TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Praktikum Grundlagen der Programmierung
Prof. Dr. Helmut Seidl

WS 2007/2008
Aufgabenblatt 5
11. November 2007

Abgabe: Montag, 26.11.07 per Mail an den jeweiligen Tutor

Praktikum Grundlagen der Programmierung

Aufgabe 27 (Ü) Übersetzung von MiniJava nach MiniJVM

```
int x, y, z, res;
x = 1;
y = 2;
z = 5*x - y/2;
if (z < y)
    res = z;
else
    res = y;
write(res);
```

Übersetzen Sie das folgende MiniJava-Programm in ein MiniJVM-Programm **nach den Regeln** aus der Vorlesung. Die Variable `x` soll dabei in Speicherzelle `0`, Variable `y` in Speicherzelle `1`, Variable `z` in Speicherzelle `2` und Variable `res` in Speicherzelle `3` stehen.

Aufgabe 28 (Ü) Waschmaschine

Modellieren Sie mit UML und implementieren Sie mit Java geeignete Datenstrukturen zur Simulation einer Waschmaschine. In unserem Szenario finden Sie:

- Münzen (mit Wert),
- Waschmaschinen (mit Platz für 10 Socken),
- Socken (mit einer Zahl als Farbwert).

Implementieren Sie die Funktionalität der Waschmaschine so, dass sie die Beladung mit maximal 10 Socken zulässt, einen Einwurf von Münzen im Wert von 2 Einheiten erfordert und nach dem Waschgang Socken zurückgibt, die als Farbwert den durchschnittlichen Farbwert aller gewaschenen Socken haben. Testen Sie Ihre Implementation mit einem geeigneten `main()`-Programm.

Aufgabe 29 (Ü) Matrix (optional)

Definieren Sie sich eine Klasse zur Repräsentation von $m \times n$ Matrizen. Implementieren Sie geeignete Konstruktoren zur Erzeugung einer Matrix und definieren Sie zu deren Darstellung eine `toString()`-Methode. Zudem wäre es sinnvoll geeignete `get-` und `set-` Methoden zu schreiben, um die Matrix zu manipulieren.

Fügen Sie Ihrer Klasse eine `add`-Methode hinzu, welche 2 Matrizen miteinander addiert.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \dots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \dots & a_{2n}+b_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \dots & a_{mn}+b_{mn} \end{pmatrix}$$

Aufgabe 30 (H) Übersetzung von MiniJava nach MiniJVM (3 Punkte)

Übersetzen Sie das nachfolgende MiniJava-Programm in ein MiniJVM-Programm **nach den Regeln** aus der Vorlesung.

```
int n, w, sum, i;
n = read();
w = 0;
sum = 0;
i = 1;
while (sum < n) {
    w = w + 1;
    sum = sum + i;
    i = i + 2;
}
write(w);
```

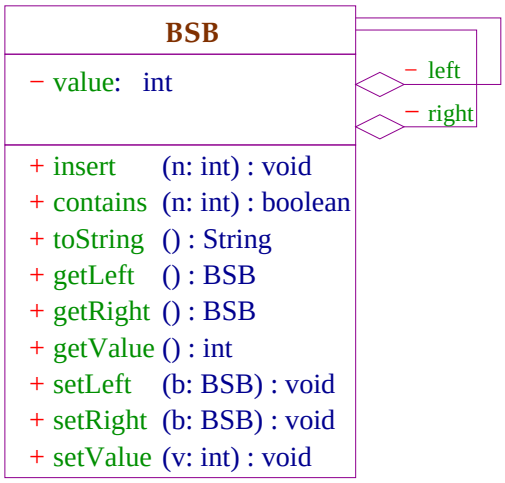
Aufgabe 31 (H) Kreise (1+5+2 Punkte)

Ein Kreis soll durch seinen Mittelpunkt (x- und y-Koordinate) und seinen Radius definiert werden.

a) Definieren Sie eine geeignete Datenstruktur zur Repräsentation von Kreisen.
b) Implementieren Sie folgende Methoden für Kreis-Objekte:
   - `getDiameter`: gibt den Durchmesser des Kreises zurück.
   - `move`: bewegt den Kreis an eine neue Koordinatenposition.
   - `getDistance`: berechnet die Distanz zwischen dem Kreis und einem anderen Kreis.
   - `hasIntersection`: prüft, ob der Kreis einen anderen Kreis schneidet.
   - `smaller`: prüft, ob der Kreis kleiner als ein anderer Kreis ist.

