Rekursive Funktionen *in Java*

- Motivation
- Beispiel für Funktionen: Berechnung der Fakultätsfunktion
- Beispiel für Prozeduren: Türme von Hanoi

Motivation

- Viele mathematische Funktionen sind *rekursiv* definiert, das bedeutet, dass in der Definition der Funktion die Funktion selbst wieder verwendet wird
- Beispiel: Fakultätsfunktion

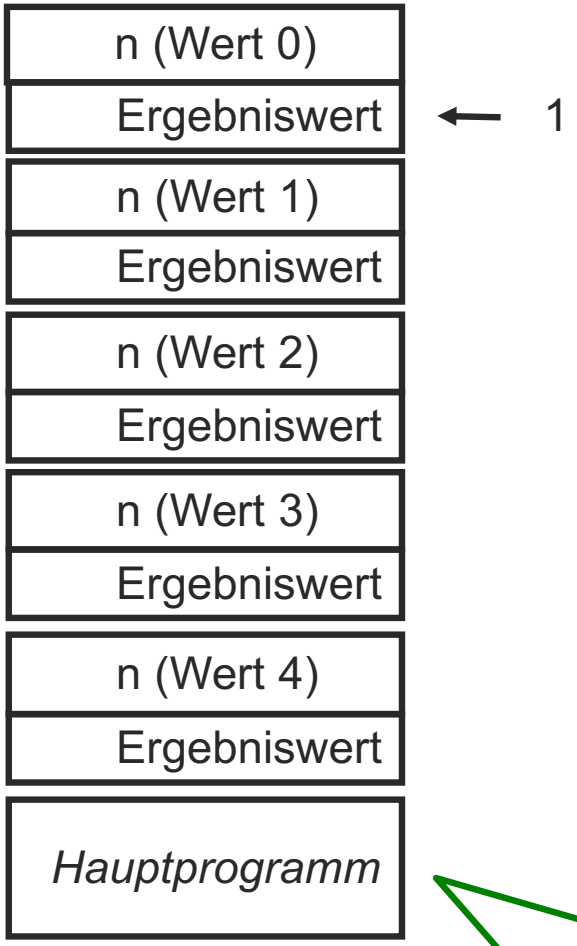
$$n! = \begin{cases} n * (n-1)! & \text{falls } n >= 1 \\ 1 & \text{falls } n = 0 \end{cases}$$

Implementierung in Java

$$n! = \begin{cases} n * (n-1)! & \text{falls } n >= 1 \\ 1 & \text{falls } n = 0 \end{cases}$$

```
static int fakultät( int n ) {
        if (n >= 1)
                return n * fakultät( n – 1 );
        else if (n == 0)
                return 1;
        else
                throw new PRException(”Fakultät ... undefiniert ... : “ + n);
}
```

Berechnung der Fakultätsfunktion



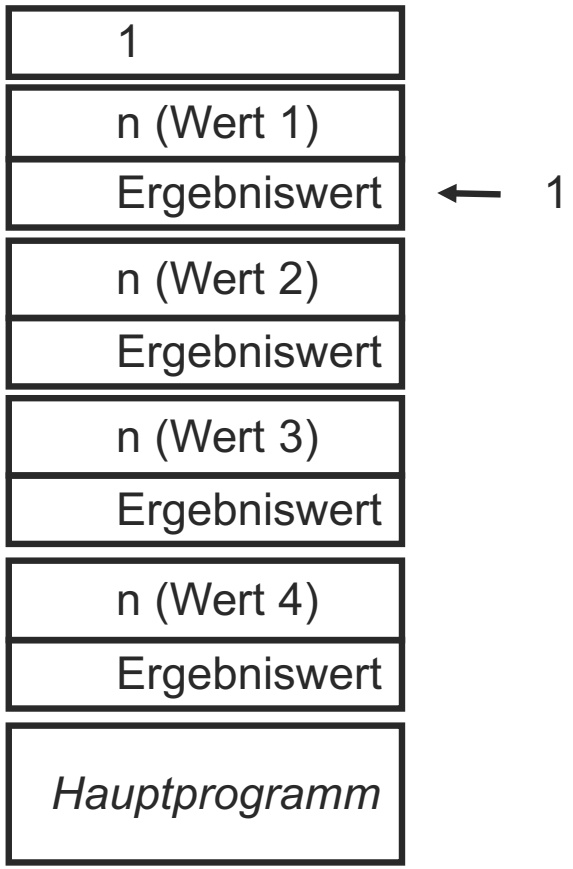
Beim rekursiven Aufruf einer (Fakultäts-) Funktion wird (wie bereits bekannt) *für jeden Aufruf ein Aktivierungssatz* angelegt; Beispiel-Aufruf: fakultät(4)

