sum f points;;

let f x = x*.x;;

let result = integrate f (0.0, 1.0, 100);;
```

b) Numerische Integration mittels Java:

```
class NumerischeIntegration {

        public static double f (double x) {
                return x*x;
        }

        public static double[] equidistantPoints (double x0,
           double xn, int n){
                double[] points = new double[n];
                double dist = Math.abs(x0 - xn)/n;
                for (int i=0; i<n; i++){
                        points[i] = x0 + dist * i;
                }
                points[n-1] = xn;
                return points;
        }

        public static double integrate (double[] f, double[] p){
                double result = 0;
                int n = p.length;
                for (int i = 0; i < n-1; i++){
                        result = result + 0.5*(p[i+1]-p[i])*(f[i]+
                           f[i+1]);
                }
                return result;
        }

        public static void main (String[] args){
                double x0 = 0;
                double xn = 1;
                int n = 100;
                double[] points = equidistantPoints(x0, xn, n);
                double[] ff = new double[n];
                for (int i=0; i<n;i++){
                        ff[i] = f(points[i]);
                }
                System.out.println(integrate(ff, points));
        }
}
```