Lösungsvorschlag zur

Klausur zu Allgemeine Informatik II

15. Juli 2006 (SS 2006)

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)	2		xxxxx
(c)	2		xxxxx
(d)	2		xxxxx
(e)	2		xxxxx
2	**10**	xxxxx	
3	**8**	xxxxx	
4	**8**	xxxxx	
(a)	4		xxxxx
(b)	4		xxxxx
5	**6**	xxxxx	
6	**8**	xxxxx	
(a)	4		xxxxx
(b)	4		xxxxx

Nr	Max	Bewertung	
7	**12**	xxxxx	
(a)	6		xxxxx
(b)	6		xxxxx
8	**14**	xxxxx	
(a)	4		xxxxx
(b)	4		xxxxx
(c)	6		xxxxx
9	**6**	xxxxx	
10	**10**	xxxxx	
(a)	4		xxxxx
(b)	2		xxxxx
(c)	4		xxxxx
11	**8**	xxxxx	
Summe	**100**		

Generelle Hinweise zur Bearbeitung dieser Klausur:

- 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.

Viel Erfolg!

Aufgabe 1 **(10 Punkte)**

(a) 2 Punkte
Was ist ein Konstruktor?

Lösungsvorschlag:

Ein Konstruktor legt ein neues Objekt an. Dazu wird einerseits der benötigte Speicherplatz reserviert. Üblicherweise werden darüberhinaus auch Default-Initialisierungen des Objektes durch den Konstruktor vorgenommen.

Ein Konstruktor heißt wie die Klasse selbst und besitzt keinen Rückgabetyp. Es kann pro Klasse mehrere Konstruktoren geben. Diese werden aufgrund verschiedener Parameterlisten voneinander unterscheiden.

(b) 2 Punkte
Erklären Sie den Unterschied zwischen Klassen- und Objektmethoden! Wie kann man im Java-Quellcode erkennen, ob es sich bei einer Methode um eine Klassen- oder Objektmethode handelt?

Lösungsvorschlag:

Klassenmethoden operieren auf der Klasse selbst und setzen kein existierendes Objekt voraus. Im Unterschied dazu gibt es Objektmethoden, die auf einem Objekt, d.h. einer Instanz dieser Klasse, arbeiten und – eine entsprechende Implementierung vorausgesetzt – je nach Objekt unterschiedliches Verhalten zeigen können.

Im Quellcode sind Klassenmethoden am Schlüsselwort `static` erkennbar.

(c) 2 Punkte
Erläutern Sie kurz, was man unter einer Wrapper-Klasse versteht und wozu derartige Klassen benötigt werden!

Lösungsvorschlag:

Wrapperklassen dienen dazu, primitive Datentypen (wie `int`, `boolean`, `char` usw.) wie Objekte verwenden zu können. Dies ist zum Beispiel im Zusammenhang mit Datenstrukturen erforderlich, welche nur Objekte aufnehmen können.

Wrapper-Klassen “verpacken” die primitive Variable in ein Objekt und bieten entsprechende Zugriffsmethoden auf die enthaltene Variable.

(d) 2 Punkte
Was ist (in Java) ein Interface?

Lösungsvorschlag:

Ein Interface ist eine besondere Klasse, die lediglich Deklarationen von Methoden enthält, nicht aber deren Implementierung. Klassen, welche ein bestimmtes Interface implementieren, müssen alle der im Interface genannten Methoden enthalten. Bildlich gesprochen können Interfaces als ein "Kontrakt" gesehen werden, welche Methoden eine Klasse (die dieses Interface implementiert) mindestens bereitstellen.

(e) 2 Punkte
Erläutern Sie kurz die Bedeutung der Java-Schlüsselworte `public` und `private`!

Lösungsvorschlag:

Auf Methoden und Felder, die als *private* deklariert sind, kann nur innerhalb der jeweiligen Klasse unmittelbar zugegriffen werden. Für den Zugriff von außen benötigt man hier entsprechende Methoden. Im Unterschied dazu sind als *public* deklarierte Felder und Methoden auch von außen unmittelbar (für Lese- und Schreibzugriffe) zugänglich.

