Bisher:

- Warteschlangen als Feld
- Einfach verkettete Listen
  (erst ohne, dann mit Abschluss;
  rekursive Methoden, Kompositum),
  insb. Warteschlange und Stapel
- Heterogene Listen
  (abstrakte Klasse *Datenelement*,
  saubere Trennung von Struktur und Inhalt)
- Sortierte Listen
  (sortiertes Einfügen, Entnehmen, Suchen)

Ist eine sortierte Liste zum Verwalten geordneter Daten günstig?

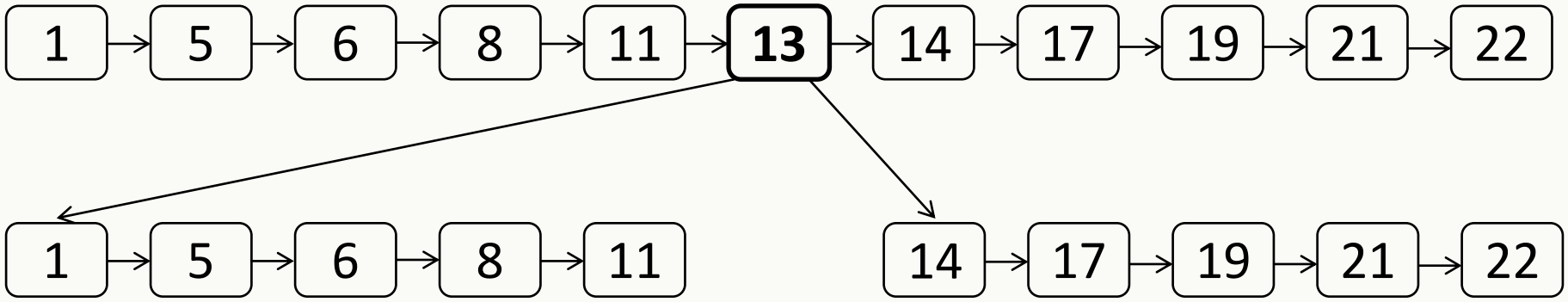
Ist ein bestimmter Wert in der Liste enthalten?

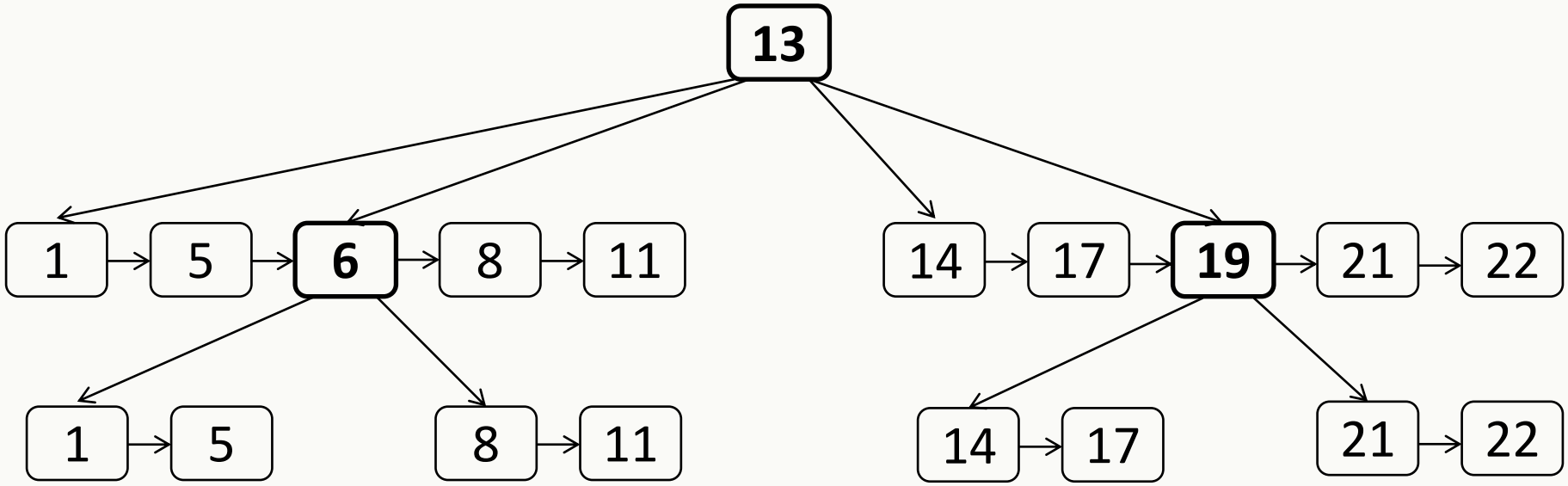
Die unsortierte Liste wird, falls der Wert nicht vorhanden ist, stets vollständig durchlaufen.

