# Fakultät Elektrotechnik/Informatik

# Klausur Einführung in die Informatik I für Elektrotechniker 22. Februar 2002

**Name:** .....

**Matr.-Nr.** .....

Bearbeitungszeit: 120 Minuten

**Bewertung**
(bitte offenlassen :–)

| **Aufgabe** | Punkte | Erreichte Punkte |
|---|---|---|
| 1 | 5 | |
| 2 | 6 | |
| 3 | 4 | |
| 4 | 2 | |
| 5 | 4 | |
| 6 | 11 | |
| 7 | 12 | |
| 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.
- Bitte schreiben Sie 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!
- Wir weisen noch einmal darauf hin, daß die Benutzung von Taschenrechnern und anderen elektronischen Hilfsmitteln nicht gestattet ist.

Viel Erfolg!

**• AUFGABE 1 (5 Punkte) Theorie.**

1. (2 Punkte) Erläutern Sie die Begriffe *Attribut*, *Variable* und *Methode*. Wie stehen sie untereinander in Beziehung?

2. (1 Punkt) Was ist ein *Cache* und wozu dient er?

3. (2 Punkte) Welche der folgenden Aussagen sind richtig (möglicherweise mehrere)?
   - o Ein Programm kann mehr Klassen als Objekte besitzen.
   - o Ein Programm hat stets genausoviele Klassen wie Objekte.
   - o Ein Programm kann mehr Objekte als Klassen besitzen.
   - o Ein Programm hat nie mehr als doppelt soviele Objekte wie Klassen.

   Begründen Sie Ihre Antwort!

## • AUFGABE 2 (6 Punkte) JAVA.

1. (1 Punkt) Ersetzen Sie folgendes Code-Fragment durch äquivalenten *JAVA*-Code ohne `if`-Anweisung. (Vorsicht: Ergebnis ist nicht guter Code, sondern eine trickreiche Spielerei.)

```
int x = ...;
int y = ...;
if (y == 0) {
    Terminal.println("Fehler");
} else {
    x = x / y;
}
```

2. (2 Punkte) Geben Sie den Aufwand der Methoden `g` und `h` in *Big-O-Notation* an.

```
int g(int p) {
    int i = 1;
    int s = 0;
    while ( i <= p) {
        for (int j = p; j > 0; j--) {
            s++;
        }
        i = i + 1;
    }
    return s;
}

int h(int x) {
    int i = 0;
    int s = 0;
    do {
        s = s + x;
        i++;
    } while (i < g(x));
    return s;
}
```

3. (3 Punkte) Wo sind die Fehler im folgenden *JAVA*-Code? Beschreiben Sie die Fehler und geben Sie die Zeilenzahlen des Auftretens an.

```
 1 class Arbeit {
 2     int machSinnloses() {
 3         double[] a;
 4         Wrapper w;
 5         int i;
 6         double y;
 7         final int N = 5;
 8         a = new double[N];
 9         y = w.value;
10         i = a.length;
11         while (i >= 0) {
12             i = i-1;
13             w = new Wrapper();
14             a[i] = Terminal.askInt("a[" + i + "] = ");
15             w.value = w.value + a[i];
16             i--;
17         }
18     }
19 }
20
21 class Wrapper {
22    int value;
23
24    void newWrapper(int value) {
25        this.value = value;
26    }
27 }
```

• **AUFGABE 3 (4 Punkte) Zahlsysteme.**

1. (2 Punkte) Wandeln Sie die reelle Zahl 0.62 in eine Dualzahl mit 4 binären Nachkommastellen um und nehmen Sie dann die Umwandlung noch einmal in Gegenrichtung vor.

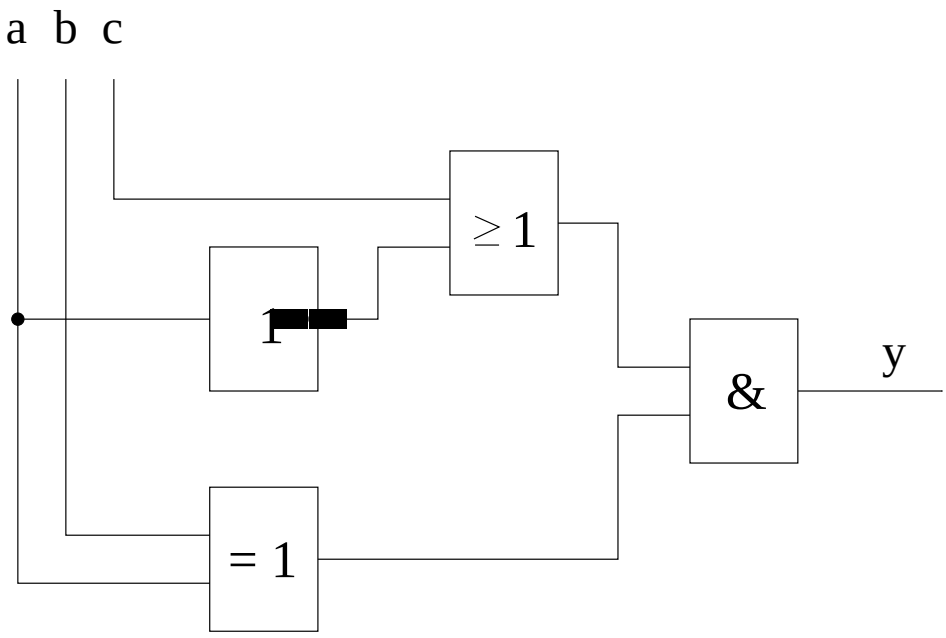
   Geben Sie den dabei auftretenden Fehler an. Lassen Sie den Lösungsweg erkennen.

