Klausur zur Vorlesung

Allgemeine Informatik II

14. Juli 2007 (SS 2007)

Prof. Dr. Franz Schweiggert / Norbert Heidenbluth

Bearbeitungszeit: 120 Minuten

NICHT MIT BLEISTIFT SCHREIBEN!

Name:	
Vorname:	
Matrikelnummer:	
Studiengang:	

Nr	Max	Bewertung	
1	**10**	xxxxx	
(a)	2		xxxxx
(b)	3		xxxxx
(c)	2		xxxxx
(d)	2		xxxxx
(e)	1		xxxxx
2	**14**	xxxxx	
(a)	3		xxxxx
(b)	3		xxxxx
(c)	4		xxxxx
(d)	4		xxxxx
3	**7**	xxxxx	
(a)	2		xxxxx
(b)	2		xxxxx
(c)	3		xxxxx
4	**12**	xxxxx	
(a)	4		xxxxx
(b)	3		xxxxx
(c)	5		xxxxx

Nr	Max	Bewertung	
5	**12**	xxxxx	
(a)	6		xxxxx
(b)	6		xxxxx
6	**15**	xxxxx	
(a)	5		xxxxx
(b)	5		xxxxx
(c)	5		xxxxx
7	**14**	xxxxx	
(a)	7		xxxxx
(b)	7		xxxxx
8	**6**	xxxxx	
9	**10**	xxxxx	
Summe	**100**		

Generelle Hinweise zur Bearbeitung dieser Klausur:

- Die Klausur muß aus 24 durchgehend nummerierten Seiten (einschließlich Deckblatt) bestehen. Bitte prüfen Sie vor Beginn der Klausur nach, ob Sie ein vollständiges Exemplar erhalten haben!
- Bitte benutzen Sie für die Lösungen den freigelassenen Platz nach der jeweiligen Angabe oder die Rückseite unter Angabe der Aufgabe.
- Sofern Sie bei der Bearbeitung der Aufgaben Annahmen treffen, geben Sie diese bitte mit an!
- Sofern in der Aufgabenstellung nichts anderes vermerkt ist, gilt:
  - Es ist Ihnen freigestellt, ob Sie zur Ein- und Ausgabe die Hilfsklassen aus **IOulm** verwenden oder auf native Java-Mechanismen zurückgreifen.
  - Es ist nicht erforderlich, mit Exception-Handling zu arbeiten.
  - Die zu schreibenden Programme in dieser Klausur müssen auf einem Java 1.4-Compiler lauffähig sein. Damit ist insbesondere die Verwendung von Auto-Inboxing und Auto-Outboxing **nicht** zulässig. Oder anders formuliert: es müssen ggf. Wrapperklassen explizit verwendet werden.
- Sofern Teilaufgaben aufeinander aufbauend sind, können Sie diese auch dann bearbeiten, wenn vorhergehende Teilaufgaben nicht gelöst wurden.
- Achten Sie auf die Aufgabenstellungen: Wenn beispielsweise eine Methode zu schreiben ist, so ist das Schreiben einer Verwendung dieser Methode (z.B. im Hauptprogramm) **nicht** erforderlich!

Viel Erfolg!

Aufgabe 1: Verständnisaufgaben **(10 Punkte)**

(a) 2 Punkte
Was versteht man in Java unter abstrakten Klassen?

Lösung:

(b) 3 Punkte
Wie funktioniert in Java das Exception-Handling?

Lösung:

(c) 2 Punkte
Warum sollten nach Möglichkeit alle Felder einer Klasse als `private` deklariert werden?

Lösung:

(d) 2 Punkte

Was versteht man im Zusammenhang mit der Programmierung von grafischen Benutzeroberflächen (GUIs) in Java unter einem *Action Listener*?

Lösung:

(e) 1 Punkte

Welche Bedeutung hat das Java-Schlüsselwort `final`?

Lösung:

Aufgabe 2: Heaps **(14 Punkte)**

Gegeben sei folgendes Array mit Integer-Elementen:

19	14	6	12	3	9	1	5	17	8

(a) 3 Punkte

Beim Heapsort-Verfahren wird das Array wie in der Vorlesung dargestellt zunächst in eine Baumstruktur überführt. Geben Sie diese zu obigem Array an!

