Diplomvorprüfung / Zwischenprüfung
Informatik I - Programmierung
25. 2. 2004

Vorname: ______________________________

Nachname: ______________________________

Matrikelnummer: ______________________________

Studiengang (bitte ankreuzen):

○ Informatik Diplom ○ Informatik Lehramt

○ Sonstige: ______________________________

- Schreiben Sie bitte auf jedes Blatt **Vorname**, **Name** und **Matrikelnummer**.
- Geben Sie Ihre Antworten bitte in lesbarer und verständlicher Form an. Schreiben Sie bitte nicht mit roten Stiften.
- Bitte beantworten Sie die Aufgaben auf den **Aufgabenblättern**. Benutzen Sie ggf. auch die Rückseiten der **zur jeweiligen Aufgabe gehörenden** Aufgabenblätter.
- Antworten auf anderen Blättern können nur berücksichtigt werden, wenn **Name, Matrikelnummer und Aufgabennummer** deutlich darauf erkennbar sind.
- Was nicht bewertet werden soll, kennzeichnen Sie bitte durch **Durchstreichen**.
- Werden Täuschungsversuche beobachtet, so wird die Klausur mit **nicht bestanden** bewertet.
- Geben Sie bitte am Ende der Klausur **alle Blätter zusammen mit den Aufgabenblättern ab**.

	Anzahl Punkte	Erreichte Punkte
Aufgabe 1	14	
Aufgabe 2	12	
Aufgabe 3	14	
Aufgabe 4	26	
Aufgabe 5	18	
Aufgabe 6	16	
Summe	100	
Note	-	

Vorname	Name	Matr.-Nr.

Aufgabe 1 (Programmanalyse, 8 + 6 Punkte)

a) Geben Sie die Ausgabe des Programms für den Aufruf `java M` an. Schreiben Sie hierzu jeweils die ausgegebenen Zeichen hinter den Kommentar "`OUT:`".

```
public class A {
  public int z = 1;

  public void f(int x) {
    z = x + z;
  }
}
public class B extends A {
  public static int n = 0;
  public B() {
      n++;
  }
  public void f(int x) {
    n = x + n;
  }
}
public class M {
  public static void main(String[] args) {
    A a = new A();
    a.f(5);
    System.out.println(a.z+", "+B.n); // OUT:
    B b = new B();
    b.f(5);
    System.out.println(b.z+", "+B.n); // OUT:
    a = b;
    b.z = a.z + 1;
    a.f(5);
    System.out.println(a.z+", "+B.n); // OUT:
    a.z = b.z + 1;
    b.f(5);
    System.out.println(b.z+", "+B.n); // OUT:
  }
}
```

b) In der Klasse `B` wird die Methode `f` durch folgende Methode ersetzt. Welche drei Fehler treten beim Compilieren auf? Begründen Sie Ihre Antwort kurz.

```
  protected void f(int a) {
    A a = new B();
    B b = new B();
    b.z = new A().z;
    A.n = 1;
  }
```

Vorname	Name	Matr.-Nr.

Vorname	Name	Matr.-Nr.

Aufgabe 2 (Verifikation, 10 + 2 Punkte)

Der Algorithmus P berechnet den ganzzahligen Rest der Division zweier Zahlen `n` und `m` (“`n mod m`”).

a) Vervollständigen Sie die folgende Verifikation des Algorithmus P im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur eine Zusicherung stehen, wenn dies aus einer Regel des Hoare-Kalküls folgt.

Algorithmus: P
Eingabe: $\mathtt{n}, \mathtt{m} \in \mathbb{N} = \{0, 1, 2, \ldots\}$
Ausgabe: `res`
Vorbedingung: $\mathtt{n} \geq 0$
Nachbedingung: $\mathtt{res} = \mathtt{n} \bmod \mathtt{m}$

$\langle\ \mathtt{n} \geq 0\ \rangle$

$\langle$ ____________________ $\rangle$

```
res = n;
```

$\langle$ ____________________ $\rangle$

$\langle$ ____________________ $\rangle$

```
while (res >= m) {
```

$\langle$ ____________________ $\rangle$

$\langle$ ____________________ $\rangle$

```
  res = res - m;
```

$\langle$ ____________________ $\rangle$

```
}
```

$\langle$ ____________________ $\rangle$

$\langle\ 0 \leq \mathtt{res} < \mathtt{m}\ \wedge\ \exists \mathtt{j} : \mathtt{n} = \mathtt{j} \cdot \mathtt{m} + \mathtt{res}\ \rangle$

