Rekursion (in Java)

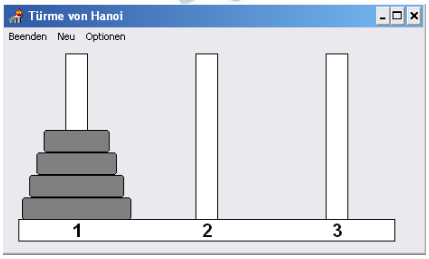
Einführung in die neue Programmiertechnik

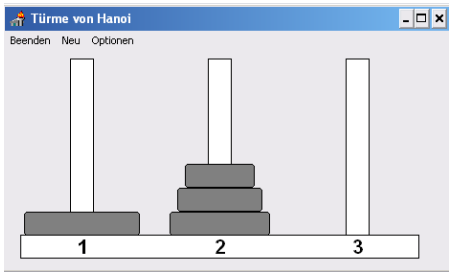
Beispiel: Türme von Hanoi

Regeln: Ein Turm aus einer gewissen Anzahl von Scheiben muss mit möglichst wenigen Zügen vom ersten Stapel auf den dritten Stapel bewegt werden, wobei eine größere Scheibe nie auf einer kleineren positioniert werden darf.

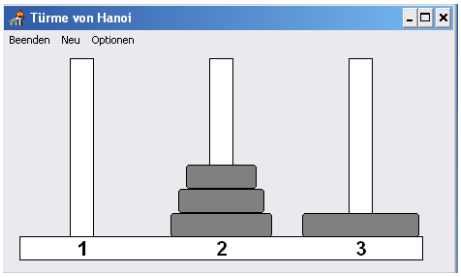
Logische Aufarbeitung:

Das beispielhaft aufgeführte Problem „Verschiebe 4 Scheiben von Stapel 1 nach Stapel 3“ (Ausgangssituation – siehe rechts) kann reduziert werden auf die folgenden leichter lösbaren Probleme:

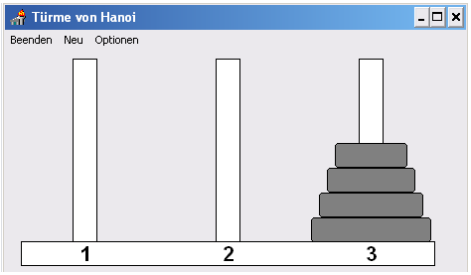




„verschiebe die drei oberen Scheiben von Stapel 1 nach Stapel 2“



„verschiebe die größte Scheibe von Stapel 1 nach Stapel 3“



„verschiebe die drei (übrigen) Scheiben von Stapel 2 nach Stapel 3“

Das erste Teilproblem kann man jetzt nach exakt dem gleichen Prinzip wiederum vereinfachen zu:

„verschiebe die zwei oberen Scheiben von Stapel 1 nach Stapel 3“
„verschiebe die größte (dritte) Scheibe von Stapel 1 nach Stapel 2“
„verschiebe die zwei (übrigen) scheiben von Stapel 3 nach Stapel 2“.

Dasselbe gilt für das dritte Teilproblem, sowie genauso für alle weiteren sich ergebenden Fälle, bei denen mehr als eine Scheibe beteiligt ist! Das Gesamtproblem wird also von Schritt zu Schritt kleiner, so dass man letztendlich immer wieder bei der elementaren Aufgabe landet, eine einzige Scheibe bewegen zu müssen.

Fachlicher Gesamt-Bezug: Die Türme von Hanoi stellen ein klassisches Beispiel einer speziellen Problemlösungsstrategie in der Informatik dar, und zwar der so genannten **Rekursion**.

Definition: Unter **Rekursion** versteht man (allgemein) die **Reduktion** eines Problems auf

- ein **leichter lösbares** Problem
- **derselben** Art.

Abgrenzung zur (bekannten) Iteration: Als Iteration bezeichnet man die **wiederholte** Durchführung eines Vorgangs. Die Anzahl der Durchführungen (Iterationen) steht entweder vorher fest oder richtet sich nach der Erfüllung eines Abbruchkriteriums.

Beispiele aus dem Alltag:

1. **„Babuschkas“**

iterativ:

```
Prozedur Zerlege(Babuschka);
 mache
   öffne die äußerste Babuschka
   wenn das Innere nicht leer ist
      dann hole es heraus und lege es zur Seite
   stecke die äußerste Babuschka wieder zusammen
   wenn das Innere nicht leer ist
     setze das Innere als neue äußerste Babuschka fest
solange, bis das Innere leer ist
```

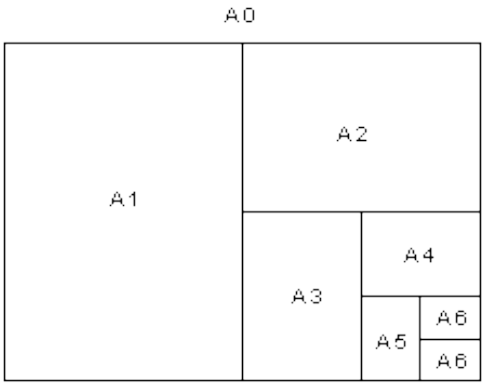
rekursiv:

```
Prozedur Zerlege(Babuschka);
 öffne Babuschka
 wenn Inneres leer
       dann schließe Babuschka
       sonst mache
                hole Inneres heraus und lege es zur Seite
                schließe Babuschka
                rufe Zerlege(Inneres) auf
```

2. **„DIN-Norm“**

iterativ:

```
Prozedur Erstelle_DINA_Blatt(n);
      besorge DINA0-Blatt
      mache n Mal
         falte Blatt und nimm eine Hälfte
```



rekursiv:

```
Prozedur Erstelle_DINA_Blatt(n);
      wenn n = 0
         dann besorge DINA0-Blatt
         sonst mache
                  rufe Erstelle_DINA_Blatt(n-1) auf;
                  falte Blatt und nimm eine Hälfte
```

Vergleich der Rekursion zur (bekannten) Iteration in der Informatik:

Iteration ist die wohl natürlichste Art zu programmieren. So ziemlich jeder Programmierer schreibt anfänglich iterative Programme – diese werden nacheinander, in einer klar ersichtlichen Reihenfolge ausgeführt. Typischerweise sind Schleifen vorzufinden und verwendete Unterprogramme verzweigen sich teils in weitere Unterprogramme, aber sie rufen sich niemals selbst auf!

Rekursion ist hingegen eine etwas elegantere Methode des Programmierens. **Rekursive Programme rufen sich selbst auf!** Dies ist der markanteste und entscheidendste Unterschied zu iterativen Programmen. Wenn es zur Lösung eines Problems immer derselben Schritte bedarf – und dies ist häufig der Fall – kann man ein relativ kurzes, übersichtliches Programm schreiben, indem man rekursiv programmiert.

Zu guter Letzt aber ein nicht zu verachtende Information!

Merke:
Jedes Problem, das rekursiv gelöst werden kann, kann auch in einer iterativen Variante formuliert werden – und umgekehrt!