



Zentralübung zu
Einführung in die Informatik I

Dr. Christian Herzog
Technische Universität München

Wintersemester 2000/2001
18. Dezember 2000

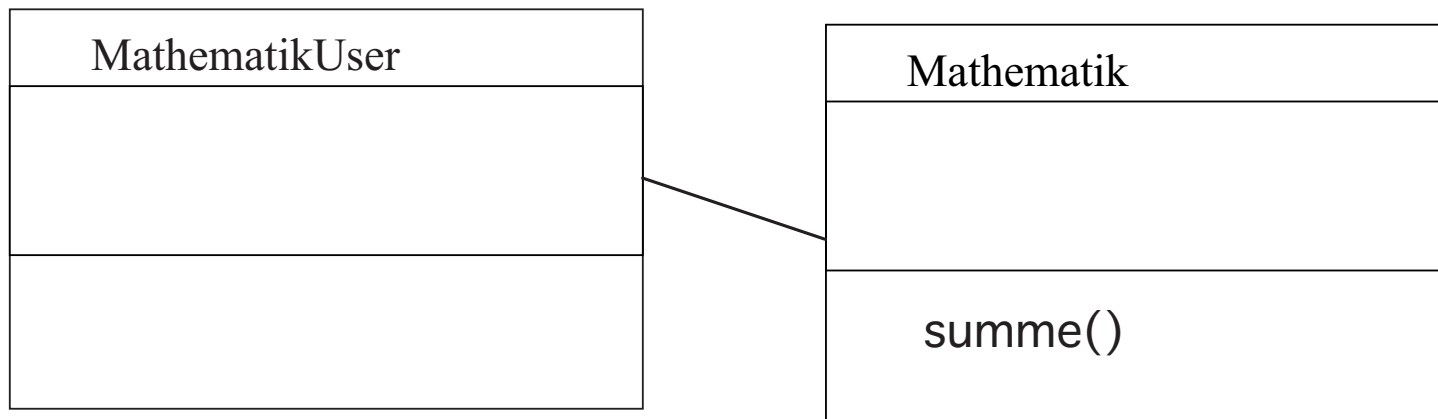
Übersicht

Programmieren mit Java:

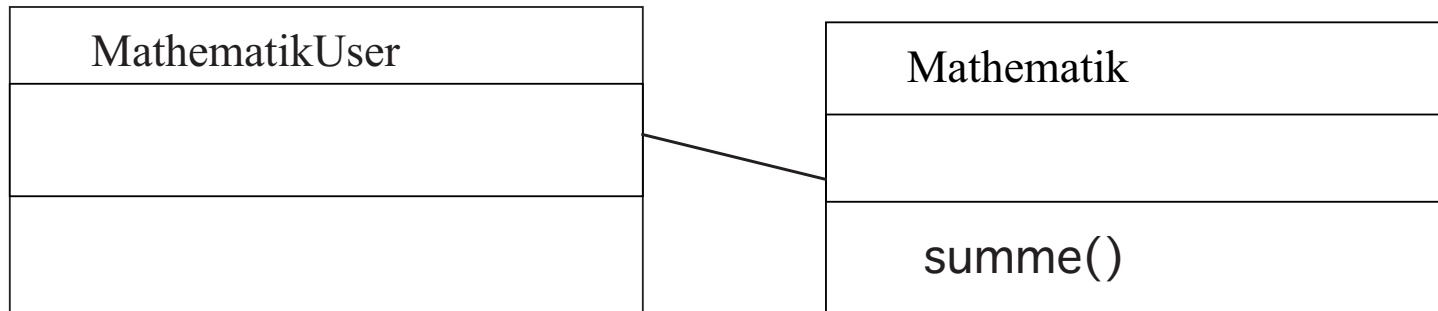
- ❖ Die Funktion `summe` eingekleidet in einen Entwurf einer Mathematik-Klasse
- ❖ eine iterative Fassung von `summe`
- ❖ Schritte zum lauffähigen Java-Programm
- ❖ Aufrufstruktur der rekursiven Funktion `fib`
- ❖ Übergang zu einer Funktion `fibNeu` mit Sprachkonzepten des imperativen Programmierparadigmas
- ❖ Ein Aufruf von `fibNeu` im Detail

Klassentwurf : Wieder ein sehr einfaches Beispiel:

