Lektion 4

for-Schleife

while-Schleife

Schleifen

Wir möchten ein Programm schreiben, dass den Wert eines angelegten Kapitals nach 10 Jahren ausrechnet.

```
float kapital = 100;
float zinssatz = 3.5f;

float kapitalNachJahr1 = kapital + kapital * zinssatz / 100.0f;
float kapitalNachJahr2 = kapitalNachJahr1 + kapitalNachJahr1 * zinssatz / 100.0f;
float kapitalNachJahr3 = kapitalNachJahr2 + kapitalNachJahr2 * zinssatz / 100.0f;
float kapitalNachJahr4 = kapitalNachJahr3 + kapitalNachJahr3 * zinssatz / 100.0f;
float kapitalNachJahr5 = kapitalNachJahr4 + kapitalNachJahr4 * zinssatz / 100.0f;
float kapitalNachJahr6 = kapitalNachJahr5 + kapitalNachJahr5 * zinssatz / 100.0f;
float kapitalNachJahr7 = kapitalNachJahr6 + kapitalNachJahr6 * zinssatz / 100.0f;
float kapitalNachJahr8 = kapitalNachJahr7 + kapitalNachJahr7 * zinssatz / 100.0f;
float kapitalNachJahr9 = kapitalNachJahr8 + kapitalNachJahr8 * zinssatz / 100.0f;
float kapitalNachJahr10 = kapitalNachJahr9 + kapitalNachJahr9 * zinssatz / 100.0f;

System.out.println("Kapital nach 1 Jahr: " + kapitalNachJahr1);
System.out.println("Kapital nach 2 Jahren: " + kapitalNachJahr2);
System.out.println("Kapital nach 3 Jahren: " + kapitalNachJahr3);
System.out.println("Kapital nach 4 Jahren: " + kapitalNachJahr4);
System.out.println("Kapital nach 5 Jahren: " + kapitalNachJahr5);
System.out.println("Kapital nach 6 Jahren: " + kapitalNachJahr6);
System.out.println("Kapital nach 7 Jahren: " + kapitalNachJahr7);
System.out.println("Kapital nach 8 Jahren: " + kapitalNachJahr8);
System.out.println("Kapital nach 9 Jahren: " + kapitalNachJahr9);
System.out.println("Kapital nach 10 Jahren: " + kapitalNachJahr10);
```

```
float kapital = 100;
float zinssatz = 3.5f;

float kapitalNachJahr1 = kapital + kapital * zinssatz / 100.0f;
float kapitalNachJahr2 = kapitalNachJahr1 + kapitalNachJahr1 * zinssatz / 100.0f;
float kapitalNachJahr3 = kapitalNachJahr2 + kapitalNachJahr2 * zinssatz / 100.0f;
float kapitalNachJahr4 = kapitalNachJahr3 + kapitalNachJahr3 * zinssatz / 100.0f;
float kapitalNachJahr5 = kapitalNachJahr4 + kapitalN      Jahr4 * zinssatz / 100.0f;
float kapitalNachJahr6 = kapitalNachJahr5 +               ahr5 * zinssatz / 100.0f;
float kapitalNachJahr7 = kapitalNach                      hr6 * zinssatz / 100.0f;
float kapitalNachJahr8 = kapit                            Jahr7 * zinssatz / 100.0f;
float kapitalNachJahr9                                    talNachJahr8 * zinssatz / 100.0f;
float kapitalNachJahr                                     + kapitalNachJahr9 * zinssatz / 100.0f;

System.out.println("Ka                ch 1 Jahr: " + kapitalNachJahr1);
System.out.println("Kapital nach 2 Jahren: " + kapitalNachJahr2);
System.out.println("Kapital nach 3 Jahren: " + kapitalNachJahr3);
System.out.println("Kapital nach 4 Jahren: " + kapitalNachJahr4);
System.out.println("Kapital nach 5 Jahren: " + kapitalNachJahr5);
System.out.println("Kapital nach 6 Jahren: " + kapitalNachJahr6);
System.out.println("Kapital nach 7 Jahren: " + kapitalNachJahr7);
System.out.println("Kapital nach 8 Jahren: " + kapitalNachJahr8);
System.out.println("Kapital nach 9 Jahren: " + kapitalNachJahr9);
System.out.println("Kapital nach 10 Jahren: " + kapitalNachJahr10);
```

zu umständlich?

