

**Praktische Informatik 1**  
(Vorlesung von Prof. Bothe)**WS 05/06**

## Übungsblatt 6

Abgabe: 12.12 - 14.12.05 in den Übungen

Die Lösung dieses Übungsblatts soll in Gruppen von je 2 Personen erfolgen.

Die Lösungen sind inklusive Lösungsweg auf Papier abzugeben. Auf jedem Lösungsblatt sind die Matrikelnummern beider Gruppenmitglieder anzugeben!

**Aufgabe 1 (insgesamt 4 Punkte)**

Betrachten Sie folgende Java-Methode:

```
public static int[] reverse(int[] feld) {           // Zeile 1
    for (int i=0; i<feld.length; i++) {          // Zeile 2
        int hilf = feld[i];                       // Zeile 3
        feld[i] = feld[feld.length-1-i];         // Zeile 4
        feld[feld.length-1-i] = hilf;           // Zeile 5
    }                                             // Zeile 6
    return eingabefeld;                          // Zeile 7
}                                               // Zeile 8
```

Diese Methode soll die Reihenfolge der Werte des Eingabefeldes umkehren und dieses zurückgeben.

**a) (1 Punkt)** Warum haben die Werte des Feldes nach dem Methodenaufruf immer noch die gleiche Reihenfolge? Wie kann das korrigiert werden?

**b) (3 Punkte)** Wie muss das Programm geändert werden, damit das Eingabefeld unverändert bleibt und dennoch ein Feld mit umgekehrter Reihenfolge zurückgegeben wird?

**Aufgabe 2 (1 Punkt)**

Ein gestresster Programmierer hat folgendes Stück Code geschrieben. Ersetzen Sie diesen Code durch eine einzige Zuweisung zur Variablen `resultat`.

```
if (x % 3 == 0) {
    if (x % 2 == 0) {
        resultat = true;
    }
    else {
        resultat = false;
    }
}
else {
    resultat = false;
}
```

### Aufgabe 3 (insgesamt 7 Punkte)

Die Multiplikation von natürlichen Zahlen ( $m, n > 0$ ) kann man wie folgt auf das Addieren, Subtrahieren und Verdoppeln zurückführen (Russische Bauernmultiplikation):

$$m \cdot n = \begin{cases} m & \text{falls } n=1 \\ m \cdot (n-1) + m & \text{falls } n>1, \text{ ungerade} \\ 2m \cdot \frac{n}{2} & \text{falls } n \text{ gerade} \end{cases}$$

a) (1 Punkt) Berechnen Sie  $257 * 37$  und geben Sie alle Zwischenergebnisse an!

b) (6 Punkte) Schreiben Sie ein Java-Programm, das die obige Rechenregel benutzt. Definieren Sie dazu die Funktionen

```
static boolean odd(int x); // wahr, falls x ungerade
static int mul2(int x); // verdoppelt x
static int div2(int x); // halbiert x
```

sowie, mit deren Hilfe und ohne Verwendung des Multiplikationsoperators, die Funktion

```
static int multiply(int m, int n); // m*n
```

### Aufgabe 4 (insgesamt 2 Punkte)

Der Programmierer aus Aufgabe 2 hat auch folgendes Stück Code geschrieben. Es soll in Variable  $s$  das Vorzeichen von  $x$  ablegen, also -1 falls  $x$  negativ, 0 falls  $x$  gleich 0 und 1 falls  $x$  positiv ist.

```
switch (x) {
  case 0:
    s = 0;
  default:
    if (x > 0) {
      s = 1;
    }
    else {
      s = -1;
    }
}
```

a) (1 Punkt) Was tut dieses Fragment tatsächlich?

b) (1 Punkt) Wie muss es korrigiert werden, um der oben gegebenen Spezifikation zu genügen?

### Aufgabe 5 (insgesamt 6 Punkte)

Die Methode `ist_prim(a)` liefert wahr, falls die Zahl `a` eine Primzahl ist, und falsch sonst. `teilbar(a, teiler)` liefert wahr, wenn die Division `a/teiler` aufgeht.

Ein Programm soll die Primzahlen aus dem abgeschlossenen Intervall `[3,30]` ausgeben, wobei jede Zahl mit Hilfe von `ist_prim` auf die Primeigenschaft getestet wird.

**a) (5 Punkte)** Welche Ausgaben wird das Programm liefern, wenn die Methode `ist_prim` wie unten implementiert ist? Geben Sie für jede der drei *Varianten* die Ausgaben des oben erwähnten Programms an. Erklären Sie die Unterschiede der zweiten und dritten Alternative gegenüber der ersten Alternative.

**b) (1 Punkt)** Ersetzen Sie die `while`-Schleife in *Variante 1* durch eine `for`-Schleife, die die gleiche Wirkung hat.

#### *Variante 1:*

```
static boolean ist_prim(int a) {
    boolean prim = true;
    int teiler = 3;
    if (a % 2 == 0) {
        prim = false;
    }
    else {
        while (teiler * teiler <= a) {
            if (teilbar(a, teiler)) {
                prim = false;
            }
            teiler = teiler + 2;
        }
    }
    return prim;
}
```

#### *Variante 2:*

```
static boolean ist_prim(int a) {
    boolean prim = true;
    int teiler = 3;
    if (a % 2 == 0) {
        prim = false;
    }
    else {
        while (teiler * teiler <= a) {
            if (teilbar(a, teiler)) {
                prim = false;
            }
            teiler = teiler + 2;
            break;
        }
    }
    return prim;
}
```

*Bitte auch nächste Seite beachten!*

**Variante 3:**

```
static boolean ist_prim(int a) {
    boolean prim = true;
    int teiler = 3;
    if (a % 2 == 0) {
        prim = false;
    }
    else {
        while (teiler * teiler <= a) {
            if (teilbar(a, teiler)) {
                prim = false;
            }
            teiler = teiler + 2;
            continue;
        }
    }
    return prim;
}
```