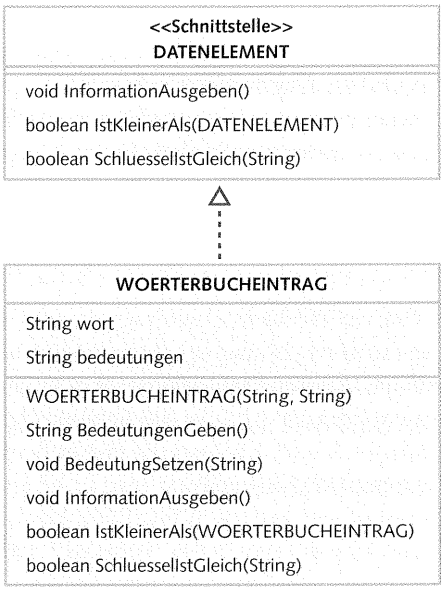
Von Listen und Bäumen

Die unsortierte Liste

Ausgangssituation ist eine unsortierte Liste:

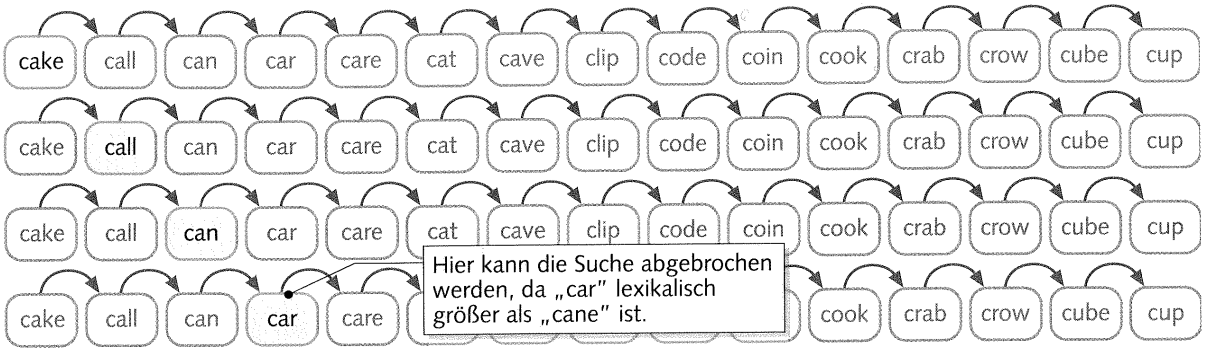


Diese Liste kann mit folgender Klassenstruktur abgebildet werden.

Der **Nachteil** ist dabei der erhöhte Suchaufwand, da im schlimmsten Fall die gesamte Liste durchlaufen werden muss.

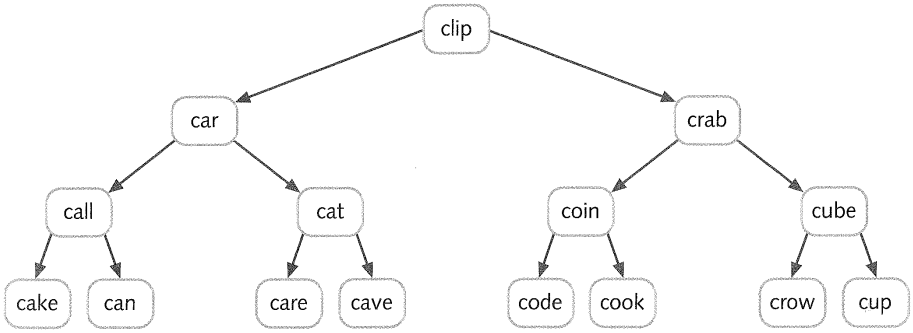
Dieser Nachteil kann beseitigt werden, wenn man die Liste sortiert aufbaut.

Die sortierte Liste



Dennoch werden Begriffe, welche am Ende der Liste stehen, erst nach einem vollständigen Durchlauf gefunden. Dies ließe sich nur vermeiden, wenn man ähnlich zum Telefonbuch die Liste weiter strukturiert.