Lösung:

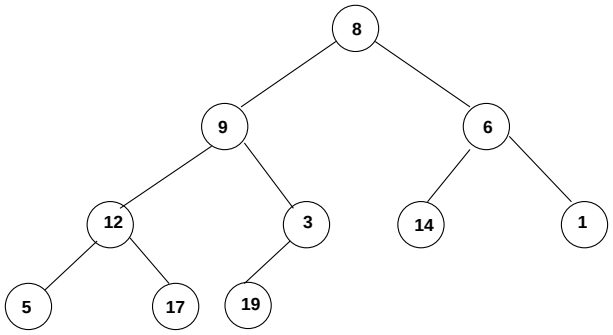
(b) 3 Punkte

Gegeben sei ein Array mit 256 Elementen. Kreuzen Sie in folgender Tabelle an:

Aussage	ist wahr	ist falsch
Der Knoten zu Index 116 ist ein Blattknoten		
Der Knoten zu Index 168 ist ein innerer Knoten		
Der Knoten zu Index 128 ist ein Blattknoten		

(c) 4 Punkte

Gegeben sei folgender Baum:



Stellen Sie für diesen Baum nach dem in der Vorlesung vorgestellten Verfahren die Heap-Eigenschaft her! Nach jedem Tauschschritt ist der Baum komplett in den unten vorgegebenen Graphen wiederzugeben! Die Zahl der angegebenen Graphen / Tauschschritte kann größer sein als die tatsächlich benötigte!

Lösung:

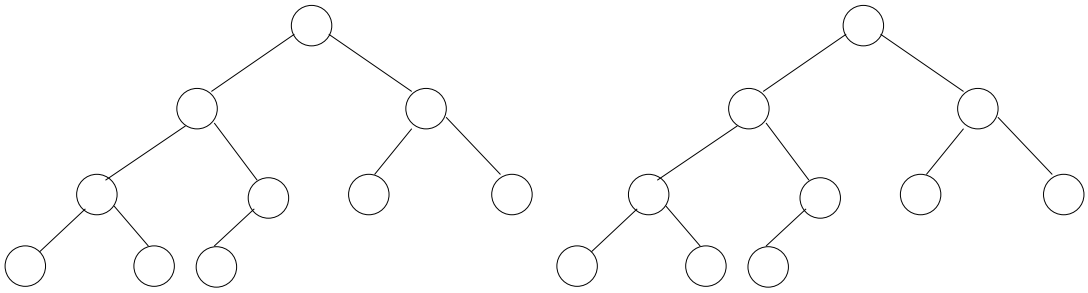


Abbildung 1: Nach dem 1. und 2. Tauschschritt

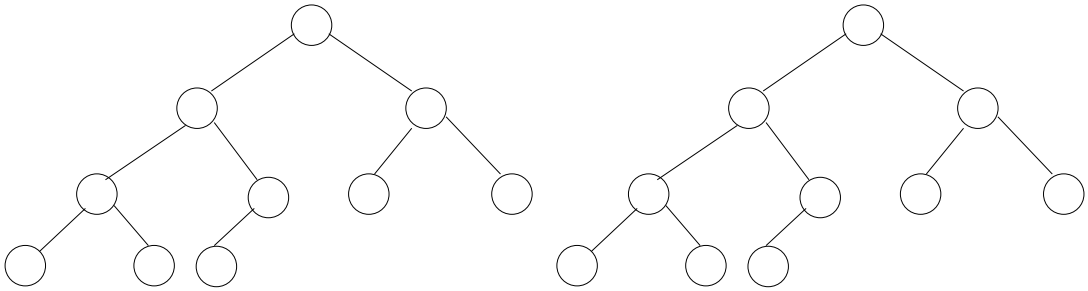


Abbildung 2: Nach dem 3. und 4. Tauschschritt

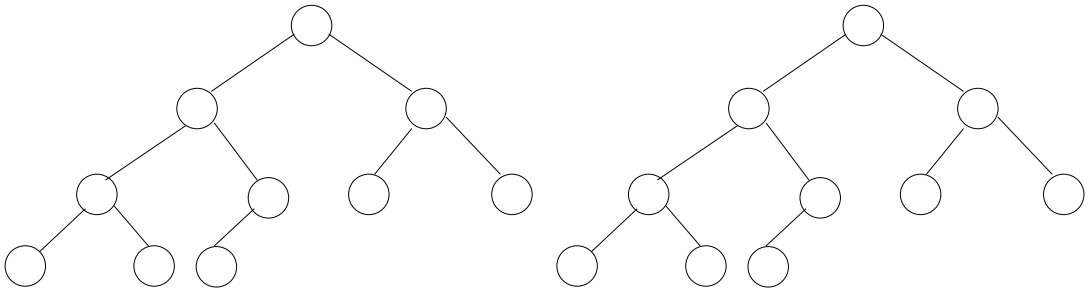
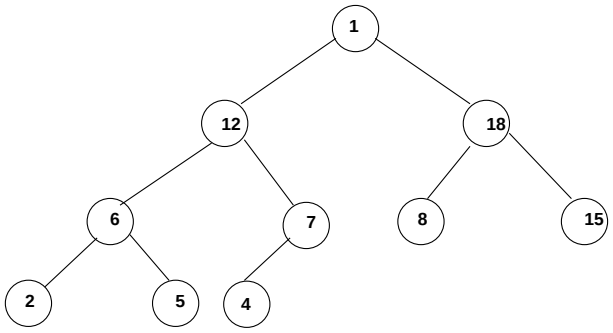


Abbildung 3: Nach dem 5. und 6. Tauschschritt

(d) 4 Punkte

Der folgende Baum erfüllt die Semi-Heap-Eigenschaft:



Stellen Sie für diesen Baum nach dem in der Vorlesung vorgestellten Verfahren die Heap-Eigenschaft her! Nach jedem Tauschschritt ist der Baum komplett in den unten vorgegebenen Graphen wiederzugeben! Die Zahl der angegebenen Graphen / Tauschschritte kann größer sein als die tatsächlich benötigte!

Lösung:

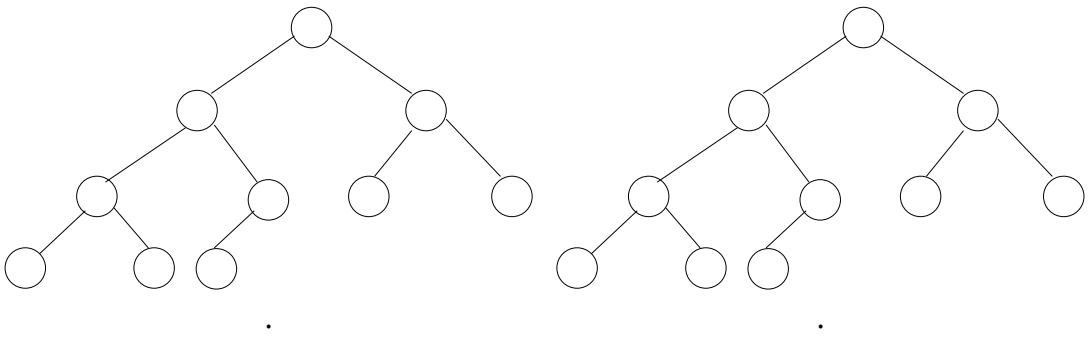


Abbildung 4: Nach dem 1. und 2. Tauschschritt

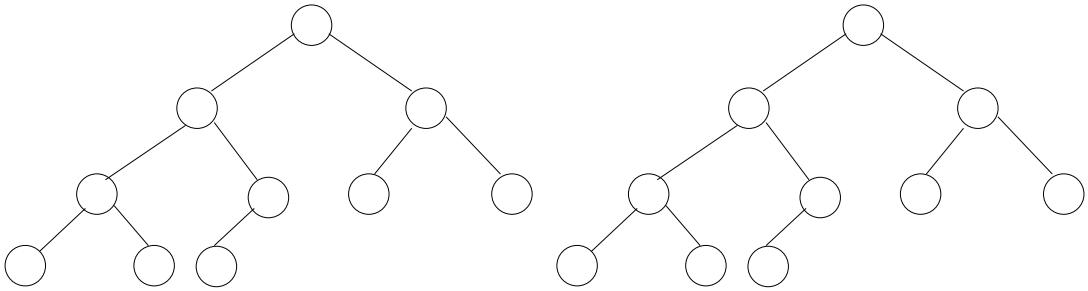


Abbildung 5: Nach dem 3. und 4. Tauschschritt

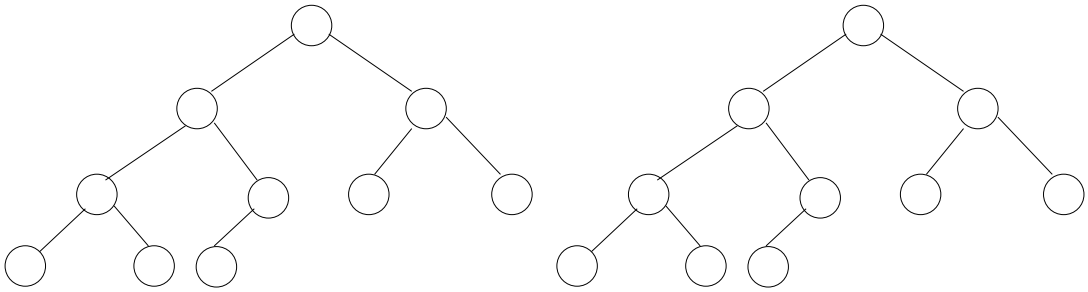


Abbildung 6: Nach dem 5. und 6. Tauschschritt

Aufgabe 3: Der Quicksort-Algorithmus **(7 Punkte)**

Gegeben sei folgendes Array mit Integer-Elementen:

19	14	6	12	3	29	1	5	17	8

(a) 2 Punkte
Begründen Sie kurz, aber präzise (!), warum das Element 29 (Index 5) für den ersten Teilungsschritt beim Quicksort-Verfahren ungünstig ist!

Lösung:

(b) 2 Punkte
Welches Element wäre (für den ersten Schritt beim Quicksort-Verfahren) ideal? Begründen Sie Ihre Antwort!

Lösung:

(c) 3 Punkte

Verwenden Sie nun das Element 19 (Index 0) für den ersten Teilungsschritt. Tragen Sie in die vorgegebenen Leerfelder den Inhalt des Arrays nach jedem Tauschschritt komplett ein!

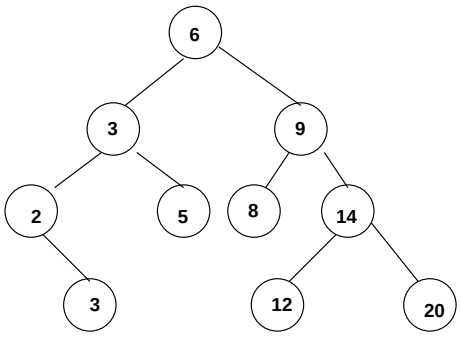
Lösung:

19	14	6	12	3	29	1	5	17	8

Aufgabe 4: Binärbäume **(12 Punkte)**

(a) 4 Punkte

Gegeben sei folgender Binärbaum:



(1.) Ist dieser Baum nach Gewicht ausgeglichen? Begründen Sie Ihre Antwort!

Lösung:

(2.) Ist dieser Baum nach Höhe ausgeglichen? Begründen Sie Ihre Antwort!

Lösung:

(b) 3 Punkte

Für den Baum aus Teilaufgabe (a) sei die folgende Java-Klasse definiert:

```
public class BinTree{
   private class Node {
      Integer content;
      Node left, right;
      Node(Integer i){ content = i; left = right = null; }
   }

   private int count;
   private Node root;
   public BinTree(){
      count = 0;
      root  = null;
   }

// hier koennten diverse Methoden stehen

// jetzt kommt der spannende Teil:

   StringBuffer sb;
   private void collect(Node x) {
      if ( x == null) return;
      collect(x.right);
      sb.append(x.content.toString());
      sb.append(" ");
      collect(x.left);
   }

   public String toString() {
      sb = new StringBuffer();
      collect(root);
      return sb.toString();
   }
}
```

Geben Sie exakt an, was die Anwendung der *toString()* Methode auf den in der Grafik in Teilaufgabe a) gegebenen Baum liefert!

Lösung:

(c) 5 Punkte

Ergänzen Sie die Klasse aus Teilaufgabe b) um eine Methode, die die Häufigkeit des Vorkommens einer als Argument übergebenen Zahl in einem Baum liefert! Diese Methode **muss rekursiv arbeiten**!

