Klausur zur Vorlesung

Allgemeine Informatik II

Prof. J. Fürnkranz
Technische Universität Darmstadt — Sommersemester 2007
Termin: 6. 9. 2007

Name: **Vorname:** **Matrikelnummer:**

Fachrichtung: **Wiederholer:** ☐ ja ☐ nein

☐ Diplom ☐ Master ☐ Bachelor

Punkte:	(1) ...	(2) ...	(3) ...	(4) ...	(5) ...	(6) ...	**Summe:**

- **Aufgaben:** Diese Klausur enthält auf den folgenden Seiten 5 Aufgaben zu insgesamt 80 Punkten. Jede Aufgabe steht auf einem eigenen Blatt. Kontrollieren Sie *sofort*, ob Sie alle sechs Blätter erhalten haben!
- **Zeiteinteilung:** Die Zeit ist knapp bemessen. Wir empfehlen Ihnen, sich zuerst einen kurzen Überblick über die Aufgabenstellungen zu verschaffen, und dann mit den Aufgaben zu beginnen, die Ihnen am besten liegen.
- **Papier:** Verwenden Sie nur Papier, das Sie von uns ausgeteilt bekommen. Bitte lösen Sie die Aufgaben auf den dafür vorgesehenen Seiten (auch auf den Rückseiten). Falls der Platz nicht ausreicht, vermerken Sie dies bitte und setzen die Lösung auf einem Zusatzblatt fort. Brauchen Sie zusätzlich Papier (auch Schmierpaper), bitte melden.
- **Fragen:** Sollten Sie Teile der Aufgabenstellung nicht verstehen, bitte fragen Sie!
- **Abschreiben:** Sollte es sich (wie in den letzten Jahren leider immer wieder) herausstellen, daß Ihre Lösung und die eines Kommilitonen über das zu erwartende Maß hinaus übereinstimmen, werden beide Arbeiten negativ beurteilt (ganz egal wer von wem in welchem Umfang abgeschrieben hat).
- **Ausweis:** Legen Sie Ihren *Studentenausweis* und *Lichtbildausweis* sichtbar auf Ihren Platz. Füllen Sie das Deckblatt sofort aus!
- **Hilfsmittel:** Zur Lösung der Aufgaben sind keine Unterlagen erlaubt. Gedruckte Wörterbücher sind für ausländische Studenten erlaubt, elektronische Hilfsmittel (Taschenrechner, elektronische Wörterbücher, Handy, etc.) sind verboten! Sollten Sie etwas anderes verwenden wollen, bitte klären Sie das *bevor* Sie zu arbeiten beginnen.
- **Aufräumen:** Sonst darf außer Schreibgerät, Essbarem, von uns ausgeteiltem Papier und eventuell Wörterbüchern nichts auf Ihrem Platz liegen. Taschen bitte unter den Tisch!

Gutes Gelingen!

Aufgabe 1 (4 Punkte)

Erklären Sie, wie Ihr Programm-Code vom Java-System bearbeitet wird. Wird er kompiliert? Wird er interpretiert? Was ist der Byte-Code? Was ist die Java Virtual Machine?

Aufgabe 2 (12 Punkte)

Gegeben seien folgende beiden Definitionen von Exceptions

```
1  public class ExcA extends Exception {
2     public String getMessage() {
3        return "Exception A";
4     }
5  }
```

```
1  public class ExcB extends ExcA {
2     public String getMessage() {
3        return "Exception B";
4     }
5  }
```

und folgendes Programmstück:

```
1     public static String f (int n) throws ExcA {
2        if (n >= 0) {  return "abc";  }
3        else {  throw new ExcA();  }
4     }
5
6     public static String g (int n) throws ExcB {
7        if (n <= 0) {  return "xyz";  }
8        else {  throw new ExcB();  }
9     }
10
11    public static void testExceptions (int i) {
12       try {
13          System.out.println(f(i));
14          System.out.println(g(i));
15       }
16       catch (ExcA e) {
17          System.out.println(e.getMessage());
18       }
19    }
```

2–a Geben Sie für die folgenden Methoden-Aufrufe die Ausgabe des Programms an und begründen Sie diese.

(i) `testExceptions( 1);`

(ii) `testExceptions( 0);`

(iii) `testExceptions(-1);`

2–b Was würde passieren, wenn man in Zeile 16 `ExcA` durch `ExcB` ersetzen würde und den Rest des Programmes unverändert läßt?

Aufgabe 3 (12 Punkte)

Schreiben Sie drei verschiedene Versionen einer Methode `static int sum(int m, int n)`, die $\sum_{i=m}^{n} i$ (die Summe aller Zahlen von m bis n) berechnet.