Daher gibt es keine „Probleme“ obwohl (im Beispiel) 5 Integer-Variablen namens "n" gleichzeitig aktiv sind

Die Aktivierungssätze werden wieder entfernt, wenn die entsprechenden Funktionsaufrufe beendet sind und der Ergebniswert vorliegt

5 Aktivierungssätze von „fakultät“ auf dem Stack

Berechnung der Fakultätsfunktion

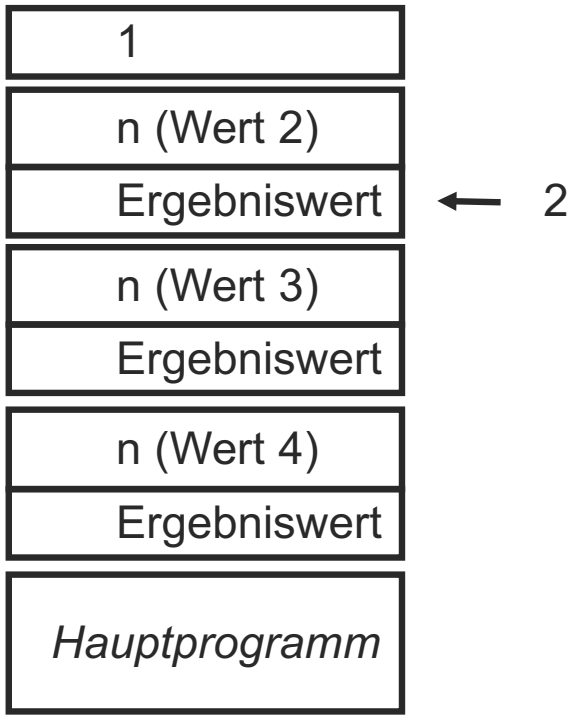


Beim rekursiven Aufruf einer (Fakultäts-) Funktion wird (wie bereits bekannt) für jeden Aufruf ein Aktivierungssatz angelegt; Beispiel-Aufruf: fakultät(4)

Daher gibt es keine „Probleme“ obwohl (im Beispiel) 5 Integer-Variablen namens n gleichzeitig aktiv sind

Die Aktivierungssätze werden wieder entfernt, wenn die entsprechenden Funktionsaufrufe beendet sind und der Ergebniswert vorliegt

Berechnung der Fakultätsfunktion

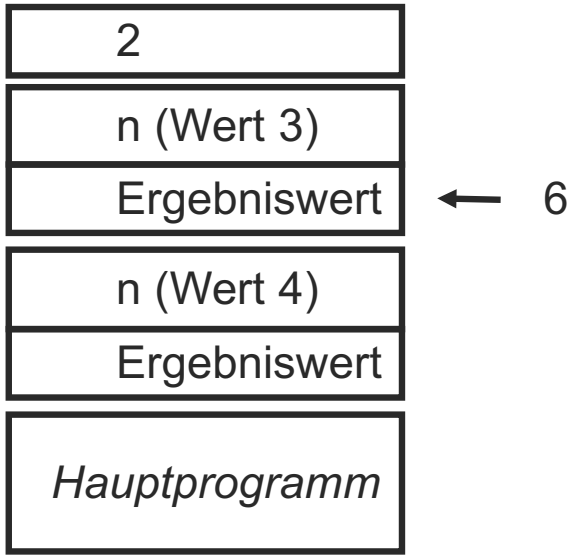


Beim rekursiven Aufruf einer (Fakultäts-) Funktion wird (wie bereits bekannt) für jeden Aufruf ein Aktivierungssatz angelegt; Beispiel-Aufruf: fakultät(4)

Daher gibt es keine „Probleme“ obwohl (im Beispiel) 5 Integer-Variablen namens n gleichzeitig aktiv sind

