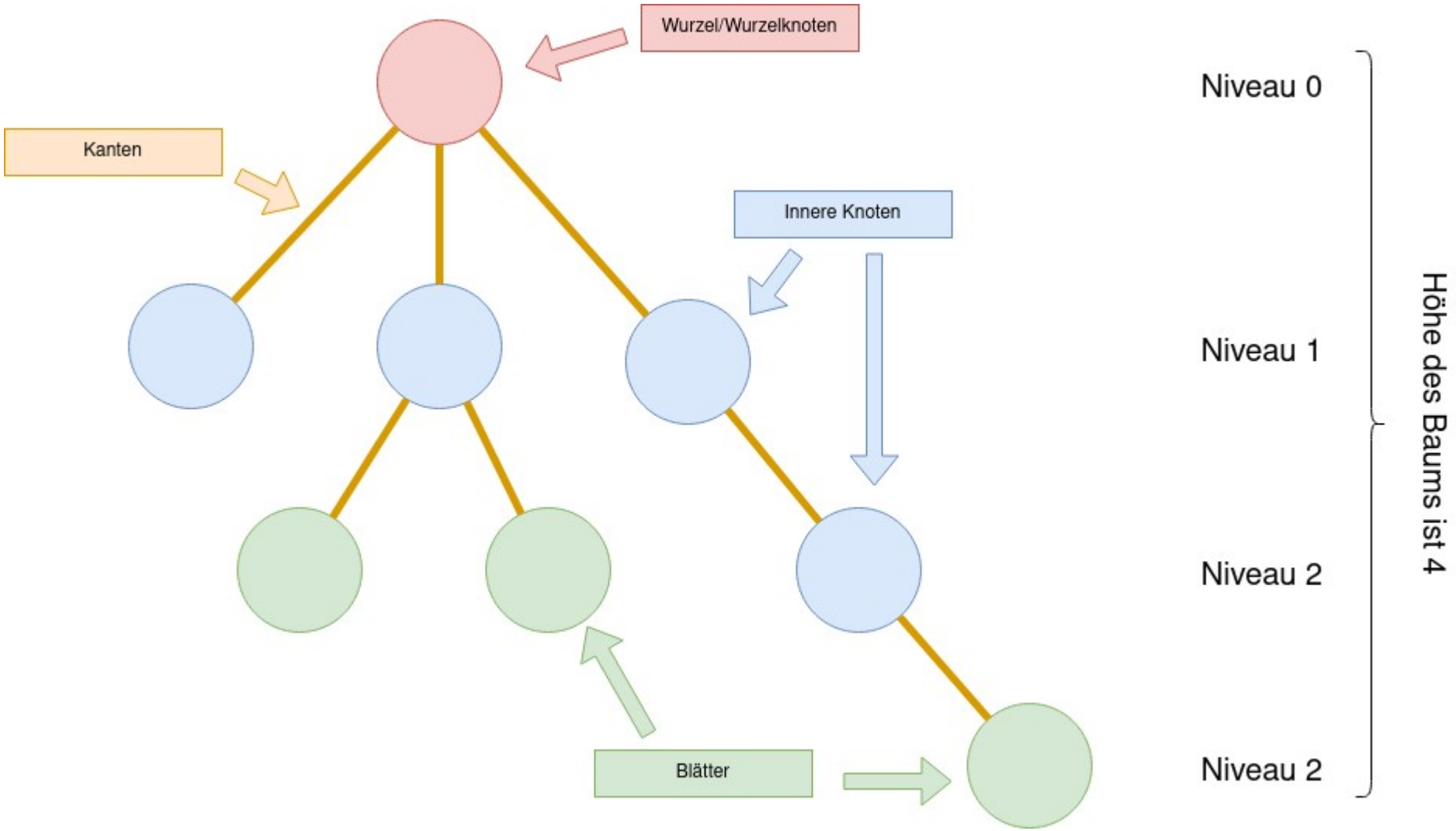
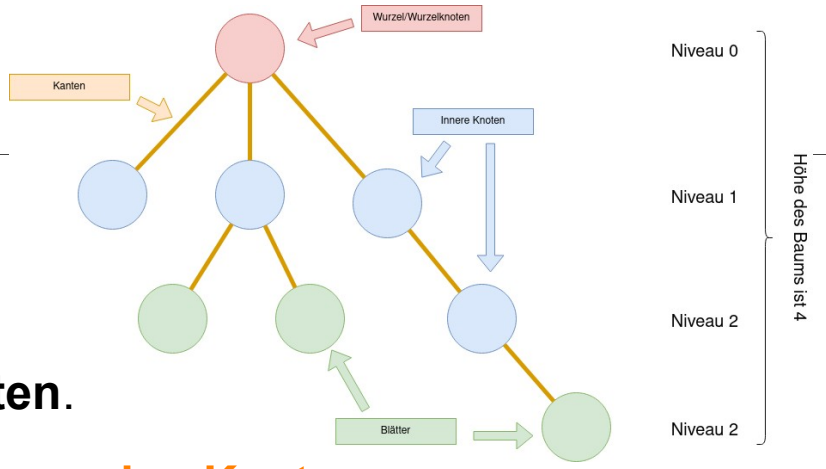
Bäume



