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

27. September 2006

# Schriftliche Prüfung zu der Vorlesung Grundlagen der Informatik I September 2006

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

**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 (16 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 27. September 2006 .....
(Unterschrift)

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

ja: ☐ nein: ☐

Erlangen, der 27. September 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 in Bezug auf die primitiven Datentypen?
   (a) Einer Variable vom Typ `boolean` kann in Java statt `true` auch eine positive ganze Zahl zugewiesen werden. ☐
   (b) Wird ein Wert vom Typ `double` in den Typ `int` explizit gecastet, wird die Fließkommazahl auf die nächste ganze Zahl gerundet. ☐
   (c) Unabhängig von der Rechnerarchitektur ist der Typ `int` in Java immer 32 Bit groß. ☐

2. Welche Aussagen bzgl. der Operatoren in Java stimmen?
   (a) Mit dem =-Operator lassen sich zwei Variablen eines primitiven Datentyps auf Wertgleichheit testen. ☐
   (b) Die Priorität von Operatoren bestimmt, in welcher Reihenfolge komplexe Ausdrücke ausgewertet werden ☐
   (c) Der +-Operator kann nur auf numerische Datentypen angewendet werden. ☐

3. Welche Aussagen treffen bei der Kodierung von Zeichen zu?
   (a) In UTF-8 sind alle Zeichen mit 8 Bit kodiert. ☐
   (b) ASCII-Dokumente sind zu UTF-8 aufwärtskompatibel. ☐
   (c) Mit der ISO 8859-Zeichensatzfamilie können alle im Unicode definierten Zeichen dargestellt werden. ☐

4. Welche Aussagen gelten für die Darstellung von Gleitkommazahlen im IEEE-Format?
   (a) Die Zahl der Bits der Mantisse hängt von der Größe der Gleitkommazahl ab. ☐
   (b) Die Zahl der Bits des Exponenten bestimmt die Genauigkeit, mit der eine Zahl gespeichert wird. ☐
   (c) Die Gleitkommazahl muss ins Binärsystem konvertiert und normiert werden. ☐

5. Welche Aussagen stimmen beim Übersetzen eines Programmes?
   (a) Die lexikalische Analyse zerlegt ein Quellprogramm in eine Folge terminaler Symbole. ☐
   (b) Bei der syntaktischen Analyse wird überprüft, ob das Programm terminiert. ☐
   (c) In der Optimierungsphase werden beispielsweise Kommentare entfernt, da sie die Programmausführung verlangsamen würden. ☐

6. Welche Aussagen treffen auf binäre Sortierbäume zu?

   (a) Binäre Sortierbäume haben den Vorteil, dass sie nicht entarten können; die Komplexität beim Einfügen ist daher immer $\mathcal{O}(\log n)$. ☐

   (b) Rotationen müssen bei AVL-Bäumen nur beim Einfügen, nicht aber beim Löschen eines Elements durchgeführt werden. ☐

   (c) Bei AVL-Bäumen gibt es nur vier mögliche Typen von Rotationen. ☐

7. Welche Aussagen treffen auf Modifikatoren von Attributen zu?

   (a) Als `static` deklarierte Attribute sind Klassenvariablen, die von allen Objekten der Klasse gemeinsam genutzt werden. ☐

   (b) Auf Attribute, die als `private` deklariert sind, kann von außen nicht zugegriffen werden, wohl aber von abgeleiteten Unterklassen aus. ☐

   (c) Wird bei einem Attribut kein Modifikator angegeben, entspricht dies in Java dem Modifikator `public`. ☐

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

   (a) Methoden in abgeleiteten Klassen überladen Methoden gleicher Signatur in der Basisklasse. ☐

   (b) Gibt es in der Basisklasse den parameterlosen Standardkonstruktor nicht, muss im Konstruktor der abgeleiteten Klasse explizit der Konstruktor der Basisklasse aufgerufen werden. ☐

   (c) Ein Objekt einer abgeleiteten Klasse kann in Java nicht einer Variablen der Basisklasse zugewiesen werden. ☐

9. Welche Aussagen bzgl. Suchalgorithmen stimmen?

   (a) Die Lineare Suche kann bei selber Komplexität sowohl auf Felder als auch auf verkettete Listen angewendet werden. ☐

   (b) Die Binäre Suche kann auch auf unsortierte Felder angewendet werden. ☐

   (c) Die Binäre Suche eignet sich besonders gut für verkettete Listen. ☐

10. Welche Aussagen treffen auf Sortierverfahren zu?

    (a) BubbleSort hat eine durchschnittliche Komplexität von $\mathcal{O}(n \log n)$. ☐

    (b) MergeSort und Quicksort sind beides Divide-and-Conquer-Verfahren, da das Problem in Teilprobleme zerlegt wird, die einzeln gelöst und deren Lösungen anschließend kombiniert werden. ☐

