RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

LEHR- UND FORSCHUNGSGEBIET INFORMATIK II
RWTH Aachen · D-52056 Aachen · GERMANY
http://www-i2.informatik.rwth-aachen.de/lufgi2
Prof. Dr. Jürgen Giesl

Prof. Dr. Jürgen Giesl
Carsten Kern, Peter Schneider-Kamp, René Thiemann

Semestralklausur
Informatik I - Programmierung
11. 1. 2006

Vorname: ______________________________

Nachname: ______________________________

Matrikelnummer: ______________________________

Studiengang (bitte ankreuzen):

○ Informatik Diplom ○ Informatik Lehramt ○ Mathematik Diplom
○ CES ○ Werkstoffinformatik ○ Computermathematik
○ 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 oder Bleistiften.
- 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	13	
Aufgabe 3	14	
Aufgabe 4	25	
Summe	66	
Prozentzahl		

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 static int x = 1;
    public A() {
        x += 2;
    }
    public int f(int y) {
        return y + x;
    }
    public int f(double y) {
        x = 0;
        return (int)y;
    }
}


public class B extends A {
    public int y = 4;
    public B(int x) {
        super();
        y += x;
    }
    public int f(double x) {
        y += 5;
        return (int)x * y;
    }
}


public class M {
    public static void main(String[] args) {
        A a = new B(1);
        System.out.println(A.x+" "+((B)a).y);       // OUT:
        int z = a.f(3.0);
        System.out.println(((B)a).y+" "+z+" "+A.x); // OUT:
        B b = (B)a;
        z = b.f(4);
        System.out.println(b.y+" "+z+" "+A.x);      // OUT:
    }
}
```

Vorname	Name	Matr.-Nr.

b) Wir schreiben zusätzlich zu `A` und `B` eine neue Klasse `C`. Welche drei Fehler treten beim Compilieren auf? Begründen Sie Ihre Antwort kurz.

```
public class C extends B {

    private int f(double x) {
        A a = new A();
        A.x = x;
        return B.y;
    }
}
```

Vorname	Name	Matr.-Nr.

Aufgabe 2 (Verifikation, 11 + 2 Punkte**)**

Der Algorithmus P berechnet für eine Zahl $a \in \mathbb{R} \setminus \{1\}$ und eine Zahl $n \in \mathbb{N}$ den Wert $\frac{a^n-1}{a-1}$.

a) Vervollständigen Sie die folgende Verifikation des Algorithmus P im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Zwei Zusicherungen dürfen 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: $n \in \mathbb{N}$, $a \in \mathbb{R} \setminus \{1\}$
Ausgabe: res
Vorbedingung: $n \geq 0, a \neq 1$
Nachbedingung: $res = \frac{a^n-1}{a-1} \wedge factor = a^n$

$\langle\ n \geq 0 \wedge a \neq 1\ \rangle$

$\langle$ ____________________________ $\rangle$

```
res = 0;
```

$\langle$ ____________________________ $\rangle$

```
factor = 1;
```

$\langle$ ____________________________ $\rangle$

```
i = 0;
```

$\langle$ ____________________________ $\rangle$

```
while (n > i) {
```

$\langle$ ____________________________ $\rangle$

$\langle$ ____________________________ $\rangle$

