Fakultät Elektrotechnik/Informatik

Klausur Einführung in die Informatik II für Elektrotechniker 15. Juli 2004

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung
(bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	7	
2	8	
3	5	
4	11	
5	12	
Summe	43	

Spielregeln (**Jetzt lesen!**):

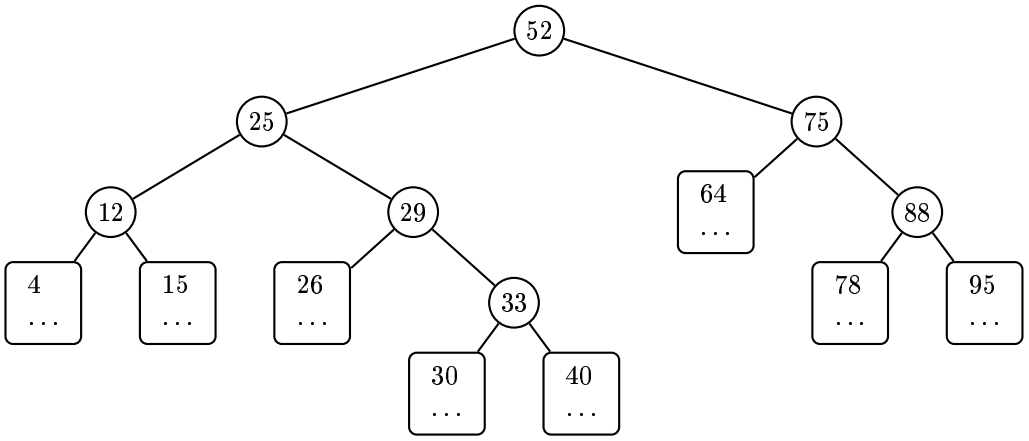
- Benutzen Sie für die Lösung der Aufgaben **nur** das mit diesem Deckblatt ausgeteilte Papier. Lösungen, die auf anderem Papier geschrieben werden, können **nicht** bewertet werden. Schreiben Sie ihre Lösung auch auf die Rückseiten der Blätter; benötigen Sie für eine Lösung mehr als ein Blatt, finden 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. (2 Punkte) Nennen Sie zwei für *effizientes Suchen* wichtige Eigenschaften von Bäumen.

2. (1 Punkt) Eignet sich der folgende Baum zum effizienten Suchen? Begründen Sie Ihre Antwort.



3. (2 Punkte) Welcher der in der Vorlesung behandelten Sortieralgorithmen eignet sich zum Sortieren extrem großer Datenmengen? Begründen Sie Ihre Antwort.

4. (1 Punkt) Was verstehen Sie unter einem *abstrakten Datentyp*?

5. (1 Punkt) Was verstehen Sie unter einer *abstrakten Klasse*?

• AUFGABE 2 (8 Punkte) Java.

1. (1 Punkt) Wie heißt in Java die Oberklasse *aller anderen* Klassen?

2. (2 Punkte) Nehmen Sie an, Sie verwenden in Java einen abstrakten Datentyp `Stack` (Stapel), welcher folgende Methoden besitzt:
   - `Stack()` — erzeugt einen leeren Stapel.
   - `void push(Object x)` — fügt das Element `x` am Anfang des Stapels ein.
   - `Object pop()` — entfernt das erste Element des Stapels und liefert es zurück.

   Was müssen Sie beachten, wenn Sie Objekte einer bestimmten Klasse `A`

   (a) in einem `Stack` speichern?

   (b) aus dem `Stack` entnehmen und anschließend auf Attribute oder Methoden von `A` zugreifen wollen?

3. (2 Punkte) Welche Werte haben die Variablen `t` und `z` am Ende des folgenden Programms:

```
class I {
    int i;
    I (int i) {
        this.i = i;
    }
}
class P {
    void test (I x, int y) {
        y = 3;
        x.i = 2 * x.i + y;
    }
    public static void main (String[] args) {
        P p = new P ();
        int t = 2;
        int z = 4;
        I i = new I (2);
        p.test (i, z);
        t = i.i;
        // t = ?, z = ?
    }
}
```

