Die rekursive Datenstruktur Baum

Peter Brichzin

Oldenbourg

Hinweis

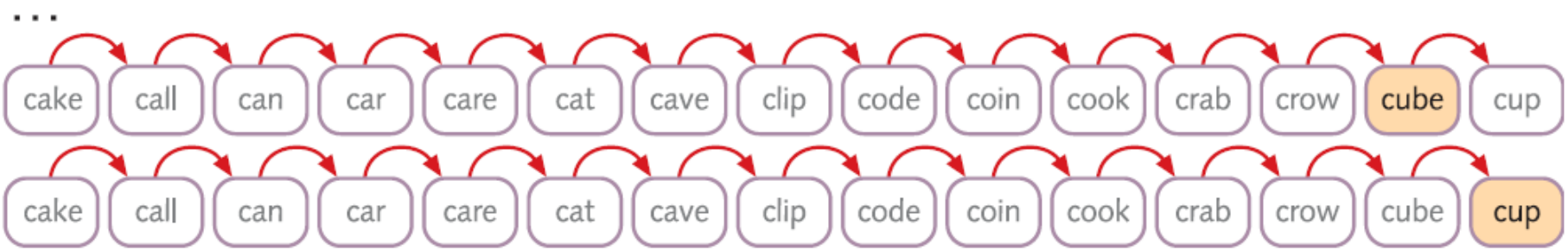
Die Abbildungen sind dem Buch Informatik Oberstufe 1 entnommen und unterliegen dem Copyright des Oldenbourg Schulbuchverlags!

II Die rekursive Datenstruktur Baum

6 Eine Liste umstrukturieren

Grenzen einer Datenstruktur aufzeigen

Suche in einem Wörterbuch:
Suche des Worts „cup“ in einem als Liste implementierten englisch-deutschem Wörterbuch.



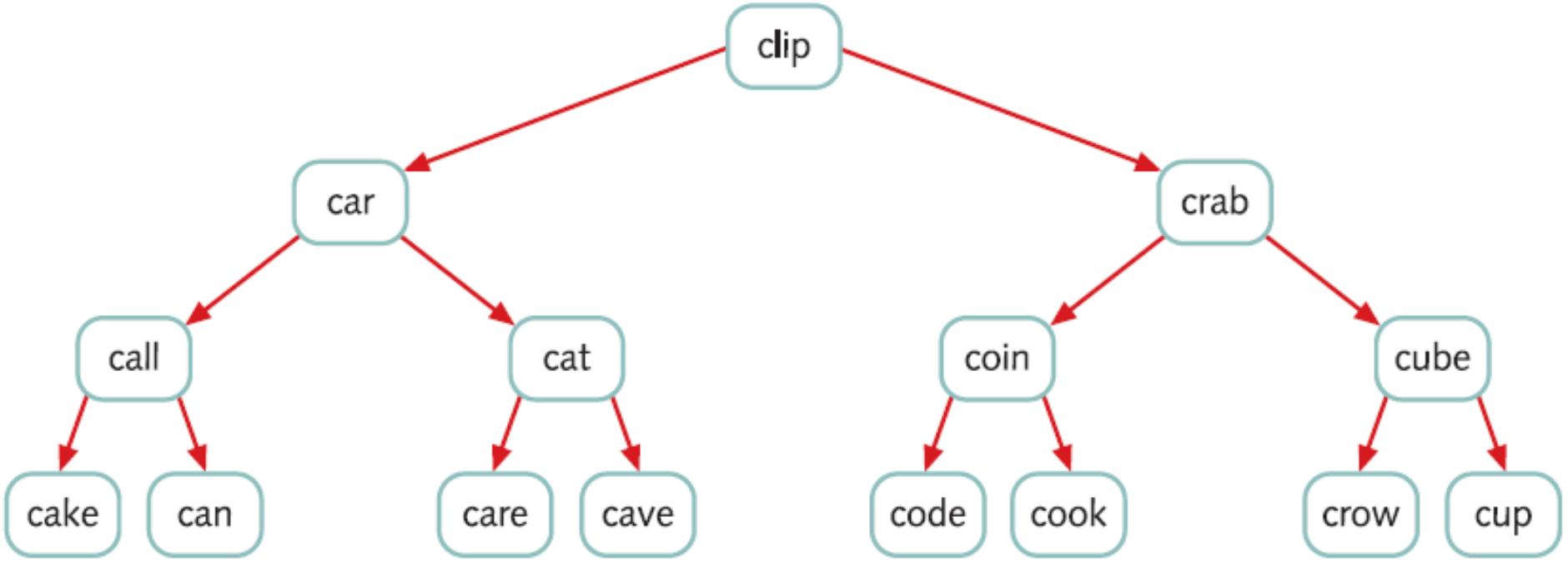
7 Suche nach dem Wort „cup“: 15 Vergleiche

II Die rekursive Datenstruktur Baum

6 Eine Liste umstrukturieren

Datenstruktur verändern

Suche in einem Wörterbuch:
Suche des Worts „cup“ in einem als Liste implementierten englisch-deutschem Wörterbuch.



10 Minimale Anzahl der Vergleiche durch eine Baumstruktur: vier Vergleiche bei der Suche nach dem Wort „cup“

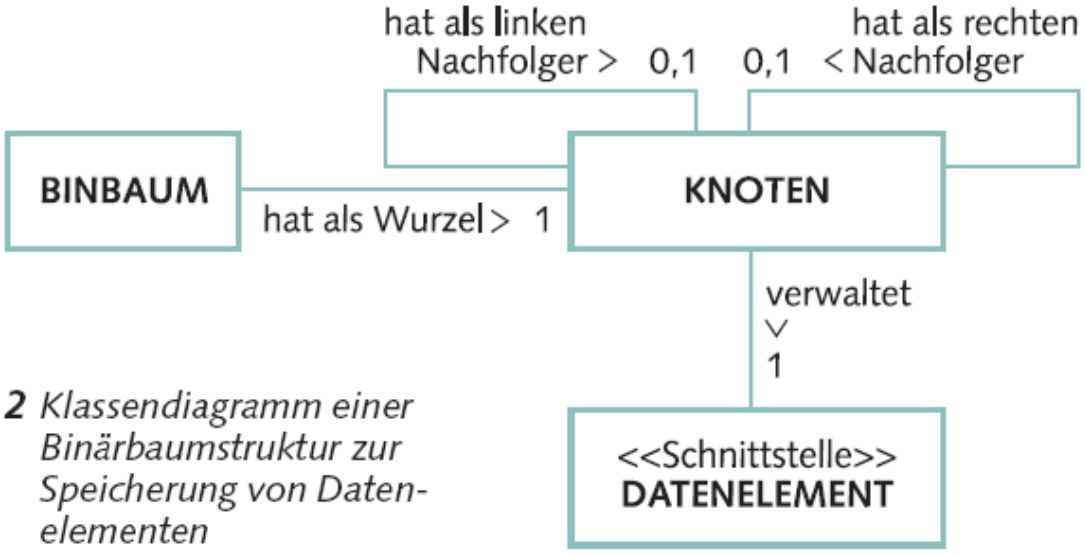
II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum

Spezialisieren

Geordneter Binärbaum als ein Sonderfall der Datenstruktur Baum

Implementierung zunächst nicht mit dem Entwurfsmuster Kompositum



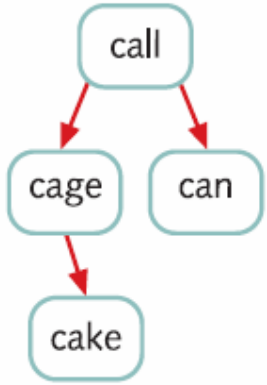
2 Klassendiagramm einer Binärbaumstruktur zur Speicherung von Datenelementen

II Die rekursive Datenstruktur Baum

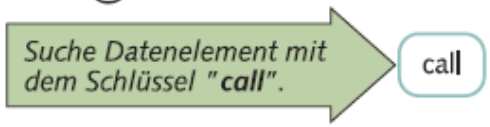
7 Geordneter Binärbaum

Systematische Analyse

Methode Suchen

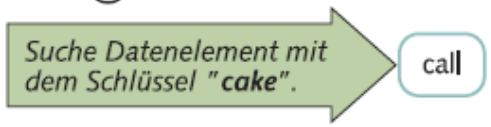


Fall ①



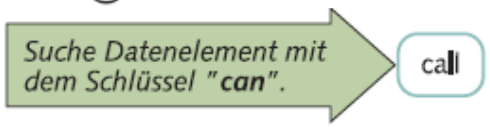
Das aufgerufene Knoten-Objekt vergleicht den Schlüssel "call" seines eigenen Datenelements mit dem eingegebenen Suchschlüssel "call". Es stellt die **Gleichheit** fest und gibt deshalb eine **Referenz auf sein Datenelement als Rückgabewert** aus. Der Methodenaufruf ist damit beendet.

Fall ②



Da der eigene Schlüssel "call" **größer** als der gesuchte Schlüssel "cake" ist, kann der Knoten die Suchanfrage nicht direkt beantworten und **gibt die Botschaft mit dem Suchauftrag an den linken Nachfolger** (Knoten "cage") **weiter**. Dieser liefert als Rückgabewert eine Referenz auf ein Datenelement, das der Knoten "call" seinerseits als Antwort zurückgeben kann.

Fall ③



Da der eigene Schlüssel „call" **kleiner** als der gesuchte Schlüssel "can" ist, kann der Knoten die Suchanfrage nicht direkt beantworten und **gibt die Botschaft mit dem Suchauftrag an den rechten Nachfolger** (Knoten "can") **weiter**. Dieser liefert als Rückgabewert eine Referenz auf ein Datenelement, das der Knoten "call" seinerseits als Antwort zurückgeben kann.

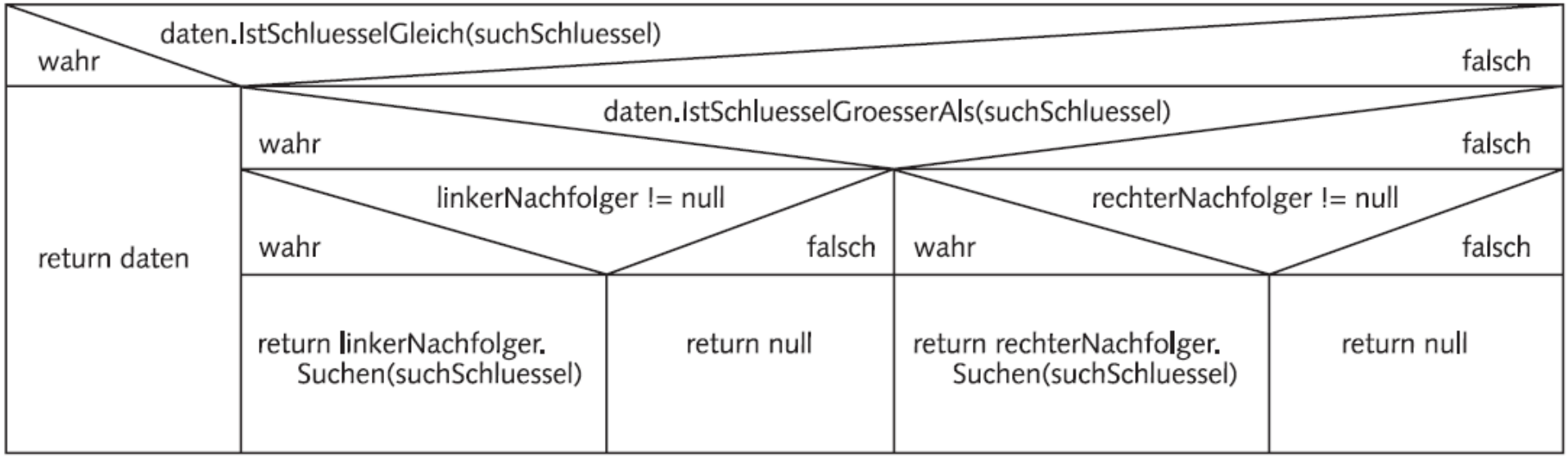
II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum

Unterschiedliche Darstellung der Algorithmen

Klasse KNOTEN
Methode DATENELEMENT Suchen(String suchSchluessel)