```
  res = res + factor;
```

$\langle$ ____________________________ $\rangle$

```
  factor = factor * a;
```

$\langle$ ____________________________ $\rangle$

```
  i = i + 1;
```

$\langle$ ____________________________ $\rangle$

```
}
```

$\langle$ ____________________________ $\rangle$

$\langle\ res = \frac{a^n-1}{a-1} \wedge factor = a^n\ \rangle$

Vorname	Name	Matr.-Nr.

b) Beweisen Sie die Terminierung des Algorithmus P.

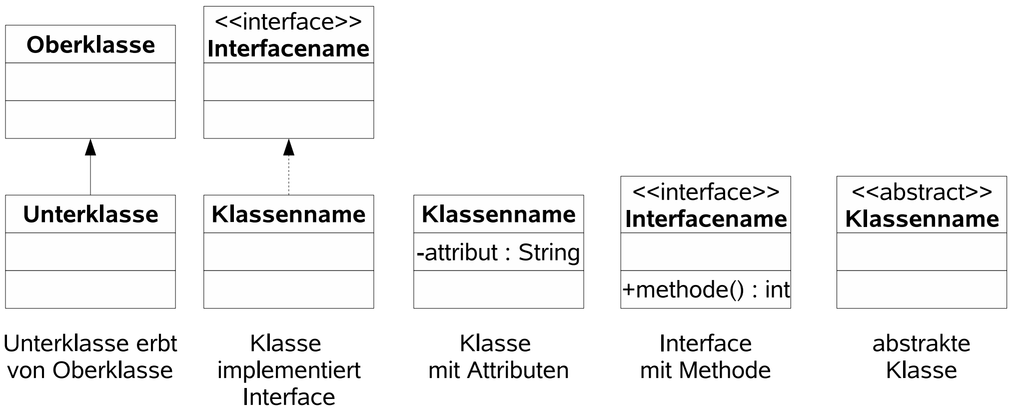
Vorname	Name	Matr.-Nr.

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

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

- Ein Rucksack ist ein Transportbehälter, der von Menschen getragen werden kann und evtl. kindgerecht ist. Jeder Rucksack hat ein Volumen und eine bestimmte Anzahl von Fächern.
- Ein Wanderrucksack ist ein Transportbehälter mit Fächern und einer Anzahl von äußeren Schnallen. Die Fächer bestimmen das Volumen eines Wanderrucksacks. Wanderrucksäcke werden von Menschen getragen, manche sind sogar für Kinder geeignet.
- Eine Kühlbox ist ein Transportbehälter mit bestimmtem Volumen. Eine Kühlbox wird von Menschen getragen und es gibt auch kleinere Ausführungen für Kinder. Kühlboxen gibt es mit einer unterschiedlichen Anzahl von Kühlakkus.
- Ein Koffer ist ein Transportbehälter für Menschen. Manche Koffer sind sogar für Kinder geeignet. Neben dem Volumen zeichnet sich jeder Koffer durch sein Außenmaterial aus.
- Ein Container ist ein Transportbehälter, welcher einem Transportunternehmen gehört. Es gibt Container mit verschiedenen Volumen.
- Ein Kühlcontainer ist ein Transportbehälter, welcher sich durch seine Kühlleistung (in Grad Celsius) auszeichnet. Jeder Kühlcontainer ist Eigentum eines Transportunternehmens und hat ein individuelles Volumen.
- Kühlboxen und Kühlcontainer sind gekühlte Behälter. Jeder gekühlte Behälter kann aufgrund seiner Kühlleistung bzw. seiner Kühlakkus berechnen, wie lange seine Kühlzeit ist.

a) Entwerfen Sie unter Berücksichtigung der Prinzipien der Datenkapselung eine Klassenhierarchie für die oben aufgelisteten Arten von Transportbehältern. 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 in der Klasse für Transportbehälter eine Methode `lieferBKB` (BKB = bester Kühlbehälter). Die Methode bekommt als Parameter ein Array von Transportbehältern übergeben und soll daraus einen Kühlbehälter mit der längsten Kühlzeit berechnen. Zudem soll auf der Konsole die Anzahl der Kühlbehälter in dem Array ausgegeben werden.

Gehen Sie dabei davon aus, dass das übergebene Array nicht der `null`-Wert ist. Kennzeichnen Sie die Methode mit dem Schlüsselwort "`static`", falls angebracht.

Vorname	Name	Matr.-Nr.

Aufgabe 4 (Programmierung in Java, 7 + 10 + 8 Punkte)

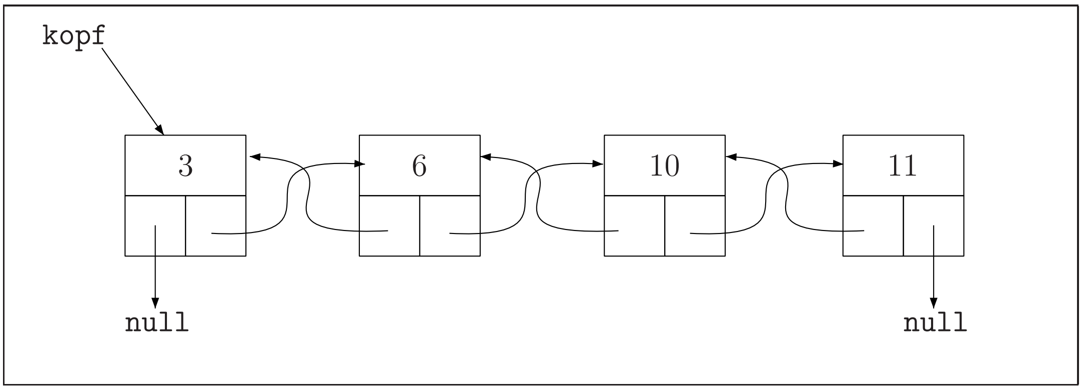
Gegeben ist das folgende Interface.