4. (3 Punkte) Welche Fehler enthält folgendes Java-Codefragment? Geben Sie jeweils die Zeilennummer an und beschreiben Sie den Fehler. Folgefehler werden ignoriert.

```
 1 class Auto {
 2     int kilometer = 0;
 3
 4     void Auto (int k) {
 5         this.kilometer = k;
 6     }
 7
 8     void fahren () {
 9         Terminal.println ("Brummm!");
10     }
11 }
12 class Haus {
13     String adresse = "Bahnhofsgasse 12";
14
15     abstract void heizen ();
16 }
17 class Wohnwagen extends Auto, Haus {
18     void heizen () {
19         Terminal.println ("Warm!");
20     }
21 }
22 class Main {
23     void start () {
24         Auto a = new Auto (2000);
25     }
26 }
```

• AUFGABE 3 (5 Punkte) Numerik.

In dieser Aufgabe soll ein Programm zur Berechnung der Maßzahl φ des *goldenen Schnitts* programmiert werden. Diese Zahl ist definiert als der Grenzwert der Folge:

$$\begin{aligned} x_1 &= 1 \\ x_{n+1} &= 1 + \frac{1}{x_n} \end{aligned}$$

1. (3 Punkte) Schreiben Sie die Methode

   ```
   double phi (double eps)
   ```

   welche die Zahl φ näherungsweise mit obiger Formel berechnet, wobei `eps` die gewünschte Genauigkeit angibt.

2. (2 Punkte) Schreiben Sie ein Hauptprogramm, das die Zahl φ mit der Methode `phi` auf 10 Nachkommastellen genau berechnet und das Ergebnis auf dem Bildschirm ausgibt.

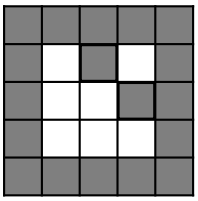
• AUFGABE 4 (11 Punkte) Robotersimulation.

In dieser Aufgabe soll ein Roboter simuliert werden, der sich auf einem Spielfeld bewegen kann.

Das Spielfeld wird durch eine Matrix von Wahrheitswerten dargestellt, wobei freie Felder durch den Wert `false` und blockierte Felder durch den Wert `true` repräsentiert werden. Dabei sind immer alle Felder am Rand blockiert.

Beispiel: Die folgende Matrix stellt das nebenstehende Spielfeld dar, in dem alle blockierten Felder grau dargestellt sind:

$$\begin{pmatrix} \textbf{true} & \textbf{true} & \textbf{true} & \textbf{true} & \textbf{true} \\ \textbf{true} & \text{false} & \textbf{true} & \text{false} & \textbf{true} \\ \textbf{true} & \text{false} & \text{false} & \textbf{true} & \textbf{true} \\ \textbf{true} & \text{false} & \text{false} & \text{false} & \textbf{true} \\ \textbf{true} & \textbf{true} & \textbf{true} & \textbf{true} & \textbf{true} \end{pmatrix}$$



Der Roboter hat eine aktuelle Position, die durch (x, y)-Koordinaten angegeben wird, sowie eine aktuelle Richtung, die durch einen Richtungsvektor $(delta\_x, delta\_y)$ dargestellt wird. Der Richtungsvektor gibt die Differenz in x/y-Richtung zum nächsten Feld (*oben*, *unten*, *links* oder *rechts*) an und kann dabei die folgenden Werte annehmen:

	delta_x	*delta_y*
oben	0	-1
unten	0	1
links	-1	0
rechts	1	0

Die Simulation wird in der folgenden Klasse implementiert, die das Spielfeld sowie die Position (x, y) und die Richtung (*delta_x*, *delta_y*) des Roboters als Attribute verwaltet.

```
class Simulation {
    int x;
    int y;
    int delta_x;
    int delta_y;
    boolean[][] spielfeld;
    // weiteres siehe unten...
}
```

Erweitern Sie die Klasse `Simulation` um folgende Methoden:

1. (1 Punkt) Schreiben Sie eine Methode

   ```
   boolean test ()
   ```

   die prüft, ob das nächste Feld in der aktuellen Richtung blockiert ist.

2. (1 Punkt) Schreiben Sie eine Methode

   ```
   void go ()
   ```

   die den Roboter in der aktuellen Richtung um ein Feld bewegt. Sollte das Feld in Bewegungsrichtung blockiert sein, soll nichts passieren.

3. (1 Punkt) Schreiben Sie eine Methode

   ```
   void turnaround ()
   ```

   die den Roboter um 180^o dreht.

