

VL05, Aufgabe 1 (Übung)

Welche Ausgaben bzw. Rückgaben werden beim Aufruf der folgenden beiden Methoden erzeugt? Nehmen Sie als Beispiel die Werte 1234 und 5678 für n an.

Wie groß ist die Rekursionstiefe (maximale Zahl gleichzeitig aktiver Aufrufe) der Methoden für n = 1234? Wie groß ist die Rekursionstiefe für allgemeines n?

- a)

```
public static void rev1(int n)
{
    System.out.print(n % 10);
    if (n > 9)
        rev1(n / 10);
}
```
- b)

```
public static int rev2(int n)
{
    if (n <= 9)
        return n;

    int logn = (int)Math.log10(n);
    int zehnHochLogn = (int)Math.pow(10, logn);

    return (n % 10) * zehnHochLogn + rev2(n / 10);
}
```
- c) Programmieren Sie eine iterative Methode, die dieselbe Ausgabe wie rev1 bzw. dieselbe Rückgabe wie rev2 erzeugt. Falls Sie Ihre Lösung testen wollen, finden Sie ein Rahmenprogramm in UEB05.zip.

VL05, Aufgabe 2 (Übung)

Bestimmen Sie für die nachstehenden Methoden folgende Informationen:

- Berechnen Sie die Anzahl der Aufrufe von tuwas() für n=3
- Bestimmen Sie die Anzahl der Aufrufe von tuwas() als Funktion von n
- Bestimmen Sie die asymptotische Zeitkomplexität (O-Notation) unter der Annahme, dass die asymptotische Zeitkomplexität von tuwas() O(1) ist

<pre>static int funk1(int n) { tuwas(); if (n <= 1) { return n; } else { return n + funk1(n - 1); } }</pre>	<pre>static void proz2(int n) { tuwas(); if (n > 0) { proz2(n - 1); proz2(n - 1); } }</pre>
--	--

Geben Sie zusätzlich jeweils die Rekursionstiefe als Funktion von n an.

VL05, Aufgabe 3 (Übung)

Die Ulam-Funktion für natürliche Zahlen ist wie folgt definiert:

$$\text{Ulam}(n) = \begin{cases} 1 & \text{für } n = 1 \\ \text{Ulam}\left(\frac{n}{2}\right) & \text{für } n \text{ gerade } \wedge n > 1 \\ \text{Ulam}(3 * n + 1) & \text{für } n \text{ ungerade } \wedge n > 1 \end{cases}$$

- a) Berechnen Sie folgende Funktionswerte auf dem Papier, und geben Sie alle Zwischenschritte an, die für die Berechnung jeweils notwendig sind.

$$\text{Ulam}(2) =$$

$$\text{Ulam}(3) =$$

Beispiel: $\text{Ulam}(18) = \text{Ulam}(9) = \dots$

- b) Programmieren Sie eine rekursive und eine iterative Funktion in Java, die die Werte der Ulam-Funktion berechnet. Falls Sie Ihre Lösung testen wollen, finden Sie ein Rahmenprogramm in `UEB05.zip`.

VL05, Aufgabe 4 (Praktikum)

Die Folge der Fibonacci-Zahlen ist rekursiv definiert durch:

1. $\text{fib}(0) = 0, \text{fib}(1) = 1$
2. $\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)$ für $n \geq 2$

Die Fibonacci-Zahlen ergeben somit die Folge 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

Programmieren Sie je eine Java-Funktion, die die n -te Fibonacci-Zahl rekursiv bzw. iterativ berechnet, ohne den Programmcode aus Aufgabe 5 zu verwenden. Wie ist das Laufzeitverhalten (asymptotische Zeitkomplexität) beider Varianten?

Hinweis: Ein Rahmenprogramm finden Sie in `UEB05.zip`.

VL05, Aufgabe 5 (Übung)

Im Sommersemester 2014 hat ein Student die Berechnung der n -ten Fibonacci-Zahl wie folgt rekursiv berechnet. Wie ist das Laufzeitverhalten dieser Implementierung? Wodurch wird es ausgelöst?

```
public static long fibRekursiv(int grenze, long fib1, long fib2)
{
    // fib1 enthält den Vorgänger, fib2 das Ergebnis
    if (grenze <= 1)
        return fib2;

    return fibRekursiv(grenze - 1, fib2, fib1 + fib2);
}
```