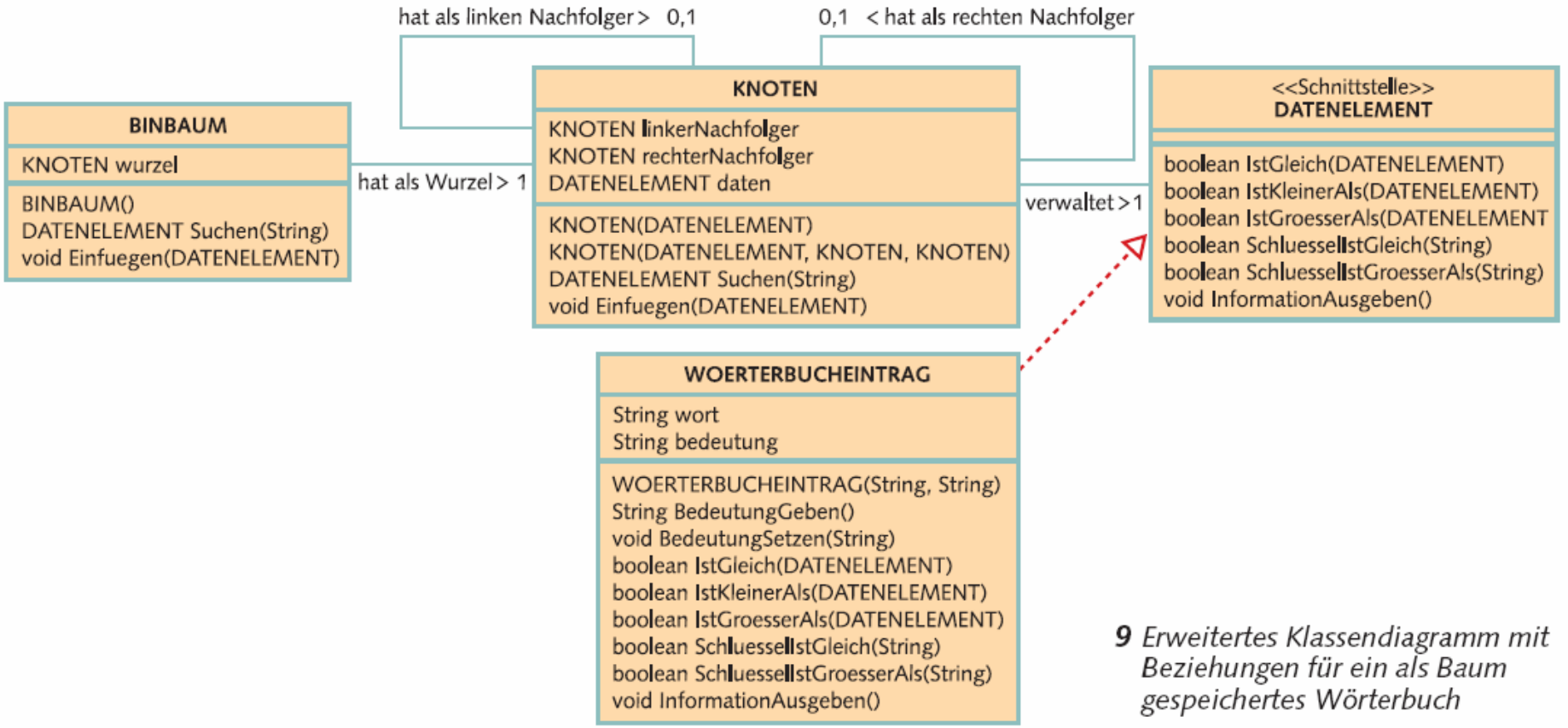
```
wenn (daten.IstSchluesselGleich(suchSchluessel))
dann
   return daten
sonst
   wenn (daten.IstSchluesselGroesserAls(suchSchluessel))
```




```
   endewenn
endewenn
```

II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum



9 *Erweitertes Klassendiagramm mit Beziehungen für ein als Baum gespeichertes Wörterbuch*

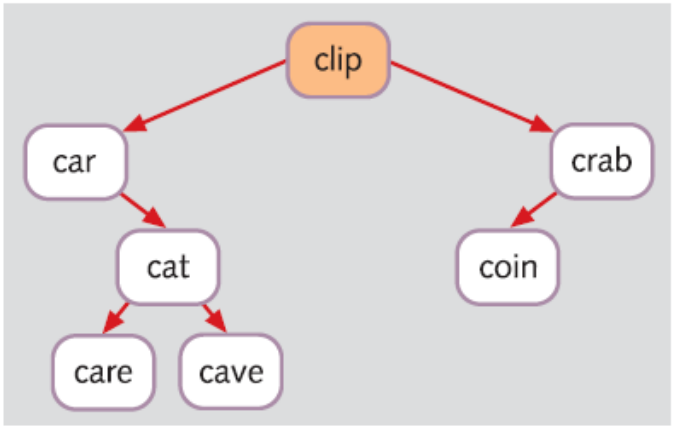
II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum

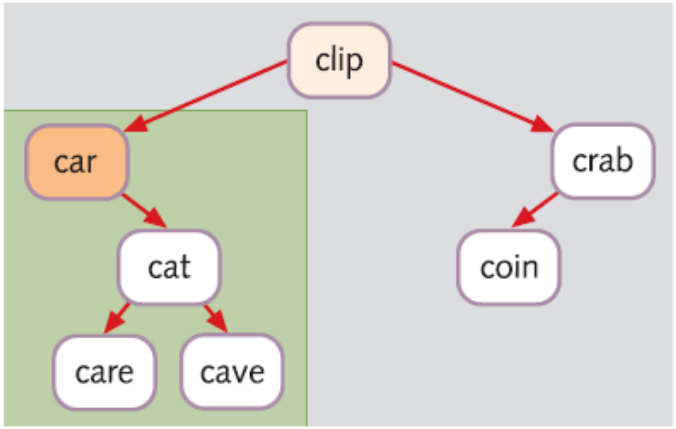
Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.Suchen("cat")

wurzel.Suchen("cat")
Schlüssel stimmt überein? nein
Ist eigener Schlüssel größer als der gesuchte Schlüssel? ja
Gibt es einen linken Nachfolger? ja
Frage linken Nachfolger.



wurzel.liNf.Suchen("cat")
Schlüssel stimmt überein? nein
Ist eigener Schlüssel größer als der gesuchte Schlüssel? nein
Gibt es einen rechten Nachfolger? ja
Frage rechten Nachfolger.