    (c) Die Wahl des Pivot-Elements bei Quicksort hat keinen Einfluss darauf, wie schnell das Feld sortiert wird. ☐

## Aufgabe 2 (14 Punkte)

1. Folgendes Programm soll dazu dienen, die Vielfachen von 3 und 5 im Intervall $[1; 300]$ aufzusummieren und diese Summe auf `stdout` auszugeben. Verbessern Sie die darin enthaltenen Fehler, sodass es sich danach um ein fehlerfreies Java-Programm handelt, welches die gestellte Aufgabe erfüllt. Schreiben Sie das Programm nicht neu.

```
public class Fehlerteufel {

    public int spezial {
        int sum;
        for (int i = 0; i < 300; i++) {
            if (i % 3 = 0) {
                sum = i;
            }
            if (i % 5 = 0) {
                sum = i;
            }
        }
        return sum;
    }

    public void main(String[] args) {
        System.Out.println("Summe: " + spezial());
    }
}
```

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

```
public class Scope {

    public static int tmp = 0x15;

    public static void tausche(int a, int b) {
        tmp = a;
        a = b;
        b = tmp;
        System.out.println("a = " + a);
        System.out.println("b = " + b);
    }

    public static void main(String[] args) {
        int zahlen[] = {4, 7, 6};
        int i = 2;

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

        tausche(zahlen[1], zahlen[2]);

        while (i > 0) {
            i--;
            System.out.println("zahlen[" + i + "] = " + zahlen[i]);
        }

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

}
```

Auf `stdout` wird ausgegeben:

## Aufgabe 3 (10 Punkte)

Bei der ganzzahligen Division zweier Zahlen $a, b \in \mathbb{N}$ bleibt ein Rest übrig, der wie folgt berechnet werden kann:

$$a \text{ modulo } b := \text{modulo}(a, b) = \begin{cases} \text{modulo}(a-b, b) & \text{falls } a \geq b \\ a & \text{sonst} \end{cases}$$

Geben Sie eine iterative und eine rekursive Implementierung der Modulofunktion in Java an. Die Verwendung des Divisions- und Moduloperators von Java ist nicht erlaubt. Berechnen Sie in der `main`-Methode der Klasse durch Aufruf der rekursiven Variante den Rest, der bei der Division von 256 und 37 entsteht und geben Sie diesen auf `stdout` aus.

```
public class Modulo {
    // Rekursive Variante der Modulofunktion
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

    // Iterative Variante der Modulofunktion
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

    // main-Methode
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
}
```

## Aufgabe 4 (16 Punkte)

Ein Keller (engl. Stack) ist eine Datenstruktur, die es erlaubt, Werte oben auf den Stack zu legen und den obersten Wert auszulesen und wieder zu entfernen. Stacks können über einfach verkettete Listen implementiert werden. Dabei genügt es, Elemente vorne in die Liste einzufügen bzw. vorne in der Liste wieder zu löschen. Verwenden Sie für diese Aufgabe folgende Definition eines einzelnen Listenelements:

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

    ListElement(int value) {
        this.value = value;
        this.next = null;
    }
}
```

Implementieren Sie eine Klasse `Stack`, um Werte des Typs `int` zu speichern. Stellen Sie folgende Methoden zur Verfügung:

- die Methode `push`, um einen neuen Wert auf den Stack zu legen, und
- die Methode `pop`, um den obersten Wert vom Stack zu nehmen und zurückzugeben. Ist der Stack leer, soll eine `StackEmptyException` geworfen werden. Die Klasse `StackEmptyException` ist von Ihnen mit den beiden für Exceptions typischen Standardkonstruktoren zu implementieren.

```
// Klasse StackEmptyException
-------------------------------------------------------------


    // Standard-Konstruktor ohne Parameter
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------


    // Standard-Konstruktor mit String-Parameter
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
}
```

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

    // Methode push
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

    // Methode pop
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
}
```

## Aufgabe 5 (16 Punkte)

1. Eine formale Grammatik ist definiert als 4-Tupel $(V_N, V_T, S, R)$. Benennen und definieren Sie die vier Bestandteile einer allgemeinen Grammatik. Welche Form haben die Elemente von $R$?

2. Welche Form haben die Grammatikregeln, wenn es sich um eine kontextfreie Grammatik handelt?

3. Gegeben sind folgende Grammatikregeln mit dem Startsymbol `S`:

   ```
   S →  X Y
   X →  a b          Y →  c d
   X →  a b X        Y →  c d Y
   ```

   (a) Geben Sie die Mengen $V_N$ und $V_T$ an.

   (b) Zeichnen Sie ein einziges Syntaxdiagramm, welches äquivalent zu den obigen fünf Regeln ist.

