Fakultät IV
Elektrotechnik/Informatik

Klausur Einführung in die Informatik II für Elektrotechniker

27. Juli 2005

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung (bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	8	
2	9	
3	3	
4	10	
5	8	
6	6	
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 (8 Punkte) Theorie.

1. (4 Punkte) Stellen Sie in den folgenden Tabellen den Verlauf einer Sortierung mittels Quicksort dar.

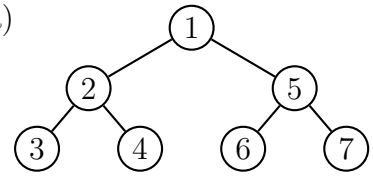
   Schreiben Sie dabei hinter jede Tabelle, welches Pivot-Element gerade ausgewählt ist, vertauschen Sie zwei Elemente bezüglich des Pivot-Elements und schreiben Sie in die nächste Zeile das Resultat dieser Vertauschung. Sobald das Array sortiert ist, können Sie aufhören.

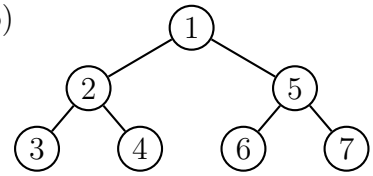
   Markieren Sie in jeder Tabelle die Grenzen des Teilarrays, das gerade bearbeitet wird.

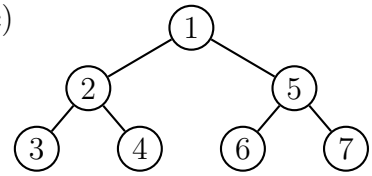
1	4	2	6	1	5	3	Pivot-Element: _____
2							Pivot-Element: _____
3							Pivot-Element: _____
4							Pivot-Element: _____
5							Pivot-Element: _____
6							Pivot-Element: _____
7							Pivot-Element: _____
8							Pivot-Element: _____
9							Pivot-Element: _____

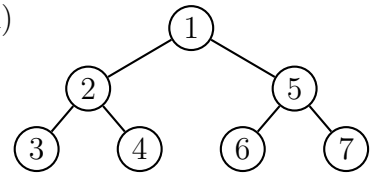
   (Hinweis: Sie müssen nicht alle Tabellen verwenden, wenn das Array schon vorher sortiert ist.)

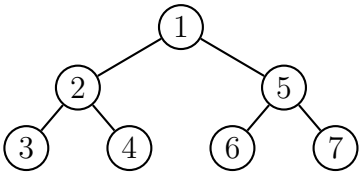
2. (3 Punkte) Traversieren Sie den folgenden Graphen, beginnend mit dem Knoten mit der Nummer 1. Verwenden Sie dafür *Tiefensuche* mit *Graphfärbung*. Markieren Sie dabei in jedem Schritt alle grauen und schwarzen Knoten.

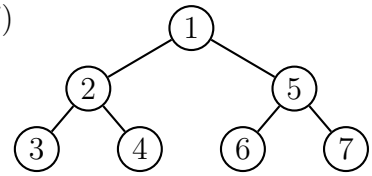
(a) 

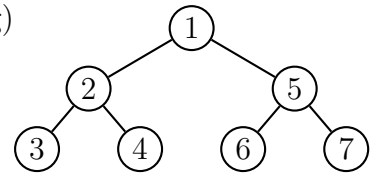
(b) 

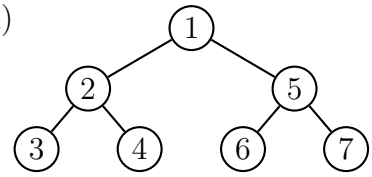
(c) 

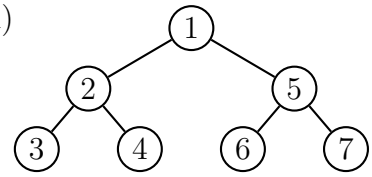
(d) 

(e) 

(f) 

(g) 

(h) 

(i) 

(Hinweis: Sie müssen nicht alle Bäume markieren, wenn die Traversierung schon vorher abgeschlossen ist.)

3. (1 Punkt) Erläutern Sie den Unterschied zwischen den Schlüsselwörtern `public` und `protected` bei der Deklaration von Attributen und Methoden in Java.

Aufgabe 2 (9 Punkte) Java.

1. (4 Punkte)

   Gültigkeitsbereiche in Java-Programmen lassen sich folgendermaßen grafisch darstellen. Die Gültigkeitsbereiche werden durch Rechtecke dargestellt, die den Code des Bereichs umfassen. Außerdem werden die einzelnen Bereiche durch Nummern markiert.

