Lehrstuhl für Praktische Informatik IV
Holger Füßler
A5, 6, Raum B 219
68131 Mannheim
Telefon: (0621) 181–2605
Email: fuessler@informatik.uni-mannheim.de

Robert Schiele
B6, 29, Raum C0.04
68131 Mannheim
Telefon: (0621) 181–2214
Email: rschiele@uni-mannheim.de

UNIVERSITÄT MANNHEIM

Praktische Informatik I — Scheinklausur
Wintersemester 2005/2006 — 9. Februar 2006

Name: _____

Matrikelnummer: _____

Studienfach: _____

Semester: _____

Tutor: _____

Hinweise

1. Überprüfen Sie bitte Ihr Klausurexemplar auf Vollständigkeit (18 Seiten).
2. Bearbeiten Sie die Aufgaben *ausschließlich* auf dem Aufgabenblatt der jeweiligen Aufgabe.
3. Schreiben Sie auf jedes Blatt, das bewertet werden soll, oben ihren Namen und ihre Matrikelnummer.
4. Verwenden Sie nur dokumentenechte Stifte (z. B. keinen Bleistift) und keine roten Stifte.
5. Es sind keine Hilfsmittel zugelassen.
6. Bearbeitungszeit: 90 Minuten.

Korrekturzeile — Bitte *nicht* ausfüllen!

1	2	3	4	5	gesamt
20	15	15	20	20	90

Name: Matrikelnummer:

Name: Matrikelnummer:

Aufgabe 1 — 20 Punkte

Aufgabe 1 a) — 2 Punkte

Beschreiben Sie kurz, wie man in Java die Sichtbarkeit von Instanzvariablen beeinflussen kann. Nennen Sie die verschiedenen Sichtbarkeitsstufen und ordnen Sie sie nach ihrer Restriktivität. Kennzeichnen Sie hierbei die restriktivste Stufe.

Name: Matrikelnummer:

Aufgabe 1 b) 4 Punkte

Im folgenden sehen sie die Deklaration von zwei Java-Klassen. Was gibt Java aus, wenn ein Objekt vom Typ `XX` erzeugt wird?

```
1  public class X {
       private int a = 1;
       public int b = 9;
       protected int c = 13;
       static public int D = 43;
6      public int getA() { return a; }
       public void setA(int a) { this.a = a; }
   }
   public class XX extends X {
       private static int D = 99;
11     private int b = 12;
       private int a = 42;
       XX() {
           D++;
           setA(19);
16         b = super.b;
           System.out.println("Werte: " + a + " : " + b + " : " + c + " : " + X.D);
       }
   }
```

Name: Matrikelnummer:

Aufgabe 1 c) 7 Punkte

Im Folgenden finden Sie die Deklarationen mehrerer Java-Klassen und Interfaces. Skizzieren Sie zunächst die Vererbungsbeziehung mit Kästchen und "Ist-Subtyp-von"-Pfeilen (also `XX` → `X` nach dem obigen Beispiel). Nennen Sie dann für jeden Methodenaufruf der mit $Z1$-$Z5$ gekennzeichneten Zeilen die Zeilennummer der Methode, die durch die Aufrufe aufgerufen wird.

```
1  public class Obst {
       public static void pfluecken() { /* ... */ }
       public String getGeschmack() { /* ... */ }
       public static void main(String[] args) {
           Obst o1 = (Obst) new Pflaume();
6          Melone o2 = (Melone) new Wassermelone();
           Wassermelone o3 = (Wassermelone) o2;
           o1.pfluecken();       // Z1
           o1.getGeschmack(); // Z2
           o2.pfluecken();       // Z3
11         o2.getGeschmack(); // Z4
           o3.pfluecken();       // Z5
       }
   }
   public class Pflaume extends Obst {
16     public static void pfluecken() { /* ... */ }
       public String getGeschmack() { return new String("pflaumig");}
   }
   public class Melone extends Obst {
       public String getGeschmack() { return new String("saftig");}
21 }
   public class Wassermelone extends Melone {
       public static void pfluecken() { /* ... */ }
       public String getGeschmack() { return new String("waessrig");}
   }
```

Aufgabe 1 d) 5 Punkte

Schreiben Sie eine Methode `double[][] initMatrix(int zeilen, int spalten)`, die eine *zeilen* × *spalten* double-Matrix mit Einsen initialisiert und zurück gibt.