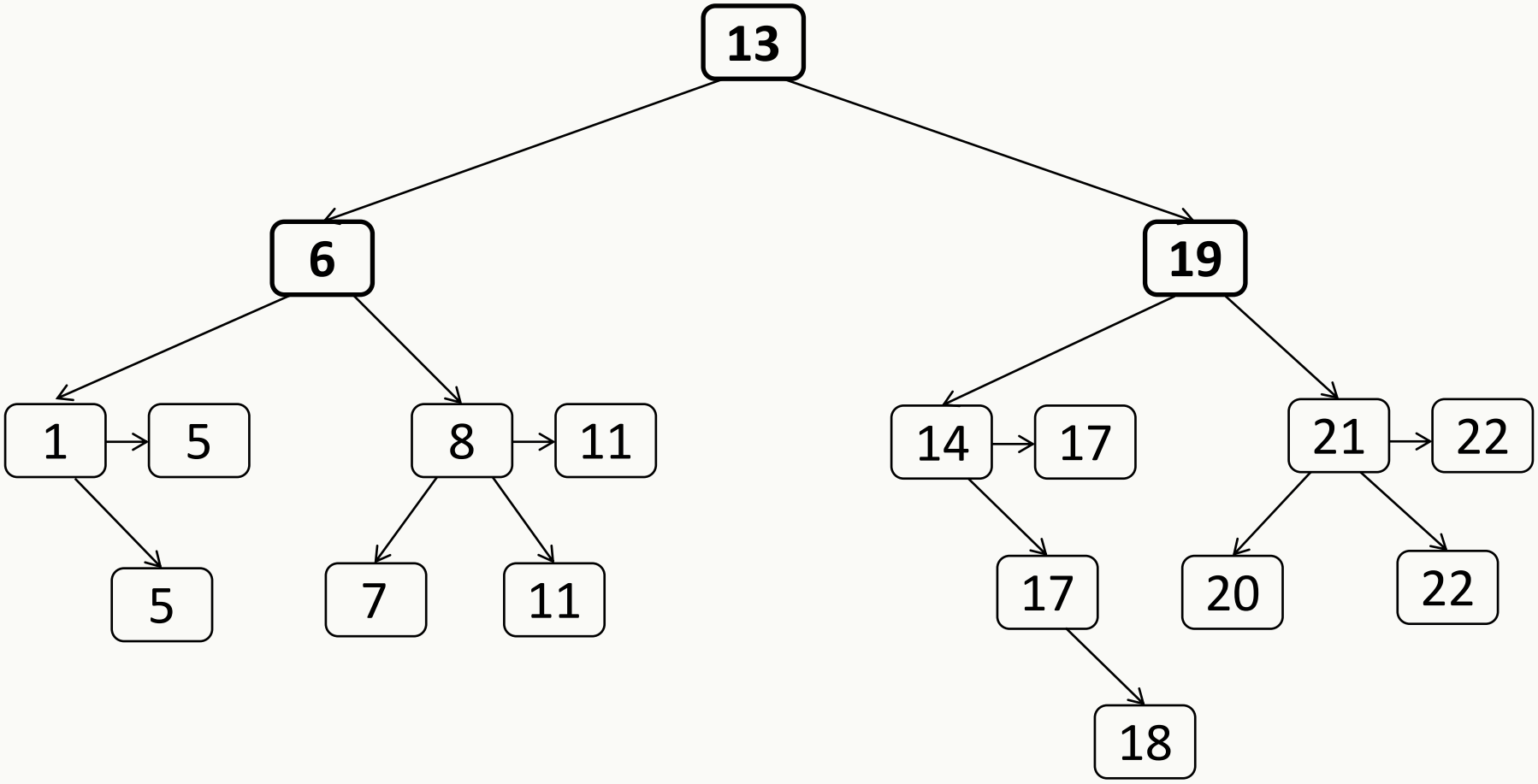
n Einträge ➔ n Vergleiche

Die sortierte Liste wird, falls der Wert nicht vorhanden ist, bis zu dem ersten Wert durchlaufen, der größer ist, als der Suchwert.

n Einträge ➔ (n+1)/2 Vergleiche


1
5
6
8
11
13
14
17
19
21
22
1
5
6
8
11
14
17
19
21
22


13
1
5
6
8
11
14
17
19
21
22
1
5
8
11
14
17
21
22



Wo muss 7 eingefügt werden?

Wo muss 18 eingefügt werden?

Wo muss 20 eingefügt werden?