(c) Geben Sie die Sprache $L$ an, die von dieser Grammatik erzeugt wird. Welches ist das kürzeste Wort $w \in L$?

(d) Zeichnen Sie den Ableitungsbaum für das Wort `abcdcd` der Sprache $L$.

## Aufgabe 6 (24 Punkte)

In dieser Aufgabe geht es um die Modellierung verschiedener Elektrogeräte. Allen Geräten gemeinsam ist, dass sie einen Firmennamen und eine Gerätebezeichnung besitzen. Außerdem wird ein Wert für die maximale elektrische Leistungsaufnahme (in kW) angegeben.

1. Schreiben Sie eine Oberklasse `Elektrogeraet`, die die Gemeinsamkeiten modelliert. Über den Konstruktoraufruf sollen sich die Attribute setzen lassen können. Der lesende Zugriff auf die Attribute ist von außen nicht erlaubt, wohl aber in abgeleiteten Klassen. Überladen Sie die Methode `toString()`, sodass diese die folgende Zeichenkette liefert: [1]

   ```
   <Firmenname> <Gerätebezeichnung> (<max. Leistungsaufnahme> kW)
   ```

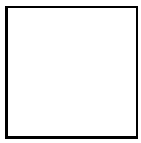
   Der tatsächliche Energieverbrauch hängt vom Elektrogerät selbst ab, beispielsweise davon, ob das Gerät eingeschaltet ist. Stellen Sie sicher, dass jedes von `Elektrogeraet` abgeleitete Gerät eine Methode `getLeistungsaufnahme` zur Verfügung stellt, die die momentane Leistungsaufnahme des Gerätes zurückliefert.

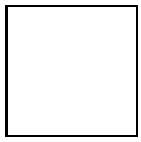
   Die Modifikatoren von Attributen und Methoden/Konstruktoren sind so restriktiv wie möglich zu wählen. Die Klassen sollen sich in beliebigen Packages nutzen lassen.

```
// Klasse Elektrogeraet

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

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

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

[1] Für die Ausdrücke in spitzen Klammern sind die entsprechenden Attributwerte einzusetzen.

```
    // Methode toString
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------


    // Methodendeklaration getLeistungsaufnahme
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
}
```





2. Als spezielles Elektrogerät soll eine Herdplatte modelliert werden. Neben den Eigenschaften eines jeden Elektrogerätes besitzt eine Herdplatte als weitere Eigenschaft die Anzahl der Stufen, in denen sich die Herdplatte einschalten lässt.

   Die Eigenschaften sollen sich (nur) bei der Objekterzeugung setzen lassen. Die Anzahl der Stufen muss dabei nicht zwingend angeben werden. Fehlt diese, soll von zwölf Stufen ausgegangen werden.

   Stellen Sie zwei Methoden zur Verfügung, um die Herdplatte aus- und mit der gewünschten Stufe einzuschalten. Gibt der Anwender beim Einschalten eine zu hohe oder zu niedrige Stufe an, soll die Methode abbrechen und eine Exception der bereits vorhandenen Klasse `InvalidParameterException` mit folgendem Text werfen:

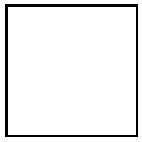
   ```
   Stufe <Stufe> nicht vorhanden
   ```

   Implementieren Sie die Methode `getLeistungsaufnahme`, wie sie von der Oberklasse gefordert wird. Die elektrische Leistungsaufnahme hängt linear von der gewählten Stufe ab und beträgt maximal 2,3 kW.

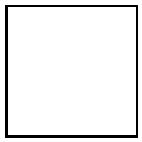
   Modellieren Sie die Klasse `Herdplatte` unter Verwendung der bereits implementierten Klasse `Elektrogeraet`.

```
// Klasse Herdplatte

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

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

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

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

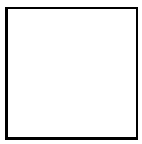
```
    // Methode einschalten
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
```



```
    // Methode ausschalten
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
```



```
    // Methode getLeistungsaufnahme
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
```



```
}
```

3. Einzelne Elektrogeräte (max. 20) können nun an den Stromkreis und damit an einen Stromzähler angeschlossen werden. Die Klasse `Stromzaehler` soll drei Methoden enthalten: eine Methode, um ein als Parameter übergebenes Elektrogerät anzuschließen, eine Methode, um das gewünschte Gerät wieder vom Stromkreis zu trennen, und eine Methode, um den aktuellen Zählerstand abzufragen.

   Zeichnen Sie für die Klassen `Stromzaehler`, `Elektrogeraet` und `Herdplatte` ein UML-Klassendiagramm, das alle Attribute und Methoden der einzelnen Klassen enthält und aus dem die Beziehungen der Klassen untereinander hervorgehen. Die Konstruktoren müssen darin nicht enthalten sein.