VL02, Lösung 1

Methode	Anzahl Aufrufe für $n=100$	Als Funktion von n	O-Notation
`proz1`	200	$f(n) = 2n$	$O(n)$
`proz2`	10000	$f(n) = n^2$	$O(n^2)$
`proz3`	11	$f(n) = \lfloor\sqrt{n}\rfloor + 1$	$O(\sqrt{n})$
`proz4`	98990100	$f(n) = 100(n^2-1)(n-1)$	$O(n^3)$
`proz5`	5050	$f(n) = 1+2+3+...+n = \frac{n(n+1)}{2}$	$O(n^2)$
`proz6`	7	$f(n) = \lfloor\log_2 n\rfloor + 1$	$O(\log_2 n)$

$\lfloor\ \rfloor$ bezeichnet die nächst kleinere ganze Zahl (Gaußsche Klammer).

Aufwandsfunktion $f(n)$	$f(n) = O(...)$	Beweis
`proz1`: $f(n) = 2n$	$f(n) = O(n)$	$f(n)$ ist bereits ein Vielfaches von n. Mit $k = 2$ und $n_0 = 1$ gilt: $f(n) \leq k*n$ für alle $n \geq n_0$.
`proz2`: $f(n) = n^2$	$f(n) = O(n^2)$	$f(n) = n^2$. Mit $k = 1$ und $n_0 = 1$ gilt: $f(n) \leq k*n^2$.
`proz3`: $f(n) = \lfloor\sqrt{n}\rfloor + 1$	$f(n) = O(\sqrt{n})$	Dazu müssen wir zeigen, dass es Konstanten k und n_0 gibt, so dass $f(n) \leq k*\sqrt{n}$ für alle $n \geq n_0$. Wir versuchen, $f(n)$ nach oben hin durch ein Vielfaches von $\sqrt{n}$ abzuschätzen. $\lfloor\sqrt{n}\rfloor+1 \leq \sqrt{n}+1$, da der ganzzahlige Anteil $\lfloor\sqrt{n}\rfloor$ von $\sqrt{n}$ niemals größer werden kann als $\sqrt{n}$ (die Nachkommastellen gehen beim ganzzahligen Anteil verloren). Wir haben also $\lfloor\sqrt{n}\rfloor$ nach oben hin durch $\sqrt{n}$ abgeschätzt. $\sqrt{n}+1 \leq \sqrt{n}+\sqrt{n}$, da für $n \geq 1$ gilt: $1 \leq \sqrt{n}$. Wir haben also 1 nach oben hin durch $\sqrt{n}$ abgeschätzt: $\sqrt{n}+\sqrt{n} = 2*\sqrt{n}$. Insgesamt gilt also: $f(n) \leq 2*\sqrt{n}$, für alle $n_0 \geq 1$.
`proz4`: $f(n) = 100(n^2-1)(n-1)$	$f(n) = O(n^3)$	$100*(n^2-1)(n-1) < 100*n^2*n$ für alle $n \geq 1$, da gilt: $n^2-1 < n^2$ und $n-1 < n$. Wenn Faktoren größer werden, wird auch das Produkt größer. Insgesamt gilt also: $f(n) < 100*n^3$. Also gilt mit $k = 100$ und $n_0 = 1$: $f(n) \leq k*n^3$.

Aufwandsfunktion f(n)	**f(n) = O(...)**	**Beweis**
`proz5`: $f(n) = \frac{n(n+1)}{2}$	$f(n) = O(n^2)$	$n*(n+1)/2 < n*2*(n+1)/2$; der Wert n wird verdoppelt, wird also größer. $n*2*(n+1)/2 = n*(n+1) \leq n*(n+n)$, da gilt: $1 \leq n$. Insgesamt ergibt sich also: $f(n) = n*(n+1)/2 < 2*n^2$. Somit ist $f(n) \leq k*n^2$ mit $k = 2$ und $n_0 = 1$.
`proz6`: $f(n) = \lfloor \log_2 n \rfloor + 1$	$f(n) = O(\log_2 n)$	$\lfloor \log_2(n) \rfloor + 1 \leq \log_2(n) + 1$ aufgrund derselben Argumentation wie bei `proz3`. $\log_2(n) + 1 \leq \log_2(n) + \log_2(n)$ für $n \geq 2$, da dann gilt: $1 \leq \log_2(n)$. Somit folgt: $\log_2(n) + \log_2(n) = 2*\log_2(n)$. Insgesamt gilt also: $f(n) = \lfloor \log_2(n) \rfloor + 1 \leq 2*\log_2(n)$. Also gilt $f(n) \leq k*\log_2(n)$ mit $k = 2$ und $n_0 = 2$.

