Fakultät IV
Elektrotechnik/Informatik

Klausur Einführung in die Informatik II für Elektrotechniker

2. August 2006

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung (bitte offenlassen :–)

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

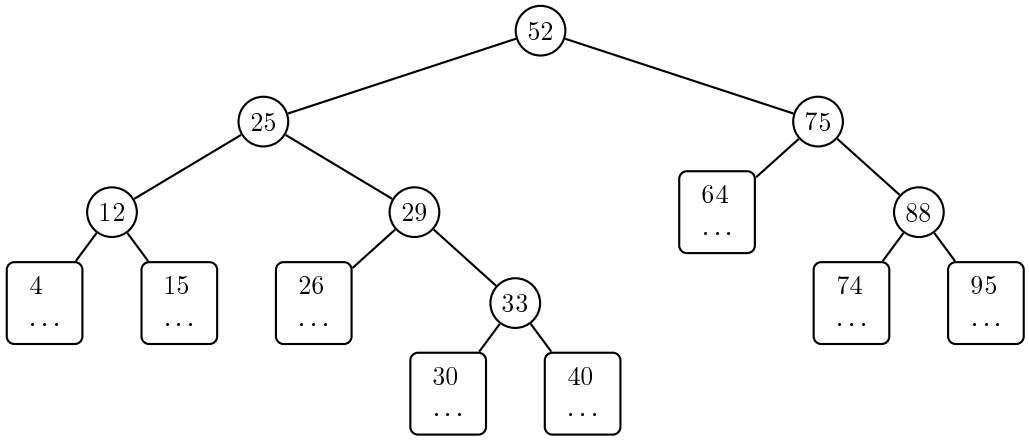
Viel Erfolg!

Aufgabe 1 (5 Punkte) Theorie.

1. (2 Punkte) Wozu dienen in Java *generische Klassen*, wie deklariert und benutzt man sie? Nennen sie einen Vorteil dieser Klassen.

2. (1 Punkt) Nennen Sie einen Vorteil und einen Nachteil des Sortieralgorithmus *Quicksort*.

3. (2 Punkte) Eignet sich der folgende Baum zum effizienten Suchen? Begründen Sie Ihre Antwort.



Aufgabe 2 (7 Punkte) Java.

1. (2 Punkte) Gegeben seien die folgenden drei Klassen:

```
class A {
    B b;
}
```

```
class B {
    C c;
}
```

```
class C {
    A a;
}
```

Stellen sie grafisch die Objektstruktur dar, die durch folgenden Java-Code erzeugt wird. Zeichnen Sie pro Objekt ein Kasten und Pfeile für die Referenzen zwischen den Objekten. Lassen Sie auch erkennen, auf welche Objekte die deklarierten Variablen verweisen.

```
A a = new A();
B b = new B();
C c = new C();
a.b = b;
b.c = new C();
b.c.a = new A();
b.c.a.b = b;
c.a = b.c.a;
```

2. (1 Punkt) Wozu dient in Java das Schlüsselwort `implements`?

3. (1 Punkt) Wozu dient in Java das Schlüsselwort `public`? An welchen Stellen im Programm kann man es benutzen?