Die Aktivierungssätze werden wieder entfernt, wenn die entsprechenden Funktionsaufrufe beendet sind und der Ergebniswert vorliegt

Berechnung der Fakultätsfunktion

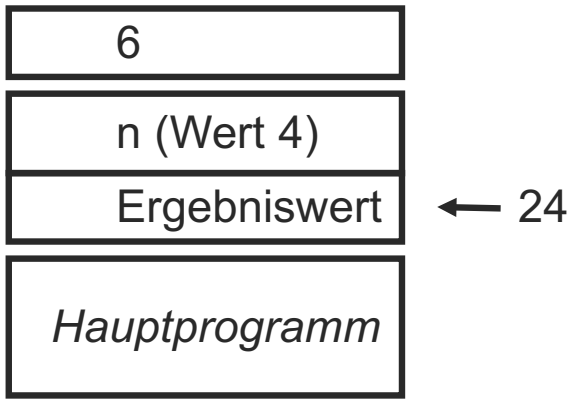


Beim rekursiven Aufruf einer (Fakultäts-) Funktion wird (wie bereits bekannt) für jeden Aufruf ein Aktivierungssatz angelegt; Beispiel-Aufruf: fakultät(4)

Daher gibt es keine „Probleme“ obwohl (im Beispiel) 5 Integer-Variablen namens n gleichzeitig aktiv sind

Die Aktivierungssätze werden wieder entfernt, wenn die entsprechenden Funktionsaufrufe beendet sind und der Ergebniswert vorliegt

Berechnung der Fakultätsfunktion



Beim rekursiven Aufruf einer (Fakultäts-) Funktion wird (wie bereits bekannt) für jeden Aufruf ein Aktivierungssatz angelegt; Beispiel-Aufruf: fakultät(4)

Daher gibt es keine „Probleme“ obwohl (im Beispiel) 5 Integer-Variablen namens n gleichzeitig aktiv sind

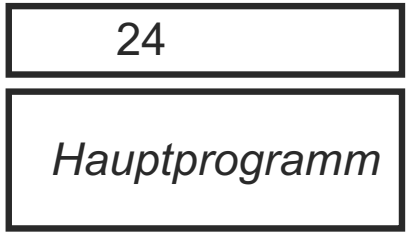
Die Aktivierungssätze werden wieder entfernt, wenn die entsprechenden Funktionsaufrufe beendet sind und der Ergebniswert vorliegt

Berechnung der Fakultätsfunktion

Beim rekursiven Aufruf einer (Fakultäts-) Funktion wird (wie bereits bekannt) für jeden Aufruf ein Aktivierungssatz angelegt; Beispiel-Aufruf: fakultät(4)

Daher gibt es keine „Probleme“ obwohl (im Beispiel) 5 Integer-Variablen namens n gleichzeitig aktiv sind

Die Aktivierungssätze werden wieder entfernt, wenn die entsprechenden Funktionsaufrufe beendet sind und der Ergebniswert vorliegt



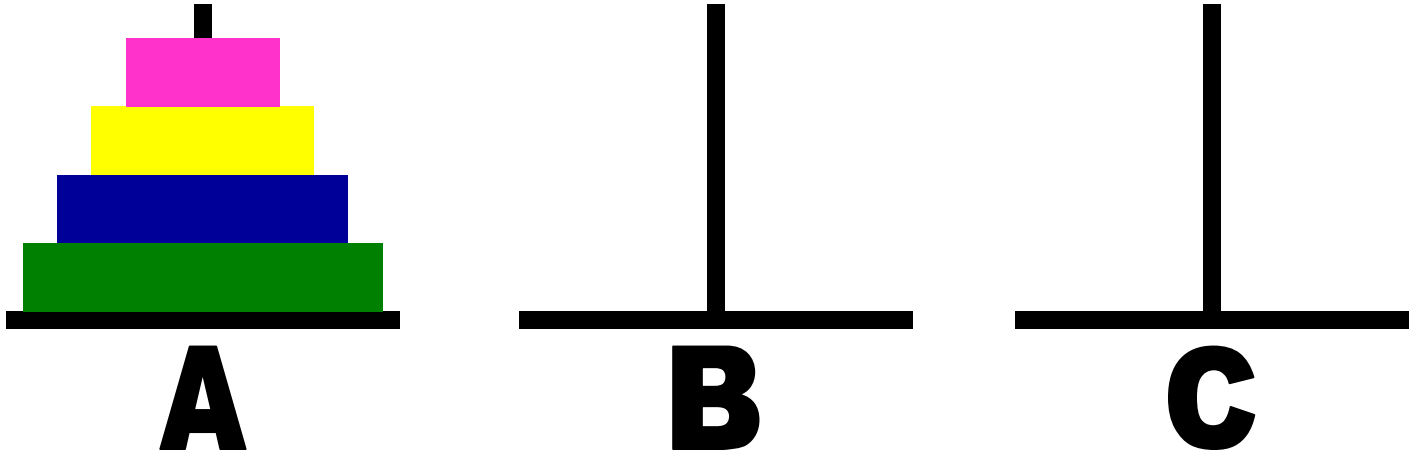
Haben Sie Fragen?