Es wäre schön, wenn man eine Anweisung häufiger ausführen könnte, ohne sie jedes Mal wieder neu hinschreiben zu müssen!

Das ist möglich durch eine sogenannte **Schleife**.
Es gibt drei Schleifentypen:

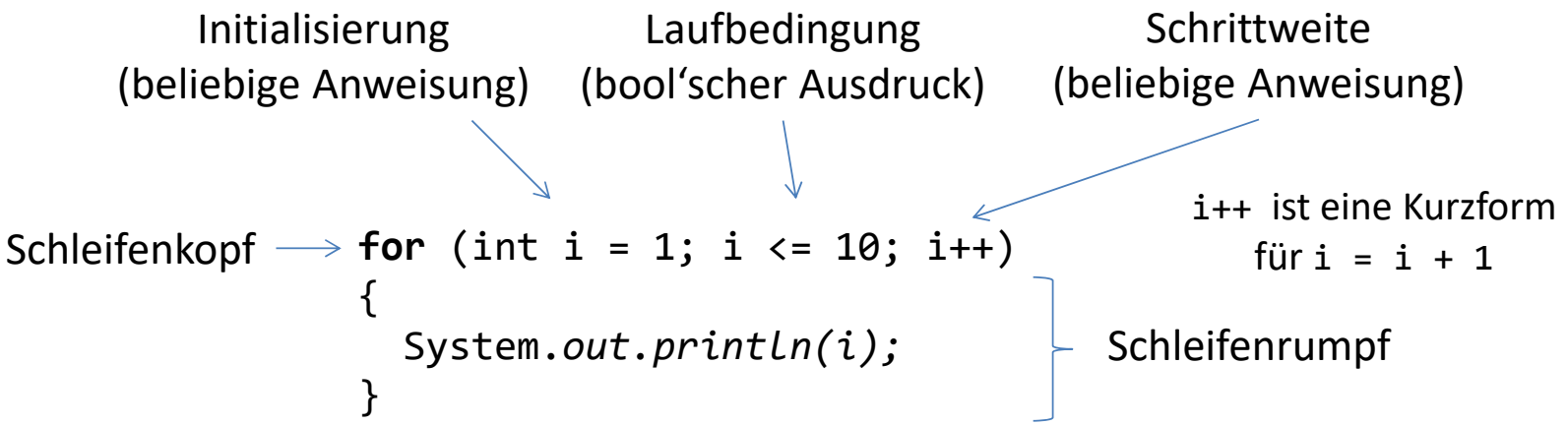
- Zählschleifen (in Java: for-Schleife)
- kopfgesteuerte Schleifen (in Java: while-Schleife)
- fußgesteuerte Schleifen (in Java: do-while-Schleife)

Eine Schleife wird verwendet, um eine oder mehrere Anweisungen mehrmals auszuführen.

Sie läuft, solange die Laufbedingung erfüllt ist.

for-Schleife

for-Schleife (I)



Eine for-Schleife ist besonders sinnvoll einsetzbar, wenn etwas abzählbar oft wiederholt werden soll.

for-Schleife (II)

```
for (int i = 1; i <= 10; i++)
{
  System.out.println(i);
}
```

bedeutet semantisch:

```
int i = 1;
```
WIEDERHOLE, SOLANGE `i <= 10` IST, FOLGENDE ANWEISUNGEN:
```
  System.out.println(i);
  i++;
```

for-Schleife (III) – Beispiel

```
public static void main(String[] args)
{
  for (int i = 1; i <= 10; i++)
  {
    System.out.println(i);
  }
}
```

