PD Dr.-Ing. Elmar Nöth
Lehrstuhl für Mustererkennung
Universität Erlangen-Nürnberg

09. April 2008

Schriftliche Prüfung zu der Vorlesung Grundlagen der Informatik I April 2008

(Prüfungsnummern 11 204 111, 11 204 112, 11 104 112, 14 104 112, 14 177 1100 , 11 177 1100, 11 179 1080, 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 (18 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 09. April 2008 .....
(Unterschrift)

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

ja: ☐ nein: ☐

Erlangen, der 09. April 2008 .....
(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 der folgenden Aussagen ist nicht gültig:
   (a) Ein Ausdruck liefert immer einen Wert zurück. ☐
   (b) Ein Ausdruck kann aus einem einzelnen Literal bestehen. ☐
   (c) Ein Ausdruck kann keine triadischen Operatoren enthalten. ☐

2. Welche Aussage trifft auf formale Sprachen zu:
   (a) Bei regulären (linearen) Grammatiken (Typ 3) unterscheidet man zwischen rechts- und linksradikalen Regeln. ☐
   (b) Die Grammatik definiert die Syntax einer Sprache und die Semantik definiert die Bedeutung der Zeichen. ☐
   (c) Uneingeschränkte Grammatiken können durch die Backus-Naur Form dargestellt werden. ☐

3. Welche der Aussagen trifft auf die Rekursion zu:
   (a) Eine Funktion heißt linear rekursiv, wenn in jedem Zweig einer Fallunterscheidung die Funktion höchstens einmal aufgerufen wird. ☐
   (b) Der Speicherbedarf ist bei rekursiven Funktionen immer geringer als bei iterativen Methoden. ☐
   (c) Als lineare Rekursion bezeichnet man die rekursive Berechnung von ausschließlich linearen Funktionen. ☐

4. Welche Aussage trifft für Induktionsbeweise zu:
   (a) Induktionsbeweise können nur für kleine Zahlen geführt werden. ☐
   (b) Induktionsbeweise besitzen eine Induktionsspannung, eine Induktionsannahme und einen Induktionsschritt. ☐
   (c) Unter der Voraussetzung, dass die Induktionsannahme für i gilt, wird im Induktionsschritt bewiesen, dass die Rekursionsformel auch für $i+1$ gültig ist. ☐

5. Für den klassischen Universalrechner (von-Neumann-Rechner) gilt folgende Aussage:
   (a) Der klassische Universalrechner besitzt auf alle Fälle eine Tastatur, einen Monitor und eine Maus bzw. Touchpad. ☐
   (b) Rechenwerk und Speicherwerk sind Bestandteile des klassischen Universalrechners und werden auch zur Zentraleinheit zusammengefaßt. ☐
   (c) Das Speicherwerk gestattet es, Informationen aufzunehmen, längere Zeit zu speichern und wiederzufinden. ☐

6. Welche der folgenden Aussagen trifft auf Unicode zu:

   (a) Bei UTF-32 werden Zeichen mit 32 Byte kodiert. ☐

   (b) Bei UTF-8 wird immer genau ein Byte zur Kodierung verwendet. ☐

   (c) UTF-16 wird in Java für die Kodierung des Grunddatentyps *Character* verwendet. ☐

7. Was trifft auf das OOP-Konzept der Abstraktion nicht zu?

   (a) Abstraktion ermöglicht die Verwendung von Objekten ohne Beachtung der genauen Implementierung. ☐

   (b) Bei der Abstraktion können nur öffentliche Klassen verwendet werden. ☐

   (c) Beschreibung von gleichartigen Objekten mittels Klassen. ☐

8. Welche Arten der Traversierung von Bäumen gibt es wirklich?

   (a) Vorordnung, Inordnung, Nachordnung ☐

   (b) Vorordnung, Mittelordnung, Nachordnung ☐

   (c) Vorordnung, Unordnung, Nachordnung ☐

9. Welche Aussage gilt bezüglich Methoden in Java:

   (a) Methoden besitzen immer einen Rückgabetyp. ☐

   (b) Methoden werden auch als Konstruktoren bezeichnet. ☐

   (c) Methoden besitzen immer mindestens einen formalen Parameter. ☐

10. Welche Aussage trifft bezüglich Sortierverfahren zu:

    (a) Mit *Greedy*-Verfahren können Felder immer *in-place* sortiert werden. ☐

    (b) *Greedy*-Verfahren sind in jedem Fall langsamer als *Divide-and-Conquer*-Verfahren. ☐

    (c) Bei Sortieren durch Mischen (*merge sort*) handelt es sich um ein *Divide-and-Conquer*-Verfahren. ☐

Aufgabe 2 (14 Punkte)