   Testen Sie Ihre Implementation mit einem geeigneten `main()`-Programm.
c) Dokumentieren Sie Ihre Klasse als UML-Diagramm.

Aufgabe 32 (H) Binärer Suchbaum (6 Punkte)



Ein *Binärbaum* ist ein Baum, in dem alle Knoten nicht mehr als 2 Kinder haben. Ein *binärer Suchbaum* (BSB) ist ein Binärbaum, der folgende Ordnungseigenschaft erfüllt:
Für jeden Knoten des Baums gilt, dass sein eigener Wert nicht kleiner ist als die Werte aller Knoten in seinem linken Teilbaum, aber echt kleiner ist als die Werte aller Knoten in seinem rechten Teilbaum.

Implementieren Sie eine Klasse, mit der man BSBs für ganze Zahlen repräsentieren kann und

a) die Methoden `BSB getLeft()`, `BSB getRight()`, **`int`** `getValue()`, die den rechten bzw. linken Teilbaum und den Wert der Wurzel (`value`) liefern, sowie die Methoden **`void`** `setLeft(BSB b)`, **`void`** `setRight(BSB b)`, **`void`** `setValue(`**`int`**` v)` zum Setzen der entsprechenden Werte,
b) eine `toString()`-Methode zur Darstellung des binären Suchbaums,
c) die Methode **`void`** `insert(`**`int`**` n)`, die - unter Erhalt der Ordnungseigenschaft - einen neuen Knoten mit dem Wert `n` in den BSB einfügt und
d) die Methode **`boolean`** `contains(`**`int`**` n)`, die ausgibt ob die Zahl `n` im BSB enthalten ist.

Testen Sie Ihre Implementierung an Hand eines geeigneten `main`-Programms.

Anhang

Regeln aus der Vorlesung zur Übersetzung von MiniJava nach MiniJVM:

prog	=	ALLOC n Übersetzung von ss HALT — sofern `prog` aus einer Deklaration von n Variablen, gefolgt von der Statement-Folge ss besteht.
x	=	LOAD i — sofern x die Variable mit Adresse i ist.
c	=	CONST c — sofern c eine Konstante ist.
$expr_1$ + $expr_2$	=	Übersetzung von $expr_1$ Übersetzung von $expr_2$ ADD — Analog für die anderen Operatoren (SUB, MUL, DIV, MOD, AND, OR, LESS, LEQ, EQ, NEQ, GT, GEQ)
-expr	=	Übersetzung von expr NEG — Analog für NOT
x = expr;	=	Übersetzung von expr STORE i — sofern x die Variable mit Adresse i ist.
x = read();	=	READ STORE i — sofern x die Variable mit Adresse i ist.
write(expr);	=	Übersetzung von expr WRITE
if (cond) stmt	=	Übersetzung von cond FJUMP A Übersetzung von stmt A: ...
if (cond) $stmt_1$ else $stmt_2$	=	Übersetzung von cond FJUMP A Übersetzung von $stmt_1$ JUMP B A: Übersetzung von $stmt_2$ B: ...
while (cond) stmt	=	A: Übersetzung von cond FJUMP B Übersetzung von stmt JUMP A B: ...
$stmt_1$ ... $stmt_k$	=	Übersetzung von $stmt_1$ ... Übersetzung von $stmt_k$

UML Referenz

```
public class A {
    private int i;
    public double d;
    public B b;
    public String s;
    public boolean[] ba;

    public A(int i, char c) {
        ...
    }

    public static String sm() {
        ...
    }

    public void m1(B b) {
        ...
    }

    public A m2() {
        ...
    }


}

public class B {
    public String sb;
}
```