Technische Universität München
Fakultät für Informatik
R. Kuntschke, I. Nagy,
Dr. M. Schneider

SS 2006
Lösungsvorschläge zu Blatt 9
7. Juli 2006

Übungen zu Einführung in die Informatik I

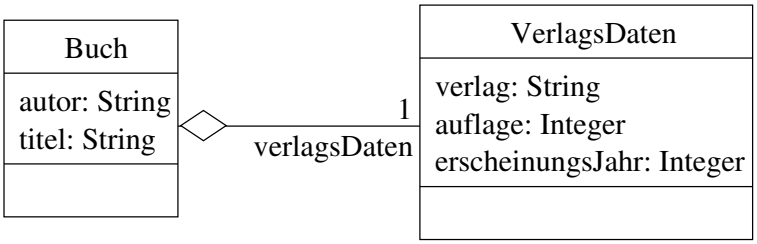
Aufgabe 36 Quicksort in Java (Lösungsvorschlag)

```
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 37 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.

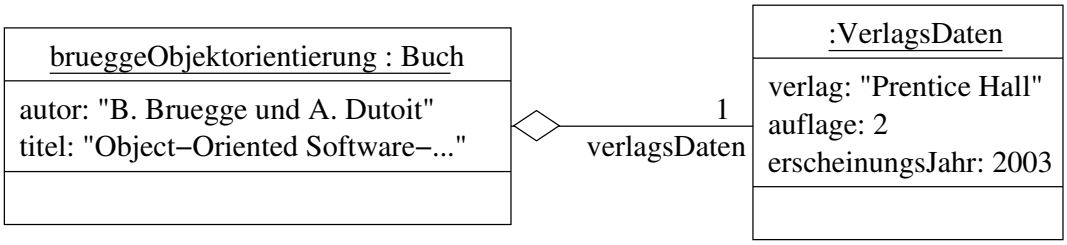


In Java wird das Buch dann mit der Definition der folgenden beiden Klassen realisiert.

```
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 38 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));
        }
    }
```

Aufgabe 39 Rekursive Berechnung des Tangens (Lösungsvorschlag)

Eine mögliche Lösung könnte folgendermaßen aussehen:

```
class Tangent
{
  public static double tg_1(double x)
  {
    double tg;

    if (x > -0.0001 && x < 0.0001)
      tg = x;
    else
      tg = 2 * tg_1(x / 2) / (1 - tg_1(x / 2) * tg_1(x / 2));

    return tg;
  }

  public static double tg_2(double x)
  {
    double tg, tgOfHalfAngle;

    if (x > -0.0001 && x < 0.0001)
      tg = x;
    else
    {
      tgOfHalfAngle = tg_2(x / 2);
      tg = 2 * tgOfHalfAngle / (1 - tgOfHalfAngle * tgOfHalfAngle);
    }

    return tg;
  }

  public static void main(String argv[])
  {
    double value = 7.89;
    System.out.print("tg_1(" + value + ") = ");
    System.out.println(tg_1(value));
    System.out.print("tg_2(" + value + ") = ");
    System.out.println(tg_2(value));
  }
}
```

Aufgabe 40 Capture-Recapture Methode, oder Fische im Teich zählen (Lösungsvorschlag)

Eine mögliche Lösung könnte folgendermaßen aussehen:

```
import java.util.Random;

public class CaptureRecapture
{
  public static int undiscovered = 0;
  public static int marked = 1;
  public static int marked_and_recaptured = 2;

  public static int number_fishes = 12345;
  public static int number_captures = 1234;
  public static int number_recaptures = 123;

  public static int fishes[] = new int[number_fishes];

  public static int getRandomInt(int max)
  {
    double dr = Math.random() * max;

    int ir = (int)Math.floor(dr);

    return ir;
  }

  public static void main(String argv[])
  {
    int irand;

    // capture some fishes
    for (int i = 0; i < number_captures; i++)
    {
      irand = getRandomInt(number_fishes);

      if (fishes[irand] == undiscovered)
      {
        fishes[irand] = marked;
      }
      else
      {
        while (fishes[irand = getRandomInt(number_fishes)]
               != undiscovered)
          ; // System.out.println("Already captured, throw him back!");
        fishes[irand] = marked;
      }
    }

    // recapture some fishes
    int num_marked_recaptures = 0;
    for (int i = 0; i < number_recaptures; i++)
    {
      irand = getRandomInt(number_fishes);

      if (fishes[irand] == marked)
      {
```

```
        fishes[irand] = marked_and_recaptured;
        num_marked_recaptures++;
      }
      else if (fishes[irand] == marked_and_recaptured)
      {
        while (fishes[irand = getRandomInt(number_fishes)]
               != marked_and_recaptured)
          ; // System.out.println("Already recaptured, throw him back
              !");
        if (fishes[irand] == marked)
          num_marked_recaptures++;
      }
    }
    int estimated = num_marked_recaptures > 0 ?
        number_recaptures * number_captures / num_marked_recaptures
        : 0;

    System.out.println("Estimated␣number␣of␣fishes:␣" +
                       estimated +
                       ",deviation␣" +
                       (100 - 100 * estimated / number_fishes) + "␣%");
  }
}
```