Ausgabe:

```
1
2
3
4
5
6
7
8
9
10
```

```
float kapital = 100;
float zinssatz = 3.5f;

float kapitalNachJahr1 = kapital + kapital * zinssatz / 100.0f;
float kapitalNachJahr2 = kapitalNachJahr1 + kapitalNachJahr1 * zinssatz / 100.0f;
float kapitalNachJahr3 = kapitalNachJahr2 + kapitalNachJahr2 * zinssatz / 100.0f;
float kapitalNachJahr4 = kapitalNachJahr3 + kapitalNachJahr3 * zinssatz / 100.0f;
float kapitalNachJahr5 = kapitalNachJahr4 + kapitalN      Jahr4 * zinssatz / 100.0f;
float kapitalNachJahr6 = kapitalNachJahr5 +               ahr5 * zinssatz / 100.0f;
float kapitalNachJahr7 = kapitalNach                      hr6 * zinssatz / 100.0f;
float kapitalNachJahr8 = kapit                            Jahr7 * zinssatz / 100.0f;
float kapitalNachJahr9                                    talNachJahr8 * zinssatz / 100.0f;
float kapitalNachJahr                                     + kapitalNachJahr9 * zinssatz / 100.0f;

System.out.println("Ka                ch 1 Jahr: " + kapitalNachJahr1);
System.out.println("Kapital nach 2 Jahren: " + kapitalNachJahr2);
System.out.println("Kapital nach 3 Jahren: " + kapitalNachJahr3);
System.out.println("Kapital nach 4 Jahren: " + kapitalNachJahr4);
System.out.println("Kapital nach 5 Jahren: " + kapitalNachJahr5);
System.out.println("Kapital nach 6 Jahren: " + kapitalNachJahr6);
System.out.println("Kapital nach 7 Jahren: " + kapitalNachJahr7);
System.out.println("Kapital nach 8 Jahren: " + kapitalNachJahr8);
System.out.println("Kapital nach 9 Jahren: " + kapitalNachJahr9);
System.out.println("Kapital nach 10 Jahren: " + kapitalNachJahr10);
```

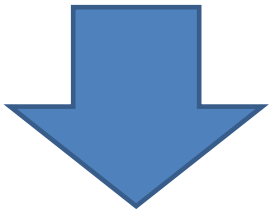
Zurück zum Kapitalbeispiel!

verzinstes Kapital nach 10 Jahren

Was wollen wir tun:

führe 10 mal aus:

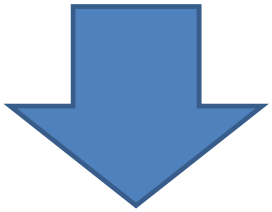
neues Kapital wird errechnet aus altem Kapital + Zinsen



```
for (int i = 1; i <= 10; i++)
{
   neues Kapital wird errechnet aus altem Kapital + Zinsen
}
```

verzinstes Kapital nach 10 Jahren

```
for (int i = 1; i <= 10; i++)
{
   neues Kapital wird errechnet aus altem Kapital + Zinsen
}
```



```
for (int i = 1; i <= 10; i++)
{
  neuesKapital = altesKapital + altesKapital * zinssatz/100.0f;
}
```

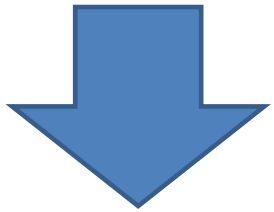
Zinsen

Im nächsten Schritt muss man mit neuesKapital weiterrechnen

verzinstes Kapital nach 10 Jahren

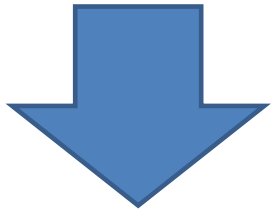
```
for (int i = 1; i <= 10; i++)
{
  neuesKapital = altesKapital + altesKapital * zinssatz/100.0f;
}
```

neuesKapital muss für nächsten Schritt zu altesKapital werden



```
for (int i = 1; i <= 10; i++)
{
  neuesKapital = altesKapital + altesKapital * zinssatz/100.0f;
  altesKapital = neuesKapital;
}
```

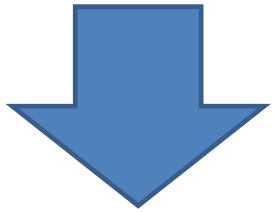
Ausgabe muss noch ergänzt werden



verzinstes Kapital nach 10 Jahren

```
for (int i = 1; i <= 10; i++)
{
  neuesKapital = altesKapital + altesKapital * zinssatz/100.0f;
  altesKapital = neuesKapital;
  System.out.println(altesKapital);
}
```

folgende Vereinfachung ist möglich

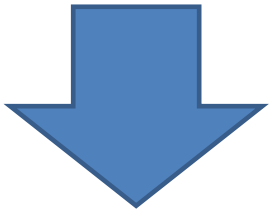


```
for (int i = 1; i <= 10; i++)
{
  altesKapital = altesKapital + altesKapital * zinssatz/100.0f;
  System.out.println(altesKapital);
}
```

verzinstes Kapital nach 10 Jahren

```
for (int i = 1; i <= 10; i++)
{
  altesKapital = altesKapital + altesKapital * zinssatz/100.0f;
  System.out.println(altesKapital);
}
```