Rekursive Prozeduren

- Wir haben rekursive *Funktionen* kennen gelernt
- Diese haben in der Mathematik ihre Entsprechung, sind also auch in Programmiersprechen wie Java vertreten
- Es gibt aber auch Fälle, in denen *rekursive Prozeduren* sinnvoll eingesetzt werden können
- Java unterstützt rekursive Prozeduren genau so wie rekursive Funktionen
  - Für jeden Aufruf einer Prozedur wird ein eigener Aktivierungssatz angelegt
  - Diese Aktivierungssätze werden nach Ablauf der Prozedur automatisch wieder entfernt
  - Üblicherweise vollbringen Prozeduren ihre Aufgaben durch *Seiteneffekte* (die sich auf die Parameter oder Ein- und Ausgabe beschränken sollten)

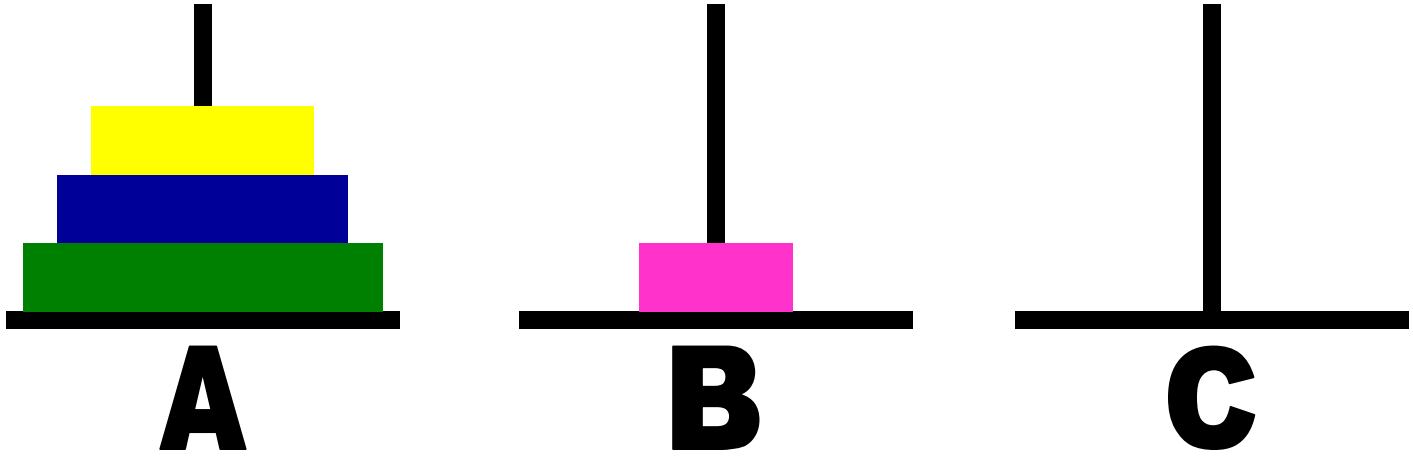
Die Türme von Hanoi

- Eines der prominentesten Beispiele für Rekursion bei Prozeduren ist das Problem der Türme von Hanoi
- Die Aufgabe besteht darin, n Scheiben verschiedener Durchmesser von einem Platz A zu einem Platz C zu transportieren
- Dabei dürfen Scheiben auf einem Platz C zwischengelagert werden
- Es ist verboten, eine Scheibe auf eine andere Scheibe kleineren Durchmessers zu legen



