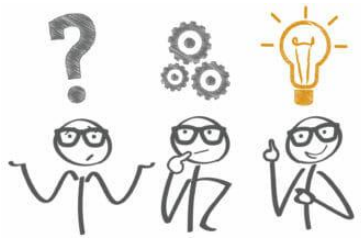
Rekursion selber entwickeln



Einleitung:

Sie haben nun schon einige Rekursionen angewandt.

Zumeist konnten Sie sich an einem Beispiel orientieren und mussten dieses noch an Java und an die gegebene Aufgabe anpassen.

Nun sollen Sie selbst eine Rekursion entwickeln.
Als Basis dient der Grundsatz, dass man eine Schleife in einen rekursiven Aufruf umformen kann. (Und umgekehrt).

Gegeben sind also Aufgabe mit Schleifen.
Gesucht ist Ihre Lösung mit Hilfe einer Rekursion.

Und Sie sollen nicht einfach die Lösung im Internet Googlen.

Gar nicht so einfach… 😉

Beispiel mit der Berechnung der Fakultät:

Das Beispiel mit der Fakultät dient dazu, systematisch von der Schleife bis zur Rekursion zu kommen.

Als n! Fakultät bezeichnet man die Berechnung der Multiplikation bis n.

Zum Beispiel 5! =5*4*3*2*1 = 120

Die Lösung mit Hilfe eine Schleife:

```
public int byLoop(int value){
   int ergebnis = 1;
   for(int i=1; i<=5; i++){
      ergebnis = ergebnis * i;
   }
   return ergebnis;
}
```

Was macht das genau?

```
System.out.println( byLoop(5));
```

Das sind die Rechenschritte im For-Loop

i=1 ➔ ergebnis = 1 * 1;

i=2 ➔ ergebnis = 1 * 2; eigentlich ➔ ergebnis = (1*1) * 2;

i=3 ➔ ergebnis = 2 * 3; eigentlich ➔ ergebnis = ((1*1)*2) * 3;

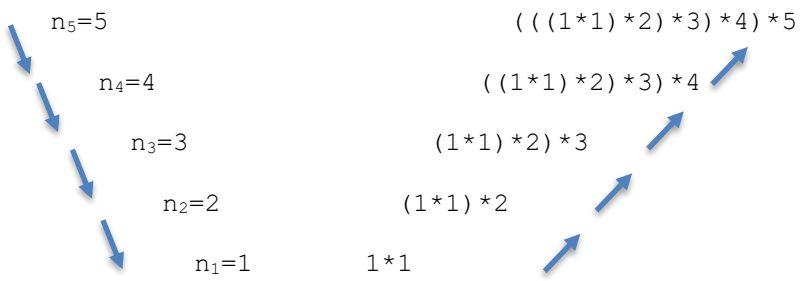
i=4 ➔ ergebnis = 6 * 4; eigentlich ➔ ergebnis = (((1*1)*2)*3) * 4;

i=5 ➔ ergebnis = 24 * 5; eigentlich ➔ ergebnis = ((((1*1)*2)*3)*4) * 5;

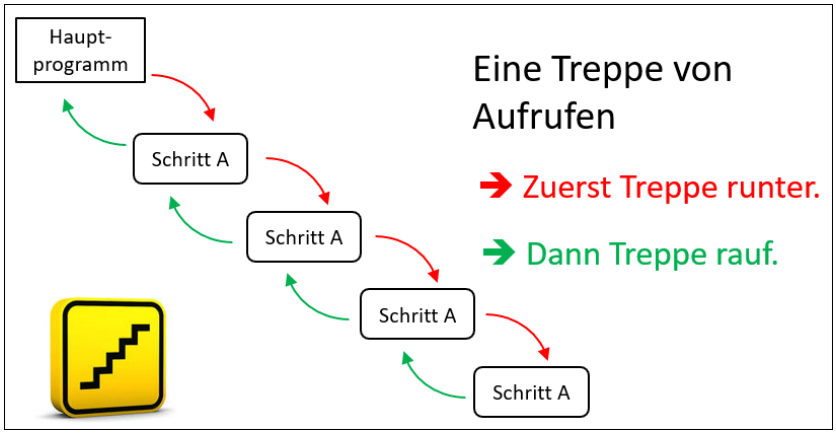
Betrachten wir die letzte Zeile mit der Verschachtelung der Multiplikationen:

i=5 ➔ ergebnis = 24 * 5; eigentlich ➔ ergebnis = ((((1*1)*2)*3)*4) * 5;

Etwas anders dargestellt:



Treppe runter und die Treppe wieder rauf… genau das Merkmal einer Rekursion



Während auf der linken Seite **n** auf dem Weg **nach unten** immer um 1 abnimmt.
Wird auf der rechten Seite auf dem Weg **nach oben** immer das *ergebnis* mit n_n multipliziert

Immer wenn ich die Rekursive Funktion aufrufe, gehe ich eine Treppenstufe runter.