sinnvolle Variablenbennung

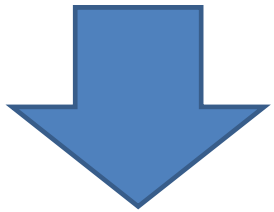


```
float kapital = 100;
float zinssatz = 3.5f;
float neuesKapital = kapital;
for (int i = 1; i <= 10; i++)
{
  neuesKapital = neuesKapital + neuesKapital * zinssatz/100.0f;
  System.out.println(neuesKapital);
}
```

verzinstes Kapital nach 10 Jahren

```
float kapital = 100;
float zinssatz = 3.5f;
float neuesKapital = kapital;
for (int i = 1; i <= 10; i++)
{
  neuesKapital = neuesKapital + neuesKapital * zinssatz/100.0f;
  System.out.println(neuesKapital);
}
```

verbesserte Ausgabe



```
float kapital = 100;
float zinssatz = 3.5f;
float neuesKapital = kapital;
for (int i = 1; i <= 10; i++)
{
  neuesKapital = neuesKapital + neuesKapital * zinssatz/100.0f;
  System.out.println("Kapital nach Jahr " + i + ": " + neuesKapital);
}
```

Reihen

$\sum$ Summenzeichen

$\prod$ Produktzeichen

$$\sum_{k=1}^{n} k = 1+2+3+\ldots+n-2+n-1+n \quad \text{(Reihe, Summe)}$$

$$\prod_{k=1}^{n} k = 1\cdot 2\cdot \ldots \cdot (n-1)\cdot n \quad \text{(Produktreihe, Produkt)}$$

$$\sum_{k=1}^{10} k = 1+2+3+4+5+6+7+8+9+10$$

$$\sum_{k=1}^{n} \frac{1}{k} = \frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{n-1}+\frac{1}{n}$$

Unendliche Reihen

$$\sum_{k=1}^{\infty} k = 1 + 2 + 3 + \ldots = \infty$$

$$\prod_{k=1}^{\infty} k = 1 \cdot 2 \cdot 3 \cdot \ldots = \infty$$

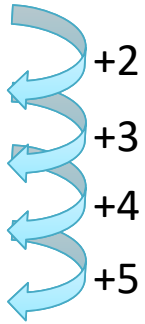
$$\sum_{k=0}^{\infty} \frac{1}{2^k} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots = 2$$

$$\cos(x) := \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$$

Beispiel:
Wie bilde ich die Summe der ersten n Zahlen?

Summe der ersten n Zahlen

- mathematisch:
  - Summe der ersten Zahl: 1
  - Summe der ersten zwei Zahlen: 3
  - Summe der ersten drei Zahlen: 6
  - Summe der ersten vier Zahlen: 10
  - Summe der ersten fünf Zahlen: 15
  - …



- ähnlich wie beim Kapital:

```
zwischenergebnis = 1;
zwischenergebnis = zwischenergebnis + 2;
zwischenergebnis = zwischenergebnis + 3;
…
```

Summe der ersten n Zahlen

- ähnlich wie beim Kapital:

```
zwischenergebnis = 1;
zwischenergebnis = zwischenergebnis + 2;
zwischenergebnis = zwischenergebnis + 3;
…
```

- Schleife bis 10 kennen wir:

```
for (int i = 1; i <= 10; i++)
{
   …
}
```

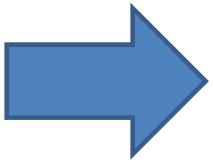
Summe der ersten n Zahlen

- ähnlich wie beim Kapital:

```
zwischenergebnis = 1;
zwischenergebnis = zwischenergebnis + 2;
zwischenergebnis = zwischenergebnis + 3;
…
```

- Schleife bis 10 kennen wir:

```
for (int i = 1; i <= 10; i++)
{
   …
}
```



```
for (int i = 1; i <= n; i++)
{
   …
}
```

Wie lässt man die Schleife
von 1 bis n laufen?