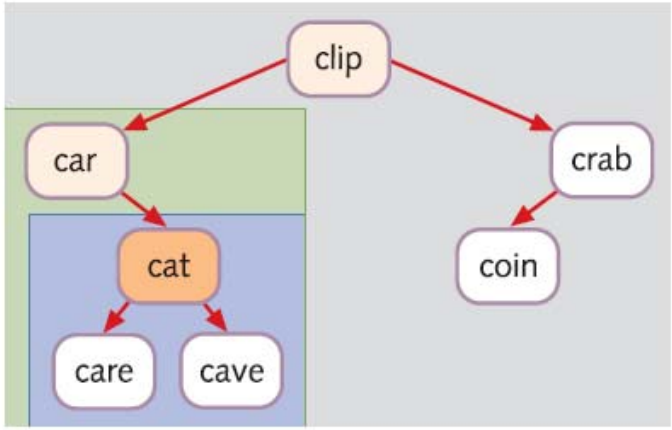
II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum

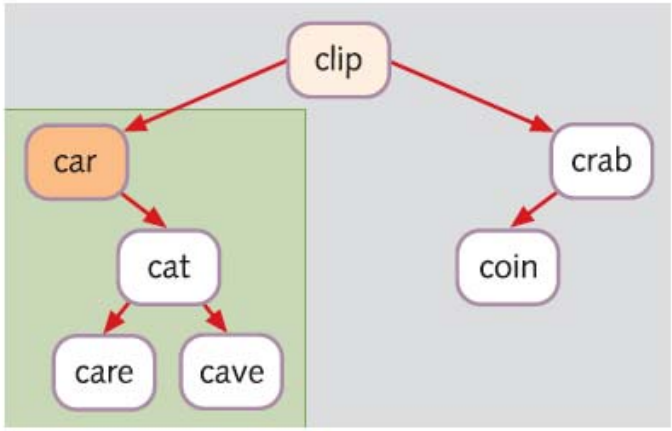
Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.Suchen("cat")

wurzel.liNf.reNf.Suchen("cat")
Schlüssel stimmt überein? ja
Gib Referenz auf Datenelement mit „cat | Katze“ zurück



Gib erhaltene Referenz auf Datenelement zurück.



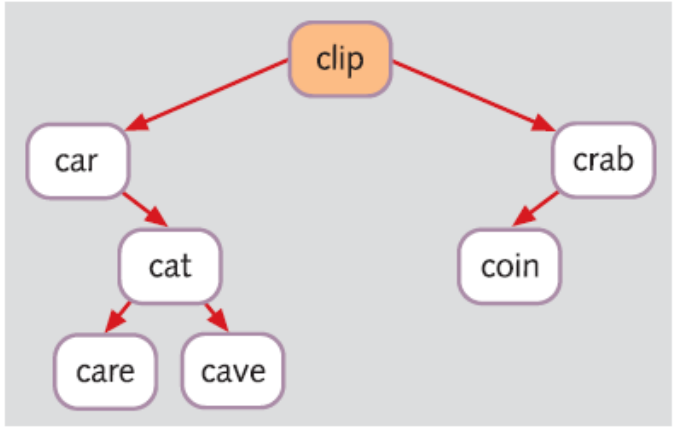
II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum

Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.Suchen("cat")

Gib erhaltene Referenz auf Datenelement zurück.