Faustregeln:

1) Ist $f(n)$ ein Polynom, so ist $f(n)$ von der Ordnung der größten Potenz, die in diesem Polynom vorkommt. **Beispiel:** $200n^4 + 3n^3 + n + 20 = O(n^4)$.
2) Ist $f(n) = k_1 * \log_2 n + k_2$ für zwei Konstanten k_1 und k_2, so gilt: $\mathbf{f(n) = O(\log_2 n)}$. **Beispiel:** $\mathbf{f(n) = 10 * \log_2 n + 32 = O(\log_2 n)}$.
3) Ist $\mathbf{f(n) = k_1 * \sqrt{n} + k_2}$ für zwei Konstanten k_1 und k_2, so gilt: $\mathbf{f(n) = O(\sqrt{n})}$. **Beispiel:** $\mathbf{f(n) = 1050 * \sqrt{n} + 113 = O(\sqrt{n})}$.

VL02, Lösung 2

1) Gilt $7n \log_2 n = O(n)$?

   Die Aussage ist falsch, denn $\lim_{n \to \infty} \frac{7n \log_2 n}{n}$ divergiert bestimmt gegen ∞.

2) Gilt $n^3 + 2^n = O(2^n)$?

   Die Aussage ist richtig, denn $\lim_{n \to \infty} \frac{n^3 + 2^n}{2^n}$ konvergiert gegen 1. Da auch $\lim_{n \to \infty} \frac{2^n}{n^3 + 2^n}$ konvergiert (reziproker Bruch), ist $O(2^n)$ sogar die genaue Ordnung.

3) Gilt $n^2 + 15n - 3 = O(n^3)$?

   Die Aussage ist richtig. $f(n)$ ist von der Ordnung des Summanden, der am schnellsten wächst. Daher gilt: $n^2 + 15n - 3 = O(n^3)$. Außerdem gilt: $O(n^2) \subseteq O(n^3)$. Alternativ stellen wir fest, dass $\lim_{n \to \infty} \frac{n^2 + 15n - 3}{n^3}$ gegen 0 konvergiert. Umgekehrt divergiert $\lim_{n \to \infty} \frac{n^3}{n^2 + 15n - 3}$ jedoch gegen ∞, so dass $n^2 + 15n - 3$ **nicht** die Ordnung $O(n^3)$ hat.

VL02, Lösung 3

```
import java.util.Scanner;
public class Zeitmessung
{
	private static long zaehler;

	private static double tuwas()
	{
		zaehler++;
		return Math.random();
	}

…

	public static void main(String[] args)
	{
		Scanner sc = new Scanner(System.in);
		System.out.println("Geben Sie die Problemgröße (n) ein: ");

		// n enthält die Problemgröße
		int n = sc.nextInt();

		StopUhr meineUhr = new StopUhr();

		meineUhr.start();
		Zeitmessung.func1(n);
		meineUhr.stop();
		System.out.println("Func1 (lineare Laufzeit): " +
			meineUhr.getDuration()/1000000.0 + " msec");
		System.out.println("Anzahl der Aufruf von tuwas(): " + zaehler);

		zaehler = 0;
		meineUhr.start();
		Zeitmessung.func2(n);
		meineUhr.stop();
		System.out.println("Func2 (quadratische Laufzeit): " +
			meineUhr.getDuration()/1000000.0 + " msec");
		System.out.println("Anzahl der Aufruf von tuwas(): " + zaehler);

		zaehler = 0;
		meineUhr.start();
		Zeitmessung.func6(n);
		meineUhr.stop();
		System.out.println("Func6 (logarithmische Laufzeit): " +
			meineUhr.getDuration()/1000000.0 + " msec");
		System.out.println("Anzahl der Aufruf von tuwas(): " + zaehler);
	}
}
```

Die Lösung der Aufgabe3 finden Sie zusätzlich in der Datei ML02-Aufgabe3.zip.