Summe der ersten n Zahlen

- Wir schauen uns erstmal das erste Element unabhängig von der Schleife an und belegen es vor:

```
int zwischenergebnis = 1;
```

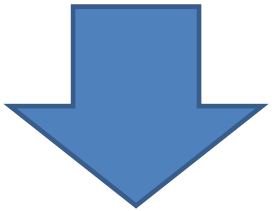
- Die Schleife lassen wir beim zweiten Element loslaufen

```
int zwischenergebnis = 1;
for (int i = 2; i <= n; i++)
{
  zwischenergebnis = zwischenergebnis + 2;
}
```

im nächsten Schritt
müssen wir 2 addieren,
danach 3, danach 4, etc

Summe der ersten n Zahlen

```
int zwischenergebnis = 1;
for (int i = 2; i <= n; i++)
{
  zwischenergebnis = zwischenergebnis + 2;
}
```

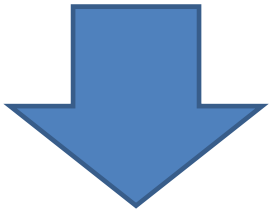


im nächsten Schritt beinhaltet i den Wert 3 und es muss 3 zum Zwischenergebnis addiert werden.

```
int zwischenergebnis = 1;
for (int i = 2; i <= n; i++)
{
  zwischenergebnis = zwischenergebnis + i;
}
```

Summe der ersten n Zahlen

```
int zwischenergebnis = 1;
for (int i = 2; i <= n; i++)
{
  zwischenergebnis = zwischenergebnis + i;
}
```



Es ist auch möglich, die Schleife ohne Vorbelegung des ersten Elements zu schreiben.

```
int zwischenergebnis = 0; //neutrales Element der Addition
for (int i = 1; i <= n; i++)
{
  zwischenergebnis = zwischenergebnis + i;
}
```

Eine Vorbelegung des ersten Elements kann bei komplizierteren Berechnungen sinnvoll sein.
Probieren Sie es für sich aus!

Summe der ersten n Zahlen

- Da wir eine Summe berechnen, könnten wir die Variable auch entsprechend benennen:

```
int summe = 0;
for (int i = 1; i <= n; i++)
{
  summe = summe + i;
}
```

- oder nach dem Wert einer Reihe:

```
int reihenwert = 0;
for (int i = 1; i <= n; i++)
{
 reihenwert = reihenwert + i;
}
```

Summe der ersten n Zahlen

- Die Summe der ersten n Zahlen lässt sich auch explizit berechnen:

$$\sum_{k=1}^{n} k = 1 + 2 + 3 + \cdots + n - 1 + n = \frac{n \cdot (n+1)}{2}$$

Blöcke nach Schleifen

- Übrigens gilt für den Block von Schleifen dasgleiche wie für den Block nach einem if.

```
int summe = 0;
for (int i = 1; i <= n; i++)
{
  summe = summe + i;
}
```

```
int summe = 0;
for (int i = 1; i <= n; i++)
  summe = summe + i;
```

```
int summe = 0;
for (int i = 1; i <= n; i++) summe = summe + i;
```

```
int summe = 0;
for (int i = 1; i <= n; i++);
  summe = summe + i;
```

Kurzschreibweisen

Kurzform	Langform
`i++ oder ++i`	`i=i+1`
`i-- oder --i`	`i=i-1`
`i+=n`	`i=i+n`
`i-=n`	`i=i-n`
`i*=n`	`i=i*n`
`i/=n`	`i=i/n`
`i%=n`	`i=i%n`
`i^=n`	`i=i^n`
`i&=n`	`i=i&n`
`i|=n`	`i=i|n`
`i<<=n`	`i=i<<n`
`i>>=n`	`i=i>>n`
`i>>>=n`	`i=i>>>n`

arithmetische Operatoren

bitweise Operatoren (später mehr dazu)

Auswertung von Ausdrücken

Operator(en)	Priorität
[ ] . (*<parameters>*) expr++ expr--	1 (höchste)
++expr --expr +expr -expr ~ ! (*<type>*) new	2
* / %	3
+ -	4
<< >> >>>	5
< <= > >= instanceof	6
== !=	7
&	8
^	9
|	10
&&	11
||	12
? :	13
= += -= *= /= %= <<= >>= >>>= &= ^= |=	14 (niedrigste)

Sichtbarkeit von Variablen in Blöcken

```
public static void main(String[] args)
{
  int summe = 0;
  for (int i = 1; i <= n; i++)
  {
    summe = summe + i;
  }
}
```

i ist sichtbar

Sichtbarkeit von Variablen in Blöcken

```
public static void main(String[] args)
{
  int summe = 0;
  int i;
  for (i = 1; i <= n; i++)
  {
    summe = summe + i;
  }
  System.out.println(i);
}
```

i ist sichtbar

for-Schleife ohne Initialisierung

```
public static void main(String[] args)
{
  int summe = 0;
  int i = 1;
  for ( ; i <= n; i++)
  {
    summe = summe + i;
  }
}
```

while-Schleife

Wie berechne ich die Quersumme einer Zahl?