⇨ Und dabei muss ich immer als Parameter n_n*-1* mitgeben. **func(n_n-1)**

Immer wenn ich aus der Rekursiven Funktion zurückkomme, gehe ich eine Treppenstufe rauf.

⇨ Und dabei muss ich immer mit den Return-Wert mit n multiplizieren: **func(n_n-1) n_n**

Fehlt nur noch das Abbruchkriterium: Wenn n_n***==1***, dann muss ich aufhören.

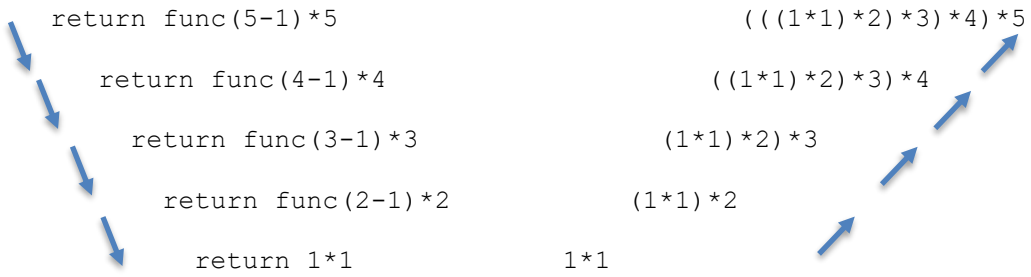
Das ergibt diese Rekursion:

```
public int func(int n){
   if (n == 1){
      return 1*1;
   }
   return (func(n-1) * n);
}
```

Der Check:

```
public int func(int n){
   if (n == 1){
      return 1*1;
   }
   return (func(n-1) * n);
}

System.out.println.func(5);
```



Zusammenfassend

Ausgangslage:

5! =5*4*3*2*1

Uns so sind wir auf die Rekursion gekommen:

1. Rechnung so darstellen, dass es aussieht wie eine raufgehende Treppe.
2. Festhalten, was man beim Runtersteigen der Treppe mit n machen muss.
3. Bestimmen was das Abbruchskriterium ist.

Aufgabe "einfache Schleife in Rekursion umwandeln":

Und nun sind Sie an der Reihe.

Versuchen Sie aus einer Schleife eine Rekursion abzuleiten.
Fangen Sie mit einfachen Beispielen an und mit einem nicht zu grossem n.

- Implementieren Sie zuerst die Schleife.
- Notieren Sie dann aufgrund der Schleife
  "raufgehende Treppe" "runtersteigende Treppe" "Abbruchkriterium"
- Leiten Sie daraus die rekursive Methode ab.
- Implementieren Sie Ihre rekursive Lösung.

Ideen:

- Addition von 0 bis n (Dreieckszahl)
- Alle ungeraden Zahlen bis n ausgeben.
- 4er Reihe (1x4 bis 10x4).
  Und dann anpassen, dass es für n statt 4 geht.
- Index-Position von x in einem Array finden
- Zwei Wörter vergleichen, ob sie gleich sind
- Kontrolle ob eine Zahl eine Primzahl ist.

Aufgabe "doppelte Schleife in Rekursion umwandeln":

Und wie sieht es mit einer doppelten Schleife aus?

Ein einfaches Beispiel ist das Ausgeben der Reihen 1 bis 5

```
Ausgabe des 1x1 von n=1 bis n=5
1:  1  2  3  4  5  6  7  8  9 10
2:  2  4  6  8 10 12 14 16 18 20
3:  3  6  9 12 15 18 21 24 27 30
4:  4  8 12 16 20 24 28 32 36 40
5:  5 10 15 20 25 30 35 40 45 50
```

Hier der Code mit zwei Loops:

```
public void einMalEinsLoop(int nMin, int nMax) {
      for(int i=nMin; i<=nMax; i++) {
            System.out.print(i + ": ");
            for(int j=1; j<=10; j++) {
                  System.out.printf("%2d ", i*j);
            }
            System.out.println();
      }
}
```

Hier der Aufruf wobei min und max eine gewisse Flexibilität ermöglichen.

```
public static void main(String[] args) {
      int min = 1;
      int max = 5;
      System.out.println("Ausgabe des 1x1 von n=" + min + " bis n=" + max);
      Calculator myCalcultor = new Calculator();
      myCalcultor.einMalEinsLoop(min, max);
}
```

Hmm und jetzt das ganze als Rekursion.. Doch wie macht man das?