- ❖ Problemstellung: Ein Mathematikstudent möchte die Summe der ersten n natürlichen Zahlen berechnen.
- ❖ Analyse: Wir definieren eine Klasse Mathematik, die eine Operation `summe()` bereitstellt.
- ❖ Systementwurf: Das System besteht aus 2 Klassen:
 - MathematikUser
 - Mathematik mit der Schnittstelle `summe()`



Implementierung der Mathematik-Klassen



```
class Mathematik {
    public Mathematik () {} // Standard-Konstruktor
    public int summe (int n) { // unser Standard-Beispiel
        return n == 0 ? 0 : summe(n-1) + n;
    }
}
```

```
class MathematikUser {
    public static void main (String[] args) {
        Mathematik mathe = new Mathematik(); // Instantiierung
        System.out.println(mathe.summe(10)); // ein Aufruf
    }
}
```

Iterative Berechnung von summe

Mathematik
summe() summeNeu() // iterativ

```
class Mathematik {  
    public Mathematik () {}           // Standard-Konstruktor  
  
    public int summe (int n) {        // unser Standard-Beispiel  
        return n == 0 ? 0 : summe(n-1) + n;  
    }  
  
    public int summeNeu (int n) {     // iterative Fassung  
        int ergebnis = 0;  
        for (int i=1; i<=n; i = i+1)  
            ergebnis = ergebnis + i;  
        return ergebnis;  
    }  
}
```

Aufrufe in der Klasse MathematikUser

```
class MathematikUser {
    public static void main (String[] args) {

        Mathematik mathe = new Mathematik(); // Instantiierung

        System.out.println(mathe.summe(10)); // ein Aufruf
        System.out.println(mathe.summe(20)); // ein Aufruf
        System.out.println(mathe.summeNeu(10)); // ein Aufruf
        System.out.println(mathe.summeNeu(20)); // ein Aufruf

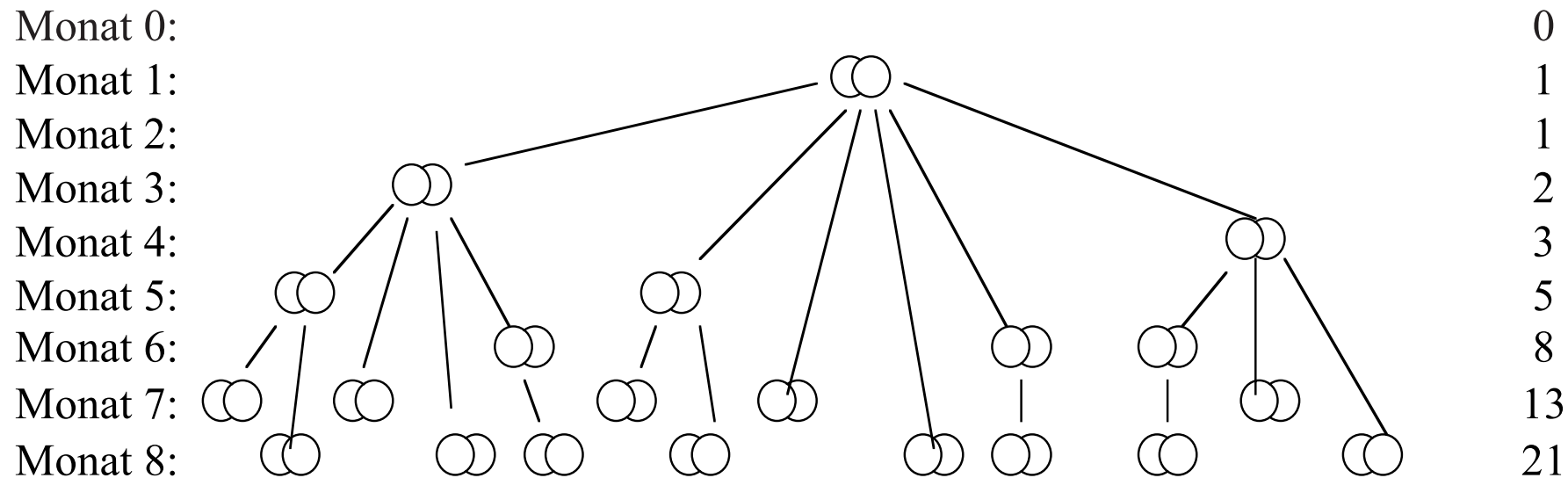
    }
}
```

Schritte zum lauffähigen Programm

- ❖ Klasse `Mathematik` wie vorgegeben in eine Datei `Mathematik.java` schreiben.
- ❖ Compilieren mittels `javac Mathematik.java` (liefert die Datei `Mathematik.class`)
- ❖ Klasse `MathematikUser` wie vorgegeben in eine Datei `MathematikUser.java` schreiben.
- ❖ Compilieren mittels `javac MathematikUser.java` (liefert die Datei `MathematikUser.class`)
- ❖ Ausführen mittels `java MathematikUser`
 - Dies veranlasst das Java-System, die in der Datei `MathematikUser.class` enthaltene Methode `main` auszuführen.
 - Dies führt zu Aufrufen der Methoden aus `Mathematik.class`.

