Humboldt-Universität zu Berlin
Grundlagen der Programmierung
(Vorlesung von Prof. Bothe)

Institut für Informatik
WS 15/16

Übungsblatt 5: Sichtbarkeit und Rekursion

Abgabe: bis **9:00 Uhr** am **11.01.2016** über **Goya**

Die Lösung dieses Übungsblatts soll in Gruppen von je 2 Personen erfolgen. Die Abgabe der Lösungen erfolgt durch Hochladen **einer Datei für jede Aufgabe** im Goya-System. Verwenden Sie exakt den in der Aufgabe angegebenen Dateinamen! **Java-Lösungen** müssen im UTF-8 Format abgegeben werden und auf dem Institutsrechner **`gruenau5`** korrekt kompilierbar sein (ansonsten wird die Aufgabe mit 0 Punkten bewertet).

Aufgabe 1 (Semantik → `FixIt.java`) — 6 Punkte

Betrachten Sie die folgende Klasse `FixIt`. Der Aufruf der Methode `FixIt.m(k)` soll den folgenden Wert berechnen:

$$m(k) = \prod_{x=1}^{k} \sum_{y=1}^{x} y = \left(\sum_{y=1}^{1} y\right) \cdot \left(\sum_{y=1}^{2} y\right) \cdot \ldots \cdot \left(\sum_{y=1}^{k} y\right)$$

Zum Beispiel ergibt:

$$m(1) = 1$$
$$m(2) = (1) \cdot (1+2) = 3$$
$$m(4) = (1) \cdot (1+2) \cdot (1+2+3) \cdot (1+2+3+4) = 180$$

```
1  public class FixIt {
2      private static int i;
3      private int j;
4      public int a( int k ) {
5          j = 0;
6          for( i = 1; i < k; ++i )
7              j += 1;
8          return k;
9      }
10     public static int m( int k ) {
11         j = 0;
12         for( i = 1; i <= k; ++i )
13             j *= a( i );
14         return j;
15     }
16 }
```

Leider haben sich ein paar Fehler eingeschlichen. Finden Sie alle Fehler und korrigieren Sie diese geeignet.

Aufgabe 2 (Binomialkoeffizienten → `blatt5_aufgabe2.txt`) **6 Punkte**

Der Binomialkoeffizient kann mit $\binom{n}{k} = \frac{n!}{k!\ (n-k)!}$ berechnet werden, aber auch ein rekursiver Lösungsweg ist möglich ($0 \leq n \geq k \geq 0$):

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{mit} \quad \binom{n}{n} = \binom{n}{0} = 1$$

Verfolgen Sie die Berechnung von $\binom{3}{2} = 3$ mit der rekursiven Methode `binomi`. Protokollieren Sie die Werteveränderungen der sichtbaren Variablen in einer Variablenbelegungstabelle.

```
1 public static int binomi( int n, int k ) {
2     if( k == n || k == 0 )
3         return 1;
4     int a = binomi( n-1, k-1 );
5     int b = binomi( n-1, k );
6     return a + b;
7 }
```

Dokumentieren Sie die Variablenbelegungstabelle für die Variablen `n, k, a, b`. Die Spalten der Tabelle seien die Variablen; jede Zeile stehe für eine Zuweisung und die nach der Zuweisung vorliegenden Werte der Variablen. Trennen Sie die Werte in einer Zeile mit einem Komma. Der Anfang dieser Tabelle könnte zum Beispiel so aussehen:

```
n, k, a, b
3, 2, -, -
2, 1, -, -
```

Vervollständigen Sie die Datei `blatt5_aufgabe2.txt`.

Hinweis: Natürlich gibt es im Falle von rekursiven Aufrufen mehrere Instanzen lokaler Variablen im Arbeitsspeicher. Protokollieren Sie in jeder Zeile stets die Werte, die zur Zeit der Ausführung gerade sichtbar sind.

Aufgabe 3 (Entrekursivierung → `Hanoi.java`) **8 Punkte**

In der Vorlesung (II.11, 2. Teil, S. 8) haben Sie die Türme von Hanoi kennen gelernt. Ebenfalls wurde eine Beispielimplementierung vorgestellt, die das Problem löst. Die Beispielimplementierung kann auch von der Website geladen werden. (https://www2.informatik.hu-berlin.de/swt/lehre/GdP-WS-15/java_beispiele/TEIL_II/Hanoi.java).
Schreiben Sie dieses Programm derart um, dass es ohne rekursive Aufrufe funktioniert. Benutzen Sie dazu den Stack, der durch die folgende Klasse `IntStack` implementiert wird:

```java
public class IntStack {
    private int[] stackElements;
    private int top;

    public IntStack( int n ) {
        stackElements = new int[n];
        top = -1;
    }

    public boolean isEmpty() {
        return top == -1;
    }

    public void push( int x ) {
        stackElements[++top] = x;
    }

    public int pop() {
        if( isEmpty() ) {
            System.out.println( "Stack leer" );
            return -1;
        }
        else {
            return stackElements[top--];
        }
    }
}
```