Lösung:

```
public int count (Node root, int candidate) {

}
```

Aufgabe 5: Arbeiten mit Strings **(12 Punkte)**

(a) 6 Punkte

Schreiben Sie eine Methode `dateValidator`, welche einen String übergeben bekommt. Aufgabe der Methode soll es sein, den übergebenen String daraufhin zu untersuchen, ob er ein korrekt formatiertes Datum enthält.

Ein korrekt formatiertes Datum habe dabei die Form "TT.MM.JJJJ" – die Angabe des Tages und des Monats seien also stets zweistellig, die Angabe des Jahres vierstellig.

Abhängig davon, ob das Datum eine gültige *Form* hat oder nicht, soll `TRUE` bzw. `FALSE` zurückgegeben werden.

Die Gültigkeit des Datums braucht in dieser Teilaufgabe noch nicht validiert zu werden!

Ob Sie diese Aufgabe unter Verwendung von regulären Ausdrücken lösen oder nicht, ist Ihnen freigestellt.

Lösung:

(b) 6 Punkte

Ergänzen Sie Teilaufgabe (a) nun dahingehend, dass Sie auch die Gültigkeit des Datums überprüfen. Der 31.12.2000 wäre z.B. ein gültiges Datum, der 31.02.2000 hingegen nicht.

Dabei seien hier (nur!) die Jahre von 1900 bis 2100 (jeweils einschließlich) als gültig zu betrachten.

Die Berücksichtigung von Schaltjahren ist hier ausnahmsweise **nicht** erforderlich!

Beachte: Schreiben Sie hier nur den Teil auf, der zu (a) ergänzend ist. Markieren Sie ggf. in Ihrer Lösung zur Teilaufgabe (a), wo dieser Ergänzungsteil einzufügen wäre.

Lösung:

Aufgabe 6: Exception Handling **(15 Punkte)**

(a) 5 Punkte

Schreiben Sie eine Methode `yesOrNo`, welche einen String übergeben bekommt. Sofern der String den Inhalt “yes” (in exakt dieser Schreibweise) besitzt, soll die Methode `TRUE` zurückliefern, und wenn der String den Inhalt “no” hat, soll `FALSE` zurückgeliefert werden.

Darüberhinaus sollen alle anderen Inhalte des übergebenen Strings eine Exception der allgemeinen Klasse `Exception` mit dem Inhalt “Wrong content” auslösen.

Lösung:

(b) 5 Punkte

Gegeben sei die folgende `main()`-Methode:

```
public static void main (String[] args) {

   if (!Urc.readString()) {
      System.err.println("Fehler beim Einlesen!");
      System.exit(1);
   }

   String text = Urc.getString();

   boolean b;
   // Ihre Loesung wuerde an dieser Stelle
   // in das Programm eingefuegt!

}
```

Ergänzen Sie diese `main()`-Methode um die Anwendung der Methode aus (a). Im Erfolgsfall soll die Variable `b` nach der Ausführung der Methode Rückgabewert der Methode besitzen. Andernfalls soll das Programm mit der erhaltenen Exception-Meldung mit Exit-Code 2 terminieren.

Sie können diese Aufgabe auch dann bearbeiten, wenn Sie keine Lösung für die Teilaufgabe (a) gefunden haben.

Lösung:

(c) 5 Punkte

Gegeben sei der folgende Ausschnitt aus der Java-Doc-Dokumentation der Klasse `String`:

> `String toLowerCase()`
> Converts all of the characters in this String to lower case using the rules of the default locale.
> `String toUpperCase()`
> Converts all of the characters in this String to upper case using the rules of the default locale.

Verändern Sie die Methode aus Teilaufgabe (a) dahingehend, dass sie unabhängig von der Gross- und Kleinschreibung von “yes” bzw. “no” funktioniert. So soll beispielsweise die Eingabe von “YEs” genauso zum Rückgabewert `TRUE` führen wir “yEs”, “YES” usw. Gleiches gilt auch für “No”, “nO”, usw..

Es genügt, wenn Sie die wesentlichen Änderungen an der Methode erneut schreiben. Eine Wiederholung des Exception-Handlings ist für diese Teilaufgabe nicht erforderlich.

Beachte: Schreiben Sie hier nur den Teil auf, der zu (a) ergänzend ist. Markieren Sie ggf. in Ihrer Lösung zur Teilaufgabe (a), wo dieser Ergänzungsteil einzufügen wäre.

