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

28. März 2007

Schriftliche Prüfung zu der Vorlesung Grundlagen der Informatik I März 2007

(Prüfungsnummern 11 204 111, 11 204 112, 14 104 112, 11 104 112, 11 177 1100 , 11 179 1080, 14 177 1100, 27 348 110, 11 248 143)

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 (17 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 28. März 2007 .....
(Unterschrift)

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

ja: ☐ nein: ☐

Erlangen, der 28. März 2007 .....
(Unterschrift)

Aufgabe 1 (10 Punkte)

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

1. Um den Kontrollfluss steuern zu können gibt es zwei Kontrollstrukturen:
   (a) Verzweigungen und Schleifen ☐
   (b) Kreuzungen und Schleifen ☐
   (c) Einfach- und Mehrfachschleifen ☐

2. Welche Aussage gilt für Felder (Arrays):
   (a) Hat das Feld *length* Elemente, laufen die Indizes von 0 bis *length*-1. ☐
   (b) Es können nur Felder von Grunddatentypen erzeugt werden. ☐
   (c) Auf einzelne Elemente kann mit dem Array-Operator `{}` zugegriffen werden. ☐

3. Welche der Aussagen trifft auf Unterprogramme zu:
   (a) Ein Unterprogramm hat einen Namen und kann über diesen aufgerufen werden. ☐
   (b) Einem Unterprogramm kann nur **ein** Parameter übergeben werden. ☐
   (c) Ein Unterprogramm gibt immer einen Wert zurück. ☐

4. Ein Induktionsbeweis besteht aus folgenden Schritten:
   (a) Induktionsanker, Induktionsbug und Induktionsheck ☐
   (b) Induktionanker, Induktionannahme und Induktionsschritt ☐
   (c) Induktionanker, Induktionsannahme und Induktionsspule ☐

5. Welche Aussage gilt bezüglich polyadischen Zahlensystemen:
   (a) Das Binärsystem ist kein polyadisches Zahlensystem, weil die 2 als Basis dieses Zahlensystems eine Primzahl ist. ☐
   (b) Das Dezimalsystem und das Binärsystem sind zwei wichtige polyadische Zahlensysteme: das Dezimalsysten wegen seiner Nutzung im Alltag, das Binärsystem wegen seiner einfachen elektrischen Realisierbarkeit. ☐
   (c) Zahlen können nicht vom Binärsystem in das Dezimalsystem umgerechnet werden. ☐

6. Welche Aussage gilt bezüglich des klassischen Universalrechners:

   (a) Das erste Modell wurde 1976 von IBM auf den Markt gebracht. ☐

   (b) Er enthält unter anderem eine Eingabeeinheit, ein Rechenwerk und eine Ausgabeeinheit. ☐

   (c) Das Rechenwerk dient der Koordinierung der Aktivitäten der Module. ☐

7. Welche Aussage gilt bezüglich eines Stapelspeichers (Kellerspeichers, stack):

   (a) Man kann auf das zuletzt und zuerst gespeicherte Element zugreifen. ☐

   (b) Der Stapelspeicher funktioniert mach den **last-in-first-out**-Prinzip: man kann nur auf das zuletzt gespeicherte Element zugreifen. ☐

   (c) Der Stapelspeicher erlaubt eingeschränkten wahlfreien Zugriff (*restricted random access*) auf die Elemente: das Einfügen von Elementen an beliebiger Stelle ist möglich; allerdings darf nur das oberste Element entfernt werden. ☐

8. Welche Aussage gilt für die Parameterübergabe in Java:

   (a) Grunddatentypen werden *by-value* übergeben ☐

   (b) Referenzdatentypen werden *by-value* übergeben wenn sie mindestens einen Grunddatentypen enthalten ☐

   (c) die Art der Parameterübergabe ist plattformabhängig ☐

9. Welche Aussage gilt bezüglich des Traversierens von Bäumen:

   (a) es gibt die Traversierungsmethoden *preorder* (Vorordnung), *postorder* (Nachordnung) und *inorder* (Inordnung) ☐

   (b) es gibt die Traversierungsmethoden *bottom-up* (von den Blättern zur Wurzel), *top-down* (von der Wurzel zu den Blättern) und *left-right* (von links nach rechts beginnend bei dem Blattknoten links unten) ☐

   (c) Das Traversieren eines Binärbaumes ist weder mit dem *preorder*- noch mit dem *bottom-up*-Verfahren möglich. ☐

10. Welche Aussage trifft bezüglich Modifikatoren in der objektorientierten Programmiersprache Java zu:

    (a) Variablen, die als `public` deklariert wurden, sind in allen anderen Klassen, die ebenfalls public deklariert wurden, sichtbar. ☐

    (b) Variablen, die als `public final` deklariert wurden, können nur durch Methoden, die ebenfalls als `public` deklariert wurden, verändert werden. ☐