Die Fibonacci-Zahlen

❖ Gegeben sei ein neu geborenes Kaninchenpaar. Jedes Kaninchenweibchen, das mindestens zwei Monate alt ist, bringt jeden Monat ein weiteres Paar zur Welt. Kaninchen leben ewig. Wieviele Kaninchenpaare gibt es nach n Monaten?

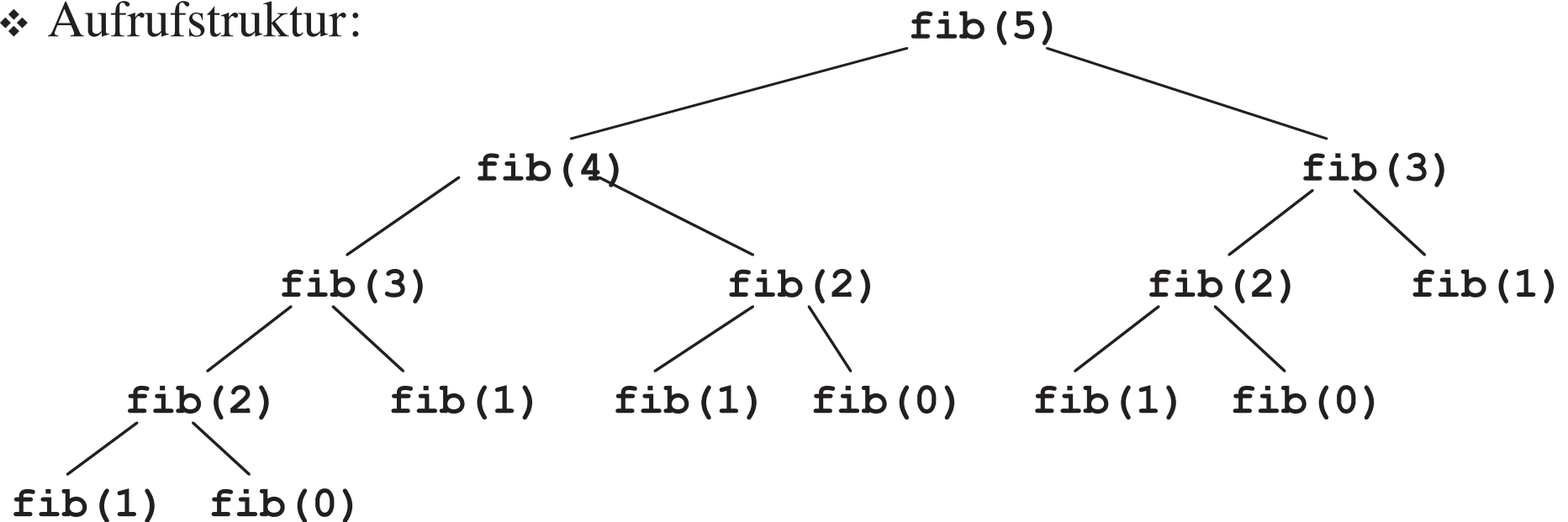


- ❖ Diese Aufgabe stellte Fibonacci im Jahre 1202.
- ❖ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ... heißen Fibonacci-Zahlen.
- ❖ Jede Fibonacci-Zahl ist die Summe ihrer beiden Vorgänger.

