Fachbereich Informatik

Klausur Einführung in die Informatik II für Elektrotechniker 24. Februar 2001

Name: .....

Matr.-Nr. .....

Bearbeitungszeit: 120 Minuten

Bewertung
(bitte offenlassen :–)

Aufgabe	Punkte	Erreichte Punkte
1	6	
2	8	
3	5	
4	5	
5	7	
6	13	
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 nicht gestattet ist.

Viel Erfolg!

• **AUFGABE 1 (6 Punkte) Theorie.**

1. (2 Punkte) Vergleichen Sie *Quicksort* und *MergeSort* bezüglich der Verteilung der Arbeit beim Aufteilen bzw. Zusammensetzen im verwendeten Divide-und-Conquer Verfahren, des Aufwands sowie des *in-situ*-Verhaltens.

2. (2 Punkte) Was versteht man unter *dynamischem Binden*? Warum benötigt man trotzdem (oder deswegen) bei Objektzugriffen manchmal Castings?

3. (2 Punkte) Erläutern Sie die Begriffe *abstrakte Klasse* und *Interface* und worin unterscheiden sich diese?

- **AUFGABE 2 (8 Punkte) JAVA.**

1. (1 Punkt) Was bewirkt das Schlüsselwort *static*?

2. (2 Punkte) Was ist eine Referenz in *JAVA*? Erläutern Sie die beiden verschiedenen Arten von *Gleichheit*, die für Referenzen definiert sind.

3. (2 Punkte) Was versteht man unter dem Konzept der *Exception*? Wie kann man sie in Java auslösen bzw. abfangen?

4. (3 Punkte) Wo sind die Fehler im folgenden *JAVA*-Code? Beschreiben Sie die Fehler und geben Sie die Zeilenzahlen des Auftretens an.

```
 1 class Number {
 2     protected int number;
 3 }
 4
 5 class AddNumber extends Number {
 6     Number(int number) {
 7         this.number = number;
 8     }
 9
10     void add(Number n) {
11         number = this.number + n.number;
12     }
13 }
14
15 class MultNumber extends Number {
16     Number mult(Number x, Number y) {
17         Number result = new Number();
20         for (i = 1; i <= y.number; i++) {
21             result.add( x );
22         }
23         return result;
24     }
25 }
```

• AUFGABE 3 (5 Punkte) Numerik.

Zur Berechnung der Kreiszahl π kann man die „Machinsche Formel“

$$\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}$$

verwenden. Durch Taylorentwicklung für den Arcus-Tangens folgt daraus die Reihenentwicklung für die Berechnung von π:

$$\frac{\pi}{4} = \frac{4}{5} \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \left(\frac{1}{5}\right)^{2k} - \frac{1}{239} \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} \left(\frac{1}{239}\right)^{2k}$$

1. (5 Punkte) Schreiben Sie eine *JAVA*-Methode `double pi()`, welche den Wert der Kreiszahl π per Approximation berechnet. Beenden Sie den Approximationsprozeß, wenn eine Genauigkeit von 10^{-12} erreicht ist. Formulieren Sie diese Bedingung mit Hilfe einer zu erstellenden Hilfsfunktion `boolean close( double x, double y, double eps )`.

   Hinweis: Für die Potenzfunktion können Sie die Methode `Math.pow` verwenden. Berechnen Sie die Summen in jedem Schritt der Approximation *nicht* jeweils vollständig neu, sondern verwenden Sie jeweils nur die letzten Summenglieder.

• AUFGABE 4 (5 Punkte) Abstrakte Datentypen.

Ein *Stack* ist eine Datenstruktur, welche Daten gemäß des LIFO–Prinzips (Last in, First out) verwaltet. Dazu dienen die beiden Operationen *push* und *pop*:

- `push` fügt ein Element hinzu
- `pop` liefert das zuletzt hinzugefügte Element (bzw. `null`, falls der Stack leer ist) und entfernt dieses Element vom Stack.

Beispiel:

```
Stack S = new Stack();          // leerer Stack
S.push(A);                      // S = [A]
S.push(B);                      // B hinzufuegen: S = [B, A]
S.push(C);                      // S = [C, B, A]
Object val = S.pop();           // liefert Element C
                                // S = [B, A]
```

Sie sollen eine Klasse `Stack`, die die obigen Methoden realisiert, implementieren. Die Implementierung soll *nicht* „zu Fuß“ über verkettete Zellen erfolgen, sondern auf der Basis eines vorgegebenen anderen abstrakten Datentyps, nämlich „Listen“ (siehe unten).