    (c) Variablen, die als `private` deklariert wurden, sind nur innerhalb der eigenen Klasse sichtbar und sind somit vor dem direkten Zugriff durch andere Klassen geschützt. ☐

Aufgabe 2 (14 Punkte)

1. Das folgende Programm soll dazu dienen, die Summe der Zahlen im Intervall von]2;5[eines Feldes zu berechnen. Korrigieren Sie die im Programm enthaltenen Fehler, so dass es sich um ein fehlerfreies Programm handelt und es die geforderte Funktionalität erfüllt. Schreiben Sie das Programm nicht neu.

```
public feldSum {

    double feldSumme(int [] feld) {
        double summe = 0.;
        for(int i=0; i<feld.length; i++) {
            if(feld[i]>2 || feld[i]<5) {
                summe += summe + feld[i];
            }
        }
        return feldSumme(feld);
    }

    public static void main(String args) {
        double [] feld = "1.1, 2.2, 3.3, 4.4, 5.5";
        System.out.println(feldSumme(feld));
    }
}
```

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

```
public class Scope {

    private static double a = 3;
    private static double b = 5.99;

    public static int Mul(int a) {
        a = a * 4;
        b = (double) b + 1;
        System.out.println("b = " + b);
        return a;
    }

    public static double Mul(double a) {
        double c = a * 3;
        b += (int) b;
        System.out.println("b = " + b);
        return c;
    }

    public static void main(String args[]) {
        int a = 4;
        int b = 2;
        System.out.println("Mul(a) = " + Mul(a));
        System.out.println("Mul(3.0) = " + Mul(3.0));
        System.out.println("a = " + a);
        System.out.println("b = " + b);
    }

}
```

Auf `stdout` wird ausgegeben:

Aufgabe 3 (10 Punkte)

Die ganzzahlige Division zweier Zahlen $m, n \in N$ kann wie folgt definiert werden:

$$m/n := \text{div(m, n)} = \begin{cases} 1 + (m-n)/n & \text{wenn} \quad m \geq n \\ 0 & \text{sonst} \end{cases} \tag{1}$$

Schreiben Sie eine rekursive und eine iterative Methode um die ganzzahlige Division zweier natürlicher Zahlen mittels des Verfahrens (1) zu berechnen und rufen Sie diese mit $m = 28$ und $n = 5$ aus einer `main`-Methode auf.

```
public class Division{
// Iterative Variante der Funktion

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

// Rekursive Variante der Funktion

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

// main Methode

-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
}
```

Aufgabe 4 (16 Punkte)

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

```
class ListElement {
    int value;                  //Element
    ListElement next;           //Nachfolger in der Liste
    ListElement(int value) {    //Konstruktor
        this.value = value;
        this.next = null;
    }
}
```

Die verkettete Liste kann Zahlen in beliebiger Reihenfolge enthalten. Schreiben Sie eine Methode `insert`, welche eine Zahl am Ende der Liste einfgt. Implementieren Sie weiterhin eine Methode `convertArrayToList(int feld[], boolean newList)`, welche ein ihr übergebenes Feld `feld` in eine Liste kovertiert. Der Parameter `newList` gibt an ob eine bestehende Liste gelöscht oder das Feld hinten an die Liste angehängt werden soll.

```
public class VerketteteListe {

    // Attribute der Klasse Verkettete Liste
    private ListElement head = null;
    private ListElement last = null;

    // Methode insert
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

    // Methode convertArrayToList
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
}
```

Aufgabe 5

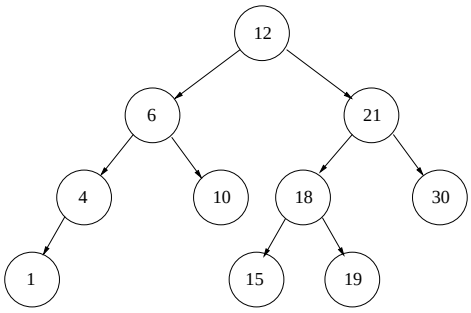
(16 Punkte)

1. Sortieren von Daten ist ein essenzieller Bestandteil der Informatik. Beschreiben Sie kurz das Prinzip von Sortieren durch Einfügen (Insertion Sort):

2. Nennen Sie zwei effizientere Sortierverfahren und das theoretische Prinzip welches diesen zu Grunde liegt:

3. Ein weiteres Problem in der Informatik ist die Suche von Daten. Hierfür werden oft Bäume verwendet. Definieren Sie, was man unter einem AVL-Baum versteht:

