Prof. Dr.-Ing. J. Hornegger
Lehrstuhl für Mustererkennung
Universität Erlangen-Nürnberg

7. April 2006

Schriftliche Prüfung zu der Vorlesung Grundlagen der Informatik I April 2006

(Prüfungsnummern 11204111, 11204112, 11248143 und 27348110)

Angaben zur Person:

Name, Vorname .....

Geburtsdatum .....

Matrikelnummer .....

Studiengang .....

Nicht von der Kandidatin/dem Kandidaten auszufüllen!

Bewertung:

Aufgabe	1	2	3	4	5	6	$\sum$	Note
Punktezahl								

Organisatorische Hinweise

Die folgenden Hinweise bitte aufmerksam lesen und die Kenntnisnahme durch Unterschrift bestätigen!

- Bitte legen Sie Ihren Studentenausweis und einen Lichtbildausweis zur Personenkontrolle bereit!
- Schreiben Sie deutlich und ausschließlich mit blauer oder schwarzer Tinte. Benutzen Sie keinen Bleistift. Unleserliche Antworten gehen nicht in die Bewertung ein.
- Hilfsmittel (außer Schreibmaterial) sind nicht zugelassen. Dies gilt auch für Taschenrechner, Mobiltelefone, elektronische Assistenten, etc.
- Fragen zu den Prüfungsaufgaben werden grundsätzlich nicht beantwortet!
- **Überprüfen Sie die Prüfungsaufgaben auf Vollständigkeit (15 Seiten inklusive Deckblatt) und einwandfreies Druckbild!** Vergessen Sie nicht, auf dem Deckblatt die Angaben zur Person einzutragen!
- Die Bearbeitungszeit beträgt 90 Minuten. Da Sie maximal 90 Punkte erreichen können, ist aus den möglichen Punkten pro Aufgabe leicht die Zeit zu berechnen, die Sie zum Lösen der jeweiligen Aufgabe investieren sollten. Die angegebene Punkteverteilung gilt unter Vorbehalt.
- Auf Ihrem Platz befindet sich 1 Blatt Schmierpapier. **Das Schmierpapier darf nicht mit abgegeben werden.** Die Lösung einer Aufgabe muss auf das Aufgabenblatt geschrieben werden und zwar in den freien Raum, der jeweils der Aufgabe folgt. Sollte der Platz nicht ausreichen, so müssen Sie bei der Aufsicht zusätzliche Formblätter anfordern und einheften lassen.
- Wenn Sie die Prüfung aus gesundheitlichen Gründen abbrechen müssen, so muss Ihre Prüfungsunfähigkeit durch eine Untersuchung in der Universitätsklinik nachgewiesen werden. Melden Sie sich in jedem Fall bei der Aufsicht und lassen Sie sich das entsprechende Formular aushändigen.

Erklärung

Durch meine Unterschrift bestätige ich den Empfang der vollständigen Klausurunterlagen und die Kenntnisnahme der obigen Informationen.

Erlangen, der 7. April 2006 .....
(Unterschrift)

Ich bin damit einverstanden, dass mein Prüfungsergebnis unter Angabe der Matrikelnummer anonymisiert veröffentlicht wird:

ja: ☐ nein: ☐

Erlangen, der 7. April 2006 .....
(Unterschrift)

Aufgabe 1 (10 Punkte)

Kreuzen Sie die richtige Antwort auf die jeweilige Frage an. Pro Frage ist immer nur eine Antwort richtig.

1. Welche Aussagen gelten in Java bei der Umwandlung von Datentypen?
   (a) Bei der Division von zwei Werten des Typs `int` wird einer der beiden Werte implizit in den Typ `double` gecastet. ☐
   (b) Wird ein Wert vom Typ `float` verlangt, kann immer auch ein Wert vom Typ `int` angegeben werden. ☐
   (c) Bei der impliziten Typumwandlung kann Information (z. B. Nachkommastellen) verloren gehen. ☐

2. Welche Aussagen bzgl. Prioritäten von Operatoren stimmen?
   (a) Operatoren mit hoher Priorität werden vom Prozessor in kürzerer Zeit ausgeführt. ☐
   (b) Die Prioritäten von Operatoren gewährleisten, dass bei der Auswertung von Ausdrücken die Regel „Punkt vor Strich“ gilt. ☐
   (c) Der Shift-Operator `<<` hat in Java eine höhere Priorität als der Additionsoperator `+`. ☐

