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

26. September 2007

Schriftliche Prüfung zu der Vorlesung Grundlagen der Informatik I September 2007

(Prüfungsnummern 11 204 111, 11 204 112, 11 104 112, 14 104 112, 14 177 1100 , 11 177 1100, 11 179 1008, 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 26. September 2007 .....
(Unterschrift)

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

ja: ☐ nein: ☐

Erlangen, der 26. September 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. Welche der folgenden Variablendeklarationen ist in der Prorammiersprache Java gültig:
   - (a) `int _1;` ☐
   - (b) `int 1a;` ☐
   - (c) `int while;` ☐

2. Welche Aussage gilt für Methoden in der Programmmiersprache Java:
   - (a) Man kann innerhalb einer Klasse zwei Methoden mit demselben Methodennamen und denselben Parametern aber verschiedenen Rückgabetypen definieren. ☐
   - (b) Eine Methode, die nichts zurückliefern soll, hat den Rückgabetyp `null`. ☐
   - (c) Methoden bestehen aus Rückgabetyp, Methodenname, einer Liste aus optionalen Parametern und dem Methodenrumpf. ☐

3. Welche der Aussagen trifft auf Rekursionen zu:
   - (a) Unter anderem werden folgende Rekursionstypen unterschieden: lineare Rekursion, kaskadenartige Rekursion, nicht-rekursive Rekursion. ☐
   - (b) Bei der kaskadenartigen Rekursion tritt eine baumartige Aufrufstruktur auf. ☐
   - (c) Als lineare Rekursion bezeichnet man die rekursive Berechnung von ausschließlich linearen Funktionen. ☐

4. Welche Aussage trifft für Induktionsbeweise zu:
   - (a) Im Induktionsschritt wird unter der Voraussetzung, dass die Induktionsannahme für i gilt, bewiesen, dass die Annahme auch für $i+1$ gilt. ☐
   - (b) Bei einfachen Induktionsbeweisen ist der Induktionsanker nicht nötig. ☐
   - (c) Induktionsbeweise können nur für Zahlen kleiner 100 geführt werden. ☐

5. Für den klassischen Universalrechner (von-Neumann-Rechner) gilt folgende Aussage:
   - (a) Windows XP kann auf dem klassischen Universalrechner nur im DOS-Modus betrieben werden. ☐
   - (b) Rechenwerk und Steuerwerk sind Bestandteile des klassischen Universalrechners und werden auch zur Zentraleinheit zusammengefasst. ☐
   - (c) Wesentliche Bestandteile des klassischen Universalrechners sind unter anderem: Rechenwerk, Steuerwerk, Eingabeeinheit und USB-Anschluss. ☐

6. 1 Byte entspricht:

   (a) 2^3 Bit. ☐

   (b) 2^4 Bit. ☐

   (c) 4 Bit. ☐

7. Welches sind die vier Grundprinzipien der objektorientierten Programmierung:

   (a) Abstraktion, Kapselung, Vererbung, Polymorphie. ☐

   (b) Abstraktion, Bijektion, Vererbung, Homomorphismus. ☐

   (c) Abstraktion, Kapselung, Rekursion, Heterogenität. ☐

8. Welche Modifikatoren sind für Variablen in Java gültig:

   (a) `public, final, static` ☐

   (b) `private, constant, super` ☐

   (c) `protected, final, this` ☐

9. Welche Aussage gilt bezüglich Konstruktoren von Klassen:

   (a) Konstruktoren dürfen eine Anweisung der Form `return;` enthalten. ☐

   (b) Bei jeder Klassendefinition muss explizit ein Konstruktor definiert werden. ☐

   (c) Eine Klasse kann nur genau eine Konstruktordefinition enthalten. ☐

10. Welche Aussage trifft bezüglich Sortierverfahren zu:

    (a) *QuickSort* ist ein *divide-and-conquer*-Verfahren, welches nicht *in-place* sortieren kann. ☐

    (b) *QuickSort* hat immer eine Komplexität von $O(\log n)$. ☐

    (c) Das optimale Pivotelement, um mit *QuickSort* eine Folge F zu sortieren, ist der Median von F. ☐

Aufgabe 2 (14 Punkte)