```
public interface Vergleichbar {
     public boolean gleich(Vergleichbar other);
     public boolean groesser(Vergleichbar other);
}
```

Für zwei Objekte `s` und `t` vom Typ `Vergleichbar` liefert `s.gleich(t)` den Wert `true` zurück, falls `s` und `t` gleich sind und `false` sonst. Die Auswertung von `s.groesser(t)` ergibt den Wert `true`, falls `s` echt größer als `t` ist und ansonsten `false`.

In dieser Aufgabe wird eine doppelt verkettete sortierte Liste betrachtet (Beispiel s. Bild). Eine solche Liste hat folgende Eigenschaften:

- Jedes Element enthält einen Wert vom Typ `Vergleichbar` und hat einen Verweis auf seinen Vorgänger und einen Verweis auf seinen Nachfolger (d.h. die Liste ist doppelt verkettet).
- Der Vorgänger des ersten Elements der Liste und der Nachfolger des letzten Elements der Liste ist `null` (der Verweis ins Leere).
- Für eine Liste mit n Elementen gilt: der Wert des $i+1$-ten Elements ist größer oder gleich dem Wert des i-ten Elements (für $1 \leq i \leq n-1$); die Liste ist also sortiert.



Die Datenstruktur sei folgendermaßen in Java implementiert:

```
public class Liste {
     private Element kopf;

     public Liste (Element kopf) {
          this.kopf = kopf;
     }
     ...
}

public class Element {
     Vergleichbar wert;
     Element vorgaenger, nachfolger;

     public Element (Vergleichbar wert){
          this.wert = wert;
     }
}
```

Vorname	Name	Matr.-Nr.

Implementieren Sie die folgenden Methoden in der Klasse `Liste`. Dabei dürfen Sie direkt (ohne Verwendung von Selektoren) auf die Attribute der Klasse `Element` zugreifen. Achten Sie darauf, dass beim Löschen und Einfügen auch die Vorgängerverweise entsprechend aktualisiert werden. Selbstverständlich dürfen Sie auch weitere benötigte Hilfsmethoden implementieren.

a) Erstellen Sie eine Methode `public void loeschen(Vergleichbar wert)`, die einen Wert aus der sortierten Liste löscht, falls dieser in der Liste existiert. Sofern der Wert mehrfach in der Liste vorkommt, wird er nur einmal gelöscht. Falls der Wert gar nicht in der Liste auftritt, wird die Liste nicht verändert.

b) Implementieren Sie eine Methode `public void einfuegen(Vergleichbar wert)`, die einen Wert in die Liste einfügt. Die Liste muss nach diesem Einfügen wieder *sortiert* sein. **Beachten Sie dabei, dass bei der Implementierung dieser Methode keine Schleifen verwendet werden dürfen. Sie dürfen aber Rekursion benutzen.**

c) Erstellen Sie eine Methode `public Liste listeKopieren()`, die eine vollständige Kopie der aktuellen Liste erzeugt, d.h. eine neue doppelt verkettete sortierte Liste, die dieselben Werte enthält, die aber keine gemeinsamen `Element`-Objekte mit der aktuellen Liste hat.

Vorname	Name	Matr.-Nr.

Vorname	Name	Matr.-Nr.