3. Welche Aussagen treffen auf die Darstellung von Zeichen im Rechner zu?
   (a) Mit 7 bit lassen sich $2^8 - 1 = 255$ verschiedene Zeichen kodieren. ☐
   (b) Für die Speicherung von Zeichenketten gibt es in Java den Datentyp `char`. ☐
   (c) Die Interpretation einer Folge von Bits hängt vom verwendeten Zeichensatz ab. ☐

4. Welche Aussagen bzgl. des klassischen Universalrechners sind richtig?
   (a) Damit das Rechenwerk eine Operation ausführen kann, müssen die Operanden in die entsprechenden Register geladen werden. ☐
   (b) Im Gegensatz zum Rechenwerk kann das Speicherwerk Rechenoperationen direkt im Hauptspeicher ausführen. ☐
   (c) Durch Pipelining lässt sich der Zugriff auf Daten im Hauptspeicher bzw. auf der Festplatte verkürzen. ☐

5. Welche Aussagen für formale Grammatiken stimmen?
   (a) Grammatiken sind durch eine Menge terminaler Symbole, einer Regelmenge und einem Startsymbol vollständig definiert. ☐
   (b) Das Startsymbol ist ein Element der Menge der terminalen Symbole. ☐
   (c) Bei Kontextsensitiven Grammatiken (Typ 1) stehen auf der rechten Seite mindestens genauso viele Symbole wie auf der linken Seite. ☐

6. Welche Aussagen treffen auf Modifikatoren von Attributen zu?

   (a) Eine als `final` deklarierte Variable kann unabhängig von der Existenz eines Objekts benutzt werden. ☐

   (b) Auf Attribute, die als `protected` deklariert sind, kann in allen abgeleiteten Unterklassen zugegriffen werden. ☐

   (c) Attribute, die als `static` deklariert sind, müssen immer auch als `public` deklariert werden, um von außen darauf zugreifen zu können. ☐

7. Welche Aussagen gelten in der objektorientierten Programmierung für den Mechanismus der Vererbung?

   (a) In abgeleiteten Klassen dürfen keine Methoden vorkommen, die dieselbe Signatur haben wie in der Basisklasse. ☐

   (b) Der Konstruktor der Basisklasse kann im Konstruktor der abgeleiteten Klasse mittels `this` aufgerufen werden. ☐

   (c) Ein Objekt einer abgeleiteten Klasse kann immer einer Variable vom Typ der Basisklasse zugewiesen werden. ☐

8. Welche Aussagen bzgl. Suchalgorithmen stimmen?

   (a) Um die Lineare Suche auf Felder anwenden zu können, müssen die Elemente sortiert sein. ☐

   (b) Bei sortierten Feldern haben die Lineare und die Binäre Suche dieselbe Komplexität. ☐

   (c) Die Binäre Suche lässt sich nicht auf verkettete Listen anwenden, da ein direkter Zugriff auf ein beliebiges Element nicht möglich ist. ☐

9. Welche Aussagen treffen auf Sortierverfahren zu?

   (a) Divide-and-Conquer-Verfahren basieren darauf, dass ein Problem in Teilprobleme zerlegt wird, die einzeln gelöst und deren Lösungen anschließend kombiniert werden. ☐

   (b) Quicksort ist ein Divide-and-Conquer-Verfahren, bei dem zwei Teillisten zunächst sortiert und anschließend nach dem Reißverschlussprinzip gemischt werden. ☐

   (c) BubbleSort hat den Nachteil, dass zum Sortieren eine Kopie des Feldes angelegt werden muss. ☐

10. Welche Aussagen sind für Kommunikationsprotokolle gültig?

    (a) Das User Datagramm Protocoll (UDP) garantiert lediglich, dass keine Daten verloren gehen; die richtige Reihenfolge wird nicht garantiert. ☐

    (b) Das Transport Control Protocol (TCP) garantiert, dass alle Daten beim Empfänger in der richtigen Reihenfolge ankommen. ☐

    (c) Im Gegensatz zu UDP basiert TCP auf dem Internet Protocol (IP). ☐

Aufgabe 2 (14 Punkte)