Zahl: 28467
Quersumme: 2+8+4+6+7=27

Wie greife ich in Java auf die einzelnen Ziffern zu?

Durch modulo 10 erhalte ich die letzte Ziffer.

```
28467 % 10 = 7
```

Wie entferne ich die hinterste Ziffer?

Ganzzahlige Division durch 10

```
28467 / 10 = 2846
```

```
28467 % 10 = 7
28467 / 10 = 2846

2846 % 10 = 6
2846 / 10 = 284

284 % 10 = 4
284 / 10 = 28
```

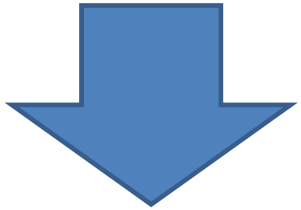
```
28 % 10 = 8
28 / 10 = 2

2 % 10 = 2
2 / 10 = 0

0 % 10 = 0
0 / 10 = 0
```

Welche Schritte muss ich also tun?

- modulo 10 berechnen
- (hinterste) Ziffer zur Quersumme addieren
- Ganzzahlige Division durch 10
- Schritte wiederholen bis alle Ziffern weg sind

Welche Schritte muss ich also tun?

- modulo 10 berechnen
- (hinterste) Ziffer zur Quersumme addieren
- Ganzzahlige Division durch 10
- Schritte wiederholen bis alle Ziffern weg sind



```
int n = 28467;
int quersumme = 0;

WIEDERHOLE, SOLANGE WEITERE ZIFFERN VORHANDEN
  int rest = n % 10;
  quersumme = quersumme + rest;
  n = n / 10;
```

while-Schleife

- Eine `while`-Schleife wiederholt die Anweisungen im Schleifenrumpf, solange die Laufbedingung eintritt.
- Vor Beginn der Anweisungen in der Schleife wird zunächst die Laufbedingung überprüft (**kopfgesteuert**).

while-Schleife

- Eine `while`-Schleife wiederholt die Anweisungen im Schleifenrumpf, solange die Laufbedingung eintritt.
- Vor Beginn der Anweisungen in der Schleife wird zunächst die Laufbedingung überprüft (**kopfgesteuert**).

```
int n = 28467;
int quersumme = 0;

while (   n > 0   )
{
  int rest = n%10;
  quersumme = quersumme + rest;
  n=n/10;
}
System.out.println(quersumme);
```

Laufbedingung

Wiederhole die Anweisungen im Rumpf solange n größer als 0 ist.

Schleifenrumpf

Äquivalenz von while- und for-Schleife

```
for (int i = 1; i <= 10; i++)
{
  System.out.println(i);
}
```

```
int i = 1;
while (i <= 10)
{
  System.out.println(i);
  i++;
}
```

Äquivalenz von while- und for-Schleife

```
int n = 28467;
int quersumme = 0;
while (n>0) {
  int rest = n%10;
  quersumme = quersumme + rest;
  n=n/10;
}
System.out.println(quersumme);
```

```
int quersumme = 0;
for(int n = 28467; n > 0; n=n/10) {
  int rest = n % 10;
  quersumme = quersumme + rest;
}
System.out.println(quersumme);
```

Reihen

$\sum$ Summenzeichen

$\prod$ Produktzeichen

$$\sum_{k=1}^{n} k = 1+2+3+\ldots+n-2+n-1+n \quad \text{(Reihe, Summe)}$$

$$\prod_{k=1}^{n} k = 1\cdot 2\cdot \ldots \cdot (n-1)\cdot n \quad \text{(Produktreihe, Produkt)}$$

$$\sum_{k=1}^{10} k = 1+2+3+4+5+6+7+8+9+10$$

$$\sum_{k=1}^{n} \frac{1}{k} = \frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{n-1}+\frac{1}{n}$$

Unendliche Reihen

$$\sum_{k=1}^{\infty} k = 1 + 2 + 3 + \ldots = \infty$$

$$\prod_{k=1}^{\infty} k = 1 \cdot 2 \cdot 3 \cdot \ldots = \infty$$

$$\sum_{k=0}^{\infty} \frac{1}{2^k} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \ldots = 2$$

$$\cos(x) := \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$$