II Bäume

1 Binärbäume

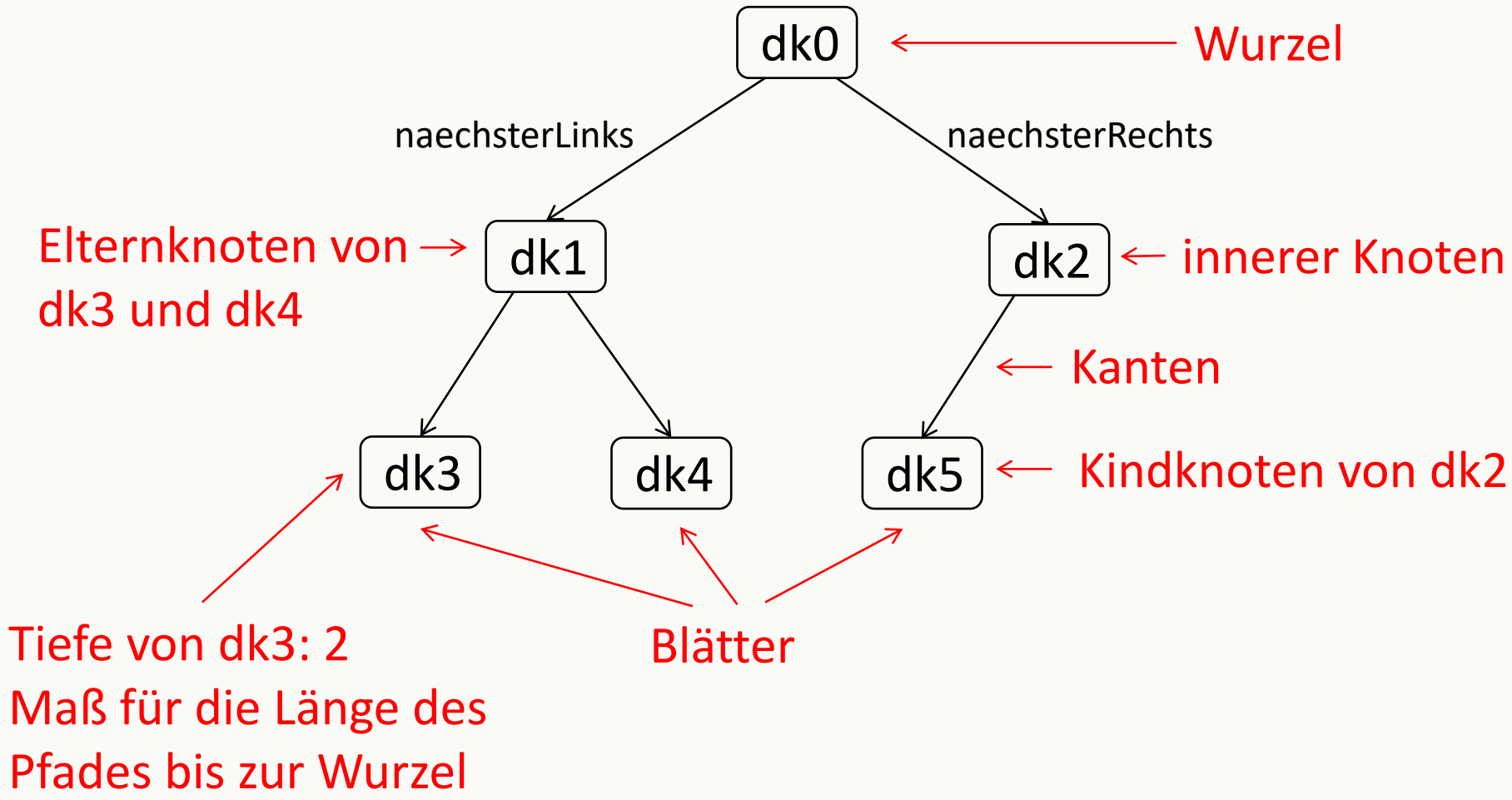
Eine Baumstruktur entsteht, wenn jedes Objekt mehrere Nachfolger haben kann und dabei gilt:

(1) Genau ein Objekt wird innerhalb der Baumstruktur **nicht** referenziert (Wurzel).

(2) Von der Wurzel kann jedes weitere Objekt der Baumstruktur auf genau einem Weg erreicht werden.

Binärbaum:
Jedes Objekt hat höchstens zwei Nachfolger.

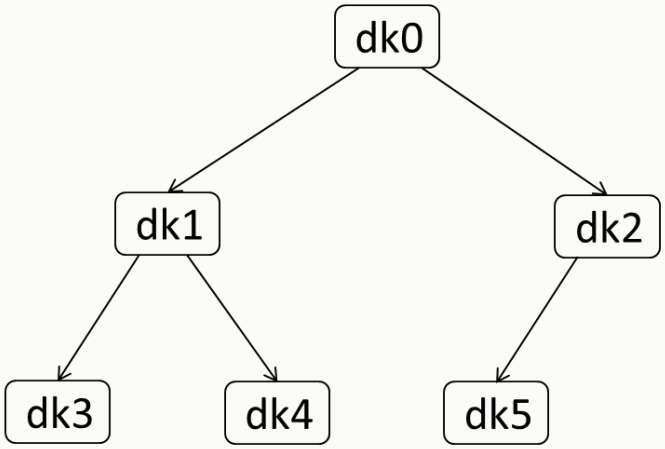
Objektstruktur



Methode: alleDatenAusgeben()