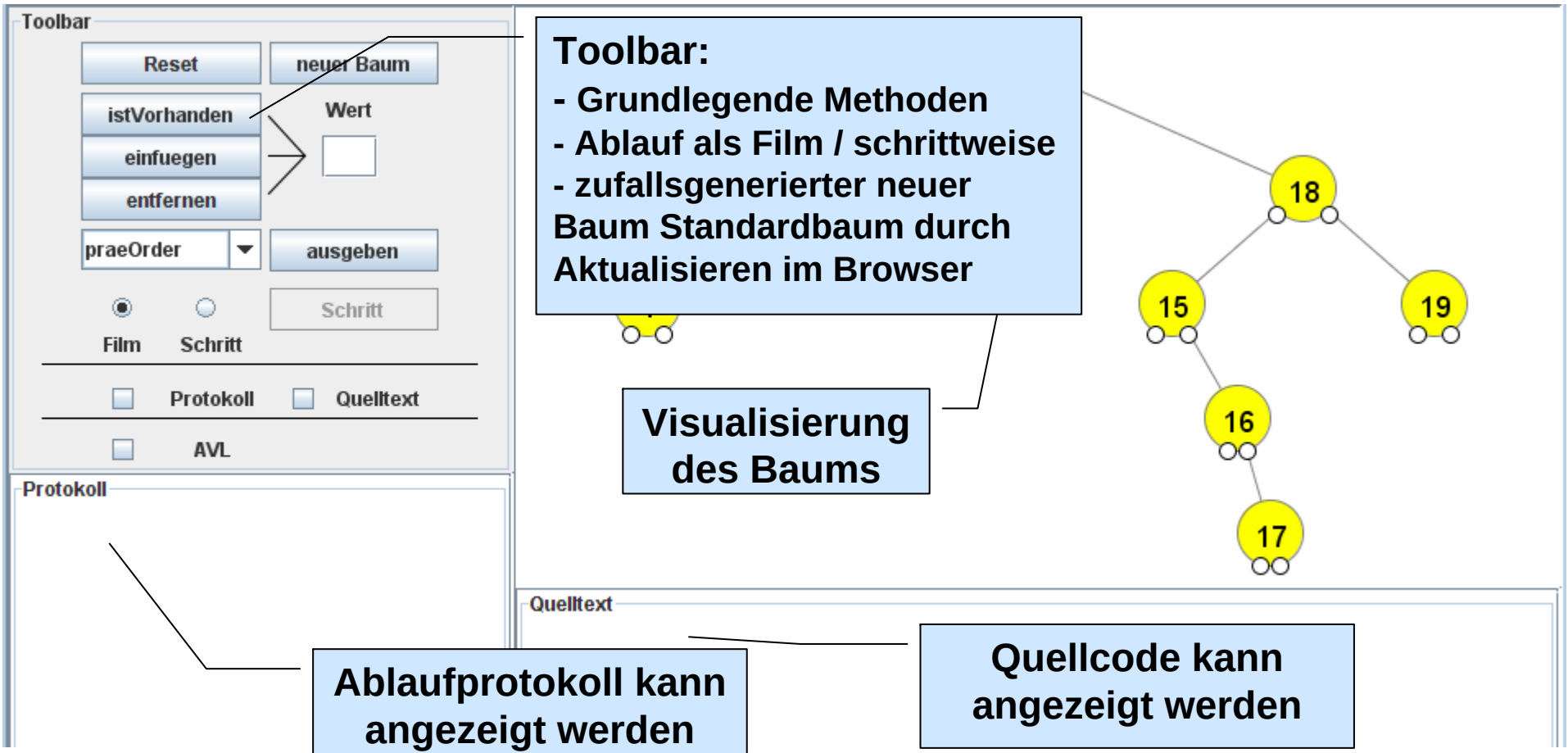


Oldenbourg

II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum

Visualisierung der Abläufe durch das BaumVisualisierungsTool

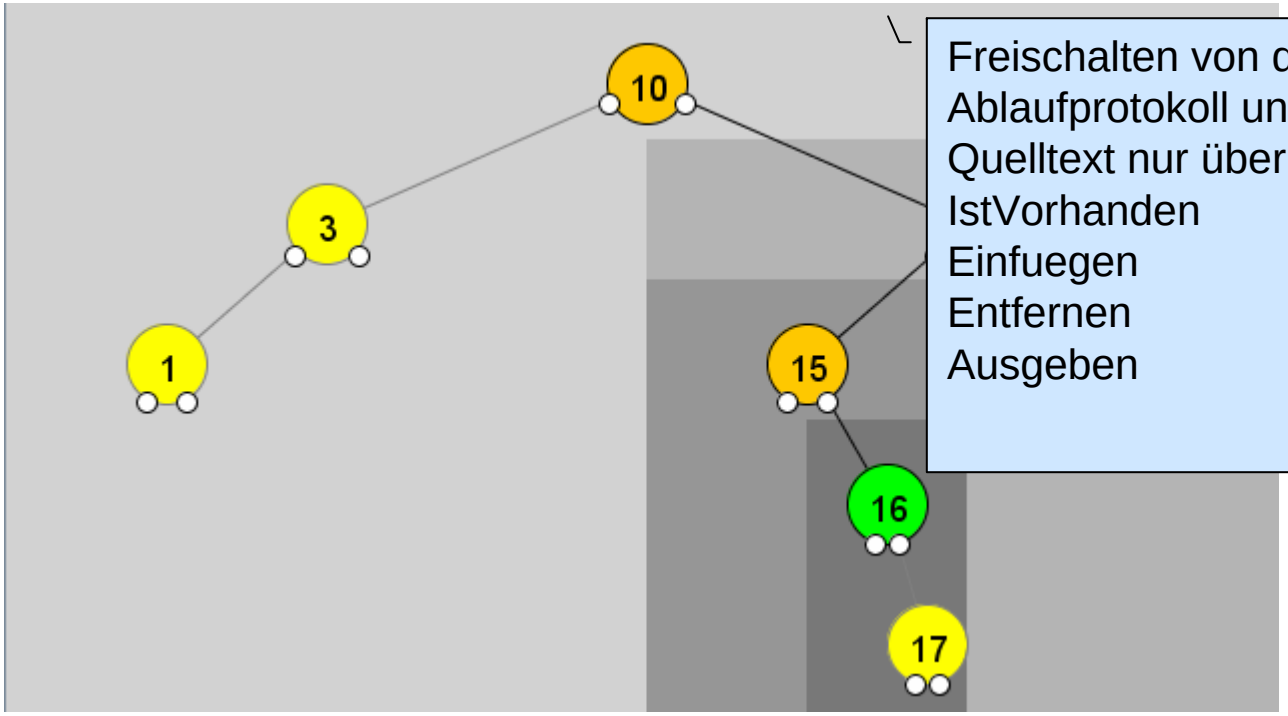


II Die rekursive Datenstruktur Baum

7 Geordneter Binärbaum

Visualisierung der Abläufe durch das BaumVisualisierungsTool

Methodenaufruf wurzel.IstVorhanden(16)



Freischalten von dem Ablaufprotokoll und dem Quelltext nur über Kennwörter:

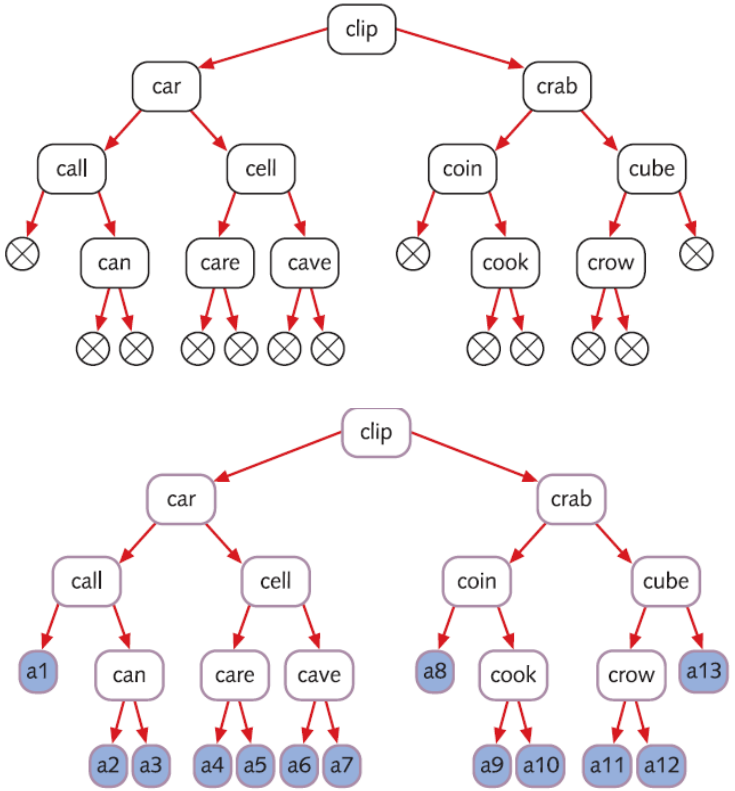
IstVorhanden	0011
Einfuegen	0121
Entfernen	1331
Ausgeben	1464

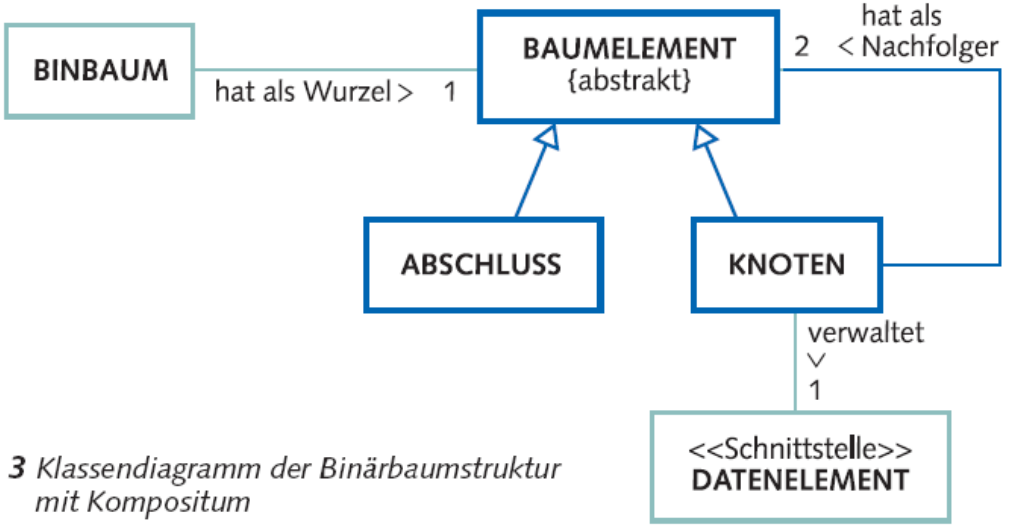
Entdeckendes Lernen ähnlich zu dynamischer Geometriesoftware im Mathematikunterricht