Aufgabe 2 **(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
Pro Klasse (in Java) darf es nur genau einen Konstruktor geben!		X
Die Anzahl der Elemente, die in eine lineare Liste eingefügt werden können, ist durch die lineare Liste nach oben beschränkt.		X
Eine Klasse kann höchstens ein Interface implementieren!		X
Der in der Vorlesung vorgestellte Suchalgorithmus "Binärsuche" geht nach dem "Divide and conquer"-Prinzip vor.	X	
Beim Exception-Handling ist die Reihenfolge der `catch`-Blöcke relevant.	X	
Ein der Höhe nach ausgeglichener Binärbaum ist immer auch nach Gewicht ausgeglichen.		X
Die Parameterübergabe erfolgt bei Java stets als *call by reference*.		X
Der Name einer Methode muss in Java pro Klasse eindeutig sein.		X
In eine Liste, deren Elemente als vom Typ `Object` deklariert sind, können Objekte von beliebigen (anderen) Typen aufgenommen werden.	X	
Italien ist Fußball-Weltmeister 2006.	X	

Aufgabe 3 **(8 Punkte)**

Ein Programm soll als Argumente beim Aufruf genau die folgenden Parameter (in der angegebenen Reihenfolge) erhalten:

1. genau einen Kleinbuchstaben aus der Menge (“d”, “h”, “n”, “m” oder “s”),
2. zwei positive ganze Zahlen sowie
3. einen String, dessen Länge genau 7 Zeichen betragen muß.

Schreiben Sie ein solches Programm gemäß der nachstehenden Spezifikation:

Sofern die bei Programmstart übergebenen Parameter **nicht** den oben genannten Anforderungen entsprechen, bricht das Programm mit einer entsprechenden Usage-Meldung, die auf die **Standardfehlerausgabe** erfolgen soll, ab. Andernfalls gibt es die Argumente einzeln aus und terminiert dann regulär.

Aufgabe 4 **(8 Punkte)**

Gegeben sei die folgende Klasse `Room`:

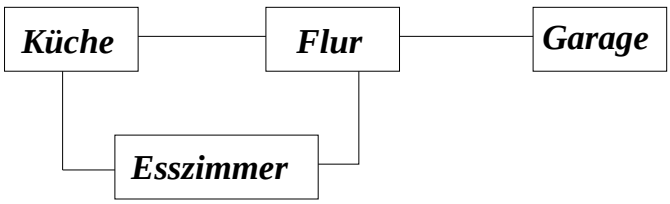
```
public class Room {

   public Room north, east, south, west;
   public String name;

   // Raum-Name wird als Parameter uebergeben, die Referenzen
   // auf Nachbarraeume werden zunaechst alle auf null gesetzt
   public Room(String name) {
      this.name = name;
      this.north = null; this.east = null;
      this.south = null; this.west = null;
   }
}
```

Instanzen dieser Klasse sollen einen Raum mit einer Bezeichnung (Namen) sowie mit maximal vier Ausgängen (jeweils einer im Norden, Osten, Süden und Westen des Raums) repräsentieren. Die Null-Referenz hat im Zusammenhang mit den Feldern `north`, `east`, `south` und `west` die Bedeutung, dass sich in die entsprechende Richtung kein weiterer Raum anschließt.

(a) 4 Punkte

Schreiben Sie den Programmcode auf, der zu dem folgenden Grundriss führt:

(Für eine Anregung, wie dieser Code aussehen könnte: siehe Teilaufgabe b).


Küche
Flur
Garage
Esszimmer

Lösungsvorschlag:

```
Room e = new Room("Esszimmer);
Room k = new Room("Kueche");
Room f = new Room("Flur");
Room g = new Room("Garage");
k.east = f; k.south = e;
f.west = k; f.south = e; f.east = g;
e.west = k; e.east = f;
g.west = f;
```

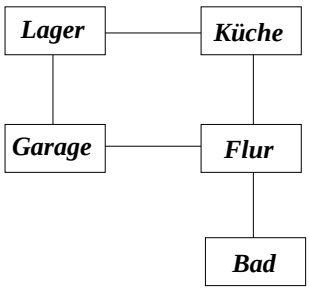
(b) 4 Punkte

Skizzieren Sie, welcher "Grundriss" durch den auf nachfolgenden Programmcode entsteht.