Die Baumstruktur



Statt 15 Vergleiche werden für `cup` nur 4 Vergleiche benötigt.

Definition: Baum
Jeder Baum hat eine eindeutige Wurzel und besteht weiter aus inneren Knoten, die mehrere Kinder haben (2 bei Binärbaume) und inneren Knoten bzw. Blätter ohne Kinder. Wurzel, innere Knoten und Blätter sind mit Kanten verbunden.

Traversieren von Binärbaumen
Wie lassen sich aus Bäumen die Listen wieder herstellen? Dafür werden drei Methoden benötigt:

- Wert des Datenelementes des aktuellen Knotens ausgeben (w)
- Gehe zum linken Kindknoten (zur Wurzel des linken Teilbaumes), falls vorhanden (LK)
- Gehe zum rechten Kindknoten (zur Wurzel des rechten Teilbaumes), falls vorhanden (RK)

Daraus ergeben sich drei Arten der Traversierung:

Preorder w-LK-RK

```
preorder(Knoten k)
   Datenwert von k bearbeiten
   Falls k einen linken Kindknoten lk hat
       preorder(lk)
   Falls k einen rechten Kindknoten rk hat
     preorder(rk)
```

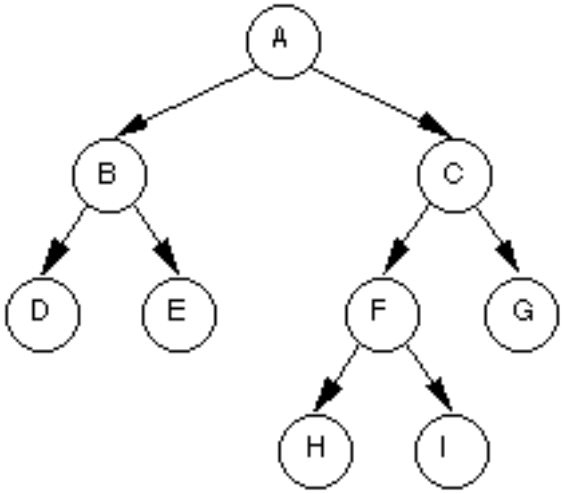
Inorder LK-w-RK

```
preorder(Knoten k)
   Falls k einen linken Kindknoten lk hat
       preorder(lk)
   Datenwert von k bearbeiten
   Falls k einen rechten Kindknoten rk hat
     preorder(rk)
```

Postorder LK-RK-w

```
preorder(Knoten k)
   Falls k einen linken Kindknoten lk hat
       preorder(lk)
   Falls k einen rechten Kindknoten rk hat
     preorder(rk)
   Datenwert von k bearbeiten
```

