Fakultät IV
Elektrotechnik/Informatik

Klausur Einführung in die Informatik II für Elektrotechniker

28. Februar 2005

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung (bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	7	
2	7	
3	5	
4	9	
5	8	
6	8	
Summe	44	

Spielregeln (**Jetzt lesen!**):

- Benutzen Sie für die Lösung der Aufgaben **nur** das mit diesem Deckblatt ausgeteilte Papier. Lösungen, die auf anderem Papier geschrieben werden, können **nicht** bewertet werden. Schreiben Sie ihre Lösung auch auf die Rückseiten der Blätter; benötigen Sie für eine Lösung mehr als ein Blatt, finden Sie am Ende der Klausur Leerblätter. Zusätzliches Papier können Sie von den Tutoren bekommen.
- Tragen Sie jetzt (vor Beginn der eigentlichen Bearbeitungszeit !!!) auf **allen** Blättern ihren Namen und ihre Matrikelnummer ein.
- Schreiben Sie deutlich! Unleserliche oder zweideutige Lösungen können nicht gewertet werden.
- Schreiben Sie **nicht** mit Bleistift und **nicht** mit rotem oder grünem Stift (das sind die Farben für die Korrektur).
- Lesen Sie die Aufgaben jeweils bis zum Ende durch; oft gibt es hilfreiche Hinweise!
- Kommentare kosten Zeit; kommentieren Sie ihr Programm nur da, wo der Code alleine nicht verständlich wäre.
- Wir weisen noch einmal darauf hin, daß die Benutzung von Taschenrechnern und anderen elektronischen Hilfsmitteln nicht gestattet ist.

Viel Erfolg!

Aufgabe 1 (7 Punkte) Theorie.

1. (1 Punkt) Was verstehen Sie unter *Rekursion*?

2. (1 Punkt) Was verstehen Sie unter dem Verfahren der *schrittweisen Approximation*?

3. (2 Punkte) Nennen Sie zwei Möglichkeiten, eine große Anzahl von Daten so in Datenstrukturen zu speichern, dass effizient nach einzelnen Daten gesucht werden kann.

4. (2 Punkte) Tragen Sie die folgenden Zahlen so in die Blätter des unten stehenden Baumes ein, dass ein sortierter Baum entsteht: 4, 8, -1, 12, 9, 2, 0. Für die inneren Knoten sind geeignete Schlüssel einzutragen.

5. (1 Punkt) Was verstehen Sie unter einer *abstrakten Methode* und wozu dient sie?

Aufgabe 2 (7 Punkte) Java.

1. (1 Punkt) Welche Klassen dürfen auf das Attribut `x` einer Klasse namens `K` zugreifen, das folgendermaßen definiert ist:

```
package p;
class K {
  protected int x = 42;
}
```

2. (1 Punkt) Welche Klassen dürfen auf das Attribut `x` einer Klasse namens `K` zugreifen, das folgendermaßen definiert ist:

```
package p;
class K {
  final int x = 42;
}
```

3. (2 Punkte) Nennen Sie mindestens zwei Anwendungsfälle für Attribute und Methoden, die mit dem Schlüsselwort `static` definiert werden.

4. (3 Punkte) Welche Fehler enthält folgende Java-Klasse? Geben Sie jeweils die Zeilennummer an und beschreiben Sie den Fehler. Folgefehler (also Fehler, die aus anderen Fehlern resultieren) sollen ignoriert werden.

```
 1 abstract class Algorithm {
 2
 3 }
 4
 5 class Sortieren implements Algorithm {
 6     public void sort(int[] a) {
 7         for (int i = 0; i < a.length; i++) {
 8             int j = i - 1;
 9             a[j] = a[i];
10         }
11         this.a = a;
12     }
13     public static void main(String[] args) {
14         int[] a = Terminal.askIntArray("a=");
15         Quick q = new Quick();
16         q.sort(a);
17     }
18 }
19
20 class Quick extends Sortieren {
21     public int[] a;
22     Quick() {
23         this.a = new int[2];
24     }
25 }
```

Aufgabe 3 (5 Punkte) Numerik.

Die *Exponentialfunktion* e^x steht in normalen Java-Systemen mit der Methode `double Math.exp(double x)` zur Verfügung. Wenn diese nicht verwendet werden kann oder soll, kann man stattdessen eine eigene Approximation programmieren. Folgende Reihenformeln für e^x lassen sich alleine mit den Grundrechenoperationen `+` und `*` umsetzen:

$$e^x = \sum_{k=0}^{\infty} \prod_{i=1}^{k} \frac{x}{i}$$

1. (2 Punkte) Schreiben Sie eine Methode `double approx(double x, int n)`, welche die n-te Teilsumme nach folgender Formel berechnet:

$$approx(x, n) = \sum_{k=0}^{n} \prod_{i=1}^{k} \frac{x}{i}$$

2. (3 Punkte) Schreiben Sie eine Methode `double exp(double x, double eps)`, welche den Wert e^x unter Verwendung von `approx` mit Genauigkeit *eps* berechnet.