4 Fügen Sie in den folgenden AVL-Baum das Element 14 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)

In dieser Aufgabe geht es um die Modellierung öffentlicher Gebäude: Dies sollen im Sinne dieser Aufgabe Gebäude sein, die keine privaten Wohnräume darstellen, sondern durch die Öffentlichkeit genutzt werden können (Universitätsgebäude beispielsweise). Jedes öffentliche Gebäude besitzt einen Namen und eine gewisse Anzahl an Stockwerken. Des Weiteren hat jedes öffentliche Gebäude Öffnungszeiten: Im Rahmen dieser Aufgabe wird idealisiert angenommen, dass die Öffnungszeiten durch zwei ganzzahlige Uhrzeitangaben (beispielsweise Öffnen: 8 Uhr, Schließen: 16 Uhr; keine Minutenangaben) hinreichend spezifiziert werden und für jeden Tag der Woche gelten. Aus Gründen der Sicherheit muss bei Betreten von öffentlichen Gebäuden gegebenfalls auch eine Zutrittsberechtigung überprüft werden (Ausweiskontrolle beispielsweise).

1. Schreiben Sie eine Oberklasse `OeffentlichesGebaeude`, die die Gemeinsamkeiten aller öffentlichen Gebäude modelliert. Der lesende Zugriff auf die Attribute ist von außen nicht erlaubt, soll aber in abgeleiteten Klassen möglich sein. Über den Konstruktoraufruf sollen sich die Attribute setzen lassen können. Überladen Sie die Methode `toString()`, so daß diese die folgende Zeichenkette zurückliefert:

   <**Gebäudename**> **Beginn:** <**Zeit:Öffnen**> **Ende:** <**Zeit:Schließen**>

   Die tatsächliche Durchführung einer Überprüfung der Zugangsberechtigung zu einem öffentlichen Gebäude hängt vom tatsächlichen Gebäude ab: Stellen Sie sicher, dass jedes von der Klasse `OeffentlichesGebaeude` abgeleitete Gebäude eine Methode `pruefeZugang(int zugangscode)` implementiert, die anhand des Parameters `zugangscode`, die Zugangsberechtigung feststellt und als Ergebnis einen `boolean`-Wert zurückgibt, der den Zutritt gewährt (`true`) oder verwehrt (`false`).
   Die Modifikatoren von Attributen und Methoden sind so restriktiv wie möglich zu wählen. Die Klasse soll sich in beliebigen Packages nutzen lassen.

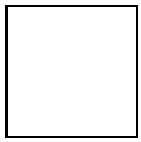
```
// Klasse OeffentlichesGebaeude

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

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

    // Konstruktor
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
```

```
    // Methoden
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
}
```



2. Als spezielles öffentliches Gebäude soll ein Forschungsinstitut modelliert werden (Klasse `ForschungsInstitut`): Neben den Eigenschaften eines jeden öffentlichen Gebäudes besitzt diese als ein weiteres Attribut ein Feld `berechtigte`, in dem die Zugangscodes (jeder Zugangscode als `int` repräsentiert) aller Forscher gespeichert sind. Alle Eingenschaften bis auf die Zugangscodes sollen sich nur bei der Objekterzeugung setzen lassen. Des Weiteren soll eine Methode `setzeZugansberechtigte(int[] zuganscodes)` implementiert werden, die das Attribut `berechtigte` der Klasse mit der korrekten Größe und den korrekten Werten initialisiert. Implementieren Sie die Methode `pruefeZugang`, wie sie von der Oberklasse gefordert wird. Dabei wird überprüft, ob das Feld `berechtigte` initialisiert wurde, und dann anschließend überprüft ob der eingegebene Zugangscode in der Liste der Berechtigten steht. Falls ja, gibt die Methode `true`, andernfalls `false` zurück. (*Hinweis: die Länge eines Feldes namens* `feld` *lässt sich mit* `feld.length` *ermitteln.*)

```
// Klasse ForschungsInstitut

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

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

    // Konstruktor
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
    -----------------------------------------------------------
```

```
    // Methoden
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------
    --------------------------------------------------------------

}
```

3. Nun soll eine Kleinstadt modelliert werden (Klasse `Kleinstadt`). Eine Kleinstadt hat einen Stadtnamen, ein Budget (auf den Cent genau spezifizert) für die Instandhaltung der öffentlichen Gebäude und kann maximal 30 öffentliche Gebäude haben. Desweiteren hat Sie eine Methode `pruefeBudget`, der als Parameter ein Datum (als String repräsentiert) übergeben wird und die dann den Stand des Budgets an diesem Datum zurückliefert. Zeichnen Sie für die Klassen `OeffentlichesGebaeude`, `Kleinstadt` und `Forschungsinstitut` ein UML-Klassendiagramm, dass alle Methoden und Attribute der einzelnen Klassen enthält und aus dem die Beziehungen der Klassen untereinander hervorgehen. Die Konstruktoren müssen dabei nicht enthalten sein.