Rekursion

Sukzessive Reduktion des Problems, bis das Problem klein genug ist, um es zu lösen.

```
if (Problem klein genug) {
      löse Problem ohne weiteren rekursiven Aufruf;
} else {
      führe rekursiven Zweig mit kleinerem Problem aus;
}
```

Beispiel Fakultät:

```
public long fact(long n) {
      if (n == 1) {
            return 1;
      } else {
            return n*fact(n-1);
      }
}
```

Direkte Rekursion:

```
void f(...) {
      ...
      f(...);
      ...
}
```

Indirekte Rekursion:

```
void f(...) {
      ...
      g(...);
      ...
}
```

```
void g(...) {
      ...
      f(...);
      ...
}
```

Bsp 1: Rekursive Ausgabe

Was geben die folgenden Methoden `print1`, `print2` und `print3` aus und warum?

```
public void print1(int n) {
      System.out.println(n);
      if (n > 0) {
            print1(n-1);
      }
}

public void print2(int n) {
      if (n > 0) {
            print2(n-1);
      }
      System.out.println(n);
}

public void print3(int n) {
      System.out.println(n);
      if (n > 0) {
            print3(n-1);
            System.out.println(n);
      }
}
```

Bsp 2: Rekursive Suche

Ergänzen Sie Ihre einfach verkettete Liste aus Blatt 12, Bsp 01 um eine Methode `boolean contains(Object o)`, die `true` zurückgibt, wenn o in der Liste enthalten ist. Verwenden Sie einen rekursiven Algorithmus zur Lösung der Problems.

Bsp 3: Verzeichnisstruktur

Schreiben Sie eine rekursive Methode, um die Verzeichnisse mit deren Dateien und Unterverzeichnisse hierarchisch - d.h. mit Einrückungen - auszugeben. Es soll der gesamte Verzeichnisbaum mit Dateien unterhalb eines Startverzeichnisses ausgegeben werden.
Verwenden Sie die Methoden `listFiles()` und `getName()` der Klasse `java.io.File`.

Beispielausgabe:

```
temp
  Datei1.txt
  Datei2.exe
  Ordner1
    Datei3.txt
    Datei4.exe
  Ordner2
    Datei5.txt
    Datei6.exe
```

Bsp 4: Labyrinth

Schreiben Sie ein Programm, dass den Weg aus einem Labyrinth findet und anzeigt. Der gefundene Weg muss nicht der kürzeste Weg sein! Diagonale Schritte sind nicht erlaubt.

Lesen Sie dazu ein Labyrinth bestehend aus ’X’ für eine Wand, ein Leerzeichen ’ ’ für ein betretbares Feld und ein ’S’ für die Startposition aus einer Datei. Die erste Zeile der Datei soll zwei Integerwerte für die Dimension des Labyrinths beinhalten.
Markieren Sie ein betretenes Feld mit einem ’.’; ein Feld, das Sie betreten haben, sich jedoch nicht als zielführendes herausgestellt hat, hingegen mit einem ’-’.
Geben Sie aus, ob es einen Weg aus dem Labyrinth gibt. Geben Sie zusätzlich auch noch das Labyrinth aus.

Beispieldatei:

```
10 10
XXXXXXXXXX
XSX X X
X X X X XX
X X   X  X
X   X  X X
X X    X X
X   X    X
X   XXXX X
X   X    X
XXXXXXXXXX
```

Eine mögliche Lösung zu obigem Labyrinth:

```
Weg aus dem Labyrinth gefunden!

XXXXXXXXXX
XSX X X...
X.X X X.XX
X.X   X..X
X.  X  X.X
X.X... X.X
X. .X....X
X. .XXXX-X
X...X----X
XXXXXXXXXX
```

Lösung zu einem Labyrinth ohne Ausweg:

```
Es gibt keinen Weg aus dem Labyrinth!

XXXXXXXXXX
XSX-X-X
X-X-X-XXXX
X-X---X--X
X---X--X-X
X-X----X-X
X---X----X
X---XXXX-X
X---X----X
XXXXXXXXXX
```