(Für eine Anregung, wie eine solche Skizze aussehen könnte: siehe Teilaufgabe a).

```
// Raume instanziieren
Room f = new Room("Flur");
Room k = new Room("Kueche");
Room b = new Room("Bad");
Room g = new Room("Garage");
Room l = new Room("Lager");

// Verbindungen zwischen Raeumen herstellen
f.south = b; f.north = k; f.west = g;
k.south = f; k.west = l;
l.east = k; l.south = g;
g.north = l; g.east = f;
b.north = f;
```

Lösungsvorschlag:


Lager
Küche
Garage
Flur
Bad

Aufgabe 5 **(6 Punkte)**

Gegeben seien die folgenden natürlichen Zahlen:

40	***18***	***99***	***36***	***48***	***72***	***76***

Sortieren Sie die oben notierten Zahlen unter Verwendung des **Quicksort**-Algorithmus, so dass sie am Ende aufsteigend sortiert sind.

Wichtig: Notieren Sie alle Zwischenschritte (d.h., das jeweilige Pivot-Element sowie die jeweilige Partitionierung) der durchgeführten Sortierung, damit das angewandte Sortierverfahren erkennbar und nachvollziehbar ist. Verwenden Sie bei der Wahl des Pivot-Elements jeweils eine der im Skript erwähnten Vorgehensweisen, und geben Sie diese auch in Ihrer Lösung an!

Lösungsvorschlag:

—

Aufgabe 6 **(8 Punkte)**

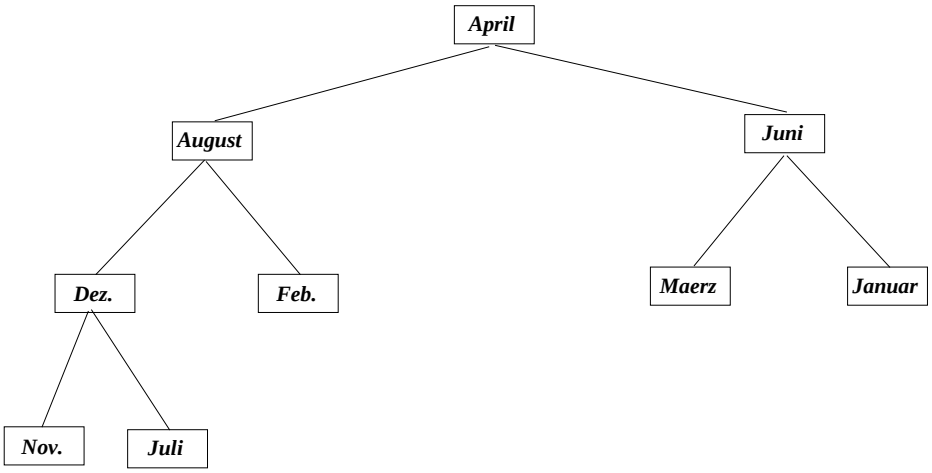
Gegeben sei das folgende String-Array:

```
String[] months = new String[9];
months[0] = "April"; months[1] = "August"; months[2] = "Juni";
months[3] = "Dezember"; months[4] = "Februar"; months[5] = "Maerz";
months[6] = "Januar"; months[7] = "November"; months[8] = "Juli";
```

(a) 4 Punkte

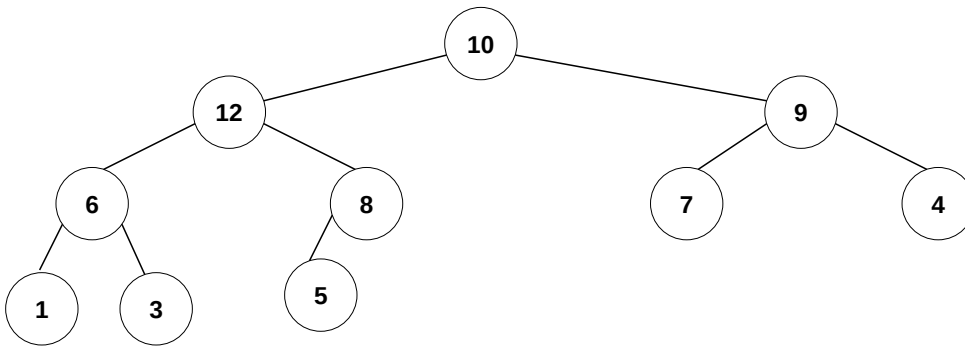
Interpretieren Sie dieses Array als Baum – so wie es im Zusammenhang mit dem Heapsort erläutert wurde – und zeichnen Sie diesen Baum.

