Rekursion

„Wer Rekursion verstehen will, muss vorher Rekursion verstehen."

Einführung

Unter einer *rekursiven Methode* versteht man eine Methode, die sich selbst wieder aufruft.
Damit diese Aufrufe nicht *unendlich* weitergehen und das Programm zu einem Ergebnis kommt, brauchen wir eine *Abbruchbedingung*.

1. Fakultät

Eine wichtige mathematische Funktion ist die Fakultät:

$$n! = 1 \cdot 2 \cdot 3 \cdots (n-2) \cdot (n-1) \cdot n$$

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

Diese kann mit einer rekursiven Methode `fak` berechnet werden.

a) Notiere dir zunächst die rekursiven Methodenaufrufe und ein konkretes Zahlenbeispiel (für $n = 5$) dafür:

b) Wann bricht die Rekursion ab, d. h. wann wird nicht mehr erneut die Methode `fak` aufgerufen?

c) Programmiere die Methode `fak` in Java. (*Hinweis: diese soll wieder direkt programmiert werden, ohne ein Objekt anzulegen, also direkt unter die `main`-Methode.*)

d) Wie muss der Methodenkopf in Java aussehen?

e) Rufe deine Methode `fak` aus der `main`-Methode auf mit folgenden Parametern: `fak(1);` (Ergebnis: 1), `fak(3);` (Ergebnis: 6), `fak(5);` (Ergebnis: 120), `fak(11);` (Ergebnis: 39 916 800)

2. Fibonacci-Folge

Die Fibonacci-Folge

$$0,1,1,2,3,5,8,13,21,34,55,\ldots$$

ist folgendermaßen definiert:

- $F(0) = 0$
- $F(1) = 1$
- $F(2) = F(0) + F(1)$
- $F(3) = F(1) + F(2)$
- oder allgemein: $F(n) = F(n-2) + F(n-1)$

Diese soll nun mit einer Methode `fibo` umgesetzt werden.

a) Notiere dir zunächst die rekursiven Methodenaufrufe dafür und das Aufrufschema für $n = 5$:

b) Wann bricht die Rekursion ab, d. h. wann wird nicht mehr erneut die Methode `fibo` aufgerufen?

c) Wie muss der Methodenkopf in Java aussehen?

d) Programmiere die Methode `fibo` in Java. (*Hinweis: diese soll wieder direkt programmiert werden, ohne ein Objekt anzulegen, also direkt unter die `main`-Methode.*)

e) Rufe deine Methode `fibo` aus der `main`-Methode auf mit folgenden Parametern: `fibo(1);` (Ergebnis: 1), `fibo(3);` (Ergebnis: 2), `fibo(5);` (Ergebnis: 5), `fibo(11);` (Ergebnis: 89), `fibo(23);` (Ergebnis: 28 657)

3. Pascal'sches Dreieck

Das PASCALsche Dreieck sieht folgendermaßen aus:

$$
\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1 \\
 & \ddots & & & \vdots & & & & \ddots & &
\end{array}
$$

Die Zahlen darin sollen mit der Methode `binom(n,k);` berechnet werden, wobei n die Zeile und k die Spalte (innerhalb dieser Zeile) angibt. (Beispiele s. unten, *Hinweis: die Zeilen- und Spaltennummerierung beginnt bei 0!*)

a) Recherchiere im Internet, wie das PASCALsche Dreieck bzw. dessen Inhalte *rekursiv* berechnet werden können.

b) Notiere dir die rekursiven Methodenaufrufe und das Aufrufschema für $n = 3$ und $k = 2$:

c) Wann bricht die Rekursion ab, d. h. wann wird nicht mehr erneut die Methode `fibo` aufgerufen?

d) Wie muss der Methodenkopf in Java aussehen?

e) Programmiere die Methode `binom` in Java. (*Hinweis: diese soll wieder direkt programmiert werden, ohne ein Objekt anzulegen, also direkt unter die `main`-Methode.*)

f) Rufe deine Methode `binom` aus der `main`-Methode auf mit folgenden Parametern: `binom(0,0);` (Ergebnis: 1), `binom(2,1);` (Ergebnis: 2), `binom(5,3);` (Ergebnis: 10), `binom(9,4);` (Ergebnis: 126), `binom(20,7);` (Ergebnis: 77 520)