Was müsste man an Ihrem Programm ggf. ändern, um eine unterschiedliche Anzahl von Spalten pro Zeile zu unterstützen?

Aufgabe 1 e) 2 Punkte

Nennen Sie zwei wesentliche Unterschiede zwischen einem Interface und einer abstrakten Klasse (in Java). Erklären Sie kurz.

Name: Matrikelnummer:

Aufgabe 2 — 15 Punkte

Aufgabe 2 a) — 4 Punkte

Zeichnen Sie je ein Struktogramm für die Java-Schleifentypen `for(;;)` und `while()`. Zeigen Sie dabei, welcher Teil der Anweisung welchem Element des Struktogramms entspricht.

Name: Matrikelnummer:

Aufgabe 2 b) 9 Punkte

Im Folgenden geht es um die Laufzeit-Komplexität von gegebenen Java-Programmen. Geben Sie zunächst für jede der u.a. Methoden die exakte Formel für die Laufzeit an und leiten sie dann die zugehörige (sinnvolle) Komplexitätsklasse in $O()$-Notation ab.

Nehmen Sie hierfür folgende Laufzeitkosten an.

c_z	:	Kosten für eine Zuweisung (ggf. inkl. Initialisierung)
c_c	:	Kosten für einen Vergleich
c_a	:	Kosten für eine arithmetische Operation, auch Inkrement / Dekrement
c_i	:	Kosten für einen Methoden-Aufruf

Operatoren, die sowohl rechnen als auch zuweisen sind wie ihre "ausgeschriebenen" Pendants zu bewerten, also z.B. `i++` wie `i = i + 1`. Die übrigen Kosten können vernachlässigt werden. Ferner können Sie annehmen, dass $n, m \geq 1$ ist und dass gilt

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

```
public static void methA(int n, int m) {
    int k = 0;
    for (int i = 1; i < n; i++){
        int j = i;
        while (--j > 0)
            k = (i*j*m)/3;
    }
}

public static void methB(int n, int m) {
    int k = 0;
    for (int i = 1; i < n; i++){
        int j = m;
        while (--j > 0)
            k = (i*n*j)/3;
    }
}

public static void methC(int n, int m) {
    int k = 0;
    for (int i = 1; i < n; i++){
        int j = 1;
        while ( (j *= 2) < (m + 2))
            k = (i*j*m)/3 - 4;
}
```

Name: Matrikelnummer:

Name: Matrikelnummer:

Aufgabe 2 c) 2 Punkte

Erklären Sie kurz den Unterschied zwischen den Laufzeitklassen P und NP und geben Sie mit Begründung an, welche Klasse eine Teilmenge der anderen ist.

Name: Matrikelnummer:

Aufgabe 3 — 15 Punkte

Aufgabe 3 a) — 5 Punkte

Gegeben sei ein deterministischer endlicher Automat über dem Alphabet $\{a, b\}$ mit fünf Zuständen $z_1 \dots z_5$, wobei z_1 sowohl Start- als auch der einzig akzeptierte Endzustand ist. Die erste Spalte beschreibt den Ausgangszustand, die rechten beiden den Zielzustand in Abhängigkeit des nächsten Symbols.

	a	b
z_1	z_2	z_5
z_2	z_3	z_5
z_3	z_4	z_5
z_4	z_1	z_5
z_5	z_2	z_4

Zeichnen Sie den Automaten.