II Die rekursive Datenstruktur Baum

8 Baum in perfekter Komposition

Entwurfsmuster Kompositum erst nach dem grundlegenden Verständnis der Funktionsweise rekursiver Methoden bei Bäumen





3 Klassendiagramm der Binärbaumstruktur mit Kompositum

II Die rekursive Datenstruktur Baum

vorher (einfacher Binärbaum)

Klasse KNOTEN
Methode
DATENELEMENT Suchen(String suchSchluessel)

```
wenn (daten.IstSchluesselGleich(suchSchluessel))
dann
   return daten
sonst
   wenn (daten.
            IstSchluesselGroesserAls(suchSchluessel))
   dann
      wenn (linkerNachfolger != null)
      dann
         return linkerNachfolger.
                  Suchen(suchSchluessel)
      sonst
         return null
      endewenn
   sonst
      wenn (rechterNachfolger != null)
      dann
         return rechterNachfolger.
                  Suchen(suchSchluessel)
      sonst
         return null
      endewenn
   endewenn
endewenn
```

nachher (Binärbaum nach Entwurfsmuster Kompositum)

Klasse KNOTEN
Methode
DATENELEMENT Suchen(String suchSchluessel)

```
wenn (daten.IstSchluesselGleich(suchSchluessel))
dann
   return daten
sonst
   wenn (daten.
            IstSchluesselGroesserAls(suchSchluessel))
   dann
      return linkerNachfolger.
               Suchen(suchSchluessel)
   sonst
      return rechterNachfolger.
               Suchen(suchSchluessel)
   endewenn
endewenn
```

Klasse ABSCHLUSS
Methode
DATENELEMENT Suchen(String suchSchluessel)

```
return null
```

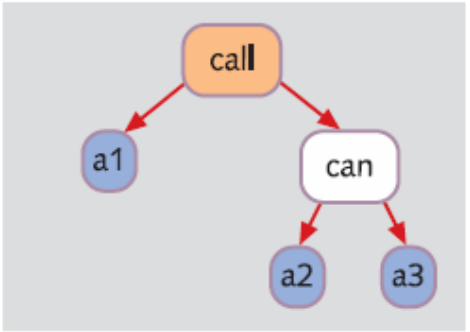
II Die rekursive Datenstruktur Baum

8 Baum in perfekter Komposition

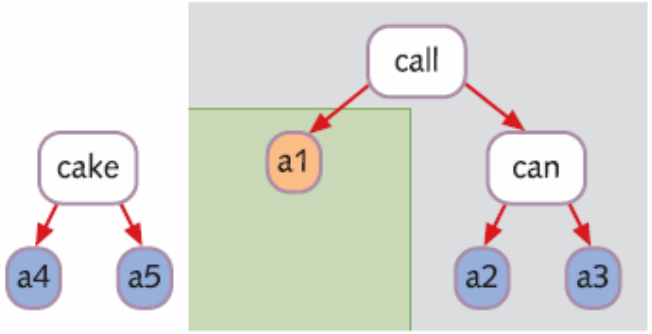
Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.Einfuegen("cake")

wurzel.Einfuegen("cake")
Daten sind gleich? nein
Sind eigene Daten größer als die neuen Daten? ja
Beauftrage linken Nachfolger mit dem Einfügen.



wurzel.liNf.Einfuegen("cake")
Erzeuge einen Knoten mit einem Datenelement "cake".



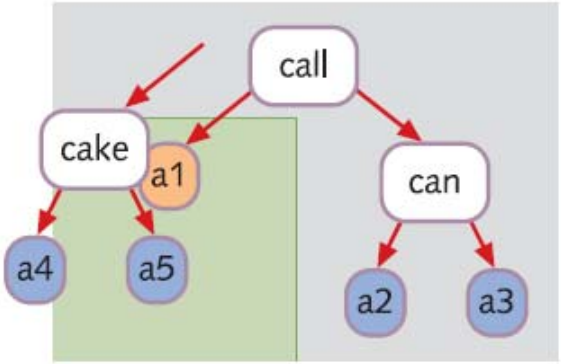
II Die rekursive Datenstruktur Baum

8 Baum in perfekter Komposition

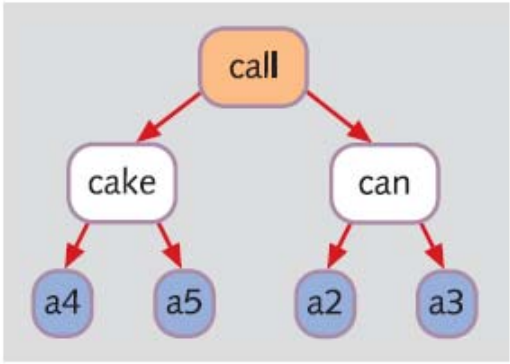
Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.Einfuegen("cake")

Gib eine Referenz auf den erzeugten Knoten zurück.



Setze den Rückgabewert als linken Nachfolger.
Gib eine Referenz auf dich selbst zurück.