3–a mit Hilfe einer `for`-Schleife

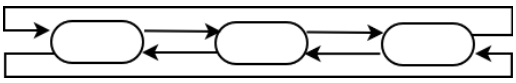
3–b mit Hilfe einer `while`-Schleife

3–c rekursiv

Aufgabe 4 (22 Punkte)

Gegeben sei folgende rekursive Datenstruktur für eine doppelt verkettete Liste:

```
1    class DList {
2       Object info;
3       DList next;
4       DList prev;
5    }
```



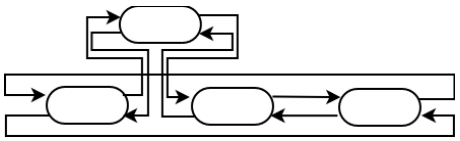
Das heißt, jedes Listenelement zeigt sowohl auf den Vorgänger als auch auf seinen Nachfolger. Es wird daher auch kein explizites Anfangs- oder Endelement gespeichert, das Ende der Liste kann man erkennen, in dem man ausgehend von einem beliebigen Element solange den Zeigern folgt, bis man wieder bei diesem angekommen ist.

Hinweis: Erinnern Sie sich daran, dass bei der Definition einer Methode das Schlüsselwort `this` einen Zeiger auf das momentane Element zurückliefert (man kann `this` aber natürlich nicht verändern). Das benötigen Sie zur Lösung der folgenden Aufgaben.

4–a Schreiben Sie einen Konstruktur, der eine doppelt verkettete Liste mit einem an den Konstruktor übergebenen Objekt initialisiert. Der Nachfolger und Vorgänger ist die Liste selbst (bzw. das erzeugte Listenelement).

4–b Schreiben Sie eine Methode `insert` mit einem Parameter `o`. Die Methode soll das Objekt `o` an der nächsten Stelle einfügen. Die Liste muß danach wieder doppelt verkettet sein.

Nebenstehende Skizze zeigt das Resultat nach Einfügen eines neuen Elements, wenn `insert` für das Element ganz links in obiger Liste aufgerufen wird.



4–c Schreiben Sie eine Methode `delete`. Die Methode soll das nächste Element aus der Liste löschen. Das darin enthaltene Objekt soll der Rückgabewert der Methode sein. Die Liste soll danach wieder doppelt verkettet sein. Sie können annehmen, daß die ursprüngliche Liste mehr als ein Element enthält.

4–d Schreiben Sie eine Methode `print_reverse`, die alle Elemente der Liste in verkehrter Reihenfolge auf dem Bildschirm ausgibt (also in obiger Skizze von rechts nach links, wenn die Methode für das Element ganz links aufgerufen wird). Die Elemente sollen in einer Zeile durch Kommas getrennt ausgegeben werden, nach dem letzten Element soll ein Newline folgen (Verwendung von `System.out.println`).

Aufgabe 5 (20 Punkte)

5–a Definieren Sie eine abstrakte Klasse `Fahrzeug`, bestehend aus

- einer Methode `radanzahl`, die die Anzahl der Räder des Fahrzeugs retournieren sollen (wird nur in Unterklassen implementiert)
- einer Datenkomponente `sitze`, die die Anzahl der Sitze des Fahrzeugs abspeichert (darf nur in dieser Klasse sichtbar sein)
- eine Methode `getSitzAnzahl`, die die Anzahl der Sitze als Rückgabewert hat (geben Sie auch die Implementation in dieser Klasse an)
- einer Methode `setSitzAnzahl`, die die Anzahl der Sitze festlegt (geben Sie auch die Implementation in dieser Klasse an)

5–b Definieren Sie ein Interface `HatMotor`, das festlegt, daß eine Klasse, die dieses Interface implementiert eine Methode `ps` haben muß die die Anzahl der PS als `double`-Zahl zurückgibt.

5–c Implementieren Sie eine Klasse `PKW`. PKWs sind Fahrzeuge (`Fahrzeug`) und haben einen Motor (`HatMotor`). Jeder PKW hat 4 Räder, die Anzahl der PS (`double`) und die Anzahl der Sitze (`int`) sind variabel.

Implementieren Sie auch alle notwendigen Methoden. Methoden die geerbt werden können sollen nicht überschrieben werden.

5–d Implementieren Sie einen Konstruktor für die Klasse `PKW`, der die Anzahl der PS und die Anzahl der Sitze des Fahrzeugs übergeben bekommt.

Aufgabe 6 (10 Punkte)

Jeder `PKW` hat eine Autonummer. Autonummern werden als Zeichenketten (`String`) gespeichert. Die Zuordnung von Autonummern zu PKWs wird in einer Hash-Tabelle abgespeichert (Schlüssel sind die Autonummern).

Schreiben Sie eine Methode

```
public static Set schnelleAutos (Map autos)
```

die eine Hash-Tabelle `autos` erhält, und die Autonummern aller darin gespeicherten Autos mit 100 oder mehr PS zurückgibt (Autos sind vom Typ `PKW`, Ihre PS-Anzahl kann mit der Methode `ps()` als `double`-Zahl ausgelesen werden). Die Rückgabe der Nummern erfolgt, wie oben angegeben, in einer Menge (Interface `Set`).

Zur Erinnerung: Die Namen der wichtigsten Methoden von Collections, Maps, und Iterators:

Set:

```
boolean add(Object obj)
boolean contains(Object obj)
boolean isEmpty()
Iterator iterator()
boolean remove(Object obj)
int size()
```

Map:

```
Object get(Object key)
Object put(Object key, Object v)
Object remove(Object key)
boolean containsKey(Object key)
boolean containsValue(Object v)
Set keySet()
```

Iterator:

```
boolean hasNext()
Object next()
void remove()
```