1. Das folgende Programm soll dazu dienen, alle Einträge eines Feldes vom Typ `int` geradzahlig zu machen. Dazu wird der Wert jedes Feldelementes modulo 2 berechnet und der Feldeintrag inkrementiert, falls er ungerade ist. 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 class Geradzahlig {

    public static void geradzahligesFeld(int feld) {
        if ( feld.length > 0 & feld != null ) {
            for (int i = 0; i < feld.length - 1; i++) {
                if (feld[i] % 2 = 0) {
                    feld[i]+=2;
                }
            }
        }
    }

    public static void main(String[] args) {

        int[] feld = 1, 2, 3, 4.0, 5, 6, 7, 8 ;
        geradzahligesFeld(feld);

}
```

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

```
public class Scope {

    int feld = 1;

    public static void inkrementieren(int[] a, int b, int c) {
        a[0]++;
        a[2]++;
        a[4]++;
        b++;
        c++;
    }

    public static void main(String[] args) {
        int a = 0x1;
        int b = 2;
        int[] feld = { 1, 2, 0x3, 011, 0x11 };

        System.out.println("a= " + a + "  b= " + b);

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

        inkrementieren(feld, a, b);
        System.out.println("a= " + a + "  b= " + b);
        for (int i = 0; i < feld.length; i++) {
            System.out.println("feld[" + i + "]= " + feld[i]);
        }
    }
}
```

Auf `stdout` wird ausgegeben:

Aufgabe 3 (10 Punkte)

Der größte gemeinsame Teiler (ggT) zweier Zahlen $m, n \in N$ kann **rekursiv** wie folgt definiert werden:

$$\text{ggT}(m,n) := \begin{cases} n & \text{wenn} \quad m \bmod n = 0 \\ \text{ggT}(n, m \bmod n) & \text{sonst} \end{cases} \tag{1}$$

Diesen Algorithmus nennt man auch den *Euklidischen Algorithmus*. Eine **iterative** Beschreibung dieses Algorithmus für die Bestimmung des ggT zweier Zahlen m und n sieht folgendermaßen aus:

1. Dividiere m durch n mit Rest r.
2. Solange $r \neq 0$: ersetze m durch n und n durch r und dann r mit $m \bmod n$.
3. Das Ergebnis ist n.

Schreiben Sie eine rekursive und eine iterative Methode um den ggT zweier natürlicher Zahlen mittels des Euklidischen Algorithmus zu berechnen und rufen Sie beide Funtionen jeweils mit $m = 25$ und $n = 8$ aus einer `main`-Methode auf. Die Ergebnisse müssen nicht ausgegeben werden. Benutzen Sie zur Lösung die Vorlage auf der nächsten Seite.

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

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

    // Rekursive Variante der Funktion

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

    // main Methode

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

Aufgabe 4 (16 Punkte)

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

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

Eine Klasse `Zahlenfolge` soll im Sinne einer verketteten Liste genutzt werden um die rekursiv definierte Zahlenfolge $a_{n+1} = 2a_n + 1$ mit $a_0 = 0$ zu modellieren. Die ersten Elemente der Zahlenfolge lauten also $a_0 = 0$, $a_1 = 1$, $a_2 = 3$, $a_3 = 7$, $a_4 = 15$, usw. In dieser Reihenfolge sollen die Elemente der Zahlenfolge als aufeinanderfolgende Folgenelemente (jeweils vom Typ `FolgenElement`) repräsentiert sein. Schreiben Sie eine Methode `naechstesFolgenElementEinfuegen()`, welche ein Folgenelement am Ende der Zahlenfolge einfügt. Dieses Element soll den korrekten nächsten Wert der Zahlenfolge beinhalten. Implementieren Sie weiterhin eine Methode `loeschen()`, welche alle Elemente der Zahlenfolge löscht. Des Weiteren implementieren Sie eine Methode `berechneSumme()`, welche die Summe aller aktuell in der Zahlenfolge enthaltenen Elemente berechnet und zurückgibt. Das Klassenattribut `anfang` soll eine Referenz auf a_0 sein; das Klassenattribut `ende` eine Referenz auf das Element mit dem Wert a_n, also das letzte aktuell vorhandene Folgenelement. Zur Lösung benutzen Sie bitte die Vorlage auf der nächsten Seite.