1. Das folgende Programm soll dazu dienen, das Maximum eines `double`-Feldes zu bestimmen. Korrigieren Sie alle syntaktischen sowie semantischen Fehler. Falls das Feld keine Werte enthält, soll der kleinst mögliche Wert zurückgegeben werden.

```
public class Fehlerteufel {

    public static double[] maximum(double[] feld) {
        if( feld = null ) {
            throw new NullPointerException();
        }

        float max = Double.MAX_VALUE;
        for( int i = 0; i<feld.length; i++ ) {
            if( max < feld[i] ) feld[i] = max;
        }

        return maximum;
    }

    public static void main( String[] args ) {
        double feld = {1.2, 3.6, 12.1, 14.5, 12};
            System.out.println( "Das Maximum ist: " + maximum( feld ) )
    }

}
```

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

```
public class Scope {

    static int a = 4;
    static int b = 1;

    public static void multFeld(int a, double[] feld) {
        feld[a] *= 0x2;
        System.out.println( "feld[" + a + "] = " + feld[a]);
        a++;
        b++;
    }

    public static void main(String[] args) {
        int a = 010;
        int b = 0;
        double[] array = {2, 3, 4, 5};
        for( int i=0; i<array.length/2; i++ ) {
            multFeld(b, array);
            System.out.println( "a = " + a );
            System.out.println( "b = " + b );
        }
    }

}
```

Auf `stdout` wird ausgegeben:

Aufgabe 3 (10 Punkte)

Die Quadratwurzel einer Zahl kann rekursiv wie folgt approximiert werden:

$$\sqrt{a} = \text{wurzel}(a, x) := \begin{cases} x & \text{wenn} \quad |x^2 - a| < \varepsilon \\ \text{wurzel}(a, \frac{x+\frac{a}{x}}{2}) & \text{sonst} \end{cases} \tag{1}$$

Dies wird auch Methode von Heron genannt. Der Parameter x stellt dabei den aktuellen Schätzwert da. Als erste Schätzung soll ein Wert $x = 1$ angenommen werden. ε repräsentiert den maximalen Fehler der Schätzung. Nehmen Sie einen Wert $\varepsilon = 0.0001$ an. Schreiben Sie eine iterative sowie rekursive Variante der Methode von Heron. Rufen Sie beide Varianten aus einer `main`-Methode auf. Die Verwendung von vordefinierten Methoden zur Berechnung der Wurzel ist nicht gestattet.

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

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

// Rekursive Variante der Funktion

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

// main Methode

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

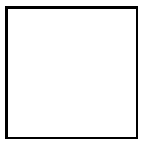
Aufgabe 4 (16 Punkte)

Die Knoten eines binären Suchbaumes sollen durch folgende Klasse modelliert werden:

```
class Knoten {
    int wert;                    // Wert an Knoten
    Knoten links;                 // Linker Nachfolger
    Knoten rechts;                // Rechter Nachfolger

    Knoten(int value) {           //Konstruktor
        this.value = value;
        this.links = null;
        this.rechts = null;
    }
}
```

Schreiben Sie eine Methode `public void clear()`, welche den Baum leert. Implementieren Sie weiterhin eine Methode `public boolean exists(int wert)`, welche `true` liefert, wenn `wert` im Baum enthalten ist und andernfalls `false`. Schließlich soll eine Methode `maximum` implementiert werden, die den maximalen Wert des Baumes zurückliefert. Falls der Baum nicht existiert, soll der Wert `Integer.MIN_VALUE` zurückgegeben werden.

```
public class Baum {

    // Attribute der Klasse Baum
    private Knoten wurzel = null;

    // Methode clear
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
```



```
    // Methode exists
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------

    // Methode maximum
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
}
```

Aufgabe 5

(16 Punkte)

1. Sortierverfahren

   (a) Beschreiben Sie kurz das Prinzip von Sortieren durch Auswählen.

   (b) Welche Komplexität besitzt dieses Sortierverfahren?

   (c) Welche zwei effizienteren Sortierverfahren kennen Sie und auf welchem Konzept basieren diese?

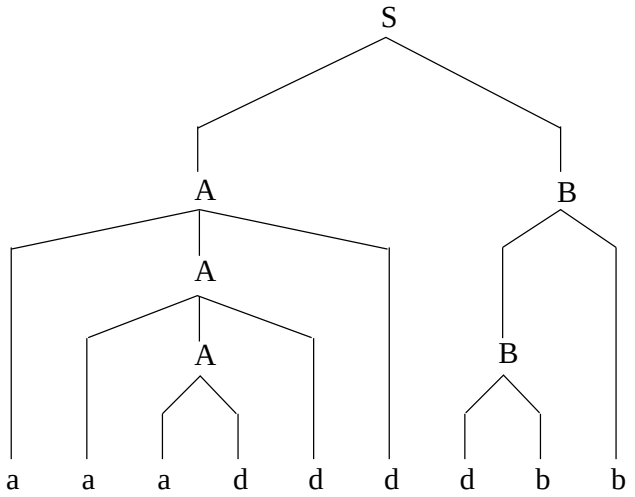
2. Gegeben sind folgende Grammatikregeln mit dem Startsymbol S:

$$R = \{S \to AB,\ A \to aAd,\ A \to ad,\ B \to dBb,\ B \to dB,\ B \to db\}$$

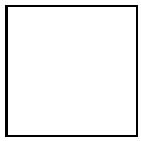
(a) Geben Sie die Mengen der terminalen Symbole V_T und der nichterminalen Symbole V_N der gegebenen Grammatik an. Wie lautet das kürzeste Wort?