Lösungsvorschlag:


April
August
Juni
Dez.
Feb.
Maerz
Januar
Nov.
Juli

(b) 4 Punkte

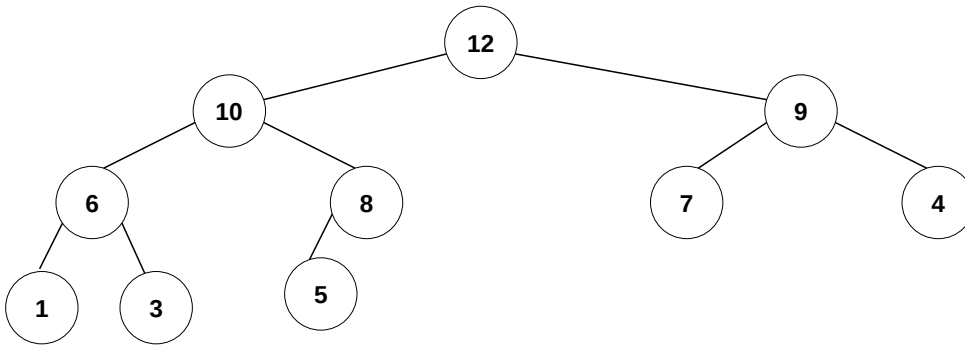
Gegeben sei der folgende Baum:



Überführen Sie diesen Baum in einen Heap und zeichnen Sie den entstandenen

Heap! Gehen Sie hierbei nach dem in der Vorlesung vorgestellten Algorithmus `buildHeap` vor.

Lösungsvorschlag:



Aufgabe 7 **(12 Punkte)**

Gegeben sei die Klasse `List` aus der Vorlesung (siehe Skript: Programm 10.1, Seite 236).

(a) 6 Punkte
Schreiben Sie eine Instanz-Methode `getLastElement()`, welche das letzte Element in der linearen Liste zurückliefert.

Lösungsvorschlag:

```
public Object getLastElement() {
   Node n = first;
   if (n == null)
      return null;   // Sonderfall: Liste war leer
   while (n.next != null)
      n = n.next;
   return n;
}
```

(b) 6 Punkte
Schreiben Sie einen Konstruktor für die Klasse `List`, welcher als Parameter ein Array von Variablen des (primitiven) Datentyps `int` übergeben bekommt und dieses Array in eine Liste umwandelt.

Lösungsvorschlag:

```
public List (int[] numbers) {
   this.count = 0;
   this.first = null;
   for (int i=0; i<numbers.length; ++i) {
      Integer i = new Integer(numbers[i]);
      add(i);
   }
}
```

Aufgabe 8 **(14 Punkte)**

Gegeben sei die sortierte lineare Liste `SortedList1` aus der Vorlesung (vgl. Programm 10.12, Seite 252). Die Methode `toString()` wurde für diese Aufgabe der besseren Optik wegen leicht modifiziert – aber dies hat keine weitere Bedeutung für diese Aufgabe!

Ferner sei das folgende Programm `Flensburg.java` gegegeben:

```
import IOulm.*;

public class Flensburg {

   public static void main (String[] args) {

      SortedList1 liste = new SortedList1();
      Auto a1 = new Auto("Porsche", "Carrera", 3800, "M-LL 4711");
      Auto a2 = new Auto("Opel", "Manta", 2000, "BM-XY 999");
      Auto a3 = new Auto("VW", "Golf", 1600, "UL-AB 123");

      liste.insertElement(a1);
      liste.insertElement(a2);
      liste.insertElement(a3);

      Write.Line("Und hier die registrierten Autos:\n" + liste);
   }
}
```