$\langle\ \mathtt{res} = \mathtt{n} \bmod \mathtt{m}\ \rangle$

Hinweis: “`n mod m`” ist die Zahl $\mathtt{r}$ mit $0 \leq \mathtt{r} < \mathtt{m}$ und $\exists \mathtt{j} : \mathtt{n} = \mathtt{j} \cdot \mathtt{m} + \mathtt{r}$.
Hierbei steht “$\exists \mathtt{j} : \mathtt{n} = \mathtt{j} \cdot \mathtt{m} + \mathtt{r}$” für: “es existiert eine ganze Zahl $\mathtt{j}$, so dass $\mathtt{n} = \mathtt{j} \cdot \mathtt{m} + \mathtt{r}$”.
Beispielsweise ist $7 \bmod 3 = 1$, da $7 = 2 \cdot 3 + 1$ gilt.

Vorname	Name	Matr.-Nr.

b) Untersuchen Sie den Algorithmus P auf seine Terminierung.

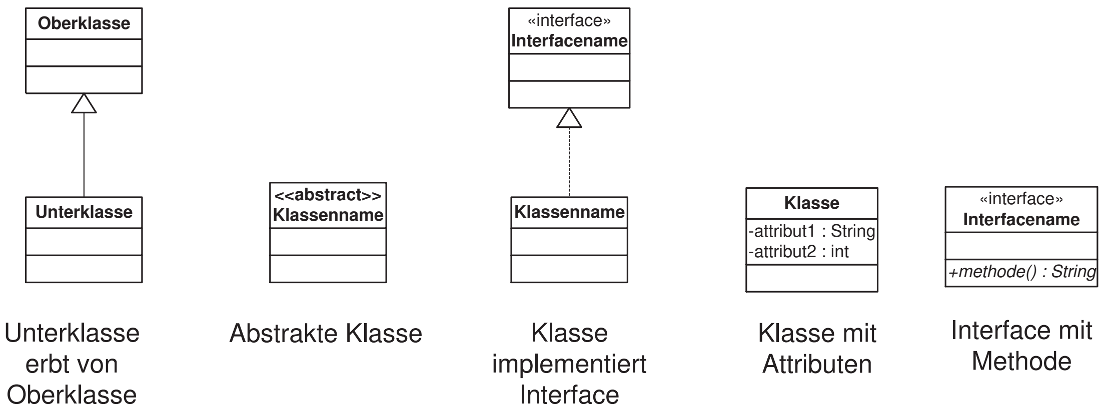
Vorname	Name	Matr.-Nr.

Aufgabe 3 (Datenstrukturen in Java, 6 + 8 Punkte)

Ihre Aufgabe ist es, eine objektorientierte Datenstruktur zur Verwaltung von Datenspeichern zu entwerfen. Bei der vorangehenden Analyse wurden folgende Eigenschaften der verschiedenen Datenspeicher ermittelt:

- Eine CD ist ein optischer Speicher, der über eine bestimmte Kapazität verfügt und eine Lead-Out-Position besitzt (d.h., eine Zeitangabe als Gleitkommazahl), die das Ende der Aufzeichnung kennzeichnet. Darüberhinaus kann eine CD im Single-Format vorliegen oder nicht.
- Eine DVD ist ein optischer Speicher mit einer bestimmten Kapazität und einer Lead-Out-Position. Darüberhinaus besitzt eine DVD eine bestimmte Anzahl von Schichten.
- Eine Diskette ist ein magnetischer Speicher mit einer bestimmten Kapazität, der durch eine Materialart (dargestellt durch einen `String`) gekennzeichnet ist. Darüberhinaus kann bei einer Diskette ein Schreibschutz aktiviert sein oder nicht.
- Ein Bandspeicher ist ein magnetischer Speicher mit einer bestimmten Kapazität, der durch eine Materialart und die Bandlänge gekennzeichnet ist.
- Alle Datenspeicher außer dem Bandspeicher sind Datenspeicher mit wahlfreiem Zugriff. Für Datenspeicher mit wahlfreiem Zugriff soll es möglich sein, ein Inhaltsverzeichnis (als `String`) berechnen zu lassen.

a) Entwerfen Sie unter Berücksichtigung der Prinzipien der Datenkapselung eine geeignete Klassenhierarchie für die oben aufgelisteten Arten von Datenspeichern. Achten Sie darauf, dass gemeinsame Merkmale in (evtl. abstrakten) Oberklassen zusammengefasst werden. Notieren Sie Ihren Entwurf graphisch und verwenden Sie dazu die folgende Notation:



Geben Sie für jede Klasse ausschließlich den jeweiligen Namen und die Namen ihrer Attribute an. Methoden von Klassen müssen nicht angegeben werden. Geben Sie für jedes Interface ausschließlich den jeweiligen Namen sowie die Namen seiner Methoden an.