1. Folgendes Programm soll dazu dienen, die Quadrate der Zahlen von 1 bis 10 zu addieren und auszugeben. Verbessern Sie die darin enthaltenen Fehler, sodass es sich danach um ein fehlerfreies Java-Programm handelt, welches die gestellte Aufgabe erfüllt.

```
public class Fehlerteufel

    public static summe_der_quadrate(void) {
        int sum;
        int i = 1;

        while (i > 10) {
            sum = i * i;
            i++;
        }

        return;
    }

    public static void main(String[] args) {
        writeln("Ergebnis: " + summe_der_quadrate);
    }
}
```

2. Geben Sie an, was beim Ablauf des Programms der Reihe nach auf `stdout` ausgegeben wird.

```
public class Scope {

    private static int tmp = 0x30
    private static int c = 5;

    public static void magic(int[] zahlen, int c) {
        int tmp = zahlen[1];
        zahlen[1] = zahlen[2];
        zahlen[2] = tmp;

        System.out.println("tmp = " + tmp);
        c++;
    }

    public static void main(String[] args) {
        int werte[] = {2, 5, 3};
        int c = 27 / Scope.c;

        magic(werte, c);

        for (int i = 0; i < werte.length; i++) {
            System.out.println("werte[" + i + "] = " + werte[i]);
        }

        System.out.println("tmp = " + (tmp++));
        System.out.println("c = " + c);
    }
}
```

Auf `stdout` wird ausgegeben:

Aufgabe 3 (10 Punkte)

Die Additionsfunktion für zwei Zahlen $m, n \in \mathbb{N}_0$ kann durch wiederholte Inkrementierung um 1 realisiert werden:

$$m + n := \text{add}(m, n) = \begin{cases} \text{add}(m, n-1) + 1 & \text{falls } n > 0 \\ m & \text{sonst} \end{cases}$$

Geben Sie eine iterative und eine rekursive Implementierung der Additionsfunktion in Java an. Verwenden Sie dabei den Inkrement- und Dekrementoperator, aber nicht den Additions- bzw. Subtraktionsoperator von Java.

```
// Iterative Variante der Additionsfunktion
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
```

```
// Rekursive Variante der Additionsfunktion
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
```

Aufgabe 4 (16 Punkte)

Die Elemente einer einfach verketteten Liste sollen mit Hilfe der folgenden Klasse modelliert werden:

```
class ListElement {
    int value;
    ListElement next;
}
```

Die verkettete Liste enthält die eingefügten Zahlen in unsortierter Reihenfolge, wobei Zahlen nicht mehrfach vorkommen. Eine Methode `insert` sei bereits vorhanden. Schreiben Sie eine Methode `remove`, die den übergebenen Wert vom Typ `int` in der verketteten Liste sucht und anschließend entfernt. Beachten Sie dabei als Sonderfall das Löschen des ersten Elements der Liste.
Ist das zu löschende Element nicht in der Liste enthalten, soll eine Exception der Klasse `ElementNotFoundException` geworfen werden, die als Grund für die Ausnahme die Zeichenkette „Element <element> nicht gefunden“ enthält (für <element> ist das nicht vorhandene Element einzutragen).

```
public class ElementNotFoundException extends Exception {
    public ElementNotFoundException() { }

    public ElementNotFoundException(String msg) {
        super(msg)
    }
}
```

```
public class VerketteteListe {
    private ListElement head = null;

    // Methode remove
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
}
```

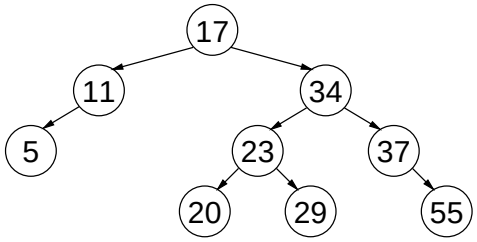
Aufgabe 5 (16 Punkte)

1. Binäre Sortierbäume können zu einer verketteten Liste entarten. Was bedeutet das für die Komplexität beim Einfügen eines Elements bzgl. der Anzahl der durchzuführenden Vergleiche im Vergleich zu einem balancierten Baum?