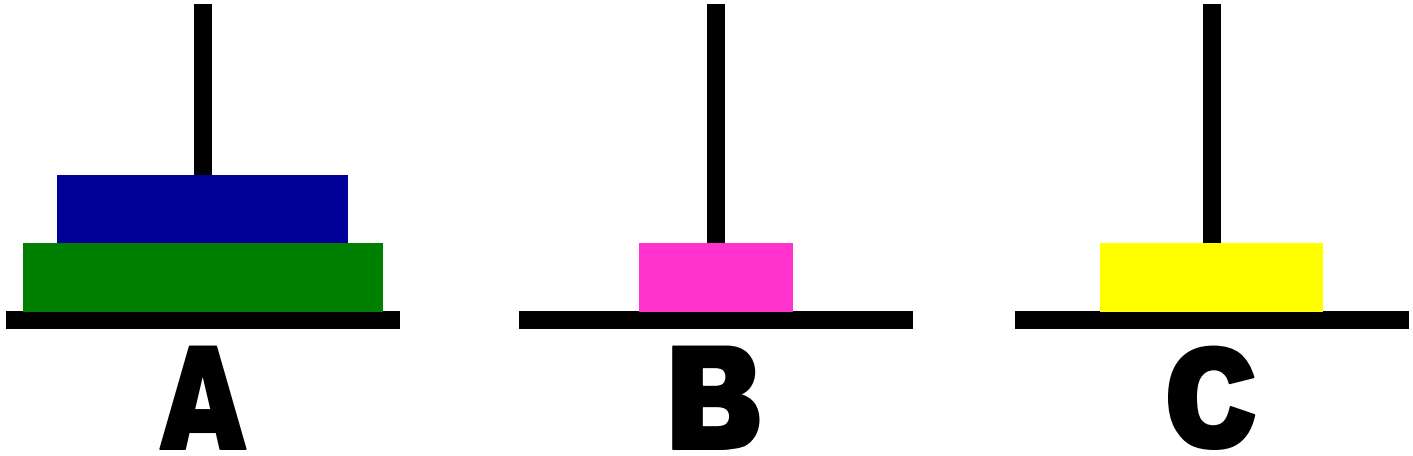
Die Türme von Hanoi

- Eines der prominentesten Beispiele für Rekursion bei Prozeduren ist das Problem der Türme von Hanoi
- Die Aufgabe besteht darin, n Scheiben verschiedener Durchmesser von einem Platz A zu einem Platz C zu transportieren
- Dabei dürfen Scheiben auf einem Platz C zwischengelagert werden
- Es ist verboten, eine Scheibe auf eine andere Scheibe kleineren Durchmessers zu legen



