Semestralklausur Einführung in die Programmierung, WS 2005/06, 6.2.2006	Seite 1/6
Name, Vorname, Matrikelnummer:	Unterschrift:

Gruppe A

1 Grundlagen (5+5 Punkte)

a) Schreiben Sie eine Klassenmethode mit einer ganzen Zahl n als Parameter, von der Sie annehmen dürfen, dass sie nicht negativ ist. Die Methode soll die Werte 2^{-i} für $i = 0, \ldots, n$ ausgeben, für $n = 3$ also so etwas:

```
1.0
0.5
0.25
0.125
```

Die Methode soll kein Funktionsergebnis zurückliefern. Denken Sie daran, dass Java keinen Operator zum Potenzieren hat: Sie werden wohl multiplizieren bzw. dividieren müssen.

```
static void tabelleA(int n) {
    double prod = 1.0;
    for (int i=0; i<=n; i++) {
        System.out.println(prod);
        prod = prod/2.0;
    }
}
```

b) Schreiben Sie nun eine Klassenmethode mit einem Parameter x vom Typ `double`. Das Funktionsergebnis soll ganzzahlig sein:

- im Fall $0 < x \leq 1$ soll es die größte ganze Zahl i sein, für die $2^i \leq x$ gilt (die dürfte wohl kleiner oder gleich Null sein),
- für alle anderen Fälle soll es 1 sein.

So soll z.B. für $x = 0.4$ der Wert -2 rauskommen ($2^{-2} = 0.25 \leq 0.4 < 0.5 = 2^{-1}$), für $x = 1$ wäre das Ergebnis 0, für $x = 7$ oder $x = -2$ wäre es 1.

```
static int logA(double x) {
    if (x>0.0 && x<=1.0) {
        int i = 0;
        double prod = 1.0;
        while (prod>x) {
            i++;
            prod = prod/2.0;
        }
        return -i;
    } else {
        return 1;
    }
}
```

2 Plus und Mal (8 Punkte)

Wir hatten folgende Klassenmethoden zur Addition zweier nichtnegativer Zahlen und zur Multiplikation zweier positiver Zahlen:

```
static int plus(int n, int m) {
    if (m == 0) {
        return n;
    } else {
        return plus(n + 1, m - 1);
    }
}
static int mal(int n, int m) {
    if (m == 1) {
        return n;
    } else {
        return plus(n, mal(n, m - 1));
    }
}
```

Schreiben Sie zwei Klassenmethoden, die genau dieselben Werte berechnen, aber die rekursiven Methodenaufrufe durch `for`-Schleifen ersetzen.
Von den arithmetischen Operationen ist dabei nur das Erhöhen um Eins (`x+1` oder `x++`) erlaubt; Methodenaufrufe sind ganz verboten!

```
static int plus_for(int n, int m) {
    int sum = n;
    for (int i=0; i<m; i++) sum++;
    return sum;
}
static int mal_for(int n, int m) {
    int prod = 0;
    for (int i=0; i<m; i++)
        for (int j=0; j<n; j++) prod++;
    return prod;
}
```

3 Felder (6 Punkte)

Schreiben Sie eine Klassenmethode mit einem Feld von ganzen Zahlen v und einer ganzen Zahl x als Parameter und ganzzahligem Ergebnis, das angeben soll, wie oft die Zahl x am Anfang des Feldes in ununterbrochener Reihenfolge auftritt. Z.B. ergibt sich für $x = 1$ folgendes:

- v mit Komponenten $1, 1, 1, 3, 4$ ergibt 3
- v mit Komponenten $1, 1, 1$ ergibt 3
- v mit Komponenten $3, 2, 1$ ergibt 0

(Die übliche Technik, die Länge eines Feldes zu bestimmen, ist übrigens kein Methodenaufruf und fällt deshalb auch nicht unter das Verbot von Fremdmethodenaufrufen)

```
static int nur_x(int[] v, int x) {
    int l = 0;
    while (l<v.length && v[l] == x) l++;
    return l;
}
```

4 Klassen und Objekte (4+8+4 Punkte)

Nun schreiben wir eine Klasse für Kaffeetrinker: mit ihr soll man seine Kaffeetassen simulieren können. Den Programmtext der folgenden Teilaufgaben a) und b) stellen wir uns dabei in den Klassenrahmen

```
public class Tasse {
// Hier denken wir die Deklarationen hin
}
```

eingebaut vor, der dazu nicht abgeschrieben werden muss.
Sie dürfen davon ausgehen, dass alle vorkommenden Kaffeemengen nicht negativ sind.

a) Deklarieren Sie Instanzvariablen vom Typ `double` für das Fassungsvermögen und den aktuellen Inhalt einer Tasse (Einheit sei ein Milliliter) und schreiben Sie einen Konstruktor, der einen Wert für das Fassungsvermögen als Parameter bekommt, dieses belegt und den Tasseninhalt auf 0 setzt.

```
double maxInhalt;
double inhalt;

Tasse(double soGross) {
    maxInhalt = soGross;
    inhalt = 0.0;
}
```

b) Schreiben Sie nun zwei Instanzmethoden zum Füllen der Tasse. Die erste bekomme eine Kaffeemenge als Parameter und simuliert das Einschenken aus einer Kanne: der Inhalt wird um die angegebene Menge erhöht, wenn diese noch in die Tasse passt, ansonsten bis zum Fassungsvermögen (der Rest läuft über).

Die zweite bekommt eine Tasse als Parameter (wir dürfen davon ausgehen, dass sie nicht das aktuelle Objekt, dessen Instanzmethode aufgerufen wird, selber ist), der Inhalt dieser Tasse soll in die aktuelle Tasse umgefüllt werden. Reicht das Fassungsvermögen der aktuellen Tasse nicht aus, soll nur soviel umgefüllt werden wie reinpasst und in der anderen Tasse bleibt ein Rest.

```
public void rein(double dazu) {
    inhalt = inhalt + dazu;
    if (inhalt > maxInhalt)
        inhalt = maxInhalt;
}
```

```
public void rein(Tasse t) {
    double dazu = t.inhalt;
    if (inhalt + dazu > maxInhalt)
        dazu = maxInhalt - inhalt;
    inhalt = inhalt + dazu;
    t.inhalt = t.inhalt - dazu;
}
```

c) Geben Sie nun Anweisungen an, wie sie z.B. in einer Methode einer anderen Klasse stehen könnten, die

- Objekte für drei Tassen mit Fassungsvermögen von 200, 300 bzw. 400ml erzeugen,
- in diese Tassen 150, 400 bzw. 200ml einfüllen und
- den Inhalt der ersten und der zweiten Tasse in die dritte (die größte) Tasse umfüllen.

```
Tasse t1, t2, t3;
t1 = new Tasse(200.0);
t2 = new Tasse(300.0);
t3 = new Tasse(400.0);
t1.rein(150.0);
t2.rein(400.0);
t3.rein(200);
t3.rein(t1);
t3.rein(t2);
```