In dieser Aufgabe sollen Sie die Klasse `Auto.java` so implementieren, dass das vorstehende Programm die folgende Ausgabe produziert:

```
Und hier die registrierten Autos:
Das Fahrzeug mit dem amtlichen Kennzeichen UL-AB 123
ist ein VW Golf und hat einen Hubraum von 1600 ccm.

Das Fahrzeug mit dem amtlichen Kennzeichen BM-XY 999
ist ein Opel Manta und hat einen Hubraum von 2000 ccm.

Das Fahrzeug mit dem amtlichen Kennzeichen M-LL 4711
ist ein Porsche Carrera und hat einen Hubraum von 3800 ccm.
```

Gehen Sie dazu nach den folgenden Teilaufgaben vor.

(a) 4 Punkte

Implementieren Sie zunächst **nur** den benötigten Konstruktor.

Lösungsvorschlag:

```
public Auto(String marke, String modell,
            int hubraum, String akz) {
   this.marke = marke;
   this.modell = modell;
   this.hubraum = hubraum;
   this.akz = akz;
}
```

(b) 4 Punkte

Implementieren Sie nun **nur** die Methode, die zur Ausgabe der Daten benötigt wird.

Lösungsvorschlag:

```
public String toString() {
   StringBuffer sb = new StringBuffer();
   sb.append("Das Fahrzeug mit dem " +
             "amtlichen Kennzeichen " + akz);
   sb.append(" ist ein " + marke + " " + modell);
   sb.append(" und hat einen Hubraum von " +
             hubraum + " ccm.");
   return sb.toString();
}
```

(c) 6 Punkte

Schreiben Sie nun den Rahmen der Klasse inkl. der Variablen und implementieren Sie nun evtl. weitere benötigte Methoden. Die Methoden der beiden vorigen Teilaufgaben brauchen hier **nicht** erneut implementiert zu werden!

Lösungsvorschlag:

```
import IOulm.*;

public class Auto implements Comparable {

   private String marke;
   private String modell;
   private int hubraum;
   private String akz;

   // Konstruktor: siehe Aufgabe a)

   // toString-Methode: siehe Aufgabe b)

   public int compareTo(Object o) {
      if (o instanceof Auto) {
         return hubraum - (((Auto) o).hubraum);
      }
      // Else
      Write.Line("Falscher Objekttyp in der Liste!");
      System.exit(1);
      return -99;
   }
}
```