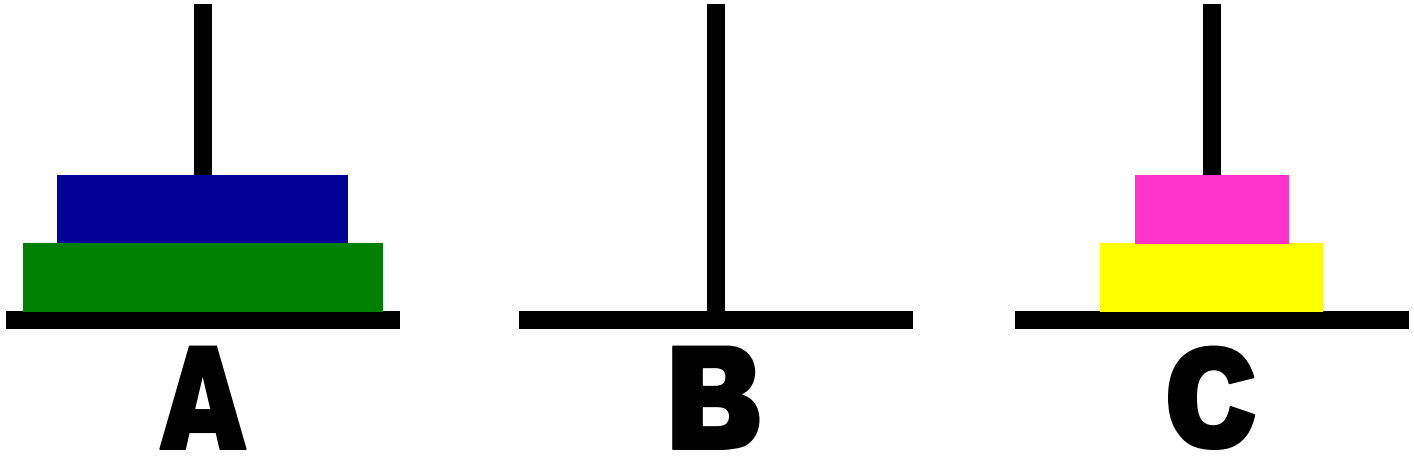
Die Türme von Hanoi

- Eines der prominentesten Beispiele für Rekursion bei Prozeduren ist das Problem der Türme von Hanoi
- Die Aufgabe besteht darin, n Scheiben verschiedener Durchmesser von einem Platz A zu einem Platz C zu transportieren
- Dabei dürfen Scheiben auf einem Platz C zwischengelagert werden
- Es ist verboten, eine Scheibe auf eine andere Scheibe kleineren Durchmessers zu legen



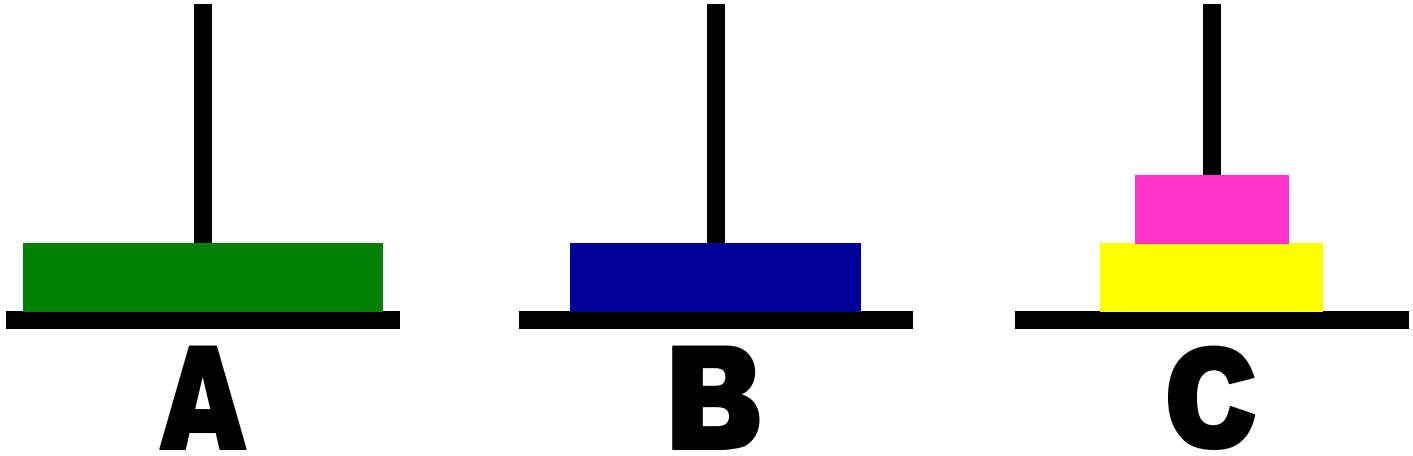
Die Türme von Hanoi

- Eines der prominentesten Beispiele für Rekursion bei Prozeduren ist das Problem der Türme von Hanoi
- Die Aufgabe besteht darin, n Scheiben verschiedener Durchmesser von einem Platz A zu einem Platz C zu transportieren
- Dabei dürfen Scheiben auf einem Platz C zwischengelagert werden
- Es ist verboten, eine Scheibe auf eine andere Scheibe kleineren Durchmessers zu legen