Aufgabe 3 b) 10 Punkte

Entwickeln Sie einen deterministischen endlichen Automaten, der über dem Alphabet $\{a, b, c\}$ genau diejenigen Wörter akzeptiert, bei denen auf jedes a direkt ein b folgt und der insgesamt eine gerade Anzahl von b's enthält.

Name: Matrikelnummer:

Aufgabe 4 — 20 Punkte

Aufgabe 4 a) — 4 Punkte

Erläutern Sie allgemein Vor- und Nachteile von rekursiven gegenüber iterativen Algorithmen.

Aufgabe 4 b) — 4 Punkte

Betrachten Sie im folgenden die Methode `quicksort`, die wie folgt definiert ist:

```
void quicksort(int a[], int p, int r) {
    if (p < r) {
        int q = partition(a, p, r);
        quicksort(a, p, q);
        quicksort(a, q + 1, r);
    }
}
```

Dabei ist die Methode `partition` derart konstruiert, dass sie eine Zahl q zurückgibt, so dass gilt $p \leq q < r$. Ansonsten kann allgemein nichts über die zurückgegebene Zahl ausgesagt werden.

Zeichnen Sie für den Methodenaufruf `quicksort(a, 0, 3)` einen Rekursionsbaum. Den Rückgabewert von `partition` dürfen Sie dabei innerhalb der gegebenen Grenzen jeweils frei wählen. Benennen Sie die Knoten im Rekursionsbaum durch die Parameter p und r des jeweiligen Aufrufs.

Aufgabe 4 c) 8 Punkte

Zeigen Sie, dass die folgende Variante sich gleich verhält als die obige Implementation:

```
void quicksort(int a[], int p, int r) {
    while (p < r) {
        int q = partition(a, p, r);
        quicksort(a, p, q);
        p = q + 1;
    }
}
```

Aufgabe 4 d) 4 Punkte

Zeichnen Sie für die neue Methode wiederum für den Methodenaufruf `quicksort(a, 0, 3)` einen Rekursionsbaum wie oben.

Name: Matrikelnummer:

Aufgabe 5 — 20 Punkte

Aufgabe 5 a) — 5 Punkte

Schreiben Sie in Scheme eine Methode `expo`, die für die übergebenen Parameter a und n als Ergebnis a^n liefert. Sie dürfen annehmen, dass a, n ganzzahlig sind und $n \geq 0$ gilt.

Name: Matrikelnummer:

Aufgabe 5 b) 10 Punkte

In Scheme sei die Prozedur `fold` definiert als:

```
(define (fold fun item lst)
  (if (null? lst)
      item
      (fun (car lst)
           (fold fun item (cdr lst)))))
```

Dabei prüft die Prozedur `null?`, ob eine Liste leer ist, `car` hat den Wert des ersten Elements einer gegebenen Liste und `cdr` hat den Wert einer Liste, die aus allen Elementen der Liste außer dem ersten besteht.

Die Prozedur `fold` wird nun folgendermaßen aufgerufen:
`(fold + 0 '(1 2 3 4))`

Werten Sie diesen Ausdruck schrittweise aus. Ob Sie dabei in applikativer oder normaler Reihenfolge vorgehen, bleibt Ihnen überlassen.

Hinweis: `'(1 2 3 4)` bezeichnet eine Liste mit den Elementen `1`, `2`, `3` und `4`. Entsprechend ist `'()` eine leere Liste.

Aufgabe 5 c) 5 Punkte

Was macht die Prozedur `fold`?

Nehmen Sie an, es existiere eine Prozedur `(insert x l)`, welche das Element `x` in die sortierte Liste `l` derart einfügt, dass die Sortierung erhalten bleibt. Wie können Sie dann leicht mit Hilfe dieser und der Prozedur `fold` eine Methode `sortnum` implementieren, welche eine ihr übergebene Liste von Zahlen in aufsteigender Reihenfolge sortiert?

Name: Matrikelnummer: