

**Einführung in die Programmierung**  
WS 2014/15

**Übungsblatt 8: Schleifen, Algorithmen**

Besprechung: 10./12./15.12.2014

Ende der Abgabefrist: Dienstag, 09.12.2014 14:00 Uhr.

**Hinweise zur Abgabe:**

Geben Sie bitte Ihre gesammelten Lösungen zu diesem Übungsblatt in einer Datei `loesung08.zip` unter <https://uniworx.ifi.lmu.de> ab.

Alle Code-Aufgaben müssen als `.java` abgegeben werden. PDF-code wird nicht akzeptiert. Nicht kompilierbare Abgaben werden mit 0 Punkten bewertet.

**Aufgabe 8-1** *Terminierung von Schleifen*

**4 Punkte**

Welche der folgenden Schleifen terminieren, welche terminieren nicht? Begründen Sie Ihre Antworten kurz. Geben Sie Ihre Lösung in einer Datei `schleifen.txt` oder `schleifen.pdf` ab.

(a) 

```
int i = 1;
int j = 1;
do
{
    i = i + j;
    j++;
}while(i < 200);
```

(c) 

```
int i = 100;
int j = 27;
while (i != j)
{
    i = i / 20;
    j = j / 3;
}
```

(b) 

```
int i = 1;
int j = 20;
while(i + j > i)
{
    i = i + 2;
    j--;
}
```

(d) 

```
int i = 26;
int j = 24;
for (int x = 0; x < 1000; x++)
{
    i = i / 12 + 23 * x;
    j = (x--) + j + 5;
}
```

**Aufgabe 8-2**     *Kontrollstrukturen***0 Punkte**

Im folgenden sind zwei Programmausschnitte (a), (b) und (c) gegeben. Betrachten Sie alle vorkommenden Variablen als deklariert und initialisiert.

```
(a)  int i = 0;
      int j = 42;
      while (i != j) {
          i++;
      }

(b)  while (a < b) {
      c = a;
      a = b;
      b = c;
    }

(c)  while (a < b)
      c = a;
      a = b;
      b = c;
```

Überlegen Sie, was in den obigen Schleifen (a), (b) und (c) passiert. Wie können Sie den gleichen Effekt ohne Verwendung von Wiederholungsanweisungen erzielen? Geben Sie ein entsprechendes Code-Fragment an.

(d) Schreiben Sie für die folgende **do-while**-Schleife jeweils eine äquivalente **while**- und **for**-Schleife.

```
int n = 1;
double x = 0;
double s;
do {
    s = 1.0 / (n * n);
    x = x + s;
    n++;
} while (s > 0.01);
```

Geben Sie Ihre Lösungen als Datei `kontrollstrukturen.txt` oder `kontrollstrukturen.pdf` ab.

**Aufgabe 8-3**     *Iteration in Java***1+1+2+2+2 Punkte**

Implementieren Sie die folgenden Berechnungen, welche Sie aus dem letzten Übungsblatt kennen, nun **auf iterative Weise** in der Datei `Iteration.java` und erstellen Sie für jede Unteraufgabe eine Methode. Eine rekursive Berechnung oder die Verwendung von Bibliotheksfunktionen wie beispielsweise `Math.pow()` oder `Math.sqrt()` wird nicht akzeptiert. Eigene Hilfsmethoden dürfen verwendet werden. Die Abgabedatei enthält eine `main`-Methode mit der Sie Ihre Lösungen überprüfen können.