Aufgabe 4 (9 Punkte) Abstrakte Datentypen.

In dieser Aufgabe soll ein abstrakter Datentyp `FlexArray` erstellt werden, der ähnlich den normalen Java-Arrays Werte speichern kann, auf die dann mit einem Index zugegriffen wird.

Ein `FlexArray` kann auf zwei Arten eingesetzt werden:

- Für dichte Arrays, wenn also wenig Nullen enthalten sind, speichert es die Werte in einem Array,
- für dünn besetzte Arrays, wenn viele Nullen enthalten sind, speichert es die Werte in einer einfach verketteten Liste. Dabei müssen nur von Null verschiedene Werte tatsächlich gespeichert werden, so dass die Liste wesentlich kürzer sein kann.

Bei der Erzeugung eines `FlexArray`-Objektes gibt man an, ob es für dichte oder dünne Arrays eingesetzt werden soll. Wenn auf ein Element zugegriffen wird, das noch nicht mit einem Wert belegt wurde, dann soll der Wert 0 zurückgegeben werden.
Die folgende Klasse für einfach verkettete Listenzellen ist bereits gegeben:

```
class LongCell {
    int index;
    long value;
    LongCell next;

    LongCell (int index, long value, LongCell next) {
        this.index = index;
        this.value = value;
        this.next = next;
    } // Konstruktor
} // LongCell
```

Beispiel:

- Ein Array mit wenigen Nullen, z.B. das Array {`3L, 9L, -1L, 12L, 0L, 3L, 6L`} wird in der Klasse `FlexArray` als folgendes Array dargestellt:

  3L 9L -1L 12L 0L 3L 6L

- Ein Array mit vielen Nullen, z.B. {0L, 0L, 0L, 8L, 0L, 0L, 4L} wird durch diese Liste dargestellt:

  3 8L 6 4L

  oder durch diese nicht optimale Liste:

  6 4L 1 0L 3 8L

1. (3 Punkte) Schreiben Sie eine Klasse `FlexArray`, die die folgenden Attribute besitzt:

   - Die Länge des Arrays (als `int`-Wert),
   - einen Wahrheitswert, der angibt, ob ein dichtes oder dünnes Array dargestellt wird,
   - den eigentlichen Inhalt des Arrays. Dieser wird über zwei Attribute dargestellt, von denen das nicht benutzte `null` ist:
     - Ein `long`-Array der richtigen Länge, welches für dichte Arrays eingesetzt wird,
     - der Anfang der Liste, die für dünn besetzte Arrays benutzt wird.

   Die Attribute sollen von außen nicht zugreifbar sein.

   Schreiben Sie außerdem einen geeigneten Konstruktor, dem die Länge des Arrays und ein Wahrheitswert übergeben wird, der `false` für ein dicht besetztes Array und `true` für ein dünn besetztes Array sein soll.

2. (1 Punkt) Schreiben Sie für die Klasse `FlexArray` eine Methode `int length ()`, die die Länge des Arrays zurückgibt.

3. (2 Punkte) Schreiben Sie für die Klasse `FlexArray` eine Methode `long get (int index)`, die das gespeicherte Element am gegebenen Index zurückliefert. Diese Methode muss die zwei möglichen internen Darstellungen berücksichtigen. Gehen Sie davon aus, dass die Methode immer mit einem gültigen Index aufgerufen wird.

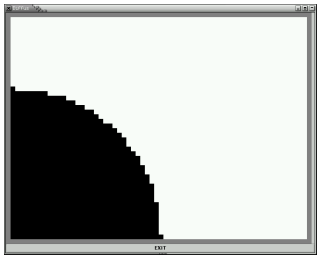
4. (3 Punkte) Schreiben Sie für die Klasse `FlexArray` eine Methode `void set (int index, long value)`, die das Element mit dem gegebenen Index auf den Wert `value` setzt. Diese Methode muss die zwei möglichen internen Darstellungen berücksichtigen. Gehen Sie davon aus, dass die Methode immer mit einem gültigen Index aufgerufen wird.

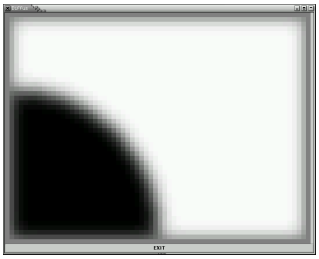
Diese Seite wurde absichtlich freigelassen.

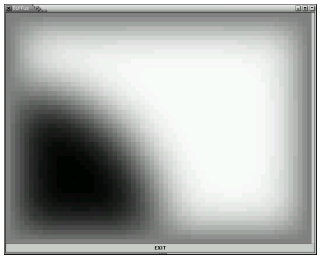
Aufgabe 5 (8 Punkte) Matrixdiffusion.