Beispiel

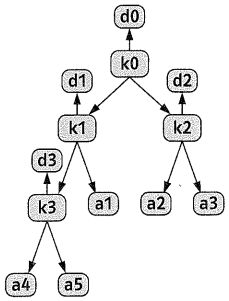


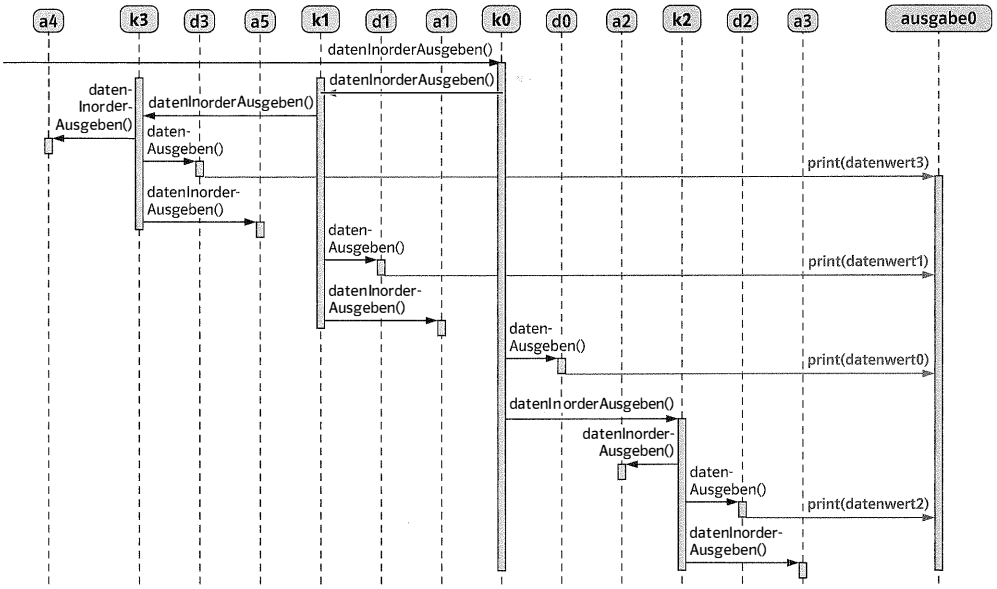
Inorder: D B E A H F I C G

Preorder: A B D E C F H I G

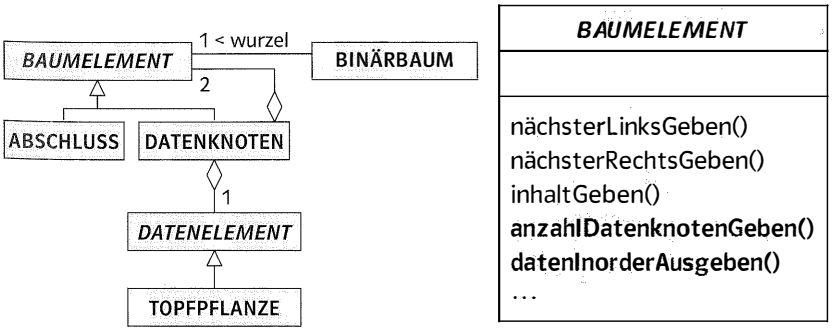
Postorder: D E B H I F G C A

Beispiel: Daten inorder() ausgeben





Implementierung
Mit Hilfe des Kompositum-Musters wird der Baum implementiert:



Aufgaben 1

Strukturieren Sie die Liste schrittweise in eine Binärbaum um. Welches Problem tritt auf?

Aufgabe 2

a) Strukturiere deine Mitschüler nach dem Nachnamen in einen Binärbaum.
b) Gib die Namen in Preorder, Inorder und Postorder aus.

Aufgaben 3
Nehmen Sie Stellung zu den Aussagen "Jeder Baum ist eine Liste" und "Jede Liste ist ein Baum".

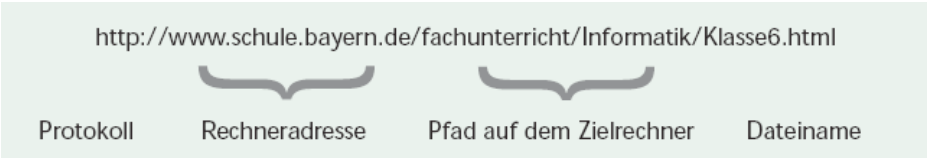
Aufgabe 4
Geben Sie das Klassendiagramm an, wenn die Kindknoten einen Verweis auf ihren Vater mit speichern sollen.

Aufgabe 5
Warum hat jeder Baum mit n Knoten genau n-1 Kanten.

Aufgabe 6

a) Welche Vorteile hat die Datenstruktur Baum im Vergleich zur Liste?

b) Nennen Sie die wichtigste Begriffe im Zusammenhang mit der Baumstruktur und erklären sie ihre Bedeutung am Beispiel von den Rechneradressen bei URL



c) Warum ist ein Baum eine rekursive Datenstruktur?