Vorname	Name	Matr.-Nr.

Vorname	Name	Matr.-Nr.

b) Implementieren Sie in Java eine Methode `getKapazitaeten`, die ein Array von Datenspeichern übergeben bekommt. Die Methode soll ein zwei-elementiges Array zurückliefern, welches die Gesamtkapazität aller optischen Speicher und aller magnetischen Speicher enthält. Die einzelnen Positionen des Arrays sind dabei wie folgt festgelegt:

- Position 0: Gesamtkapazität aller optischen Speicher
- Position 1: Gesamtkapazität aller magnetischen Speicher

Gehen Sie dabei davon aus, dass für alle Attribute geeignete Selektoren existieren und verwenden Sie für den Zugriff auf die benötigten Attribute die passenden Selektoren. Kennzeichnen Sie die Methode mit dem Schlüsselwort “`static`”, falls angebracht.

Vorname	Name	Matr.-Nr.

Aufgabe 4 (Programmierung in Java, 5 + 4 + 2 + 7 + 8 Punkte)

In dieser Aufgabe sollen Programme unter Beachtung der Prinzipien der Datenkapselung entworfen werden. Kennzeichnen Sie Methoden mit dem Schlüsselwort “`static`”, falls angebracht.

a) Gegeben ist das folgende Interface.

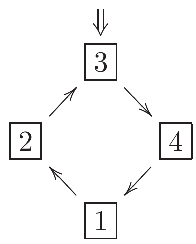
```
public interface Test {
    public boolean check(Object obj);
}
```

Falls `t` ein Objekt vom Typ `Test` ist und `obj` ein beliebiges `Object` ist, so sagen wir, dass “`obj` den Test `t` erfüllt”, falls `t.check(obj)` den Wert `true` liefert. Schreiben Sie zwei Klassen `SmallerTest` und `InverseTest` in Java, die beide das Interface `Test` implementieren. Objekte der Klasse `SmallerTest` kapseln eine ganze Zahl. Falls `s` ein Objekt der Klasse `SmallerTest` ist, so soll `obj` genau dann den Test `s` erfüllen, wenn `obj` eine ganze Zahl vom Typ `Integer` ist, deren Wert kleiner als die in `s` gekapselte Zahl ist. Hierbei können Sie die Methode `public int intValue()` der Klasse `Integer` verwenden.

Objekte der Klasse `InverseTest` kapseln ein Objekt vom Typ `Test`. Falls `i` ein Objekt der Klasse `InverseTest` ist, so sollen genau die Objekte `obj` den Test `i` erfüllen, die den in `i` gekapselten Test nicht erfüllen. Sie brauchen keine Konstruktoren für die Klassen zu schreiben.

Vorname	Name	Matr.-Nr.

b) Die folgenden beiden Klassen dienen zur Darstellung von *Ringen* (zyklischen Listen). In diesen Ringen können beliebige Objekte (vom Typ `Object`) gespeichert werden. Ein Beispiel ist der folgende Ring `r1`:



Die einzelnen Ringelemente werden durch Objekte der Klasse `Element` implementiert, die jeweils ein Attribut `value` für den Wert und ein Attribut `next` für das nächste Element des Rings besitzen. Ein Objekt der Klasse `Ring` besitzt nur ein Attribut `position`, das auf das Element an der *aktuellen* Position zeigt (im Beispiel durch einen Pfeil "$\Downarrow$" gekennzeichnet). Für den Ring `r1` wäre `r1.position` das Element `e` mit dem `value` 3. Es muss sicher gestellt sein, dass das "letzte" Ring-Element wieder auf das erste Element des Rings verweist. Wenn man also vom Element `e` aus viermal dem `next`-Verweis folgt, so erhält man wieder dasselbe Element, d.h., es gilt `e.next.next.next.next == e`. Die Klassen `Ring` und `Element` sind wie folgt implementiert. Der Konstruktor `Ring()` dient hierbei zur Erzeugung des leeren Rings.