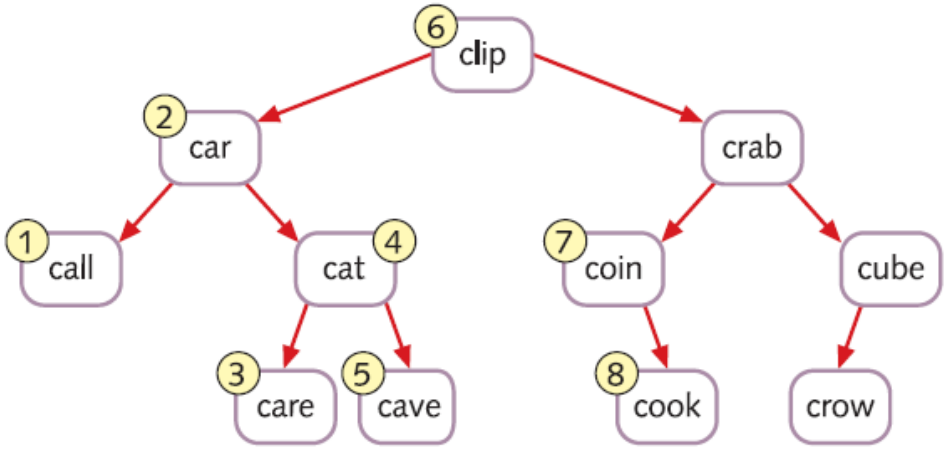


II Die rekursive Datenstruktur Baum

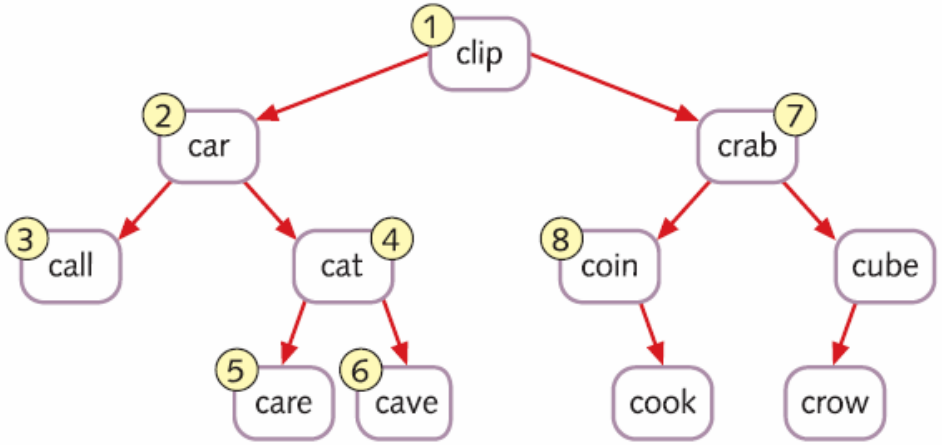
9 Bäume mit rekursiven Aufrufen durchlaufen

Unterschiedliche Anwendungsfälle

1 call
2 car
3 care
4 cat
5 cave
6 clip
7 coin
8 cook
…

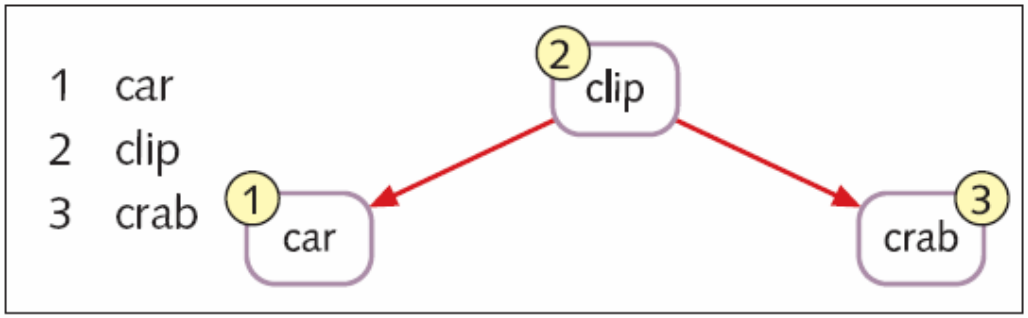


1 clip
2 car
3 call
4 cat
5 care
6 cave
7 crab
8 coin
…



II Die rekursive Datenstruktur Baum

9 Bäume mit rekursiven Aufrufen durchlaufen



Sortierte Ausgabe mithilfe eines Inorder-Durchlaufs …

Die rekursive Formulierung einer Ausgabe der Baumelemente mithilfe eines Inorder-Durchlaufs hat für Knoten folgende drei Bestandteile:

- Rufe die Methode *InorderAusgeben* des linken Nachfolgers auf.
- Gib die eigenen Daten aus.
- Rufe die Methode *InorderAusgeben* des rechten Nachfolgers auf.

II Die rekursive Datenstruktur Baum

9 Bäume mit rekursiven Aufrufen durchlaufen

Klasse BINBAUM
Methode void InorderAusgeben()

```
wurzel.InorderAusgeben()
```

Klasse KNOTEN
Methode void InorderAusgeben()

```
linkerNachfolger.InorderAusgeben()
daten.InformationAusgeben()
rechterNachfolger.InorderAusgeben()
```

Klasse ABSCHLUSS
Methode void InorderAusgeben()

```
// tue nichts
```

II Die rekursive Datenstruktur Baum

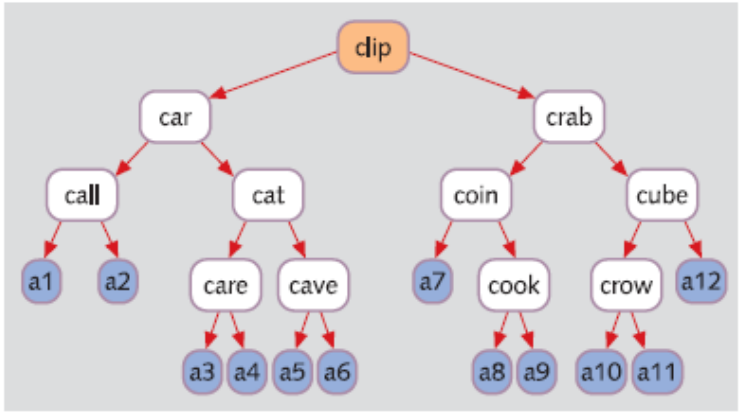
9 Bäume mit rekursiven Aufrufen durchlaufen

Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.InorderAusgeben()

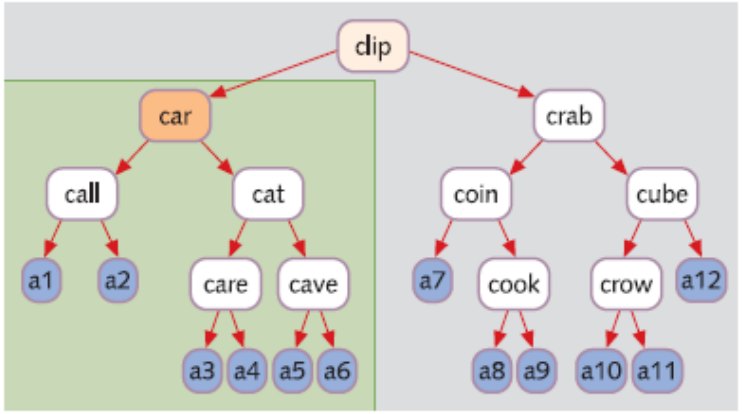
---

wurzel.InorderAusgeben()
Beauftrage linken Nachfolger mit Inorder-Ausgabe.



---

wurzel.liNf.InorderAusgeben()
Beauftrage linken Nachfolger mit Inorder-Ausgabe.