2. Geben Sie ein Beispiel für das Einfügen von fünf Zahlen in einen leeren binären Sortierbaum an, bei dem der Baum entartet. Zeichnen Sie den entarteten Baum nach dem Einfügen aller fünf Zahlen.

3. Definieren Sie, was man in der Informatik unter einem AVL-Baum versteht:

4. Fügen Sie in folgenden AVL-Baum das Element `27` ein und führen Sie die notwendigen Rotationen aus. Kennzeichnen Sie dabei, an welchen Knoten Sie welche Rotation durchführen müssen und zeichnen Sie den Baum nach jeder Einzelrotation. Geben Sie bei allen Bäumen für jeden Knoten die Höhendifferenz zwischen dem rechten und dem linken Teilbaum an.



Aufgabe 6 (24 Punkte)

1. Modellieren Sie einen quaderförmigen Behälter, der durch seine Breite, seine Höhe und seine Tiefe gegeben ist, durch eine geeignete Java-Klasse `Behaelter`. Die Maße des Behälters werden bei der Erzeugung des Objekts in der Einheit `cm` angegeben und sollen sich danach von außen weder auslesen, noch verändern lassen. Die Anwenderschnittstelle soll lediglich eine Methode `getVolume` zur Verfügung stellen, die das Volumen des Behälters in der Einheit Liter ($1000\,\mathrm{cm}^3$) zurückliefert.

```
public class Behaelter {

    // Attribute
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

    // Konstruktor
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

    // Methode getVolume
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

}
```

2. Ein Wasserbehälter ist ein spezieller Behälter, der mit Wasser gefüllt und aus dem Wasser wieder entnommen werden kann. Er besitzt einen Überlaufabfluss, der verhindert, dass der Behälter mit mehr als 90 % seines Volumens gefüllt wird.

   Wie bei einem gewöhnlichen Behälter sollen die Maße bei der Objekterzeugung angegeben werden können. Möchte man keine Maße angeben, so soll ein Wasserbehälter mit den Standardmaßen 100 cm x 35 cm x 25 cm (b x h x t) erzeugt werden.

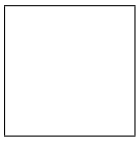
   Stellen Sie eine Methode `fuellen` zum Befüllen des Behälters mit Wasser zur Verfügung, der die gewünschte Wassermenge in der Einheit Liter als Parameter übergeben wird. Der Rückgabewert der Methode ist die Füllmenge in Liter nach dem Befüllen.

   Des Weiteren soll ein Wasserbehälter eine Methode `entnehmen` besitzen, mit der sich eine gewünschte Wassermenge (Einheit Liter) wieder entnehmen lässt. Da sich unter Umständen nicht genügend Wasser im Behälter befinden, gibt der Rückgabewert die tatsächlich entnommene Wassermenge an.

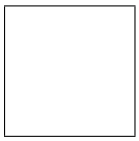
   Modellieren Sie eine Klasse `Wasserbehaelter` unter Verwendung der bereits implementierten Klasse `Behaelter`.

```
// Klassendefinition Wasserbehaelter

-------------------------------------------------------------

    // Attribute
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------

    // Konstruktoren
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------

    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
```

```
    // Methode fuellen
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
```



```
    // Methode entnehmen
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
```



```
}
```

3. Ein Pool ist wiederum ein spezieller Wasserbehälter, der sich dadurch auszeichnet, dass mit einer Methode `add_disinfectant` eine gewünschte Menge Desinfektionsmittel hinzugegeben werden kann. Außerdem lässt sich mit Hilfe der Methode `getConcentration` die Konzentration des insgesamt hinzugefügten Desinfektionsmittels in der vorhandenen Wassermenge bestimmen. Stellen Sie dabei über möglichst restriktive Modifikatoren der Attribute der Klasse `Wasserbehaelter` sicher, dass die Klasse `Pool` auf die Wassermenge der Klasse `Wasserbehaelter` zugreifen kann. Die Klasse `Pool` selbst ist nicht zu implementieren!

   Zeichnen Sie für die Klassen `Behaelter`, `Wasserbehaelter` und `Pool` ein UML-Klassendiagramm, das alle Attribute und Methoden der einzelnen Klassen enthält und aus dem die Beziehungen der Klassen untereinander hervorgehen.