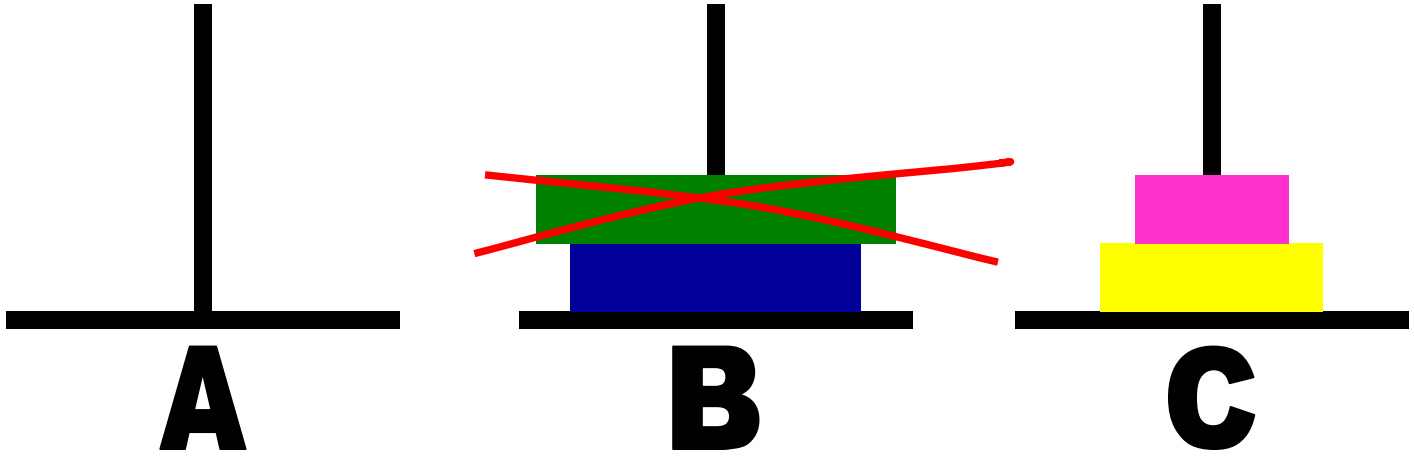
Die Türme von Hanoi

- Eines der prominentesten Beispiele für Rekursion bei Prozeduren ist das Problem der Türme von Hanoi
- Die Aufgabe besteht darin, n Scheiben verschiedener Durchmesser von einem Platz A zu einem Platz C zu transportieren
- Dabei dürfen Scheiben auf einem Platz C zwischengelagert werden
- Es ist verboten, eine Scheibe auf eine andere Scheibe kleineren Durchmessers zu legen



Die Türme von Hanoi

- Eines der prominentesten Beispiele für Rekursion bei Prozeduren ist das Problem der Türme von Hanoi
- Die Aufgabe besteht darin, n Scheiben verschiedener Durchmesser von einem Platz A zu einem Platz C zu transportieren
- Dabei dürfen Scheiben auf einem Platz C zwischengelagert werden
- Es ist verboten, eine Scheibe auf eine andere Scheibe kleineren Durchmessers zu legen



Rekursiver Lösungsalgorithmus

- Das Problem lässt sich mit folgender Strategie lösen
  1. Schaffe N-1 Scheiben von Platz A nach Platz B
  2. Schaffe 1 Scheibe von Platz A nach Platz C
  3. Schaffe N-1 Scheiben von Platz B nach Platz C

Das sind alle außer der unteren, größten Scheibe

Das ist die untere, größte Scheibe

Das sind wieder alle außer der unteren, Größten

- Das Problem ist kleiner geworden:
  Schafft man die Lösung für N-1 Scheiben, dann hat man eine Anleitung gefunden, wie das Problem für N Scheiben zu lösen ist
- Es handelt sich um eine rekursive Lösung:
  Für die Lösung des Problems in Teil 1 darf Platz C und in Teil 3 darf Platz A für die Zwischenlagerung von Scheiben benutzt werden
- Ist das Problem entsprechend weit reduziert, muss nur eine einzelne Scheibe transportiert werden (Rekursionsstart mit N = 1)

Java-Programm

- In Java kann das Problem wie folgt rekursiv gelöst werden

```
static void hanoi ( int n, String start, String ziel, String zwischen )
{
    if ( n == 1 )
        println( "Schaffe Scheibe von "+start+" nach "+ziel );
    else {
        hanoi( n-1, start, zwischen, ziel );
        println( "Schaffe Scheibe von "+start+" nach "+ziel );
        hanoi( n-1, zwischen, ziel, start );
    }
}
```