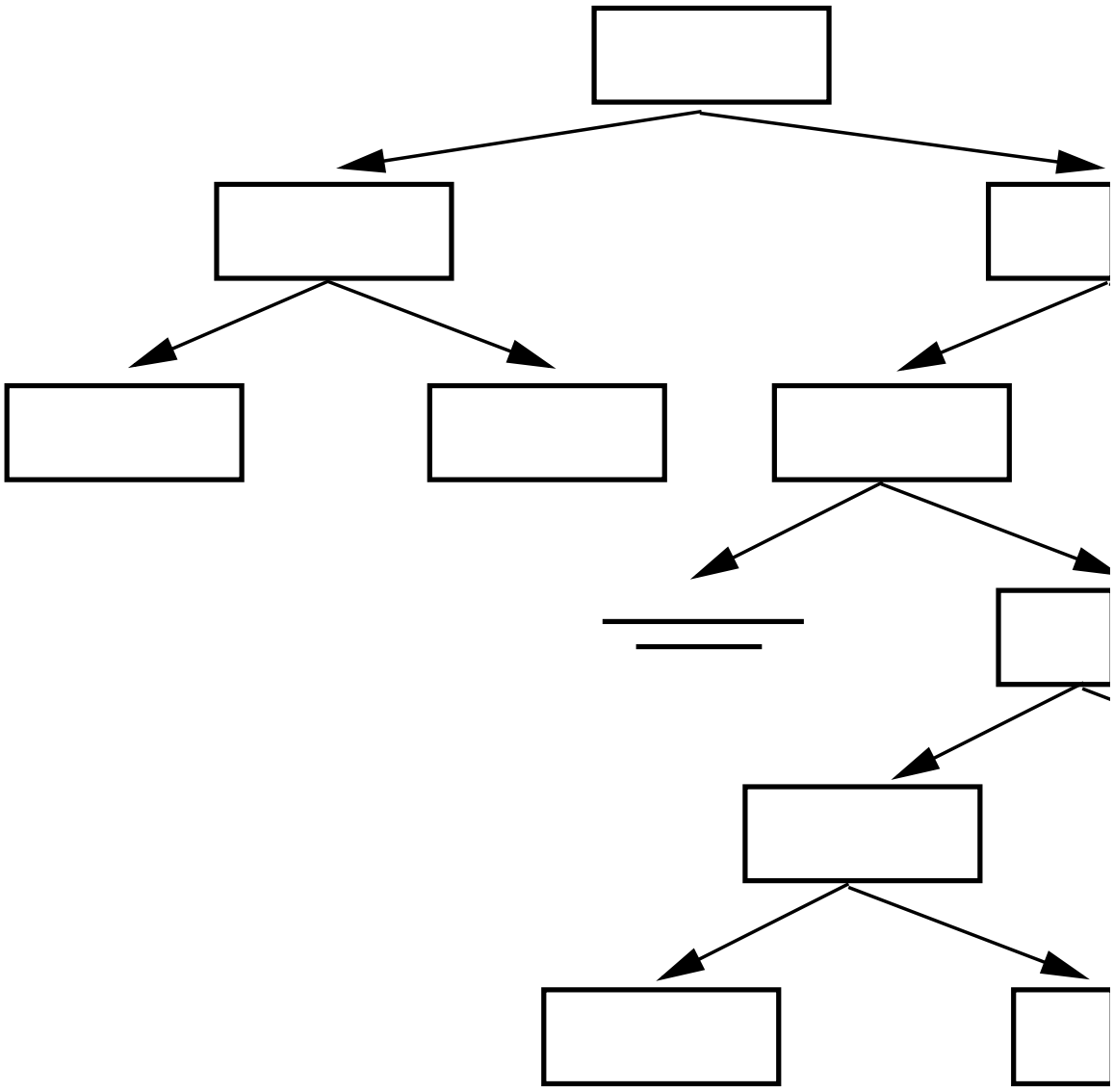
Aufgabe 9 **(6 Punkte)**

Gegeben sei der folgende Baum:

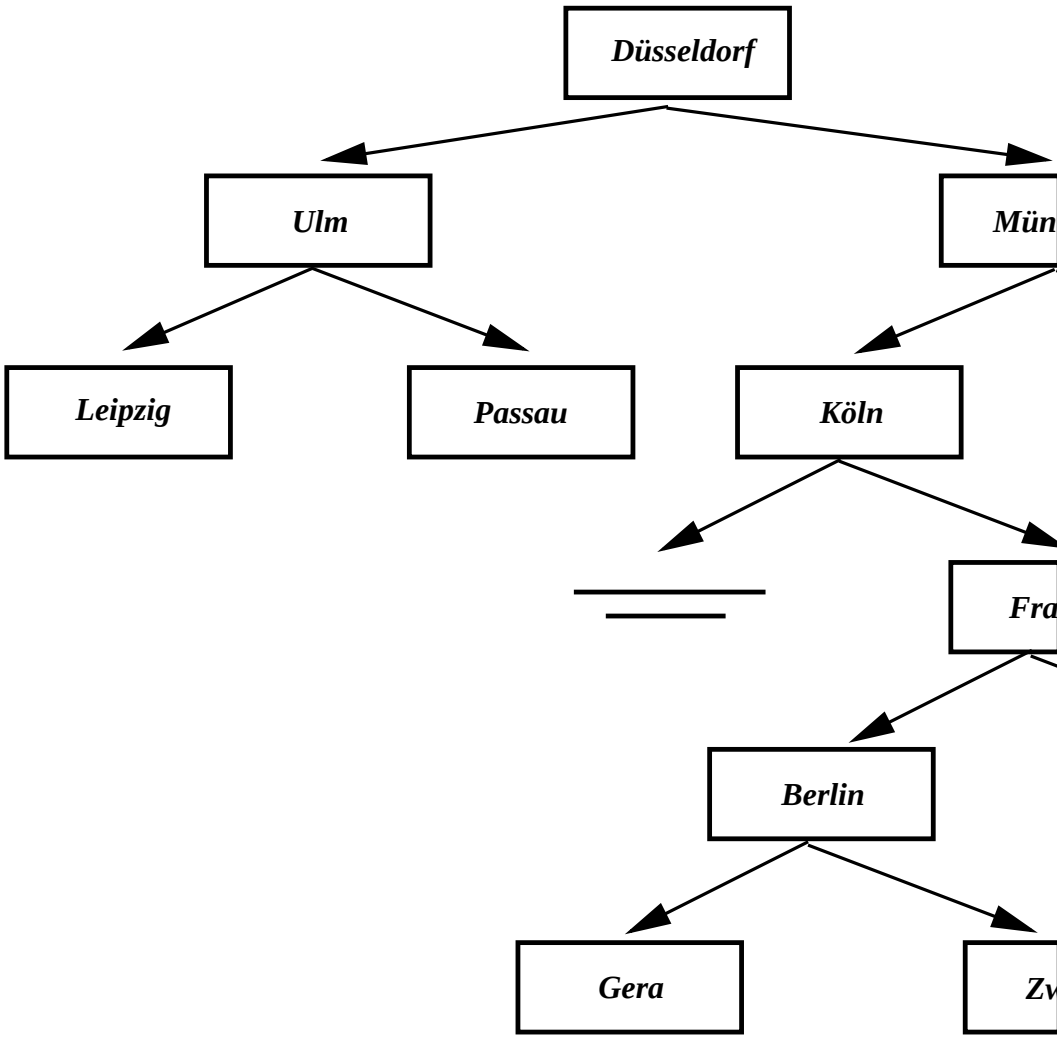
Fügen Sie die nachstehenden Städtenamen so in den gegebenen Baum ein, dass sie, wenn der Baum **postorder** traversiert wird, in der selben Reihenfolge wie in dieser Aufgabenstellung ausgegeben werden:

Leipzig, Passau, Ulm, Gera, Zwickau, Berlin, Frankfurt, Köln, Stuttgart, München, Düsseldorf

Hinweis: Der besseren Übersicht wegen sind die `null`-Referenzen der Blätter in dieser Skizze nicht dargestellt.



Lösungsvorschlag:



Aufgabe 10 **(10 Punkte)**

Gegeben sei der folgende sortierte Binärbaum:

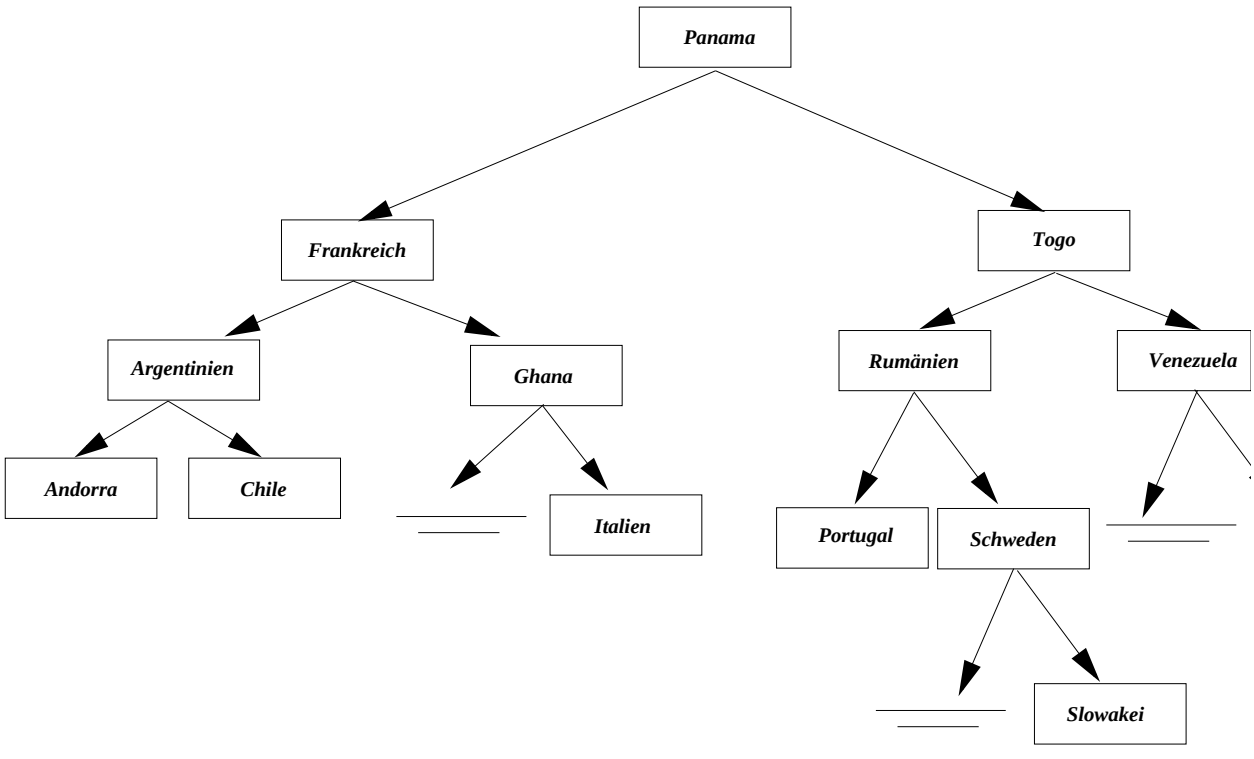
Hinweis: Auch hier sind der besseren Übersicht wegen die `null`-Referenzen der Blätter nicht dargestellt.