4. (3 Punkte) Welche Fehler enthalten die folgenden Java-Klassen? 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 Stift {
2       private int farbe;
3       Stift(int farbe) {
4           this.farbe = farbe;
5       }
6   }
7
8   class Bleistift implements Stift {
9       Bleistift(int farbe) {
10          super(farbe);
11      }
12  }
13
14  class Hauptklasse {
15      public static void main(String[] args) {
16          Bleistift b = new Bleistift(0);
17          Stift s = new Stift(1);
18          b = s;
19      }
20  }
```

Aufgabe 3 (6 Punkte) Numerik.

Die Kosinusfunktion kann folgendermaßen definiert werden:

$$\cos(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$$

1. (4 Punkte) Schreiben Sie eine Methode

   ```
   double cos(double x, int n)
   ```

   welche den Kosinus von x nach der oben genannten Formel berechnet. Der Approximationsprozess soll abgebrochen werde, sobald eine Genauigkeit von n Nachkommastellen erreicht ist.

   Hinweis: Sie dürfen die Methode `double pow(double x, double n)` *aus der Klasse* `Math` *benutzen um die Potenz x^n zu berechnen. Die Fakultätsfunktion*

   $$\begin{aligned} 0! &= 1 \\ (n+1)! &= (n+1) \cdot n! \end{aligned}$$

   müssen Sie aber selbst definieren.

2. (2 Punkte) In Java gibt es die Funktion `double Math.cos(double x)`, welche ebenfalls die Kosinusfunktion berechnet.

   Schreiben Sie eine Methode

   ```
   boolean testCos(double x, int n, double eps)
   ```

   welche prüft, ob die Differenz der Ergebnisse der Funktionen `Math.cos(x)` und `cos(x, n)` (aus der vorigen Unteraufgabe) weniger als der Betrag `eps` ist.

Aufgabe 4 (8 Punkte) Abstrakte Datentypen.

In dieser Aufgabe soll ein abstrakter Datentyp `AuftragsListe` zum Speichern von Bestellungen erstellt werden, welcher Objekte der folgenden Klasse verwalten kann:

```
class Bestellung {
    boolean express;
    String ware;
    int anzahl;
    Bestellung(boolean express, String ware, int anzahl) {
        this.express = express;
        this.ware = ware;
        this.anzahl = anzahl;
    }
}
```

Eine Bestellung ist entweder normal (Attribut `express` = `false`) oder eine Expressbestellung (Attribut `express` = `true`). Dies muss in den zu programmierenden Methoden beachtet werden.

Eine `AuftragsListe` besitzt drei Methoden, die weiter unten zu programmieren sind:

`boolean istLeer()` Prüfen, ob die Liste leer ist.

`void rein(Bestellung b)` Eine Bestellung einfügen.

`Bestellung raus()` Eine Bestellung aus der Liste holen.

Für die Methode `raus()`, die Bestellungen aus der Liste holt, sollen folgende Regeln gelten.

1. Express-Bestellungen sollen Vorrang vor normalen Bestellungen haben, also immer zuerst herausgegeben werden.
2. Als Einschränkung gilt: nach jeweils drei Express-Bestellungen muss eine normale Bestellung herausgegeben werden, damit eine gewisse Fairness herrscht.
3. Wenn die Liste leer ist, soll `null` abgeliefert werden und angenommen werden, dass noch keine Expresslieferung rausgenommen wurde.

Hinweis: Sie können die Klasse `LinkedList` mit dem Konstruktor `LinkedList()` und den Methoden `boolean isEmpty()`, `void add(Object o)` und `Object removeFirst()` aus der Java-Standardbibliothek benutzen.

Aufgaben:

1. (2 Punkte) Schreiben Sie den Anfang der Klasse `AuftragsListe`. Definieren Sie die benötigten Attribute, um die oben beschriebenen Operationen schreiben zu können. Schreiben Sie einen geeigneten Konstruktor, mit dem man eine leere Auftragsliste erzeugen kann.

   Hinweis: Die Art und Weise, wie die Bestellungsobjekte intern von der Klasse `AuftragsListe` verwaltet werden, ist Ihnen überlassen.

2. (1 Punkt) Schreiben Sie (in der Klasse `AuftragsListe`) eine Methode

   ```
   boolean istLeer()
   ```

   welche den Wert `true` zurückgibt, wenn keine Bestellungen in der Liste sind und andernfalls den Wert `false`.

3. (2 Punkte) Schreiben Sie (ebenfalls in der Klasse `AuftragsListe`) eine Methode

   ```
   void rein(Bestellung b)
   ```

   welche der Auftragsliste die gegebene Bestellung hinzufügt.

4. (3 Punkte) Schreiben Sie (ebenfalls in der Klasse `AuftragsListe`) eine Methode

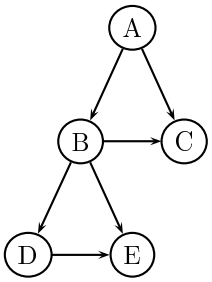
   ```
   Bestellung raus()
   ```

   welche eine Bestellung aus der Liste entfernt und als Ergebnis zurückliefert.

   Beachten Sie hierbei die Einschränkungen, die für die Herausgabe von Bestellungen aus der Liste gelten, so wie sie in der Einleitung angegeben wurden.

Aufgabe 5 (9 Punkte) Bäume.

In dieser Aufgabe soll eine erweiterte Datenstruktur für Binärbäume geschrieben werden. Zusätzlich zu den Verweisen auf die linken und rechten Nachfolgerknoten enthält jeder Knoten noch einen Verweis auf seinen rechten Nachbarn auf der gleichen Ebene.
Die folgende Abbildung zeigt, wie solch ein Baum konzeptionell aussieht. Alle Knoten haben Verweise auf bis zu zwei Nachfolger. Außerdem haben alle Knoten einen Verweis auf den rechten Nachbarn, sofern einer existiert.



1. (3 Punkte) Schreiben Sie eine Klasse `Knoten` zur Darstellung von Baumknoten. Jeder Knoten hat sowohl Referenzen auf zwei Kindknoten als auch eine Referenz auf den rechten Nachbarn. Weiterhin soll jeder Knoten eine Referenz auf ein Datenelement der Klasse `Object` enthalten.

   Schreiben Sie einen geeigneten Konstruktor, der die Initialisierung aller Attribute erlaubt.

2. (2 Punkte) Schreiben Sie Methode