```
public class Zahlenfolge {

    // Attribute der Klasse Zahlenfolge
    private FolgenElement anfang = null;
    private FolgenElement ende = null;

    // Methode naechstesFolgenElementEinfuegen
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    // Methode loeschen
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    // Methode berechneSumme
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
}
```

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. Es seien $V_N = \{S, A, B, C\}$ und $V_T = \{a, b, c, d\}$. S sei das Startsymbol. Geben Sie drei gültige Elemente von R an, wenn es sich um eine kontextfreie Grammatik handelt.

3. Gegeben sind folgende Grammatikregeln mit dem Startsymbol S:

$$R = \{S \to ASB,\ S \to AB,\ A \to abA,\ A \to ab,\ B \to bcB,\ B \to bc\}$$

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

   (b) Geben Sie ein einziges Syntaxdiagramm an, welches äquivalent zu den obigen sechs Regeln ist.

(c) Geben Sie die Sprache L an, die von dieser Grammatik erzeugt wird. Gibt es ein Wort der Länge 2? Wenn ja, geben Sie ein Beispiel. Wenn nein, geben Sie das kürzestmögliche Wort an.

(d) Geben Sie den Ableitungsbaum des Wortes `abababcbc` an.

Aufgabe 6 (24 Punkte)

1. In dieser Aufgabe soll ein Roboter modelliert werden. Als Basisfunktionalität soll dieser vorwärts fahren und sich um beliebige Winkel drehen können. Implementieren Sie hierzu die folgenden Methoden:

   - `public void drehe(double winkel)`: Dreht die Blickrichtung des Roboters um den winkel `winkel` (im Bogenmaß) und
   - `public void fahre(double strecke)`: Bewegt den Roboter um die Strecke `strecke` (gegeben in Metern) in Richtung der Ausrichtung des Roboters. Falls die zu fahrende Strecke negativ ist, soll eine `NegativeStreckeException` geworfen werden, welche mit dem parameterlosen Konstruktor und dem Konstruktor, der einen `String` übergeben bekommt, zu implementieren ist.

   Stellen Sie weiterhin einen Konstruktor zur Verfügung, welcher die initiale Position und Richtung des Roboters übergeben bekommt. Die Sichtbarkeit der verwendeten Attribute soll so restriktiv wie möglich definiert werden.

   Hinweis: Sie benötigen Attribute für die folgenden Größen: x- und y-Position sowie den Winkel der Ausrichtung des Roboters. Wenn der Roboter im Ursprung steht und nach Westen ausgerichtet ist, besitzt er die folgende Konfiguration: `xPos=0.0`, `yPos=0.0` und `winkel=`π. Die x/y Translation des Roboters lässt sich mittels `Math.cos(winkel)*strecke` bzw. `Math.sin(winkel)*strecke` berechnen.
   Nutzen Sie zur Lösung die Vorlage auf der nächsten Seite.

```
// Klasse Roboter

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

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

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

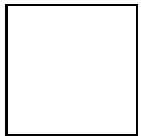
```
    // Methoden
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------
    ------------------------------------------------------------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


// Exception NegativeStreckeException

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

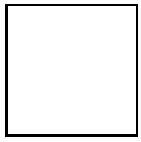
2. Eine erweiterte Version des Roboters soll eine Batterie, von welcher die Kapazität bekannt ist, besitzen. Die Entladung der Batterie soll 10 mAh pro gefahrenem Meter entsprechen. Wenn die Batterie vollständig entladen ist, muss der Roboter stehen bleiben. Überschreiben Sie hierzu die Methode `fahre`. Implementieren Sie weiterhin einen Konstruktor, welcher den Roboter im Ursprung platziert, nach Westen ausrichtet und die Batterie auf eine initiale Kapazität von 200 mAh setzt.

```
// Klasse ErweiterterRoboter
```

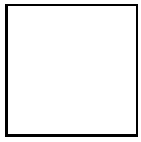


```
-------------------------------------------------------------

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



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



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

}
```

3. Um die Roboter in einem Applet anzeigen zu können, müssen Sie eine weitere Klasse `RoboterApplet` entwerfen, welche von der Klasse `Applet` abgeleitet werden muss. Weiterhin müssen die Methoden `public void paint(Graphics g)` und `public void init()` der Klasse `Applet` überschrieben werden. Die Roboter sollen in einem Feld vom Typ `ErweiterterRoboter` gespeichert werden. Insgesamt sollen bis zu 25 Roboter gleichzeitig dargestellt werden. Zeichnen Sie ein UML-Diagramm, welches die Beziehung zwischen den verwendeten Klassen darstellt. Die Klasse `RoboterApplet` soll *nicht* implementiert werden! Evtl. auftretende Exceptions und die Klasse `Applet` müssen nicht im UML-Diagramm kenntlich gemacht werden.