(b) Wie lautetet die allgemeine Form der Regeln, wenn es sich um eine kontextfreie Grammatik handelt?

(c) Läßt sich der folgende Ableitungbaum mit Hilfe der gegebenen Regeln erstellen? Begründen Sie kurz ihre Antwort.



(d) Erstellen Sie für das Wort `adddbb` einen gültigen Ableitungsbaum.



Aufgabe 6 (24 Punkte)

1. In dieser Aufgabe sollen Getränkekisten modelliert werden. Eine Kiste wird durch ein Attribut beschrieben, welches die maximale Anzahl der enthaltenen Flaschen enthält. Weiterhin wird ein Attribut benötigt, welches die Zahl der verbleibenden vollen Flaschen angibt. Jeder tausendste Kasten soll einen Preis gewinnen. Deshalb muss sichergestellt werden, dass jede Getränkekiste eine eindeutige Nummer besitzt. Modellieren Sie dies mit Hilfe entsprechender Attribute. Ein Standardkasten soll maximal 15 Flaschen enthalten.

   Modellieren Sie weiterhin die folgende Funktionalität:

   - `public int getKastenNummer()`: Gibt die Nummer der Getränkekiste zurück.
   - `public int nimmFlasche()`: Entfernt eine Flasche aus dem Kasten und gibt die Anzahl der verbleibenden vollen Flaschen zurück. Falls der Kasten bereits leer ist, soll eine `LeererKastenException` geworfen werden. Implementieren Sie auch die Klasse `LeererKastenException`.

   Stellen Sie weiterhin einen Standardkonstruktor zur Verfügung. Ein weiterer Konstruktor hat die Anzahl der Flaschen als Parameter. Die Sichtbarkeit der verwendeten Attribute soll so restriktiv wie möglich definiert werden.
   Hinweis: Verwenden Sie für Ihren Programmcode die umseitig definierten Vorlagen!

```
// Klasse Kasten

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

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

    // Konstruktoren
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
    -------------------------------------------------------------
```

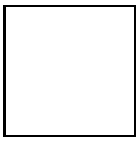
```
    // Methoden
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------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

// Exception LeererKastenException

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

2. Eine erweiterte Version der Getränkekiste stellt ein Bierkasten dar. Leiten Sie eine Klasse `Bierkasten` von `Kasten` ab. Zusätzlich zu den bestehenden Attributen soll eine Variable vom Typ `String` bereitgestellt werden, welche die Biersorte bezeichnet. Weiterhin soll das Alter des Käufers gespeichert werden.

   Stellen Sie folgende Funktionalität zur Verfügung:

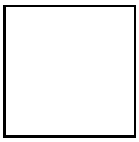
   - Stellen Sie einen Konstruktor zur Verfügung, der das Alter des Käufers sowie die Marke des Bieres übergeben bekommt. Standardmäßig sollen in alle Bierkästen maximal 20 Flaschen passen.
   - Es soll nur eine Flasche aus dem Kasten entfernt werden können, wenn das Alter des Käufers ≥ 16 ist. Überschreiben Sie hierfür die Methode `public int nimmFlasche()`.

```
// Klasse Bierkasten
```

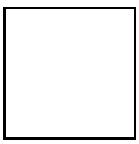


```
-----------------------------------------------------------------
```

```
    // Attribute
    -----------------------------------------------------------------
    -----------------------------------------------------------------
    -----------------------------------------------------------------
```



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



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

}
```

3. Modellieren Sie eine Klasse `Getränkemarkt`. Diese soll als Attribute die Anzahl der Mitarbeiter, die maximale Kapazität (wieviele Kisten können gelagert werden) und ein Feld mit den Getränkekisten enthalten. Stellen Sie entsprechende Konstruktoren zur Verfügung, welche einen Getränkemarkt erstellen. Weiterhin sollen folgende Methoden zur Verfügung gestellt werden: `public int verkaufe()` und `public void bestelle(int anzahl)`.

   Zeichnen Sie ein UML-Diagramm, welches die Beziehung zwischen den verwendeten Klassen darstellt. Die Klasse `Getränkemarkt` soll *nicht* implementiert werden! Evtl. auftretende Exceptions müssen nicht im UML-Diagramm kenntlich gemacht werden. Alle Attribute sollen so restriktiv wie möglich definiert werden.