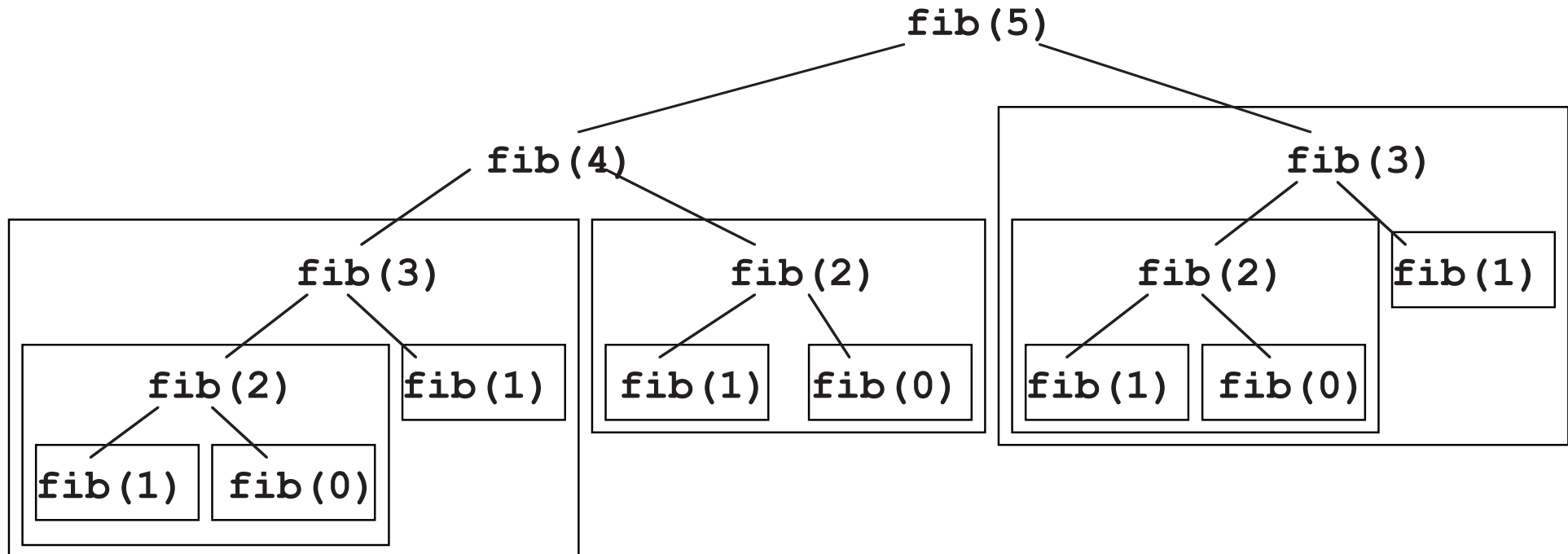
Berechnung der Fibonacci-Zahlen

```
public int fib (int n) {  
    return n==0 ? 0  
           : n==1 ? 1  
           : fib(n-1) + fib(n-2);  
}
```

❖ Aufrufstruktur:



Problem: Mehrfachauswertung



`fib(3)` wird 2 mal ausgewertet.

`fib(2)` wird 3 mal ausgewertet.

`fib(1)` wird 5 mal ausgewertet.

`fib(0)` wird 3 mal ausgewertet.

Fibonacci ohne Mehrfachauswertung

- ❖ Gesucht: Funktion fibNeu, die die n-te Fibonacci-Zahl berechnet, dabei
 - wie bisher die Summe der beiden Vorgänger bildet;
 - aber jede kleinere Fibonacci-Zahl nur einmal berechnet.
- ❖ Lösung: Zuweisung und Zählschleife (imperative Lösung)

```
public int fibNeu (int n) {
    int aktuell=0, vorg=0, vorVorg=0;
    if (n == 0) return 0;
    if (n == 1) return 1;
    vorg = 1; vorVorg = 0;
    for (int i = 2; i <= n; i = i+1) {
        aktuell = vorg + vorVorg;
        vorVorg = vorg; vorg = aktuell;
    }
    return aktuell;
}
```

Erweiterung der Klasse Mathematik

Mathematik
summe() // rekursiv
summeNeu() // iterativ
fib() // rekursiv
fibNeu() // iterativ

```
class Mathematik {
    public Mathematik () {} // Standard-Konstruktor

    public int summe (int n) { // rekursive Summe
        // Rumpf von summe wie angegeben }
    public int summeNeu (int n) { // iterative Summe
        // Rumpf von summeNeu wie angegeben }

    public int fib (int n) { // rekursive fib-Fassung
        // Rumpf von fib wie angegeben }
    public int fibNeu (int n) { // iterative fib-Fassung
        // Rumpf von fibNeu wie angegeben }
}
}
```

Erweiterung der Klasse MathematikUser

```
class MathematikUser {
    public static void main (String[] args) {

        Mathematik mathe = new Mathematik(); // Instantiierung

        System.out.println(mathe.summe(10)); // ein Aufruf
        System.out.println(mathe.summe(20)); // ein Aufruf
        System.out.println(mathe.summeNeu(10)); // ein Aufruf
        System.out.println(mathe.summeNeu(20)); // ein Aufruf

        System.out.println(mathe.fib(5)); // ein Aufruf
        System.out.println(mathe.fibNeu(5)); // ein Aufruf
    }
}
```

Diesen Aufruf von fibNeu
wollen wir
näher betrachten!