   Beispiel:

```
if (x > 0.25) {
  while (!ready) {                                    1
    ...                                               2
  }
}
```

   (a) Stellen Sie in folgendem Java-Code die verschiedenen Gültigkeitsbereiche von Variablen grafisch dar und nummerieren Sie die einzelnen Gültigkeitsbereiche.

```
class Radio {
  private double frequenz;
  private int sender;
  private double[] programmiert;
  boolean setSender(int send) {
    boolean fehler = false;
    if (send >= 0 && send < programmiert.length) {
      this.sender = send;
      this.frequenz = programmiert[send];
    } else {
      String a = Terminal.ask("Falscher Sender. Bitte Enter drücken.");
      fehler = true;
    }
    return fehler;
  }
  int getSender() {
    return sender;
  }
}
```

   (b) Geben Sie an, welche Variablen in welchen Gültigkeitsbereichen gültig sind.

2. (2 Punkte) Geben Sie für folgenden Java-Code an, welche Objekte an den (mit `(1)`, `(2)` und `(3)`) markierten Stellen existieren, wenn die Methode `f` aufgerufen wird. Geben Sie auch die Attributwerte der Objekte an.

```
class Point {
  double x;
  double y;
  Point(double x, double y) {
    this.x = x;
    this.y = y;
  }
}

void f() {
  Point p1 = new Point(2.0, 3.1);
  Point p2 = new Point(1.0, 1.0);
  // (1)
  p1.x = p2.y;
  p2.x = p1.y;
  // (2)
  p1 = new Point(p1.x, p1.y);
  Point p3 = new Point(p2.y, p1.x);
  // (3)
}
```

3. (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 Wesen {
2  }
3  abstract class Erdling implements Wesen {
4  }
5  abstract class Alien extends Wesen {
6      void beep();
7  }
8  class Marsianer extends Alien {
9      Erdling klonen() {
10         return new Erdling();
11     }
12     void beep() {
13       Terminal.println("Beep");
14     }
15 }
```

Aufgabe 3 (3 Punkte) Perfekte Zahlen.

Perfekte Zahlen sind definiert als Zahlen, die gleich der Summe aller kleineren Zahlen sind, durch die sie teilbar sind. In dieser Aufgabe soll eine Methode definiert werden, die den folgenden Algorithmus implementiert:

> Für eine gegebene Zahl n sollen alle Zahlen, die kleiner oder gleich der Zahl $\frac{n}{2}$ sind und durch die n teilbar ist, addiert werden. Wenn die Summe gleich n ist, ist n eine perfekte Zahl, sonst nicht.

Hinweis: Eine ganze Zahl a ist durch eine andere ganze Zahl b teilbar, wenn `a % b = 0` *gilt.*
(`%` *ist der Modulo-Operator in Java.)*

> **Beispiel:** Die Zahl 6 ist perfekt, weil sie durch 1, 2, und 3 teilbar ist, und $1 + 2 + 3 = 6$.
>
> Die Zahl 8 ist durch 1, 2 und 4 teilbar, aber $1 + 2 + 4 = 7 \neq 8$: 8 ist also keine perfekte Zahl.

1. (3 Punkte) Schreiben Sie eine Methode

   ```
   boolean perfekt(long n)
   ```

   die den Wert `true` zurückgibt, wenn `n` eine perfekte Zahl ist und andernfalls den Wert `false`.

Aufgabe 4 (10 Punkte) Abstrakte Datentypen.

In dieser Aufgabe geht es darum, einen abstrakten Datentyp zu programmieren. Dieser Datentyp soll eine Liste darstellen, deren Kapazität (also die Anzahl der Elemente, die maximal enthalten sein können) bei der Erzeugung einer Liste festgelegt werden soll. Insgesamt soll dieser abstrakte Datentyp `BoundedList` folgende Operationen zur Verfügung stellen:

- `BoundedList(int max)` Konstruktor, der mit der maximalen Anzahl von Elementen aufgerufen wird.
- `boolean insert(Object o, int index)` Einfügen eines Elements `o` an Position `index`. Wenn erfolgreich, soll `true` zurückgegeben werden, sonst `false`. Die Operation schlägt fehl, wenn bereits die maximale Anzahl an Elementen in der Liste enthalten ist.

Hinweise:

- Die Indizierung der Elemente beginnt bei 0, wie in Java für Arrays üblich.
- Sorgen Sie dafür, dass die Attribute der Klasse nicht von außerhalb sichtbar sind.
- Gehen Sie davon aus, dass die Methoden nur mit gültigen Parametern aufgerufen werden.

1. (2 Punkte) Schreiben Sie eine Klasse `Element`, die zur Verkettung der einzelnen Zellen der Liste dient. Diese soll beliebige Objekte als Daten aufnehmen können. Denken Sie an einen geeigneten Konstruktor

2. (2 Punkte) Definieren Sie die Attribute der Klasse `BoundedList` und schreiben Sie den Konstruktor.

3. (4 Punkte) Schreiben Sie die Methode `insert`.

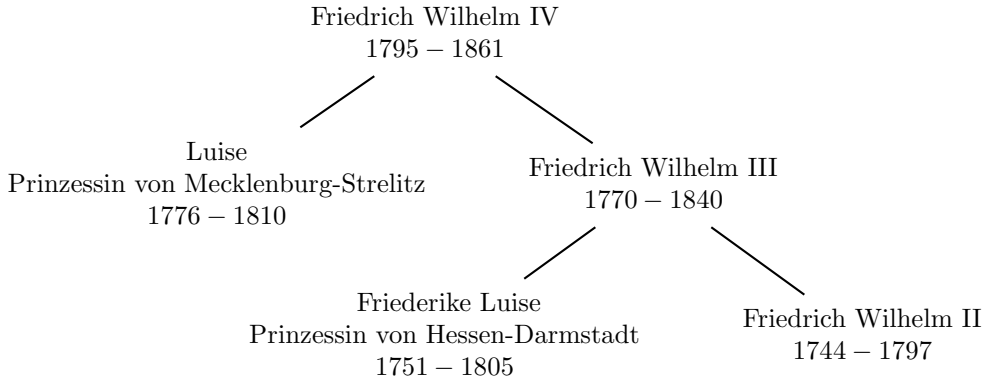
4. (2 Punkte) Schreiben Sie nun die komplette Klassendefinition der Klasse `BoundedList`. Dabei sollen alle Definitionen, die nicht zur Schnittstelle des abstrakten Datentyps gehören, von außen nicht sichtbar sein.

   Wiederholen Sie nur Attribute, Klassen und Methodenköpfe, Sie brauchen nicht die Methodenrümpfe zu wiederholen.

Aufgabe 5 (8 Punkte) Bäume.

In dieser Aufgabe betrachten wir Stammbäume.
Jeder Knoten in einem Stammbaum ist einer Person zugeordnet. Eine Person enthält neben dem Namen auch das Geburts- und das Todesjahr. Weiterhin enthält jeder Knoten eine Referenz auf die Mutter sowie eine Referenz auf den Vater. Falls Vater oder Mutter nicht bekannt sind, ist die entsprechende Referenz `null`.

Beispiel: Dies ist ein Stammbaum, der einige der Vorfahren Friedrich des IV. darstellt.



Gegeben sind die folgenden Klassendefinitionen:

```
class Person {
    String name;
    int geburtsJahr;
    int todesJahr;
    Person(String n, int g, int t) {
        name = n;
        geburtsJahr = g;
        todesJahr = t;
    }
}
class Knoten {
    Person pers;
    Knoten mutter;
    Knoten vater;
    Knoten(Person p, Knoten m, Knoten v) {
        pers = p;
        mutter = m;
        vater = v;
    }
}
```

1. (4 Punkte) Schreiben Sie eine Methode

   ```
   Person personMax(Person p1, Person p2)
   ```

   welche aus zwei Personen diejenige ermittelt, die länger gelebt hat. Dabei sollen nur Parameter verglichen werden, die nicht den Wert `null` haben (wenn also ein Parameter `null` ist, dann ist der andere die ältere Person). Wenn keine ältere Person ermittelt werden kann, soll `null` zurückgegeben werden. Wenn beide Personen gleich alt geworden sind, ist egal, welche Person zurückgegeben wird.

2. (4 Punkte) Schreiben Sie eine Methode

   ```
   Person aeltestePerson(Knoten baum)
   ```

   welche die älteste Person im gegebenen Stammbaum ermittelt. Kann keine älteste Person ermittelt werden, dann soll `null` zurückgegeben werden.

Aufgabe 6 (6 Punkte) Vererbung.

In dieser Aufgabe wird das Angebot eines Restaurants durch Java-Klassen modelliert. Das Angebot beschränkt sich zwar auf Cola und Steaks, soll aber vernünftig modelliert werden, um spätere Erweiterungen zu erlauben.

1. (1 Punkt) Schreiben Sie eine abstrakte Klasse `Ware`, welche eine abstrakte Methode `preis` definiert, die den Preis der Ware in Cent berechnet (als Integer-Wert).

2. (2 Punkte) Schreiben Sie zwei von `Ware` abgeleitete abstrakte Klassen `Getraenk` und `Essen`. Die Klasse `Getraenk` soll eine abstrakte Methode `menge` definieren, die die Menge in Litern (als `double`-Wert) angibt. Die Klasse `Essen` soll keine weiteren Eigenschaften haben.

3. (2 Punkte) Schreiben Sie eine *nicht abstrakte* Klasse `Cola`, die von `Getraenk` abgeleitet ist. Diese Klasse soll einen Wahrheitswert als Attribut haben, der angibt, ob es eine kleine oder eine große Cola ist. Das Attribut soll nicht von außen zugreifbar sein.

   Schreiben Sie einen Konstruktor, der das Attribut angemessen initialisiert.

   Definieren sie die geerbten abstrakten Methoden. Dabei gilt: eine große Cola hat eine Menge von 0.5 Litern und kostet 3.50 EUR, eine kleine Cola hat 0.33 Liter und kostet 2.00 EUR.

4. (1 Punkt) Schreiben Sie nun eine Klasse `Steak`, die von `Essen` abgeleitet ist. Sie hat als Eigenschaft ein Gewicht in Gramm, das über einen geeigneten Konstruktor initialisiert werden soll und nicht von außen zugreifbar ist. Der Preis berechnet sich als das Doppelte des Gewichts.