4. (2 Punkte) Schreiben Sie eine Methode

   ```
   int distance ()
   ```

   die die Anzahl der aufeinanderfolgenden freien Felder in der aktuellen Bewegungsrichtung zählt.

5. (3 Punkte) Schreiben Sie eine Methode

   ```
   void turnleft ()
   ```

   um den Roboter um 90° nach links (gegen den Uhrzeigersinn) zu drehen.

6. (3 Punkte) Schreiben Sie eine Methode

```
void show ()
```

die auf dem Bildschirm eine Darstellung des Labyrinths ausgibt. Für das Beispiel am Anfang dieser Aufgabe sollte folgende Darstellung ausgegeben werden (unter der Annahme, dass der Roboter sich an Position (1/1) befindet):

```
*****
*O* *
*  **
*   *
*****
```

• AUFGABE 5 (12 Punkte) Schaltungs-Modellierung.

In dieser Aufgabe sollen digitale Schaltelemente durch Java-Klassen modelliert werden. Ziel ist es, durch das Verbinden von Objekten dieser Klassen komplexe Schaltungen zu simulieren.

Jedes Schaltelement soll dabei als *Signalquelle* aufgefasst werden, welche genau einen Ausgang besitzt. Die beiden möglichen Signalpegel werden durch die Java-Wahrheitswerte `true/false` dargestellt.

1. (1 Punkt) Schreiben Sie eine abstrakte Klasse `SignalQuelle`, welche folgende Methoden vorsieht:
   - `abstract boolean korrekt()` — liefert `true`, wenn das Schaltelement korrekt angeschlossen ist.
   - `abstract boolean signal()` — liefert das momentane Ausgangssignal des Schaltelements.

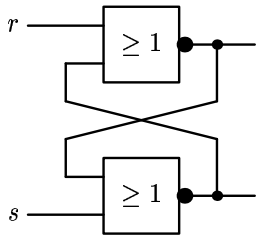
2. (2 Punkte) Schreiben Sie eine Klasse `Konstant` als Unterklasse von `SignalQuelle` zur Darstellung konstanter Signale.
   - Das Ausgangssignal soll als ein von außen *nicht zugängliches* Attribut implementiert werden. Definieren Sie einen geeigneten Konstruktor zur Initialisierung.
   - Programmieren Sie auch die geerbten abstrakten Methoden. Dabei gilt: ein konstantes Signal ist immer korrekt angeschlossen.

3. (2 Punkte) Schreiben Sie eine abstrakte Klasse `Verknuepfung` als Unterklasse von `SignalQuelle` zur Darstellung von Schaltelementen mit zwei Eingängen.

   - Implementieren Sie die beiden Eingänge als Attribute, die auf die Eingangs-Signalquellen verweisen. Diese sollen mit `null` initialisiert werden.
   - Programmieren Sie auch die Methode `korrekt()`. Dabei gilt: Eine Verknüpfung ist korrekt angeschlossen, wenn beide Eingänge an Objekte angeschlossen sind.

4. (1 Punkt) Schreiben Sie eine Klasse `Nor` als Unterklasse von `Verknuepfung`, welche als Ausgangssignal die NOR-Verknüpfung der beiden Eingangssignale liefert.

5. (3 Punkte) Schreiben Sie ein Fragment Java-Code, welches folgende Schaltung erzeugt. Die Signalquellen `r` und `s` können Sie als gegeben annehmen.



Was geschieht, wenn die Methode `signal()` für eines der beiden NOR-Elemente aufgerufen wird?

6. (3 Punkte) Schreiben Sie eine Klasse `Flipflop` als Unterklasse von `Verknuepfung`, welche ein getaktetes Flipflop modelliert. Dabei gilt, dass das Ausgangssignal eines Flipflops seinem Zustand entspricht.

   - Der aktuelle Zustand des Flipflops soll als von außen *nicht zugängliches* Attribut q gespeichert werden.
   - Programmieren Sie die Methode `signal()`.
   - Schreiben Sie eine Methode `void takt()`, welche beim Aufruf einen Zustandsübergang in Abhängigkeit von den Eingangssignalen x_1 und x_2 nach folgender Tabelle durchführt:

x_1	x_2	q_{neu}
false	false	q_{alt}
false	**true**	false
true	false	**true**
true	**true**	$\overline{q_{alt}}$