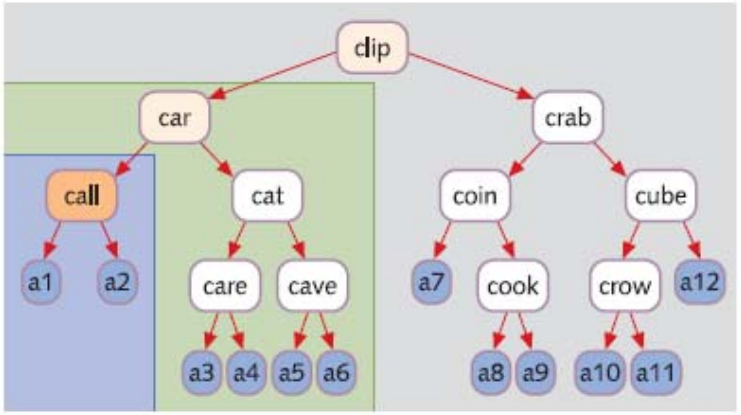
II Die rekursive Datenstruktur Baum

9 Bäume mit rekursiven Aufrufen durchlaufen

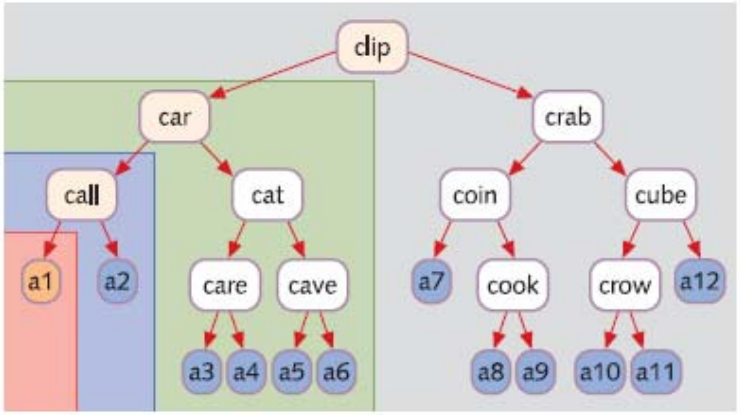
Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.InorderAusgeben()

wurzel.liNf.liNf.InorderAusgeben()
Beauftrage linken Nachfolger mit Inorder-Ausgabe.



wurzel.liNf.liNf.liNf InorderAusgeben()
//fertig



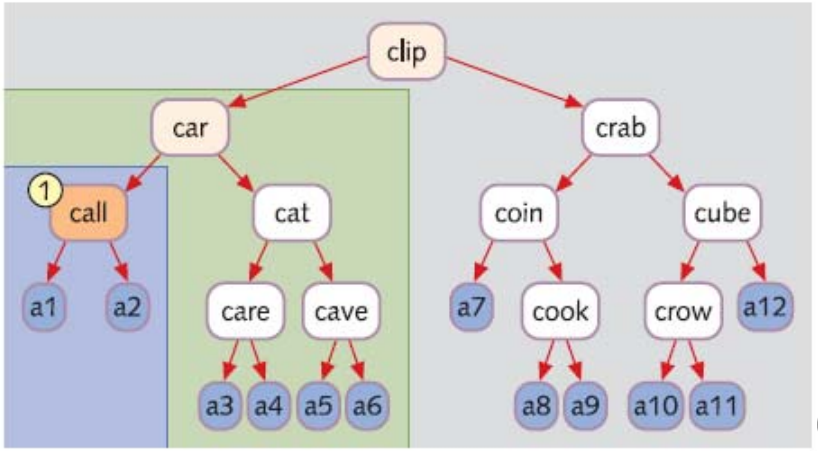
II Die rekursive Datenstruktur Baum

9 Bäume mit rekursiven Aufrufen durchlaufen

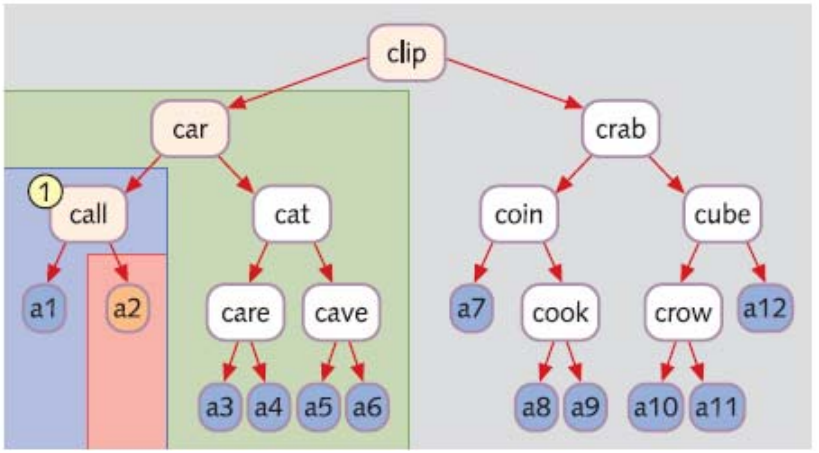
Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.InorderAusgeben()

Gib eigene Daten aus: **call**
Beauftrage rechten Nachfolger mit Inorder-Ausgabe.



wurzel.liNf.liNf.reNf.InorderAusgeben()
//fertig



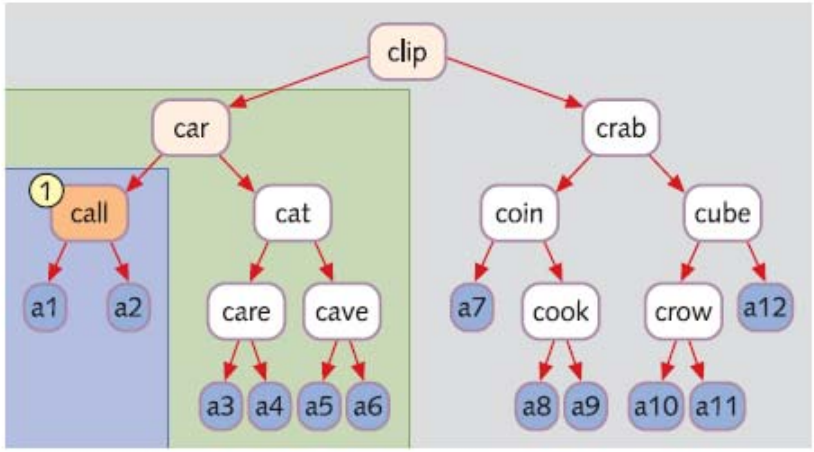
II Die rekursive Datenstruktur Baum

9 Bäume mit rekursiven Aufrufen durchlaufen

Visualisierung der Abläufe durch Ablaufprotokolle

Methodenaufruf wurzel.InorderAusgeben()

//fertig



Gib eigene Daten aus: **car**
Beauftrage rechten Nachfolger mit Inorder-Ausgabe.