1. (1 Punkt) Schreiben Sie eine Klasse `Stack`, die einen Stack von beliebigen Objekten verwaltet. Intern sollen die Elemente in einer Datenstruktur vom Typ `List` (siehe unten) gehalten werden.
2. (2 Punkte) Ergänzen Sie die Klasse `Stack` um die Methode `void push(Object o)`, welche ein Element `o` in den `Stack` einfügt.
3. (2 Punkte) Ergänzen Sie die Klasse um die Methode `Object pop()`, welche das zuletzt eingefügte Element zurückliefert und aus dem Stack entfernt.

   Wenn die Operation auf den leeren Stack angewendet wird, soll `null` zurückgegeben werden.

Hinweis: Bei der Bearbeitung dieser Aufgabe können Sie voraussetzen, daß Ihnen die folgende Klasse `List` zur Verfügung steht (*Achtung:* Nicht alle Methoden brauchen Sie zur Lösung der Stack-Aufgabe.):

```
class List {
    List();                  // leere Liste
    void insert(Object o);   // Einf"ugen vor aktueller Position
    Object currentValue();   // aktuelles Element)
    void remove();           // aktuelles Element entfernen
    void reset();            // zum Listenanfang
    void advance();          // zum n"achsten Element
    boolean finished();      // Listenende erreicht?
    boolean isempty();       // Liste leer?
}
```

• **AUFGABE 5 (7 Punkte) Währungen.**

1. (1 Punkt) Schreiben Sie eine abstrakte Klasse `Geld`, welche die abstrakten Methoden `double EuroValue()` zur Bestimmung des Geldwertes in Euro sowie `String toString()` zur Umwandlung in einen String enthält.

2. (3 Punkte) Schreiben Sie zwei Klassen `DM` und `Euro`, die beide von der Klasse `Geld` abgeleitet sind.

   Beide Klassen sollen jeweils einen Geldbetrag der jeweiligen Währung vom Typ `double` enthalten, welches im Konstruktor gesetzt wird.

   Ferner sollen jeweils die Methoden `EuroValue` und `toString` implementiert werden. Dabei gilt für die Umrechnung von DM nach Euro: $1\ EUR = 1,95583\ DM$.

   Beispiele:

   ```
   DM d = new DM(10.0);
   Euro e = new Euro(10.0);
   d.toString() == "10.0 DM"
   e.toString() == "10.0 EUR"
   d.EuroValue() == 5.113
   e.EuroValue() == 10.0
   ```

3. (3 Punkte) Schreiben Sie ferner eine Klasse `VielGeld`, welche als Attribut ein Array von Geldbeträgen enthält. Der Konstruktor soll dieses Array übergeben bekommen.

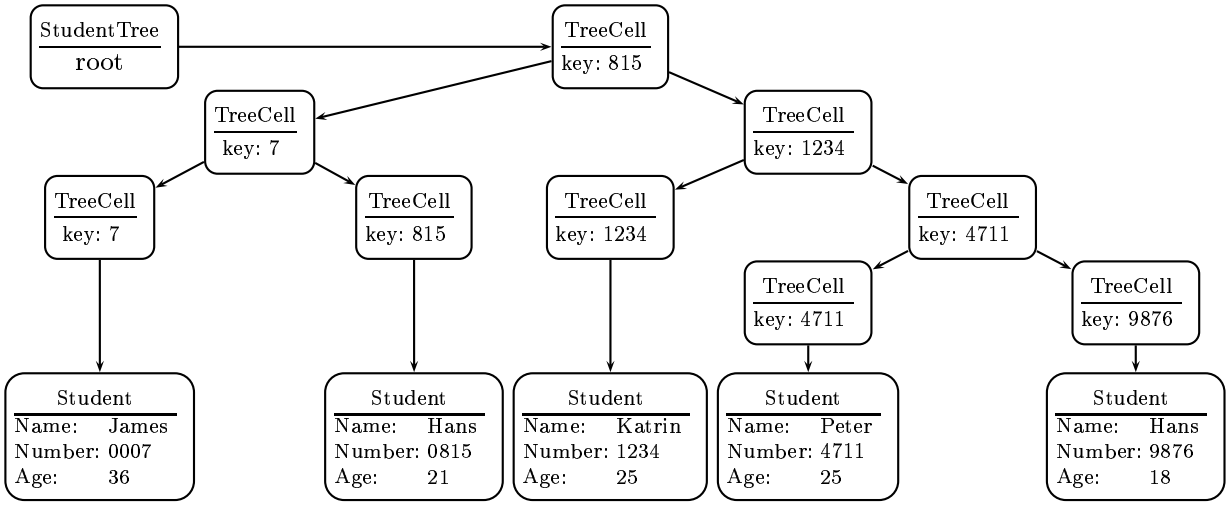
   Die Klasse `VielGeld` soll ebenfalls von der Klasse `Geld` abgeleitet sein und die dort deklarierten Methoden implementieren. Dabei soll `toString` eine in Klammern ([bzw.]) gesetzte und durch Komma getrennte Liste der einzelnen Elemente ausgeben, `EuroValue` den Gesamtbetrag aller Elemente in Euro.