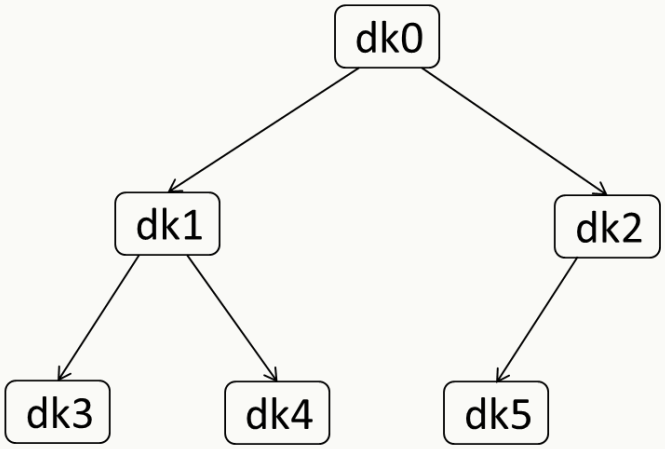
Traversieren eines Binärbaums:

Durchlaufen des Binärbaums, so dass jeder Datenknoten genau einmal erfasst wird.



In jedem Datenknoten müssen 3 Operationen durchgeführt werden:

- Ausgabe des Inhalts des aktuellen Knotens (w)
- Weiterreichen der Methode an den linken Teilbaum (LK)
- Weiterreichen der Methode an den rechten Teilbaum (RK)



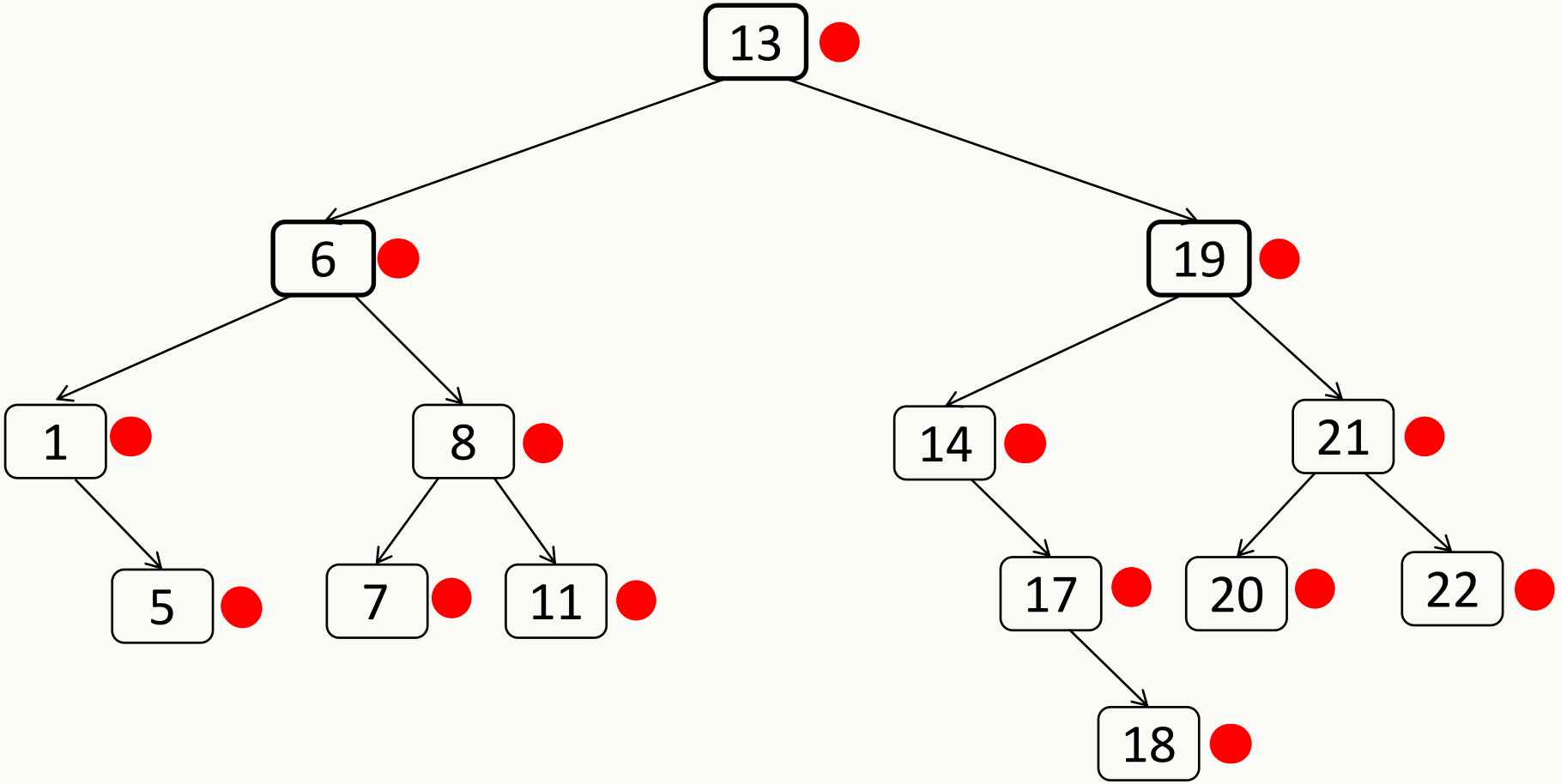
3 mögliche Strategien:

- Preorder: w – LK –RK

  dk0, dk1, dk3, dk4, dk2, dk5
- Inorder: LK – w –RK

  dk3, dk1, dk4, dk0, dk5, dk2
- Postorder: LK –RK – w

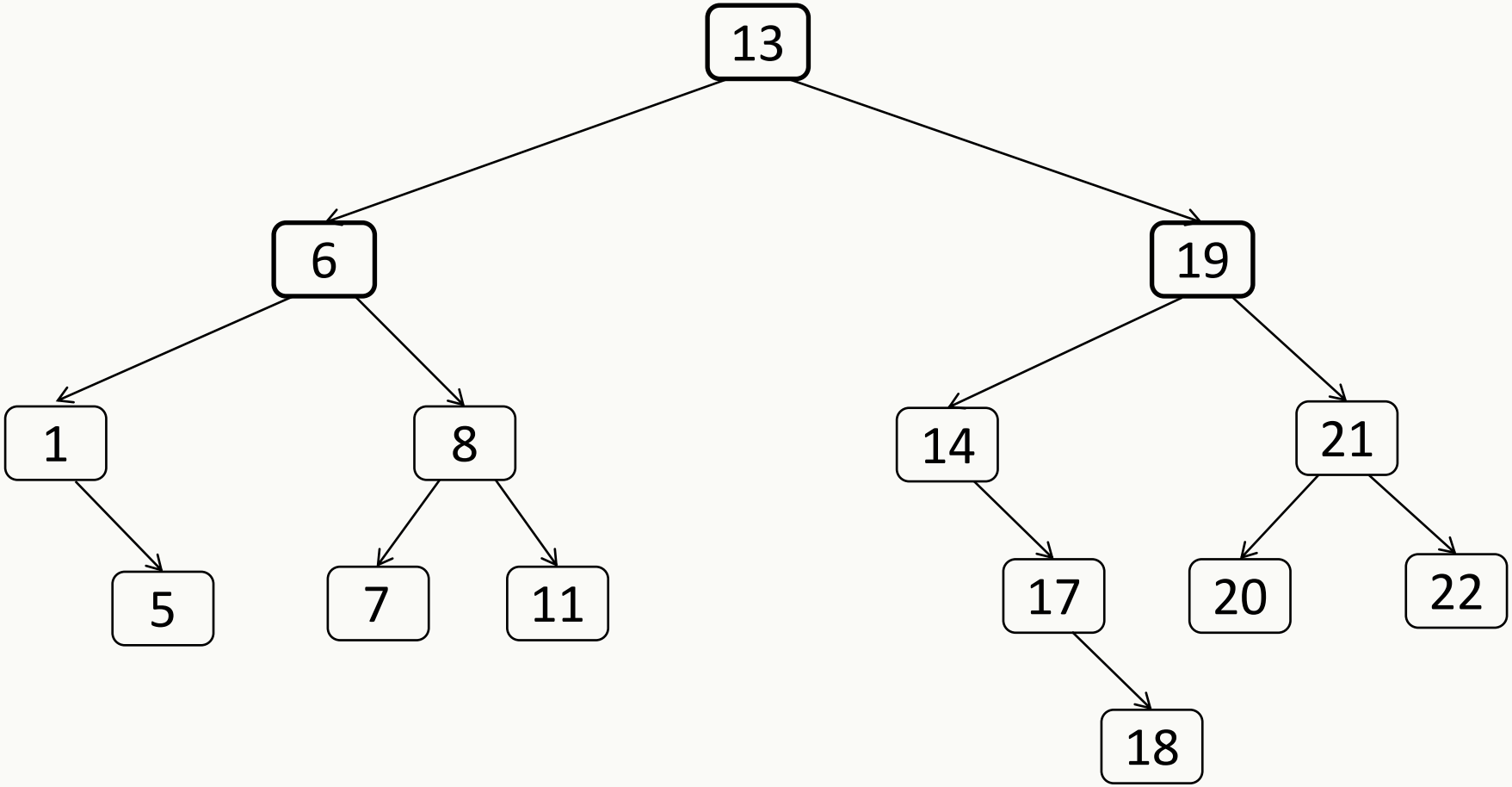
  dk3, dk4, dk1, dk5, dk2, dk0



Preorder: **13**, 6, 1, 5, 8, 7, 11, 19, 14, 17, 18, 21, 20, 22

Inorder:

Postorder:

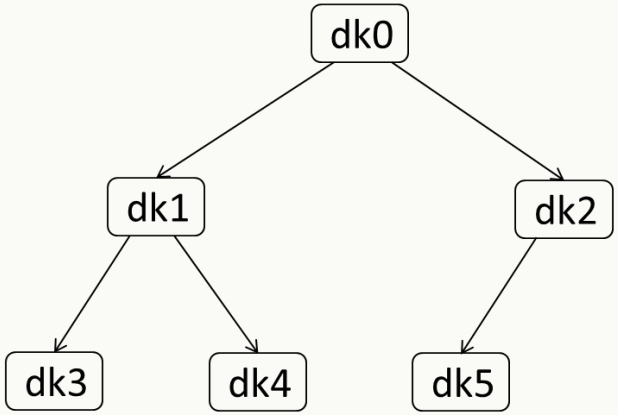


Preorder: **13**, 6, 1, 5, 8, 7, 11, 19, 14, 17, 18, 21, 20, 22

Inorder: 1, 5, 6, 7, 8, 11, **13**, 14, 17, 18, 19, 20, 21, 22

Postorder: 5, 1, 7, 11, 8, 6, 18, 17, 14, 20, 22, 21, 19, **13**

Implementierung von Binärbäumen



Woher weiß man beim Traversieren, dass man in einem Blatt angekommen ist?

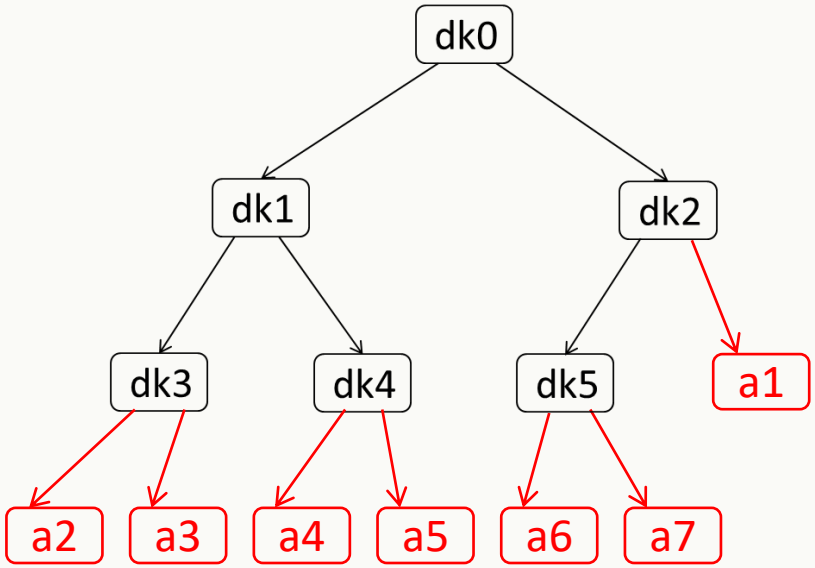
Ein Blatt hat keinen linken und keinen rechten Nachfolger.