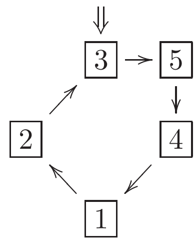
```
public class Element {

    private Object value;
    private Element next;

    public Element(Object value, Element next) {
        this.next = next;
        this.value = value;
    }

    public void setNext(Element next) {
        this.next = next;
    }

    public Element getNext() {
        return next;
    }

    public Object getValue() {
        return value;
    }
}

public class Ring {

    private Element position;

    public Ring() {
        position = null;
    }
}
```

Vorname	Name	Matr.-Nr.

Ergänzen Sie die Klasse `Ring` um eine Methode

```
public void insert(Object v) { ... }
```

die ein neues `Element` mit dem Wert `v` in den Ring einfügt und zwar an die Stelle hinter der aktuellen Position. Sofern der Ring nicht leer war, bleibt die aktuelle Position unverändert. Falls `v` der Wert `5` (als `Integer`-Objekt) ist, so verändert der Aufruf `r1.insert(v)` den Ring `r1` also zu folgendem Ring `r2`.

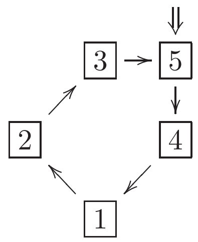


Vorname	Name	Matr.-Nr.

c) Ergänzen Sie die Klasse `Ring` um eine Methode

```
public void rotate() { ... }
```

die die aktuelle Position um eins weitersetzt. Die Anwendung von `rotate` auf `r2` ergibt also den folgenden Ring `r3`.



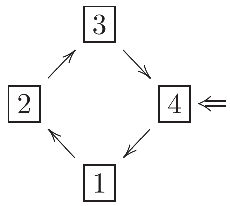
Die Anwendung von `rotate` auf einen leeren Ring oder einen Ring mit nur einem Element ändert den Ring nicht.

Vorname	Name	Matr.-Nr.

d) Ergänzen Sie die Klasse `Ring` um eine Methode

```
public void delete() { ... }
```

die das Element an der aktuellen Position löscht. Die neue aktuelle Position ist dann die darauffolgende Position. Die Anwendung von `delete` auf `r3` ergibt also den folgenden Ring `r4`.

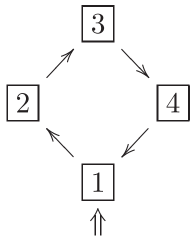


Vorname	Name	Matr.-Nr.

e) Ergänzen Sie die Klasse `Ring` um eine Methode

```
public boolean find(Test t) { ... }
```

die überprüft, ob es einen Wert im Ring gibt, der den Test `t` erfüllt. Falls kein Wert des Rings den Test erfüllt, wird nichts geändert und `false` zurückgegeben. Ansonsten wird die aktuelle Position auf das erste Element gesetzt, das den Test erfüllt und `true` zurückgegeben. (Die Suche soll hierbei an der aktuellen Position beginnen.) Falls `t` der Test der Klasse `SmallerTest` ist, der überprüft, ob ein Wert kleiner als `3` ist, so hat `r4.find(t)` das Ergebnis `true` und der Ring wird wie folgt geändert:



Die Methode `find` und ggf. benötigte Hilfsmethoden sollen ohne Verwendung von Schleifen realisiert werden. Sie dürfen aber Rekursion benutzen.

Vorname	Name	Matr.-Nr.

Aufgabe 5 (Funktionale Programmierung in Haskell, 4 + 3 + 4 + 2 + 5 Punkte**)**

a) Geben Sie den allgemeinsten Typ der Funktionen `f` und `g` an, die wie folgt definiert sind. Gehen Sie hierbei davon aus, dass `5` den Typ `Int` hat.

```
f = \x -> f x

g x y = y (x ++ [5])
```

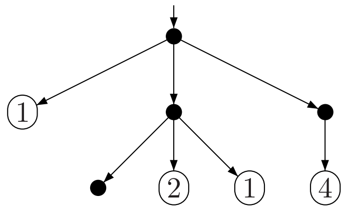
b) Bestimmen Sie das Ergebnis der Auswertung für die beiden folgenden Ausdrücke.

```
filter  (\y -> y > 0)  (map (\x -> 3 - x) [2,3,4])

(\f x -> f (f x))  (\x -> x * x)  ((\x -> 3) (\x -> 2 * x))
```

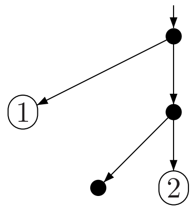
c) Schreiben Sie eine Funktion `take` in Haskell, die eine ganze Zahl `n` $\geq$ `0` und eine Liste `xs` als Eingabe bekommt und als Ergebnis die Liste der `n` ersten Elemente von `xs` liefert. Falls `xs` kürzer als `n` ist, so wird die gesamte Liste `xs` zurückgeliefert. Beispielsweise gilt `take 2 [4,5,6] = [4,5]` und `take 3 [True,False] = [True,False]`. Geben Sie auch den Typ von `take` an. (Die Funktion `take` ist in Haskell vordefiniert. Sie sollen `take` in dieser Aufgabe aber ohne Verwendung der vordefinierten Funktion selbst schreiben.)