```
Knoten testBaum()
```

welche den Baum erzeugt, der oben dargestellt ist. Die Datenobjekte sind die Strings `"A"`, `"B"`, `"C"`, usw.

3. (4 Punkte) Gegeben sei folgendes Interface:

```
interface Besucher {
    void besuche(Knoten k);
}
```

Schreiben Sie eine Methode

```
void ebene(Knoten k, Besucher b)
```

welche alle Knoten, die an dem Knoten `k` hängen, ebenenweise besucht. Für den oben dargestellten Baum soll also die `besuche`-Methode von `b` nacheinander für die Knoten A, B, C, D, E aufgerufen werden.

Hinweis: Beachten Sie, dass ein Knoten auch einen einzigen linken oder rechten Nachfolger haben kann.

Aufgabe 6 (9 Punkte) Vererbung.

In dieser Aufgabe sollen Sie ein Programm zur Planung Ihres Stundenplans schreiben.

1. (1 Punkt) Schreiben Sie eine abstrakte Klasse `Lehrveranstaltung`, welche die abstrakten Methoden `String name()`, `int sws()` und `String typ()` besitzt.

2. (3 Punkte) Schreiben Sie eine Klasse `Vorlesung`, welche von `Lehrveranstaltung` abgeleitet ist. Den Namen und die Semesterwochenstunden (SWS) soll man beim Aufruf des Konstruktors angeben können. Der Typ einer Vorlesung ist die Zeichenkette `"LV"`. Wenn Sie Attribute verwenden, sollen diese von außen *nicht* sichtbar sein.

3. (2 Punkte) Schreiben Sie weiterhin eine Klasse `Uebung`, welche ebenfalls von `Lehrveranstaltung` abgeleitet ist. Eine Übung soll die Vorlesung speichern, zu der sie gehört. Diese Vorlesung und die Semesterwochenstunden (SWS) soll man beim Aufruf des Konstruktors angeben können. Der Typ einer Übung ist die Zeichenkette `"UE"`, der Name entspricht der dazugehörigen Vorlesung. Wenn Sie Attribute verwenden, sollen diese von außen *nicht* sichtbar sein.

4. (3 Punkte) Schreiben Sie nun eine Klasse `Stundenplan`, die folgende Methoden zur verfügung stellt:

   - Einen Konstruktor, dem man ein Array von Lehrveranstaltungen übergeben kann. Der Konstruktor soll eine Kopie das übergebenen Arrays anlegen.
   - Eine Methode `int gesamtSWS(String typ)`, welche die Summe der SWS aller Lehrveranstaltungen mit dem angegebenen Typ berechnet.

   Hinweis: Beachten Sie, dass Sie zum Vergleichen von Strings die Methode `boolean equals(String other)` verwenden müssen. Der Operator `==` funktioniert bei Strings nicht.

Name: .....

Matr.-Nr. .....