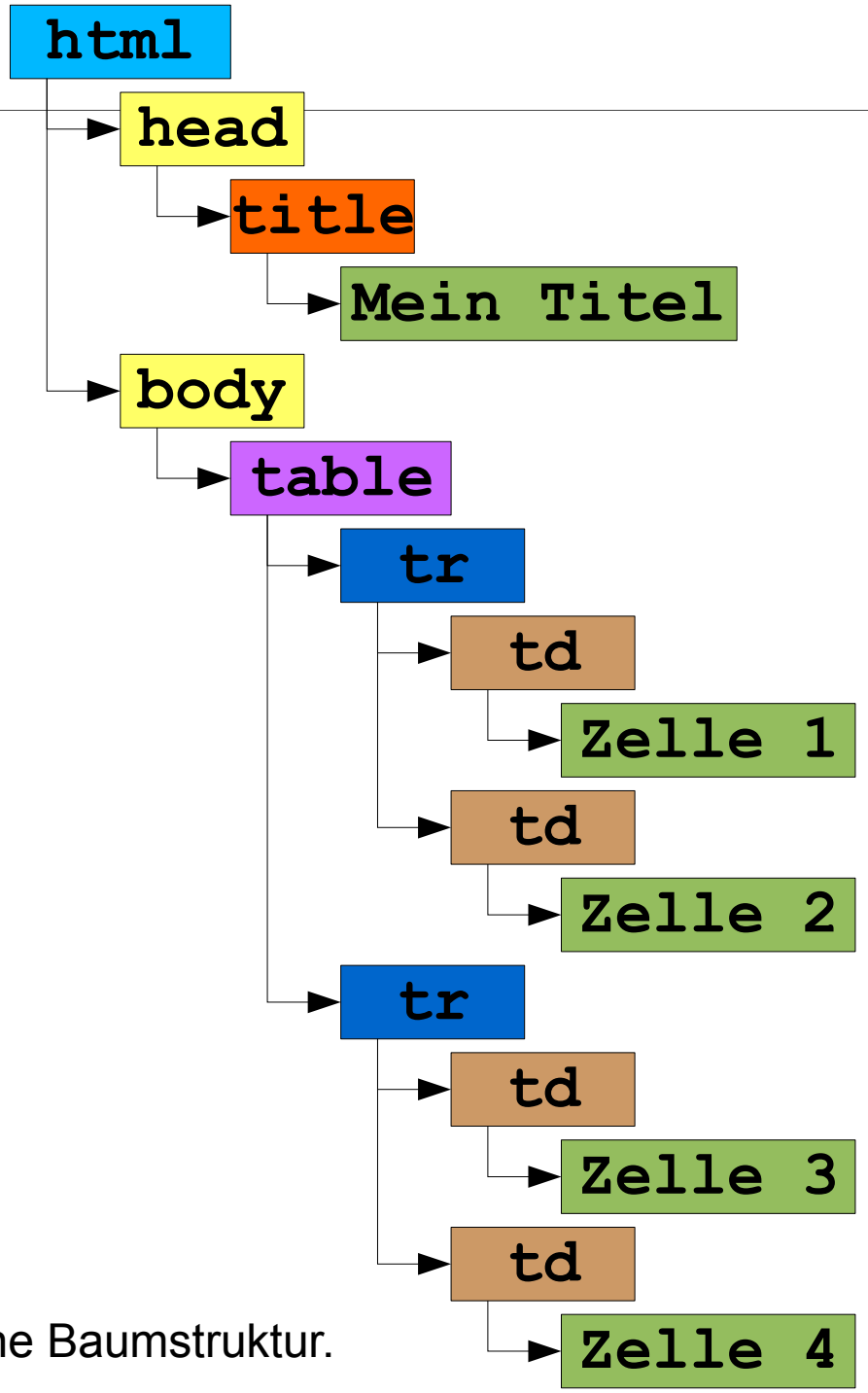
Bäume



- Der Knoten ohne Elternknoten ist der **Wurzelknoten**.
  Jeder (nicht leere) Baum hat **genau einen Wurzelknoten**.
- Jeder Knoten - außer dem Wurzelknoten - ist durch **genau eine Kante** mit seinem Elternknoten (Vaterknoten, Vorgänger) verbunden. Dieser Knoten wird häufig **Kind** oder **Nachfolger** des Elternknotens genannt.
- Ein Knoten der keine Kinderknoten hat heißt **Blatt**
- Knoten mit Eltern- und Kinderknoten heißen **innere Knoten** des Baums
- Ein **Pfad** ist eine Abfolge von Knoten, die durch Kanten miteinander verbunden sind. Bei einem Baum gibt es zwischen dem Wurzelknoten und jedem anderen Knoten **genau einen Pfad**. (Bäume sind „zyklenfrei“ und „zusammenhängend“).
- Das **Niveau** eines Knotens ist die Länge des Pfads vom Wurzelknoten zum betrachteten Knoten.
- Die **Höhe** des Baums ist die Anzahl der Knoten im längsten Pfad des Baums (oder gleichbedeutend: Das größte Niveau eines Knotens im Baum +1)
- In der Informatik zeichnet man Bäume aus praktischen Erwägungen von oben nach unten.

Ein Beispiel für Bäume: HTML

```
<html>
    <head>
        <title>Mein Titel</title>
    </head>
    <body>
        <table>
            <tr>
                <td>Zelle 1</td>
                <td>Zelle 2</td>
            </tr>
            <tr>
                <td>Zelle 3</td>
                <td>Zelle 4</td>
            </tr>
        </table>
    </body>
</html>
```



HTML-Seiten besitzen eine Baumstruktur.

Ein Beispiel für Bäume: HTML

```
<html>
   <head>
      <title>Mein Titel</title>
   </head>
   <body>
      <table>
         <tr>
            <td>Zelle 1</td>
            <td>Zelle 2</td>
         </tr>
         <tr>
            <td>Zelle 3</td>
            <td>Zelle 4</td>
         </tr>
      </table>
   </body>
</html>
```

Mit JavaScript z.B. kann man diesen Baum (den „DOM-Tree“) durchlaufen und auf einzelne Elemente zugreifen, z.B. mit

```
var html  = document.firstChild;
var body  = html.childNodes[1];
var table = body.childNodes[0];
var row   = table.childNodes[1];
var cell  = row.childNodes[0];
```

Jetzt kann man die Zelle mit dem Inhalt „Zelle 3“ z.B. verändern.

