VL02, Aufgabe 1 (Übung)

Bestimmen Sie für die nachstehenden Prozeduren folgende Informationen:

- Berechnen Sie die Anzahl der Aufrufe von `tuwas()` für $n=100$
- Bestimmen Sie die Anzahl der Aufrufe von `tuwas()` als Funktion von n
- Bestimmen Sie die asymptotische Zeitkomplexität in $O(f(n))$-Notation unter der Annahme, dass die asymptotische Zeitkomplexität von `tuwas()` $O(1)$ ist

```
void proz1(int n)
{
    for (int a=0; a<n; a++)
        tuwas();

    for (int b=0; b<n; b++)
        tuwas();
}
```

```
void proz2(int n)
{
    for (int a=0; a<n; a++)
        for (int b=0; b<n; b++)
            tuwas();
}
```

```
void proz3(int n)
{
    for (int a=0; a*a<=n; a++)
        tuwas();
}
```

```
void proz4(int n)
{
    for (int a=1; a<=100; a++)
        for (int b=1; b<n*n; b++)
            for (int c=1; c<n; c++)
                tuwas();
}
```

```
void proz5(int n)
{
    for (int a=1; a<=n; a++)
        for (int b=1; b<=a; b++)
            tuwas();
}
```

```
void proz6(int n)
{
    while (n>=1)
    {
        tuwas();
        n /= 2;
    }
}
```

VL02, Aufgabe 2 (Übung)

Geben Sie bei jeder der untenstehenden Fragen an, ob die Aussage über die asymptotische Komplexität der angegebenen Funktion richtig ist oder nicht. Begründen Sie Ihre Antwort jeweils.

1) $7n \log_2 n = O(n)$
2) $n^3+2^n = O(2^n)$
3) $3((n \bmod 3)+1) = O(1)$
4) $n^2+15n-3 = O(n^3)$

VL02, Aufgabe 3 (Übung)

Zur Lösung eines Problems stehen Ihnen drei Algorithmen mit folgendem Zeitverhalten zur Verfügung:

Algorithmus	Zeitverhalten
A_1	$g_1(n) = 4n + 32$
A_2	$g_2(n) = 6n$
A_3	$g_3(n) = n^2$

Welchen Algorithmus müssen Sie für welchen Problemumfang verwenden, damit Sie den geringsten Zeitaufwand benötigen?

Vergleichen Sie die Funktionen dafür paarweise. Beantworten Sie also beispielsweise die Frage: Ab welchem n gilt $4n+32 < 6n$? Ermitteln Sie das gesuchte n durch Lösen der Ungleichung.

VL02, Aufgabe 4 (Praktikum)

Zur Lösung eines Problems stehen Ihnen jetzt sechs **andere** Algorithmen mit folgendem Zeitverhalten zur Verfügung:

Algorithmus	Zeitverhalten
A_1	$g_1(n) = 1000n$
A_2	$g_2(n) = 100n \log_2 (n+1)$
A_3	$g_3(n) = 10n^2$
A_4	$g_4(n) = n^3$
A_5	$g_5(n) = 2^n$
A_6	$g_6(n) = n!$

Implementieren Sie die zu den Algorithmen A_1 bis A_6 gehörenden Funktionen $g_i(n)$ mit $1 \leq i \leq 6$ in der Klasse `BesterAlgorithmus` (siehe ZIP-Archiv `UEB02.zip`).

Implementieren Sie zunächst die Methode `gewinnerFuerN`, die für ein gegebenes n den schnellsten Algorithmus ermittelt. Speichern Sie dazu die Laufzeiten in einem Array ab, und ermitteln Sie den Index des Minimums.

Schreiben Sie dann eine `main`-Methode, in der für jedes n zwischen 1 und 2000 ausgegeben wird, welcher der sechs Algorithmen für das betrachtete n der Beste ist. Leiten Sie daraus die Antwort auf die Frage ab, in welchen Bereichen für n welcher der sechs Algorithmen bzgl. des Zeitverhaltens der Beste ist.

Hinweis: $\log_2(n) = \frac{\log_{10} n}{\log_{10} 2}$

VL02, Aufgabe 5 (Praktikum)

Diese Aufgabe dient dazu, Ihnen ein Gefühl für die Auswirkung der Zeitkomplexität eines Algorithmus auf die Laufzeit zu vermitteln. Messen Sie die Ausführungszeiten für die in der Klasse `Zeitmessung` angegebenen Methoden `func1`, `func2` und `func6` bei verschiedenen Werten für `n` (z.B. 1000, 10000, 100000 und 1000000). `func1` besitzt eine lineare, `func2` eine quadratische und `func6` eine logarithmische Laufzeit analog zu `proz1`, `proz2` und `proz6` aus Aufgabe 1.

Hinweis: Benutzen Sie die Klasse `StopUhr`. Die Klasse finden Sie zusammen mit der Klasse `Zeitmessung` im ZIP-Archiv `UEB02.zip`.

```
import java.util.Date;

public class StopUhr
{
    private long startTime, stopTime;

    public void start()
    {
        startTime = System.nanoTime();
    }

    public void stop()
    {
        stopTime = System.nanoTime();
    }

    public long getDuration()
    {
        return stopTime - startTime;
    }
}
```