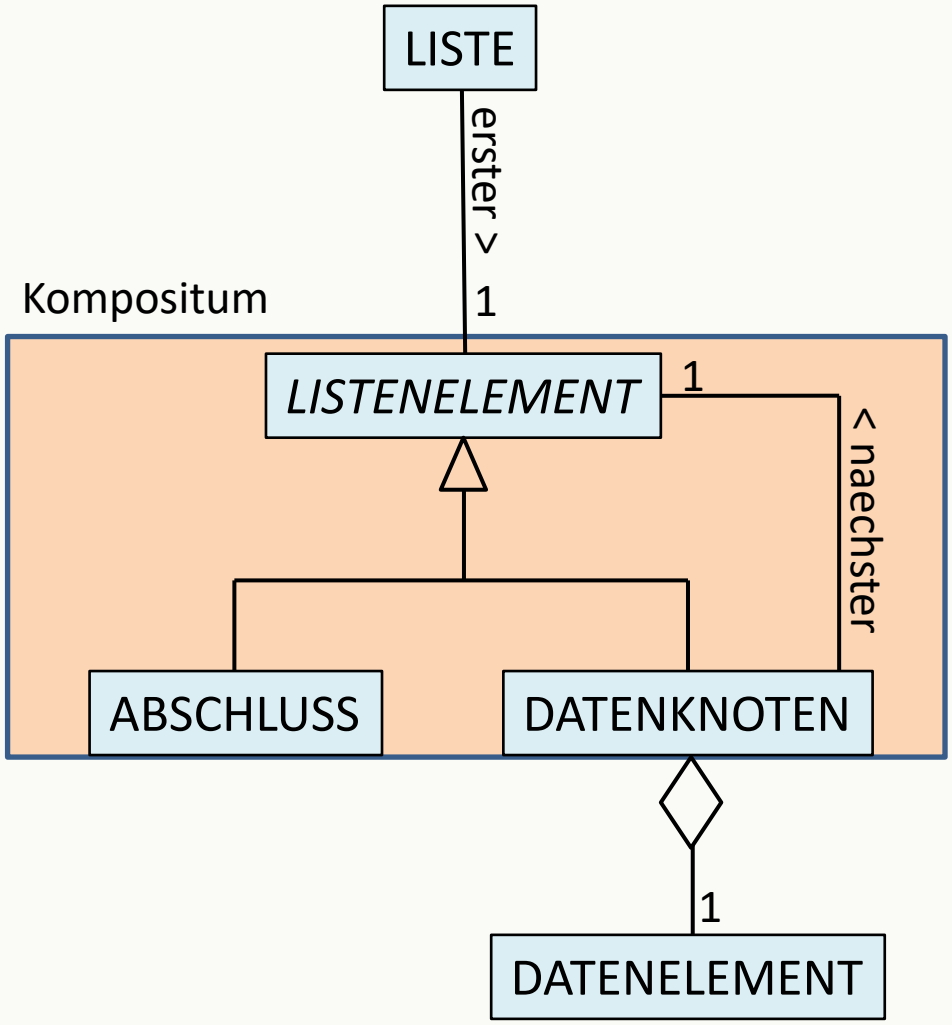
2 Möglichkeiten der Implementierung:

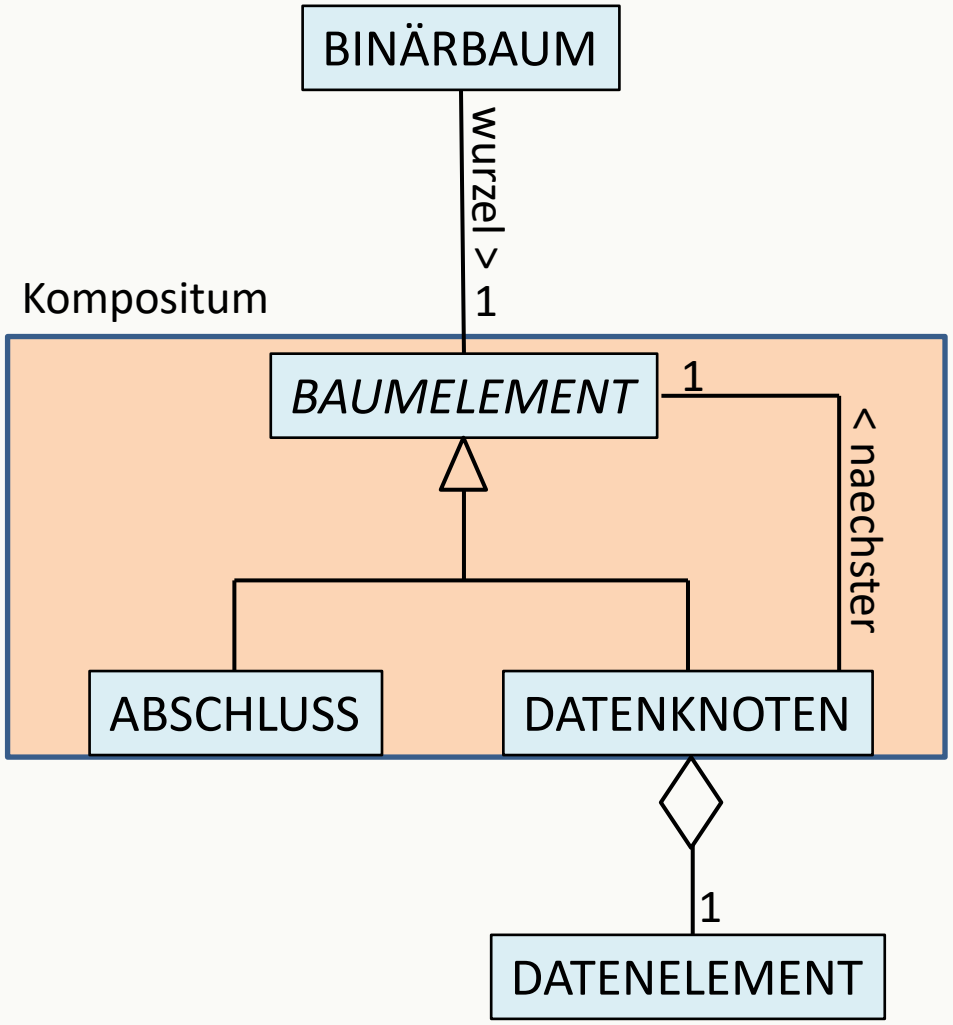
- Beide Referenzen sind null.

