zu Kapitel 5

Aufgabe 1

(Weihnachtsbaum.java)

a) Schreiben Sie eine Funktion, die eine Zeichenkette beliebiger Länge mit beliebiger Startposition (beginnend von der aktuellen Cursor-Position) in eine Zeile zeichnet. Auch das verwendete Zeichen soll frei wählbar sein. Die Funktion hat die Form:

```
static void printKette (char zeichen, int start, int laenge)
```

Der Aufruf

```
printKette('a', 3, 5);
```

bewirkt also, dass erst zwei Leerzeichen und dann 5-mal der Buchstabe a geschrieben werden. Testen Sie Ihre Funktion mit einem einfachen main()-Programm, das Sie danach wieder löschen können.

b) Mit einem main()-Programm soll jetzt folgender einfacher Weihnachtsbaum unter Verwendung der Funktion printKette() erzeugt werden:

```
       +
      +++
     +++++
    +++++++
   +++++++++
  ++++++++++
 +++++++++++
+++++++++++++
```

Die Höhe des Weihnachtsbaums (in obigem Bild ist sie 7) wird dabei in einer Integervariable zu Beginn der main()-Funktion beliebig gewählt.

Wichtig: beginnen Sie erst mit Teilaufgabe b), wenn Teilaufgabe a) fertig ist.

Aufgabe 2

(Fakult.java)

Schreiben Sie ein Programm, das berechnet wie viele Möglichkeiten es beim Zahlenlotto 6 aus 49 gibt.

a) Schreiben Sie zuerst eine iterative Variante der Fakultätsfunktion (d.h. unter Verwendung einer Laufschleife), welche die Fakultät einer vorgegebenen nicht negativen, ganzen Zahl berechnet. Die Funktion hat als Übergabeparameter eine Zahl vom Typ int und gibt die Fakultät mit dem Typ double zurück.

Hinweis:
Die Fakultät n! einer positiven ganzen Zahl ist definiert durch n! = n*(n-1)*...2*1. Weiterhin gilt 0! = 1. Da Fakultäten recht schnell recht groß werden, sollte der Rückgabewert der Funktion vom Typ double sein.

b) Schreiben Sie danach ein Programm für die Möglichkeiten beim Zahlenlotto.

Hinweis:
Die Formel für die Anzahl der Möglichkeiten 6 aus 49 beim Zahlenlotto lautet: 49! / ((49 – 6)! * 6!)

Aufgabe 3
(Power2.java)

Erstellen Sie eine rekursive Variante der power()-Funktion mit der Funktionsdeklaration

```
static long power (int basis, int exponent)
```

und testen Sie diese mit einer einfachen main()-Funktion, in der Sie "basis" und "exponent" von Tastatur einlesen.

Tipp: Es gilt: $a^n = a * a^{n-1}$ und $a^0 = 1$.

Hinweis: Dies ist eine einfache Übungsaufgabe zu rekursiven Funktionen, die iterative Variante der power()-Funktion aus der Vorlesung ist jedoch wesentlich effizienter und deshalb zu bevorzugen.

Aufgabe 4
(ZaehleHanoi.java)

Erstellen Sie eine rekursive Funktion, die zählt, wie viele Scheiben beim Spiel "die Türme von Hanoi" umgelegt werden müssen. Übergabeparameter ist die Anzahl der Scheiben, mit denen gespielt wird. Schreiben Sie ein Testprogramm (d.h. eine main()-Funktion), welches Ihre rekursive Funktion testet.

Hinweis: Natürlich gibt es eine mathematische Formel, um die Anzahl der Züge zu berechnen Die sollen Sie aber nicht verwenden, sondern stattdessen rekursiv programmieren. Das bedeutet: Um mit n Scheiben zu spielen, müssen Sie einen Turm der Höhe n-1 umsetzen, dann die letzte Scheibe bewegen und zum Schluss den Turm mit n-1 Scheiben, wieder darauf setzten. Wenn Sie also wissen, wie viele Scheiben Sie bewegen müssen, um einen Turm mit n-1 Scheiben zu bewegen, dann wissen Sie auch, wie viele Scheiben Sie bewegen müssen, um einen Turm mit n Scheiben zu bewegen.

Aufgabe 5
(Wuerfeln.java)

Sie wollen ein Würfelspiel spielen, haben aber keinen Würfel. Schreiben Sie ein Programm, das Ihnen eine Zufallszahl zwischen 1 und 6 erzeugt.

Hinweis: Zufallszahlen erzeugen Sie mittels der statischen Methode random() in der Klasse Math. Lesen Sie dazu die Beschreibung z.B. in der Java-SUN-Dokumentation durch.

Aufgabe 6
(Wochentag2.java)

Nachdem wir jetzt Zahlen von Tastatur einlesen können, können wir (endlich) kleine "Tools" erstellen, auch wenn diese noch nicht sehr komfortabel sind. Ändern Sie das Programm Wochentag.java so ab, dass Tag, Monat und Jahr von der Tastatur eingelesen werden. Machen Sie Ein- und Ausgabe so komfortabel, dass das Programm, wie ein (sehr einfaches) Tool aussieht.