Übungsaufgaben zu Einführung in das Programmieren

SS 2007

1. Bekanntlich ist $n! = n \cdot (n-1) \cdot (n-2) \ldots 3 \cdot 2 \cdot 1$.

   a) Schreiben Sie eine Applikation, die eine ganze Zahl n von der Tastatur einliest und $n!$ ausgibt.

   b) Schreiben Sie ein Programm, das mit Hilfe der Formel

   $$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

   die Zahl e berechnet. Der Benutzer soll eingeben können, wie viele Terme der Summe berechnet werden sollen.

   c) Finden Sie (im Internet) eine Näherungsformel für π und schreiben Sie ein Programm, bei dem Sie eingeben können, auf wie viele Stellen genau Sie π berechnen möchten, und berechnen Sie π dann mit dieser Genauigkeit.

2. Schreiben Sie ein Programm, das (untereinander) die folgenden Muster ausgibt:

```
*           *********       *********              *
**          ********         ********             **
***         *******           *******            ***
****        ******             ******           ****
*****       *****               *****          *****
******      ****                 ****         ******
*******     ***                   ***        *******
********    **                     **       ********
*********   *                       *      *********
```

   `*` darf dabei nur als `System.out.print('*');` angegeben werden. Ansonsten sollten Sie zur Ausgabe nur `System.out.print(' ');` und `System.out.println();` verwenden.

3. Benutzen Sie das *Sieb des Eratosthenes*, um alle Primzahlen ≤ 999 zu bestimmen und auszudrucken:

   a) Legen Sie einen Array mit 1000 `boolean` Elementen an und initialisieren Sie alle Elemente mit `true`.

b) Beginnend mit Array-Index 2 soll das Programm für jedes Array-Element, das auf `true` steht eine Schleife durchlaufen, in der alle Elemente, deren Index ein Vielfaches dieses Index ist, auf `false` gesetzt werden.

Nachdem dieser Prozeß abgeschlossen ist, sind die Primzahlen ≤ 999 genau die Indizes der verbleibenden `true`-Elemente. Diese sollen dann formatiert ausgegeben werden.

4. Schreiben Sie eine Applikation, die eine Folge von 10 ganzen Zahlen von der Tastatur einliest und sie aufsteigend sortiert wieder ausgibt.

5. Schreiben Sie eine Applikation `Zahlenraten`, bei der der Benutzer eine vom Computer intern ermittelte (Integer-) Zufallszahl zwischen `0` und `999` durch Eingaben über die Tastatur herausfinden soll. Zufallszahlen generiert man mit Hilfe der Methode `Math.random()`, die eine Zufallszahl vom Typ `double` im Wertebereich $[0, 1)$ retourniert. Die Deklaration

   ```
   z = (int) (1000*Math.random());
   ```

   liefert also eine zufällige Integer-Zahl zwischen `0` und `999`. Das Programm soll nach jedem Tipp des Users `zu hoch!`, `zu niedrig!` oder `richtig!` ausgeben. Ein- und Ausgabe des Programmes sollten über ein `JOptionPane`-Objekt, wie im Beispiel `Eingabe_Swing.java` gezeigt, erfolgen.

6. Die Folge der Fibonacci Zahlen

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

   beginnt mit den Zahlen 0 und 1 und hat nach Definition die Eigenschaft, dass jedes Folgenglied die Summe der beiden Vorgänger ist. Schreiben Sie ein Programm, das die ersten n Fibonacci-Zahlen in einer Tabelle ausgibt, wobei n vom Benutzer eingegeben werden soll.

7. *Türme von Hanoi.* Die Priester eines fernöstlichen Klosters stehen vor folgender Aufgabe: Auf einem Pflock sind Scheiben (mit einem Loch in der Mitte) der Größe nach aufgeschichtet (wobei die größten Scheiben unten liegen). Es stehen zwei weitere Pflöcke zur Verfügung. Das Ziel der Priester ist es, die Scheiben von Pflock 1 nach Pflock 3 zu transportieren (mit Pflock 2 als Zwischenlager), wobei sie sich allerdings an folgende Regeln zu halten haben:

   - Es darf in jedem Arbeitsschritt nur eine Scheibe bewegt werden.

- Niemals darf eine größere auf eine kleinere Scheibe gelegt werden.

Schreiben Sie eine Applikation, die den Benutzer zur Eingabe der Scheibenzahl auffordert und eine Lösungsstrategie etwa in der Form

```
1 -> 3
1 -> 2
3 -> 2
...
```

ausgibt.

Hinweise: Am besten löst man dieses Problem *rekursiv*, mittels folgender Überlegung: Um n Scheiben von Pflock 1 nach Pflock 3 zu transportieren, kann man wie folgt vorgehen:

a) Bewege $n-1$ Scheiben von Pflock 1 nach Pflock 2, wobei Pflock 3 als Zwischenlager dient.

b) Lege die letzte (größte Scheibe) von Pflock 1 nach Pflock 3.

c) Bewege die $n-1$ Scheiben von Pflock 2 nach Pflock 3, wobei Pflock 1 als Zwischenlager dient.

Somit kann man den Fall von n Scheiben auf den von $n-1$ Scheiben reduzieren und der Fall $n = 1$ ist ohnehin trivial.

Schreiben Sie eine rekursive (also sich selbst aufrufende) Methode `turm`, die vier `int` Parameter übernimmt:

(a) Die Gesamtzahl der Scheiben.

(b) Die Zahl des Pflocks, auf dem die Scheiben zu Beginn liegen.

(c) Die Zahl des Pflocks, auf dem die Scheiben am Ende liegen sollen.

(d) Die Zahl des Pflocks, der als Zwischenlager dienen soll.

Mit Hilfe dieser Methode ist die Applikation dann leicht zu implementieren (man kann den Benutzer sogar Anfangs- und Endstapel aussuchen lassen, wenn man möchte).

a) Implementieren Sie eine Klasse `Komplex`, die die Durchführung der Grundrechnungsarten und die formatierte Ausgabe komplexer Zahlen ermöglicht.

b) Schreiben Sie eine Applikation, um die Funktionalität ihrer Klassendefinition zu überprüfen.

8. Schreiben Sie eine Applikation, die einen String einliest und mittels einer rekursiven Methode `stringReverse` (mit Rückgabewert `void`) in umgekehrter Reihenfolge wieder ausgibt. Der Methode `stringReverse` soll als Parameter ein Character-Array, der den String enthält, übergeben werden.

   Hinweis: Sorgen Sie dafür, dass das letzte Zeichen des Character-Arrays ein '\0' ist und definieren Sie zwei Instanzvariablen `i,j`, die mit 0 initialisiert sind sowie eine Instanzvariable `s` (ein Character-Array derselben Länge wie der eingelesene String). `stringReverse( char b[])` arbeitet dann wie folgt:

   - Falls `b[i] == '\0'`: `return;`
   - Sonst: Erhöhe `i` um 1 und rufe `stringReverse(b)` auf.
   - `s[j++] = b[--i];`

   Fertigen Sie ein Struktogramm an und testen Sie zuerst von Hand, ob der Algorithmus funktioniert.