Vorname	Name	Matr.-Nr.

d) Ein Vielwegbaum besteht aus Knoten und Kanten, wobei jeder Knoten beliebig viele Nachfolger haben kann. Werte können ausschließlich in den Blättern gespeichert werden (d.h., in Knoten ohne Nachfolger), aber es kann auch Blätter ohne darin gespeicherten Wert geben. Die folgenden beiden Bäume `b1` und `b2` sind Beispiele für Vielwegbäume.



Baum `b1`



Baum `b2`

Entwerfen Sie eine Datenstruktur für Vielwegbäume in Haskell, so dass beliebige Werte in den Blättern gespeichert werden können.

e) Ein Vielwegbaum kann auf die Breite `n` beschnitten werden, indem in jedem Knoten alle bis auf die linkesten `n` ausgehenden Kanten abgeschnitten werden. Beispielsweise ist `prune 2 b1 = b2`.

Implementieren Sie eine Funktion `prune` in Haskell, die zu einer ganzen Zahl `n` $\geq$ `0` und einem Baum `b` einen Baum liefert, der durch Beschneidung von `b` auf die Breite `n` entsteht. Hierbei dürfen Sie die Funktion `take` aus Teil c) verwenden. Geben Sie auch den Typ von `prune` an.

Vorname	Name	Matr.-Nr.

Aufgabe 6 (Logische Programmierung in Prolog, 6 + 3 + 7 Punkte)

a)
- Definieren Sie in Prolog ein zweistelliges Prädikat `sublist`, welches Teillisten berechnet. Die Aussage `sublist(L1,L2)` soll genau dann wahr sein, wenn die Liste `L2` durch Streichung beliebig vieler Elemente aus der Liste `L1` entsteht. Beispielsweise gilt `sublist([2,1,3], L)` genau dann, wenn `L` eine der Listen `[]`, `[1]`, `[2]`, `[3]`, `[2,1]`, `[2,3]`, `[1,3]` oder `[2,1,3]` ist. Sie dürfen hierbei keine vordefinierten Prädikate verwenden.

- Gibt Ihre Implementierung für die Anfrage "`?- sublist([2,1,3],L).`" Lösungen mehrfach aus, wenn Sie sich durch wiederholte Eingabe von "`;`" alle Antworten berechnen lassen?

- Geben Sie eine Anfrage an, die alle Listen `L1`, `L2` berechnet, so dass `L1` und `L2` Teillisten von `[5,2,7,8]` sind und `L1` um drei Elemente kürzer ist als `L2`. Hierbei dürfen Sie das vordefinierte zweistellige Prädikat `length` benutzen, wobei `length(L,X)` genau dann wahr ist, wenn die Liste `L` die Länge `X` hat. Beispielsweise gilt also `length([5,2,7,8],4)`.

Vorname	Name	Matr.-Nr.

b) Geben Sie den allgemeinsten Unifikator für die folgenden Term-Paare an, oder begründen Sie, warum dieser nicht existiert.

- `f(g(X),a,Y,h(Z))` und `f(Y,X,g(Z),h(b))`

- `k(Y,Z,g(Z),Y)` und `k(h(X),f(U),X,U)`

Vorname	Name	Matr.-Nr.

c) Gegeben sei das folgende Prolog-Programm.

```
p(a,X) :- p(X,c).
p(X,X) :- p(X,b).
p(c,X).
```

- Stellen Sie den Beweisbaum für die Anfrage “`?- p(Y,Z).`” graphisch dar.

Vorname	Name	Matr.-Nr.

- Geben Sie alle Lösungspaare für `Y` und `Z` bei dieser Anfrage an.

- Geben Sie an, welche Lösungen Prolog in welcher Reihenfolge findet (hierbei sollten Sie Lösungen, die mehrfach gefunden werden, auch mehrfach angeben).

- Kann man das Programm durch Vertauschen von Klauseln so umschreiben, dass Prolog alle Lösungen findet? Geben Sie das neue Programm an oder begründen Sie, wieso man durch Vertauschen der Klauseln kein solches Programm erhalten kann.