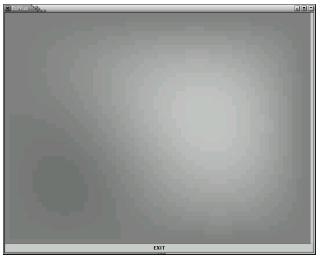
Durch ein einfaches numerisches Verfahren kann simuliert werden, wie sich eine Meßgröße (zum Beispiel Temperatur oder elektrische Ladung) im Lauf der Zeit in einem leitenden Medium verteilt.
Dazu werden Meßpunkte in einem rechteckigen Raster angeordnet. Der Zustand zu einem Zeitpunkt wird als Matrix von Meßwerten dargestellt. Der Übergang zum nächsten Zeitpunkt kann durch das Verrechnen benachbarter Matrixelemente simuliert werden.

Beispiel: Eine grafische Darstellung dieses Verfahrens liefert folgende Momentaufnahmen:


Anfangszustand


nach 80 Schritten


nach 640 Schritten


nach 5120 Schritten

1. (4 Punkte) Den Randelementen der Matrix fehlen einige Nachbarelemente. Damit diese nicht als Sonderfall behandelt werden müssen, kann man die Matrix mit zusätzlichen Randelementen ausstatten.

   Schreiben Sie eine Methode

   ```
   double[][] extend(double[][] a, double r)
   ```

   welche die Matrix a rechts, links, oben und unten mit je einer Reihe von Elementen erweitert, die alle mit dem Wert r besetzt sind. Sie können davon ausgehen, dass a tatsächlich eine rechteckige, nichtleere Matrix darstellt.

   Beispiel:

$$\begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix} \quad \text{wird zu} \quad \begin{pmatrix} r & r & \cdots & r & r \\ r & a_{11} & \cdots & a_{1m} & r \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r & a_{n1} & \cdots & a_{nm} & r \\ r & r & \cdots & r & r \end{pmatrix}$$

2. (4 Punkte) Ein Simulationsschritt besteht darin, dass jedes Matrixelement durch eine gewichtete Summe seiner Nachbarn ersetzt wird. Sei die Anordnung eines Elementes m und seiner Nachbarn l, r, o, u wie folgt:

$$\begin{pmatrix} & & \vdots & & \\ & & o & & \\ \cdots & l & m & r & \cdots \\ & & u & & \\ & & \vdots & & \end{pmatrix}$$

Dann ersetze m durch den Wert $m' = d \cdot (l + r + o + u) + (1 - 4d) \cdot m$.

- Die Konstante d hängt von der konkreten Anwendung ab.
- Das Element m darf nicht am Rand der Matrix liegen.

Dabei ist es unerheblich, ob die Nachbarn von m bereits ersetzt wurden oder nicht.

Schreiben Sie eine Methode

```
void step(double[][] a, double d)
```

welche die Ersetzung für alle inneren Elemente der Matrix a mit dem gegebenen d genau einmal durchführt.

Aufgabe 6 (8 Punkte) Vererbung.

In dieser Aufgabe sollen Klassen zur Darstellung von Gewässern, Flüssen und Meeren definiert werden. Gegeben ist folgende Klasse, die die Gemeinsamkeiten aller Gewässer beschreibt.

```
abstract class Gewaesser {
    protected String name;

    // kann von Schiffen befahren werden
    abstract boolean schiffbar ();

    // von diesem Gewaesser kann ein Schiff ein Meer erreichen
    abstract boolean meerErreichbar ();
} // class Gewaesser
```

1. (2 Punkte) Schreiben Sie eine Klasse `Meer`. Diese Klasse soll von `Gewaesser` abgeleitet sein und ein zusätzliches Attribut `flaeche` von geeignetem Typ besitzen.

   Schreiben Sie einen Konstruktor zur Initialisierung aller Attribute und programmieren Sie auch die geerbten abstrakten Methoden.

   Dabei gilt: Alle Meere sind schiffbar und offensichtlich ist auch ein Meer erreichbar.

2. (6 Punkte) Schreiben Sie eine Klasse `Fluss`. Diese Klasse ist ebenfalls von `Gewaesser` abgeleitet und besitzt zwei weitere, von außen nicht zugängliche Attribute:

   - ein Gewässer, in das der Fluss mündet
   - ein Wahrheitswert, der angibt, ob der Fluss schiffbar ist.

   Schreiben Sie außerdem eine Methode `Meer ziel()`, die das Meer bestimmt, in das der Fluss letztendlich mündet.

   Schreiben Sie einen Konstruktor zur Initialisierung aller Attribute und programmieren Sie auch die geerbten abstrakten Methoden. Für die Methode `meerErreichbar` gilt: Der Fluss muss schiffbar sein und von dem Gewässer, in das er mündet, muss ein Meer erreichbar sein.

Diese Seite wurde absichtlich freigelassen.

Diese Seite wurde absichtlich freigelassen.

Diese Seite wurde absichtlich freigelassen.