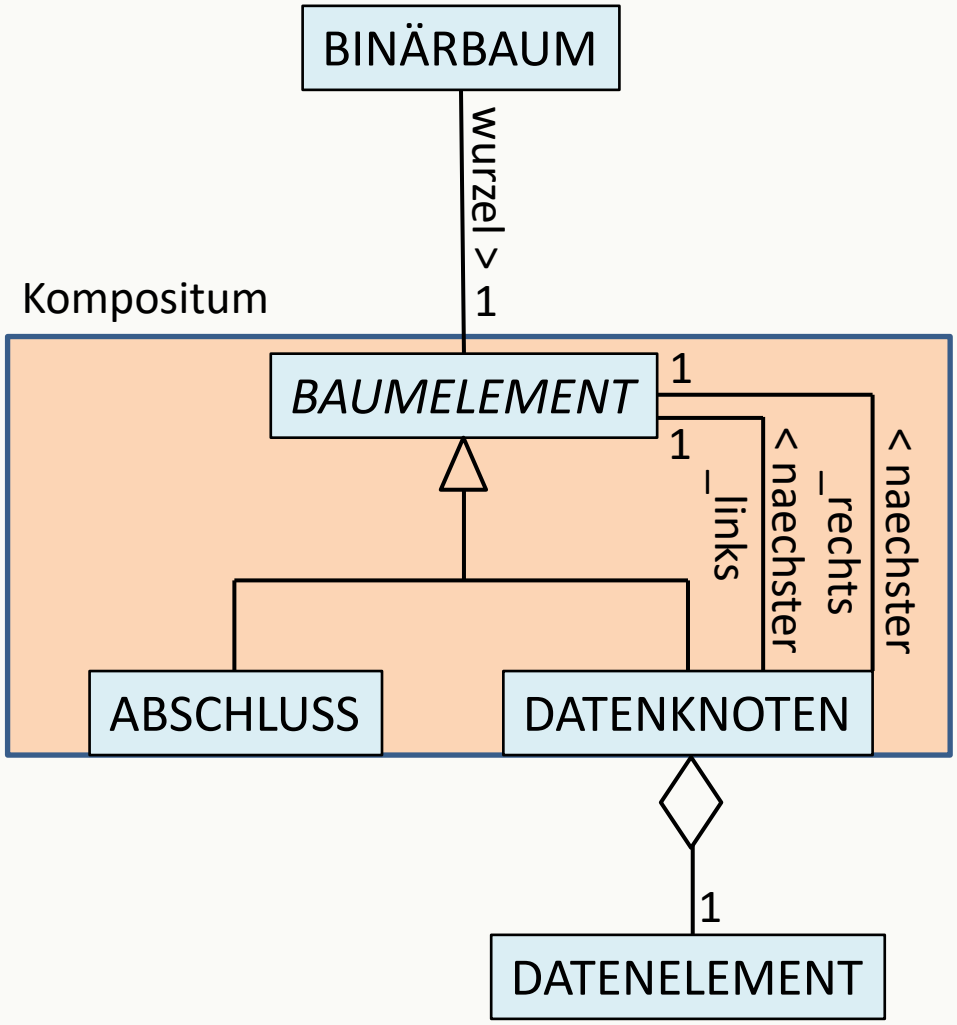
oder

- Das Blatt ist ein Abschluss.


dk0
dk1
dk2
dk3
dk4
dk5
a1
a2
a3
a4
a5
a6
a7


LISTE
erster >
1
Kompositum
LISTENELEMENT
1
< naechster
ABSCHLUSS
DATENKNOTEN
1
DATENELEMENT


BINÄRBAUM
wurzel
< 1
Kompositum
BAUMELEMENT
1
< naechster
ABSCHLUSS
DATENKNOTEN
1
DATENELEMENT


BINÄRBAUM
wurzel
< 1
Kompositum
BAUMELEMENT
1
1
_links
< naechster
_rechts
< naechster
ABSCHLUSS
DATENKNOTEN
1
DATENELEMENT

S. 59/1 Ahnengalerie (kein geordneter Binärbaum)

Einfüge-Methoden werden erst bei geordneten Bäumen näher behandelt, daher hier "Setzen per Hand" in einer Testklasse.

- Binaerbaum
  Attribut, Konstruktor, Methode wurzelSetzen
- Baumelement
  Gebe-Methoden
- Datenknoten
  Attribute, Konstruktor, Gebe-Methoden
- Abschluss
- Datenelement
  Attribute (vorname, nachname, alter), Konstruktor, datenGeben()
- Testablauf
  personenErzeugen, stammbaumSetzen (wurzel muss per Hand gesetzt werden),
  stammbaumAusgeben, personenZaehlen (erfordert entsprechende Ergänzungen in den anderen Klassen)