SAARLAND UNIVERSITY
COMPUTER SCIENCE

Arbeitsgruppe Programmierung
Jun.-Prof. Dr. Sebastian Hack
Dipl.-Inform. Christoph Mallon

Programmierung 2 — Übungsblatt 8

Abgabe: **Dienstag, 23. Juni 2009, 11.45 Uhr, Geb. E1 3, Briefkasten im EG**
Lösung mit **Name**, **Matrikelnummer** und **Name des Tutors** beschriften!

Aufgabe 8.1: Programmieren mit KELLERMEISTER

Schreiben Sie KELLERMEISTER-Programme für folgende Aufgaben:

1. Berechne den Ausdruck $((100 * (-7) + (800 - 23 - 35)) * (810 - 30))$
2. Eine Funktion, die 1 zurückgibt, wenn für ihre Parameter a, b, c die Aussage $a < b < c$ gilt, 0 sonst.
3. Eine Funktion, die den Absolutwert einer Zahl berechnet.[1]
4. Eine Funktion mit einem Parameter n, welche die $n$te Fibonaccizahl berechnet.

Aufgabe 8.2: Umgekehrte polnische Notation (UPN)

1. Wandeln Sie die folgenden Ausdrücke in UPN um:
   - $((100 * (-7) + (800 - 23 - 35)) * (810 - 30))$
   - $((x \;\&\; !y) \mid !z) \mid (!x \mid y \;\&\; z) \;\&\; (!y \;\&\; true)$
2. Wandeln Sie die folgenden UPN-Ausdrücke in Infix-Notation um:
   - $300\ 2\ \ominus\ *\ 200\ 100\ -\ 1\ \ominus\ *\ +$
   - $x\ !\ y\ !\ \&\ z\ !\ \mid x\ y \mid z\ !\ \mid \&$

Ein Ziel von UPN ist die klammerfreie Darstellung von Ausdrücken. Dies führt zu Mehrdeutigkeiten bei unären und binären Operationen, die dasselbe Symbol verwenden. Ein Beispiel: $1\ 2\ -\ -$ kann als $-(1-2)$ oder $1-(-2)$ gelesen werden. Daher wird für die unären Operationen $\oplus$ und $\ominus$ verwendet. Somit ergibt sich eindeutig entweder $1\ 2\ -\ \ominus$ oder $1\ 2\ \ominus\ -$.

[1] Zusatzaufgabe: Realisieren sie dies, ohne `JZ` und `SEL` zu verwenden.

Aufgabe 8.3: Codeerzeugung für Schleifen und bedingte Abfragen

Analysieren Sie den folgenden Code für KELLERMEISTER. Welche Arten von Java-Schleifen/Verzweigungen werden hier durch die Kellermaschine ausgeführt? Welchen Algorithmus stellt das Programm dar? Schreiben Sie den Algorithmus als Java-Methode.

```
1  L0:
2     ld    0 // Lade erstes Argument
3     jz    L3
4     ldc   0
5     st    1
6     ldc   1
7     st    2
8  L1:
9     ld    0
10    addc  -1
11    st    0
12    ld    0
13    jz    L2
14    ld    1
15    ld    2
16    add
17    st    3
18    ld    2
19    st    1
20    ld    3
21    st    2
22    jmp   L1
23 L2:
24    ld    2
25    hlt
26 L3:
27    ldc   0
28    hlt
```

Aufgabe 8.4: Organisation des Laufzeitkellers: Schachtel, Parameterübergabe, Methodenaufruf, Rückkehr aus Methodendurchführung

In der Vorlesung wurde der Zustand des Laufzeitkeller bei der Durchführung geschachtelter Methodenaufrufe detailliert dargestellt.

1. Arbeiten Sie dieses Beispiel intensiv durch und beantworten Sie die folgenden Fragen:
   - Was ist eine Schachtel (Frame), wozu dient sie? Wann und wie wird eine Schachtel aufgebaut und schließlich mit Werten ausgefüllt?
   - Kann die Schachtel-Organisation auch eingesetzt werden, wenn sich Methoden direkt oder indirekt rekursiv aufrufen?

2. Notieren Sie die zeitlich aufeinanderfolgenden Zustände des Laufzeitkellers bei der Durchführung geschachtelter Methodenaufrufe der folgenden Klasse B. Orientieren Sie sich am Vorlesungbeispiel.

```
1  class B {
2    public static void main(String[] args) {
3      B b = new.B();
4      int res = b.m(3, 4);
5    }
6    public void m(int x, int y) {
7      n(x, 2 * y, y);
8      return;
9    }
10   public void n(int x, int y, int z) {
11     q(x + y, x + z);
12     return;
13   }
14   public void q(int x, int y) {
15     int s = x * y;
16     return;
17   }
18 }
```