Aufruf: `mathe.fibNeu(5)`

```
public int fibNeu (int n) {  
    int aktuell=0,  
        vorg=0, vorVorg=0;  
    if (n == 0) return 0;  
    if (n == 1) return 1;  
    vorg = 1;  
    vorVorg = 0;  
    for (int i=2; i<=n; i=i+1) {  
        aktuell = vorg + vorVorg;  
        vorVorg = vorg;  
        vorg = aktuell;  
    }  
    return aktuell;  
}
```

Belegung der Variablen:

```
        n: 5  
  
    aktuell: 0  
        vorg: 0  
    vorVorg: 0
```

Aufruf: `mathe.fibNeu(5)`

```
public int fibNeu (int n) {  
    int aktuell=0,  
        vorg=0, vorVorg=0;  
    if (n == 0) return 0;  
    if (n == 1) return 1;  
    vorg = 1;  
    vorVorg = 0;  
    for (int i=2; i<=n; i=i+1) {  
        aktuell = vorg + vorVorg;  
        vorVorg = vorg;  
        vorg = aktuell;  
    }  
    return aktuell;  
}
```

Belegung der Variablen:

`n`: 5

`aktuell`: 0

`vorg`: ~~0~~ 1

`vorVorg`: ~~0~~ 0

`i`: 2

Aufruf: `mathe.fibNeu(5)`

```
public int fibNeu (int n) {  
    int aktuell=0,  
        vorg=0, vorVorg=0;  
    if (n == 0) return 0;  
    if (n == 1) return 1;  
    vorg = 1;  
    vorVorg = 0;  
    for (int i=2; i<=n; i=i+1) {  
        aktuell = vorg + vorVorg;  
        vorVorg = vorg;  
        vorg = aktuell;  
    }  
    return aktuell;  
}
```

Belegung der Variablen:

```
n: 5  
  
aktuell: 0 1  
        vorg: 0 1 1  
        vorVorg: 0 0 1  
  
i: 2 3
```

Aufruf: `mathe.fibNeu(5)`

```
public int fibNeu (int n) {  
    int aktuell=0,  
        vorg=0, vorVorg=0;  
    if (n == 0) return 0;  
    if (n == 1) return 1;  
    vorg = 1;  
    vorVorg = 0;  
    for (int i=2; i<=n; i=i+1) {  
        aktuell = vorg + vorVorg;  
        vorVorg = vorg;  
        vorg = aktuell;  
    }  
    return aktuell;  
}
```

Belegung der Variablen:

```
n: 5  
  
aktuell: 0 1 2  
vorg: 0 1 1 2  
vorVorg: 0 0 1 1  
  
i: 2 3 4
```

Aufruf: `mathe.fibNeu(5)`

```
public int fibNeu (int n) {  
    int aktuell=0,  
        vorg=0, vorVorg=0;  
    if (n == 0) return 0;  
    if (n == 1) return 1;  
    vorg = 1;  
    vorVorg = 0;  
    for (int i=2; i<=n; i=i+1) {  
        aktuell = vorg + vorVorg;  
        vorVorg = vorg;  
        vorg = aktuell;  
    }  
    return aktuell;  
}
```

Belegung der Variablen:

```
n: 5  
  
aktuell: 0 1 2 3  
vorg: 0 1 1 2 3  
vorVorg: 0 0 1 1 2  
  
i: 2 3 4 5
```

Aufruf: `mathe.fibNeu(5)`

```
public int fibNeu (int n) {  
    int aktuell=0,  
        vorg=0, vorVorg=0;  
    if (n == 0) return 0;  
    if (n == 1) return 1;  
    vorg = 1;  
    vorVorg = 0;  
    for (int i=2; i<=n; i=i+1) {  
        aktuell = vorg + vorVorg;  
        vorVorg = vorg;  
        vorg = aktuell;  
    }  
    return aktuell;  
}
```

Belegung der Variablen:

n: 5

aktuell:	0	1	2	3	5
vorg:	0	1	1	2	3 5
vorVorg:	0	0	1	1	2 3

Ergebnis: 5

**Ein gesegnetes
Weihnachtsfest
und einen guten
Rutsch
in das Neue Jahr 2001**