(a) 4 Punkte
Traversieren Sie den Baum in **preorder** und **inorder** und schreiben Sie jeweils die Ausgabe auf.

Lösungsvorschlag:

preorder: Panama, Frankreich, Argentinien, Andorra, Chile, Ghane, Italien, Togo, Rumänien, Portugal, Schweden, Slowakei, Venezuela, Zypern

inorder: Andorra, Argentinien, Chile, Frankreich, Ghana, Italien, Panama, Portugal, Rumänien, Schweden, Slowakei, Togo, Venezuela, Zypern


Panama
Frankreich
Togo
Argentinien
Ghana
Rumänien
Venezuela
Andorra
Chile
Italien
Portugal
Schweden
Slowakei

(b) 2 Punkte

Ist der Baum der Höhe nach ausgeglichen? Ist er nach Gewicht ausgeglichen? Begründen Sie Ihre Antwort!

Lösungsvorschlag:

Der Baum ist **der Höhe** nach ausgeglichen, aber **nicht dem Gewicht** nach.

Grund: Die Definition der Ausgeglichenheit bezieht sich nicht nur auf den Baum als solches sondern auch auf alle seine Teilbäume. Während beide Ausgeglichenheiten von "Panama" aus gesehen erfüllt sind, trifft die Gewichtsausgeglichenheit auf den Teilbaum, dessen Wurzel "Togo" ist, nicht mehr zu: dort sind im linken Teilbaum vier Knoten enthalten, wohingegen der rechte Teilbaum nur zwei Blätter enthält.

(c) 4 Punkte

Entfernen Sie (unter Verwendung des in der Vorlesung vorgestellten Vorgehens) **Togo** aus dem Baum, und machen Sie im obigen Bild (durch Durchstreichen und "Drüberschreiben") kenntlich, wie der Baum nach dieser Operation aussieht. Beachten Sie, dass der Baum nach dieser Operation weiterhin sortiert sein soll!

Lösungsvorschlag: Man ersetze beispielsweise Togo durch die Slowakei.

Aufgabe 11 **(8 Punkte)**

Gegeben sei die Klasse `BinTree.java`:

```
public class BinTree {

   private Object content;
   private BinTree left;
   private BinTree right;

   // Konstruktor und weitere Methoden dieser Klasse
   // fuer diese Aufgabe unbedeutend
}
```

Schreiben Sie eine **rekursive** Instanz-Methode `getNumberOfNodes()`, welche die Anzahl der im Baum enthaltenen Knoten zurückliefert.

Lösungsvorschlag:

```
public int getNumberOfNodes() {
    int leftNodes = 0;
    int rightNodes = 0;
    if (this.left != null) { leftNodes = left.getNumberOfNodes(); }
    if (this.right != null) { rightNodes = right.getNumberOfNodes(); }
    return 1 + leftNodes + rightNodes;
}
```