(a) Berechnen Sie für ein  $x \in \mathbb{R}$  und ein  $n \in \mathbb{N}$  das Ergebnis der Exponentialfunktion  $x^n$ . Ein Beispiel:

`exp (4.2, 3) = 74.088`

(b) Berechnen Sie für ein  $n \in \mathbb{N}$  das Ergebnis der Fakultätsfunktion  $n! = 1 \times 2 \times 3 \times \dots \times n$ . Ein Beispiel:

`fakultaet (4) = 24`

(c) Überprüfen Sie, ob es sich bei einem  $n \in \mathbb{N}$  um eine Primzahl handelt, indem Sie alle potenziellen Teiler von 2 bis  $n - 1$  testen. Ein Beispiel:

`istPrim (893) = false`

(d) Setzen Sie den euklidischen Algorithmus zur Berechnung des größten gemeinsamen Teilers zweier Zahlen  $m, n \in \mathbb{N}$  iterativ um. Ein Beispiel-Aufruf:

`ggT (42, 154) = 14`

(e) Überführen Sie ein  $n \in \mathbb{N}$  in  $p$ -adische Darstellung. Sie können einen Gültigkeitsbereich für  $p$  von  $[2, 16]$  annehmen. Die Methode soll das Ergebnis in einem `String` zurück geben. Beispiele:

`rebase (5, 2) = "101"`  
`rebase (237, 16) = "ED"`

#### Aufgabe 8-4 Algorithmen

0 Punkte

Ziel dieser Aufgabe ist es, eine Methode zu implementieren, die alle möglichen Ergebnissequenzen beim  $n$ -maligen Werfen eines  $k$ -seitigen Würfels auf der Konsole ausgibt, wobei eine Seite des Würfels zwischen 1 und  $k$  Augen hat. Die Reihenfolge der Ausgabe der Ergebnissequenzen spielt dabei keine Rolle. Gehen Sie zur Lösung der Aufgabe folgendermaßen vor.

- (a) Implementieren Sie in einer Datei `Wuerfeln.java` (die Datei kann auf der Website heruntergeladen werden) eine Methode `public static int exp(int b, int e)`, die den Wert von  $b^e$  berechnet. Eine Verwendung von `Math.pow()` ist in dieser Aufgabe *nicht* gestattet!
- (b) Implementieren Sie die Methode `public static void muenzwurf(int n)` in Java, die alle möglichen Ergebnissequenzen beim  $n$ -maligen Werfen einer Münze (Also eines zweiseitigen Würfels) ausgibt.

**Beispiel:** Ein Aufruf der Methode `muenzwurf(3)` würde (die Reihenfolge der Zeilen spielt keine Rolle) folgende Ausgabe auf der Konsole erzeugen:

```
1,1,1
1,1,2
1,2,1
1,2,2
2,1,1
2,1,2
2,2,1
2,2,2
```

**Hinweis:** Verwenden Sie die Methode `System.out.print(int x)` um eine Zahl  $x$  auf der Konsole auszugeben. Die Methode `System.out.println()` kann verwendet werden um einen Zeilenumbruch zu erzeugen. Kommas können Sie mit dem Befehl `System.out.print(", ")` ausgeben werden.

- (c) Erweitern Sie die zuvor entwickelte Methoden in der Klasse `Wuerfeln.java` eine Methode `public static void wuerfeln(int n, int k)`, die alle möglichen Ergebnissequenzen beim  $n$ -maligen Würfeln eines  $k$ -seitigen Würfels ausgibt. Mögliche Seiten eines Würfels sollen in dieser Teilaufgabe die Beschriftung  $1..k$  haben.

**Beispiel:** Die Ausgabe aller Sequenzen der Länge 3 eines zweiseitigen Würfels (`wuerfeln(3, 2)`) ist also äquivalent zur Ausgabe von `muenzwurf(3)`. Die Ausgabe aller Sequenzen der Länge 2 eines dreiseitigen Würfels (`wuerfeln(2, 3)`) würde folgende Ausgabe erzeugen:

```
1,1
1,2
1,3
2,1
2,2
2,3
3,1
3,2
3,3
```

Geben Sie die bearbeitete Datei `Wuerfeln.java` ab. Achten Sie darauf, dass die abgegebenen Klassen kompilieren; nicht kompilierbare Klassen werden nicht bewertet!