Lösung:

Aufgabe 7: Lineare Listen **(14 Punkte)**

Gegeben sei die folgende Definition einer Liste:

```
public class VerySimpleAssignmentList

   public String matnr;
   public Integer number;
   public VerySimpleAssignmentList next;

}
```

Wir gehen davon aus, dass die Liste dazu dient, Klausurergebnisse zu verwalten (d.h. pro Listenelement die Matrikelnummer jeweils eines Studierenden als String sowie das Klausurergebnis als ganze Zahl zwischen 0 und 100).

(a) 7 Punkte

Schreiben Sie eine Methode, welche ein Objekt der oben gegebenen Klasse `VerySimpleAssignmentList.java` übergeben bekommt (dies sei die Referenz auf das Startelement der Liste) und ein Histogramm-Array aus der gesamten Liste erstellt und zurückliefert!

Das Histogramm-Array soll die Häufigkeit angeben, mit der ein Ergebnis aus einem bestimmten Intervall vorkommt. Die Intervallgrenzen seien dabei 0 bis 9 Punkte, 10 bis 19 Punkte, 20 bis 29 Punkte, usw. (in 10er-Schritten).

Die Ausgabe des Histogramms soll als reine Textdarstellung erfolgen, also z.B. wie folgt:

```
[0-10]:  4
[11-20]: 2
[21-30]: 5
usw.
```

Lösung:

(b) 7 Punkte

Schreiben Sie eine Methode, die erneut eine Referenz auf das Startelement einer Liste des Typs `VerySimpleAssignmentList` sowie daüberhinaus eine ganze Zahl des *primitiven Datentyps* `int` und einen String übergeben bekommt.

Die Methode soll dann ein neues Listenelement bestehend aus den übergebenen Parametern (d.h Punktezahl und Matrikelnummer) gemäß der folgenden Spezifikation einfügen:

- Ist die Liste leer, so ist das neue Element das einzige der Liste.
- Besteht die Liste aus nur einem Element, so wird das neue Element **vorne** eingefügt.
- In allen anderen Fällen soll das neue Element an die **vorletzte** Stelle der Liste eingefügt werden.

Lösung:

Aufgabe 8: Rekursive Methoden **(6 Punkte)**

Schreiben Sie eine *rekursive* Methode `count`, die ein durchgehend (d.h. lückenlos) gefülltes Array von Objekten der Klasse `Integer` sowie einen unteren Index übergeben bekommt und die Summe der Werte der im Array enthaltenen Integer-Objekten *rekursiv* ermittelt und zurückgibt.

Beachte:

- Die Verwendung von `length` ist **nicht** zulässig!.
- Der Versuch eines Zugriffs auf Arrayelemente außerhalb des gültigen Bereichs löst eine `ArrayOutOfBoundsException` aus!

Lösung:

Aufgabe 9: Java-Grundwissen **(10 Punkte)**

Kreuzen Sie an, ob die nachstehenden Aussagen richtig oder falsch sind. Für jede korrekte Antwort gibt es einen Punkt, für jede falsche Antwort ziehen wir einen Punkt ab. Unbeantwortete Aussagen zählen nicht. Es ist also besser, bei Unsicherheit eine Aussage unbeantwortet zu lassen. Die Abzüge wirken sich nur auf diese Teilaufgabe aus. Schlimmstenfalls erhalten Sie also nur 0 Punkte.

Frage	Richtig	Falsch
Eine Klasse kann beliebig viele Konstruktoren haben (auch keinen)!		
Der Nachfolger des letzten Elements einer einfach verketteten linearen Liste darf **nicht** auf das Startelement dieser Liste zeigen!		
Interfaces können Methoden implementieren (müssen aber nicht)!		
Von abstrakten Klassen kann kein Objekt instanziiert werden.		
Jede Klasse ist stets eine unmittelbare Ableitung der Klasse `Object`.		
Ein dem Gewicht nach ausgeglichener Binärbaum kann nicht nach Höhe ausgeglichen sein.		
Die Parameterübergabe für Objekte erfolgt bei Java stets als *call by reference*.		
Der Name einer Variablen muss in Java pro Klasse eindeutig sein.		
Die Anzahl der Knoten in einem Binärbaum ist zur Compilier-Zeit bereits bekannt.		
Für das Vererben von Klassenmethoden wird Erbschaftssteuer fällig!		