2. (2 Punkte) Führen Sie folgende Berechnungen unter Verwendung der 2-Komplementdarstellung auf einem imaginären 4-Bit-Rechner aus. Lassen Sie den Lösungsweg erkennen. Wo findet ein *Over-* oder *Underflow* statt? Woran erkennt man das Auftreten eines *Over-* bzw. *Underflows*?

   - $5 + 7$
   - $-4 - 6$
   - $2 - 5$

• **AUFGABE 4 (2 Punkte) Schaltungen.**

Stellen sie die Wertetabelle für die folgende Schaltung auf.



**• AUFGABE 5 (4 Punkte) Reihenberechnung.**

Schreiben Sie eine Methode

```
double reihensum(int n)
```

welche das `n`-te Glied der Reihe $(s_n)$ berechnet.

$$s_n = \sum_{i=1}^{n} (-1)^{i+1} \cdot \frac{1}{i}$$

*Beispiel:*

$$\texttt{reihensum(5)} = \frac{1}{1} - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} = 0.78333333$$

**• AUFGABE 6 (11 Punkte) Kartons.**

In einer Verpackungsfirma werden Pappkartons auf einem Fließband verarbeitet. Es handelt sich um rechteckige Kartons, die durch Länge, Breite und Höhe charakterisiert sind.

```
class Karton {
    double laenge;
    double breite;
    double hoehe;

    // weiteres siehe unten
}
```

1. (1 Punkt) Erweitern Sie die Klasse `Karton` um eine Methode

   ```
   double volumen()
   ```

   zur Bestimmung des Volumens des Kartons.

2. (1 Punkt) Wir wollen im Folgenden mit genormten Kartons arbeiten. Ein Karton ist genormt, wenn gilt:

   $$\text{Länge} \geq \text{Breite} \geq \text{Höhe} > 0$$

   Erweitern Sie die Klasse `Karton` um eine Methode

   ```
   boolean genormt()
   ```

   um zu testen, ob ein Karton genormt ist.

3. (2 Punkte) Erweitern Sie die Klasse `Karton` um eine Methode

   ```
   boolean passt(Karton other)
   ```

   die feststellt, ob ein genormter Karton in einen anderen genormten Karton `other` hineinpaßt. (Die Dicke der Pappwände eines Kartons soll vernachlässigt werden.)

4. (2 Punkte) Schreiben Sie nun eine Methode

   ```
   boolean matrjoschka(Karton[] a)
   ```

   die feststellt, ob in einem Feld von genormten Kartons der erste in den zweiten paßt, der zweite in den dritten, usw.

5. (2 Punkte) Schreiben Sie eine Methode

   ```
   double maxVol(Karton[] a)
   ```

   um das Volumen des voluminösesten Kartons in einem nichtleeren Feld von beliebigen Kartons zu bestimmen.

6. (3 Punkte) Schreiben Sie eine Methode

   ```
   Karton[] minVolFilter(Karton[] a, double minVol)
   ```

   die aus einem Feld von Kartons diejenigen heraussucht, welche mindestens ein bestimmtes, vorgegebenes Mindest-Volumen haben.

## • AUFGABE 7 (12 Punkte) Zahlenumwandlung.

1. (4 Punkte) Schreiben Sie eine Methode

   `String dec2bin(int dec)`

   die eine ganze Zahl in die Binärdarstellung umwandelt. Bei negativen Zahlen geben Sie eine leere Zeichenkette zurück.

   *Beispiele:*

   - `dec2bin(19)` = "10011"
   - `dec2bin(5)` = "101"
   - `dec2bin(-6)` = ""

2. (4 Punkte) Schreiben Sie eine Methode

   `int bin2dec(String bin)`

   zum Umwandeln einer Binärzahl ins Dezimalsystem. Sollte die übergebene Zeichenkette keine Binärzahl repräsentieren, geben Sie zur Fehlersignalisierung die Zahl -1 zurück.

   Dazu können Sie für einen `String s` folgende Operationen von *Java* benutzen:

   - `s.length()` – Länge des Strings
   - `s.charAt(i)` – Zeichen (`char`) an der Stelle `i` ($0 \leq$ `i` $\leq$ `s.length()` $- 1$)

   *Hinweis:*

   - Führende Nullen in der Binärdarstellung sind erlaubt.

   *Beispiele:*

   ```
   bin2dec("10011") = 19          // normale Binärzahl
   bin2dec("00101") = 5           // führende Nullen ok
   bin2dec("-101") = -1           // keine Binärzahl – nur 0/1 erlaubt (ohne Vorzeichen)
   bin2dec(" 101") = -1           // keine Binärzahl – nur 0/1 erlaubt (keine Leerzeichen)
   bin2dec("10+11") = -1          // keine Binärzahl – nur 0/1 erlaubt
   bin2dec("l=/01ej,hc") = -1     // keine Binärzahl – nur 0/1 erlaubt (das ist nie eine Binärzahl)
   ```

3. (4 Punkte) Schreiben Sie nun eine Methode

   `String binadd(String bin1, String bin2)`

   die zwei Zahlen in Binärdarstellung addiert und das Ergebnis wieder als Binärzahl zurückgibt.
   (Falls einer der beiden Parameter keine Binärzahl darstellt, liefern Sie wie bei `dec2bin` eine leere Zeichenkette als Fehlerkennung zurück.)

   *Beispiele:*

   - `binadd`("101", "1001") = "1110"
   - `binadd`("10011", "10110") = "101001"
   - `binadd`("101", "123") = "" // keine Binärzahl