Innere Schleife rekursiv

Zunächst ersetzen wir mal die inneren Schleife durch eine rekursive Methode.
Denn eine Schleife… das haben wir schon gemacht.

for(int j=1; j<=10; j++) soll rekursiv werden

Vorschlag für die Signatur der rekursiven Methode:

```
public void nMalEinsBisZehnRekursiv(int i, int j) {
```

Aufruf der Methode:

```
public void einMalEinsRekursivInnerLoop(int nMin, int nMax) {
    for(int i=nMin; i<=nMax; i++) {
          System.out.print(i + ": ");
          nMalEinsBisZehnRekursiv(i, 1);
          System.out.println();
    }
}
```

Aufgabe:

Versuchen Sie die rekursive Methode zu implementieren.
(Die Lösung ist auf der nächsten Seite..)

Musterlösung

```
//inner rekursiv
public void einMalEinsRekursivInnerLoop(int nMin, int nMax) {
      for(int i=nMin; i<=nMax; i++) {
            System.out.print(i + ": ");
            nMalEinsBisZehnRekursiv(i, 1);
            System.out.println();
      }
}

public void nMalEinsBisZehnRekursiv(int i, int j) {
      //Abbruchkriterium
      if(j==(10+1)) {
            return;
      }
      //Ausgabe
      System.out.printf("%2d ", i*j);

      //j erhöhen, rekusiv aufrufen
      nMalEinsBisZehnRekursiv(i, j+1);
}
```

Äussere Schleife rekursiv

Und wie wäre es, wenn man die äussere Schleife rekursiv macht?

Vorschlag für die Signatur der rekursiven Methode:

```
public void einMalEinsRekursivOuterLoop (int i, int nMax) {
```

Aufruf der Methode:

```
public void einMalEinsLoop1MalRekursivOuterLoop(int nMin, int nMax) {
      nMalEinsBisZehnRekursivOuterLoop(nMin, nMax);
}
```

Aufgabe:

Versuchen Sie die rekursive Methode zu implementieren.
(Die Lösung ist auf der nächsten Seite..)

Musterlösung

```
public void einMalEinsRekursivOuterLoop(int nMin, int nMax) {
      nMalEinsBisZehnRekursivOuterLoop(nMin, nMax);
}

public void nMalEinsBisZehnRekursivOuterLoop(int i, int nMax) {
      if(i== (nMax+1)) {
            return;
      }
      System.out.print(i + ": ");
      for (int j=1; j<=10; j++) {
            System.out.printf("%2d ", i*j);
      }
      System.out.println();
      nMalEinsBisZehnRekursivOuterLoop(i+1, nMax);
}
```

Und nun beide Schleifen in einer Rekursion

Und wie wäre es, wenn man beide Schleifen in nur einer Methode rekursiv macht?

Vorschlag für die Signatur der rekursiven Methode:

```
public void nMalEinsBisZehnDoppeltRekursiv(int i, int j, int nMax) {
```

Aufruf der Methode:

```
public void einMalEinsRekursiv(int nMin, int nMax) {
      int i=nMin;
      int j=1;
      nMalEinsBisZehnDoppeltRekursiv(i, j, nMax);
}
```

Aufgabe:

Versuchen Sie die rekursive Methode zu implementieren.
(Die Lösung ist auf der nächsten Seite..)

Musterlösung

```
//diese Methode kapselt den ersten Aufruf der Methode
public void einMalEinsRekursiv(int nMin, int nMax) {
      int i=nMin;
      int j=1;
      nMalEinsBisZehnDoppeltRekursiv(i, j, nMax);
}

public void nMalEinsBisZehnDoppeltRekursiv(int i, int j, int nMax) {
      if(i== (nMax+1)) {
            return;
      }
      if(j==(10+1)) {
            /* Ich bin bei der Spalte 11 angekommen
             * nun muss ich die nächste Zeile machen
             * also rekursiv mit i+1 aufrufen
             */
            System.out.println();
            nMalEinsBisZehnDoppeltRekursiv(i+1, 1, nMax);
            return;
      }
      //bei der ersten Spalte voraus 'i :' ausgeben
      if(j==1) System.out.print(i + ": ");

      //Spalten behandeln, i*j ausgeben und rekursvi j+1 aufrufen
      System.out.printf("%2d ", i*j);
      nMalEinsBisZehnDoppeltRekursiv(i, j+1, nMax);
}
```

Ja, das ist gar nicht so einfach!!

Aufgabe "zweifache Schleife in Rekursion umwandeln":

Probieren Sie weitere doppelte Schleifen aus.

- Implementieren Sie zuerst die Lösung mit zwei Schleifen.
- Implementieren Sie Ihre rekursive Lösung.

Ideen:

- Zeichen von Mustern auf den Bildschirm

```
*
**
***
```

oder

```
   *
  **
 ***
****
```

- Primzahlen von 1 bis n berechnen. (n nicht zu gross wählen)
- Häufigkeit von Buchstaben in einem Wort oder kurzen Satz bestimmen.