   Beispiele:

toString	EuroValue
[10 DM]	5,113
[10 DM, 10 EUR]	15,113
[100 EUR, [10 DM, 10 EUR], 0 DM]	115,113

• AUFGABE 6 (13 Punkte) Binärbäume.

Zur Verwaltung von einer Universität sollen Sie einen Binärbaum erstellen, der die Daten von Studenten (Name, Matrikelnummer und Alter) verwaltet.



Der Baum soll nach Matrikelnummer der Studenten geordnet sein, braucht jedoch nicht balanciert zu sein. Die Studentendaten sollen lediglich in den Blättern des Baumes gespeichert werden, *nicht* jedoch in den inneren Knoten des Baumes.

Es soll möglich sein, einen Studenten in den Baum einzufügen, einen Studenten anhand seiner Matrikelnummer zu finden sowie alle Studenten um ein Jahr altern zu lassen.

1. (2 Punkte) Schreiben Sie eine Klasse `Student`, welche die persönlichen Daten eines Studenten (Name, Matrikelnummer und Alter) verwaltet. Machen Sie die Attribute privat.

   Sie benötigen jeweils eine Methode zum Lesen bzw. zum Schreiben des Alters (`int getAge()` bzw. `void setAge(int a)`) sowie zum Lesen der Matrikelnummer `int getNumber()`.

2. (3 Punkte) Schreiben Sie ferner eine Klasse `TreeCell`, die eine Zelle in einem Binärbaum definiert. Sie enthält einen Schlüssel `int key` sowie Referenzen auf den linken und rechten Unterbaum bzw. eine Referenz auf ein Objekt der Klasse `Student`.

   Als Konstruktoren sollen eine Methode zur Erzeugung eines Blattes, welche einen Schlüssel sowie ein Datenobjekt als Parameter erhält, sowie ein Konstruktor zur Erzeugung eines inneren Knotens (mit einem Schlüssel und zwei Zellen für den linken bzw. rechten Unterbaum als Parameter) realisiert werden.

   Die Klasse soll ferner eine Methode `boolean isLeaf()` zum Test auf ein Blatt, d.h. linker und rechter Unterbaum sind leer, enthalten.

3. (8 Punkte) Die Klasse `StudentTree` soll folgendes enthalten:

   - Ein Attribut `root`, das auf die Wurzel des Baumes zeigt.
   - Ein Konstruktor, der einen leeren Baum erzeugt.
   - Eine Methode `void insertStudent(Student s)`, die einen Studenten an der richtigen Stelle im Baum einfügt.

     Falls es den Studenten mit der entsprechenden Matrikelnummer bereits gibt, sollen die Studentendaten mit dem neuen Studenten überschrieben werden.
   - Eine Methode `Student getStudent(int nr)`, welche den Zugriff auf den Studenten mit der entsprechenden Nummer erlaubt.

     Falls ein Student mit der entsprechenden Nummer nicht existiert, soll eine `Exception(String message)` ausgelöst werden.
   - Eine Methode `incAge()`, die das Alter aller Studenten im Baum um eins erhöht.