Programmieren 1
Selbststudium Semesterwoche 10

Aufgaben

Kapitel 6.3.1
1. zu bearbeitende Aufgaben: 6.1 bis 6.5

Kapitel 6.4.1
2. zu bearbeitende Aufgaben: 6.10 bis 6.13

Kapitel 6.4.2
3. zu bearbeitende Aufgaben 6.14 und 6.15

Kapitel 6.4.3
4. zu bearbeitende Aufgaben 6.16 bis 6.18

Kapitel 6.4.4
5. zu bearbeitende Aufgabe 6.20

Kapitel 6.7
6. zu bearbeitende Aufgaben 6.21 bis 6.23

Kapitel 6.8.2
7. zu bearbeitende Aufgaben 6.25, 6.26 und 6.28

Kapitel 6.9
8. zu bearbeitende Aufgaben 6.31 und 6.33

Testen

9. Schreiben Sie eine Testspezifikation für ein Programm, das das Volumen V eines Quaders bei gegebenen drei Seiten a, b und c berechnet.
Verwenden Sie dazu die angegeben Tabelle.

	Testfall	Eingabe	Ausgabe	bestanden?
Normalfälle:	Nr. 1	a=1, b=2, c=3	V=6	Ja
	Nr. 2	a=100, b=1, c=1	V=100	Ja
	Nr. 3	a=1, b=2, c=300	V=600	Ja
zulässige Spezialfälle:	Nr. 4	a=1, b=1, c=1	V=1	Kein Quader mehr
	Nr. 5	a=b=c		Kein Quader mehr
	Nr. 6			
unzulässige Spezialfälle:	Nr. 7	a=-1, b=2, c=3	V=-6	Nein
	Nr. 8	a=0, b=2, c=3	V=0	Nein

Algorithmen

10. Implementieren Sie den Pseudocode "Türme von Hanoi" in Java.

```
//Pseudocode (rekursiver Alogrithmus)
funktion bewege (Zahl i, Stab a, Stab b, Stab c) {
    falls (i > 0) {
       bewege(i-1, a, c, b);
       verschiebe oberste Scheibe von a nach c;
       bewege(i-1, b, a, c);
    }
}

//Pseudocode (iterativer Algorithmus)
solange (Stab A oder B enthalten Scheiben) {
    Verschiebe S1 im Uhrzeigersinn um einen Platz;
    falls (eine von S1 verschiedene Scheibe ist verschiebbar)
        Verschiebe eine von S1 verschiedene Scheibe;
}
```

```
//Lösungsbeispiel in Java
public class TowersOfHanoi
{
    private int count = 0;

    public TowersOfHanoi()
    {

    }

    public void moveTower(int NumberOfDiscs)
    {
        count = 0;
        move(NumberOfDiscs, "a", "c", "b");
    }

    private void move(int n, String src, String dst, String by)
    {
        count++;

        if(n==1)
        {
            System.out.println("Step "+count+": Move disc "+src+" to "+dst);
        }
        else
        {
            move(n-1, src, by, dst);
            move(1, src, dst, by);
            move(n-1, by, dst, src);
        }

    }
}
```

http://en.wikipedia.org/wiki/Tower_of_Hanoi

11. Beobachten Sie im Debugger wie die "offenen" Methoden (deren Ausführung unterbrochen ist) auf dem Stack "eingefroren" werden. Verwenden Sie dazu die im BlueJ Debugger angezeigte Aufrufkette der Methodenaufrufe (Call Graph).