Das sind alle außer der unteren, größten Scheibe

Das ist die untere, größte Scheibe

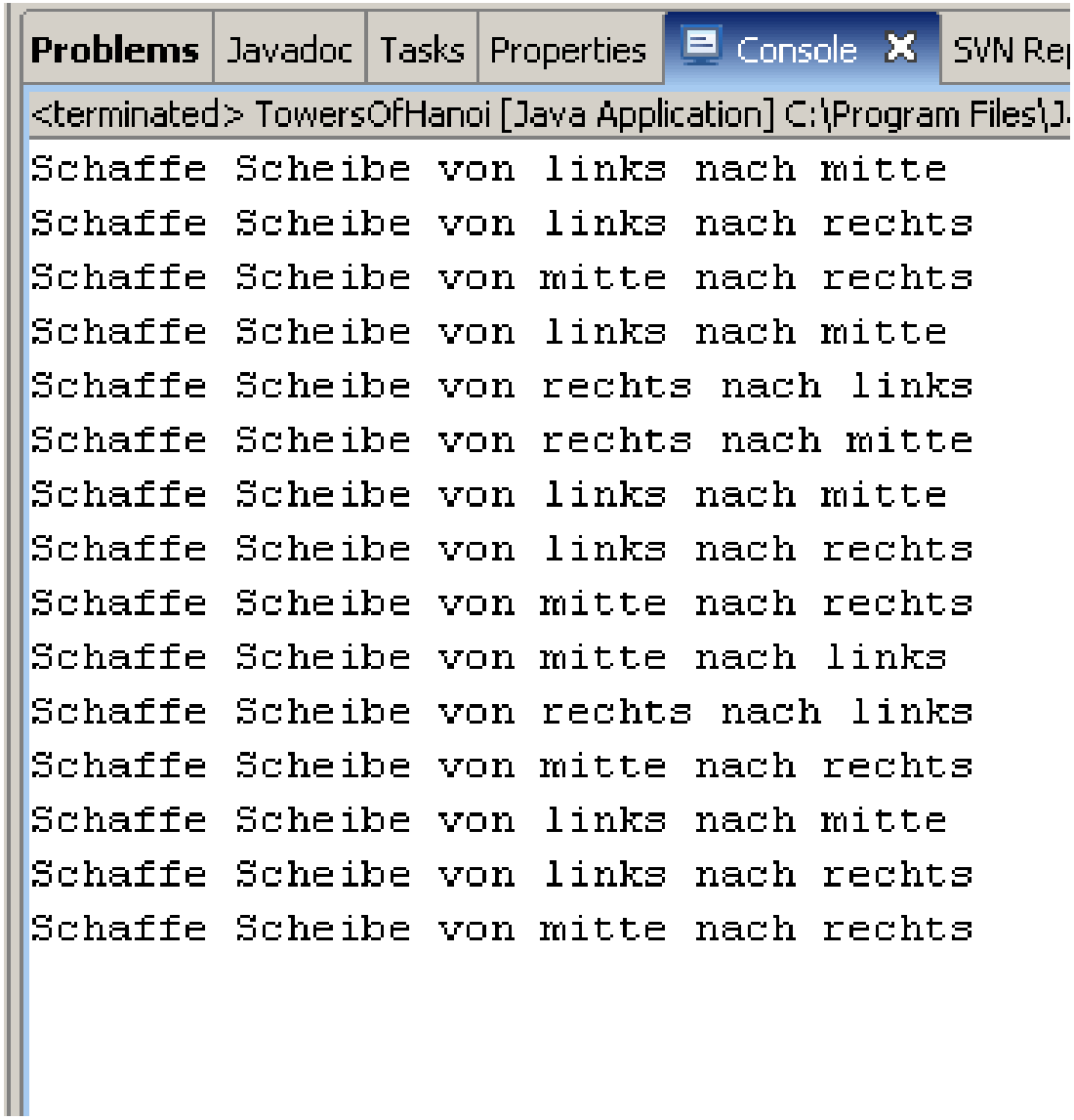
Das sind wieder alle außer der unteren, Größten

- Aufruf-Beispiel:

```
hanoi( 4, "links", "rechts", "mitte" );
```

- Der Seiteneffekt der Prozedur besteht darin, die Anweisungen zum Transportieren der Scheiben auf dem Bildschirm auszugeben

Beispiel: Ausgabe zum Aufruf hanoi(4, "links", "rechts", "mitte");



Haben Sie Fragen?