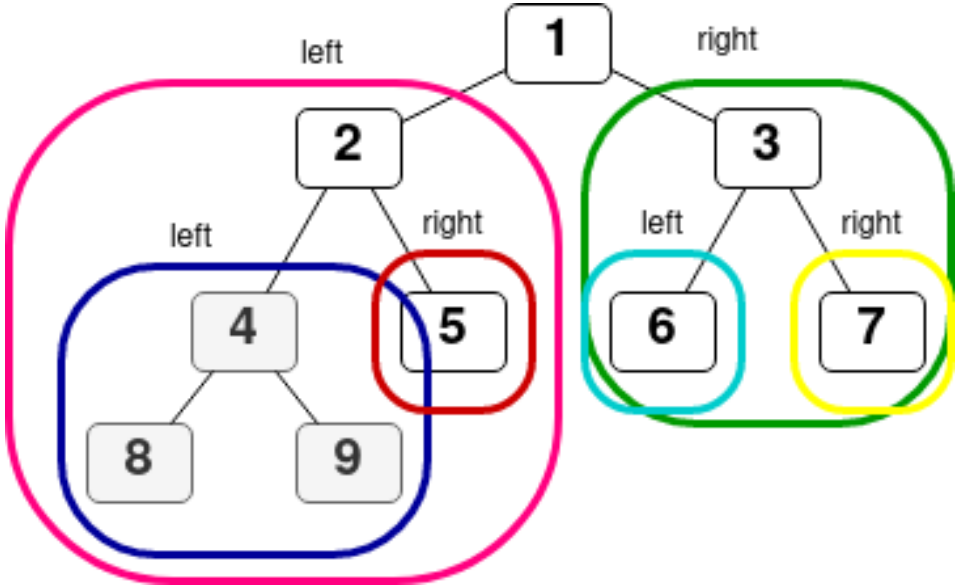
Binärbäume

Bei einem **Binärbaum** hat jeder **Knoten** maximal zwei Kindknoten.

Man kann diese als **linken (left)** und **rechten (right)** Kindknoten bezeichnen.
Wenn der rechte und der linke Kindknoten eindeutig unterschieden werden können spricht man von einem **geordneten Baum.**

Jeder Knoten kann ein Datenelement enthalten.

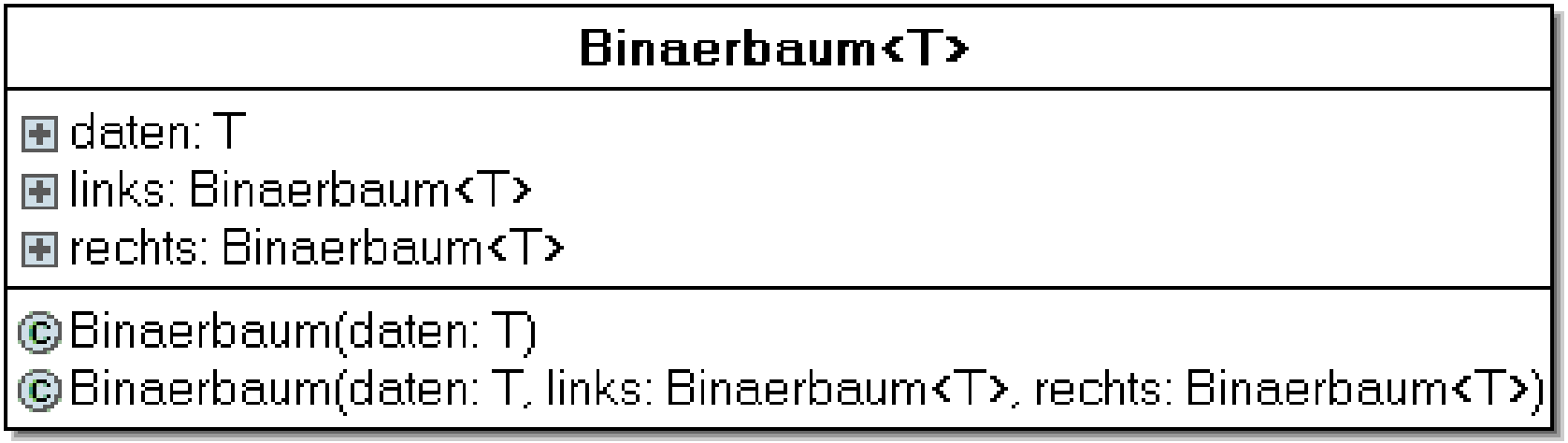
Ein (Binär)Baum ist eine rekursive Datenstruktur.
left und **right** zeigen jeweils auf einen weiteren Baum →



Die **Blätter** des Baums zeichnen sich dadurch aus, dass **left** und **right** auf null zeigen, also nicht auf weitere „Unterbäume“ verweisen.

Binärbäume

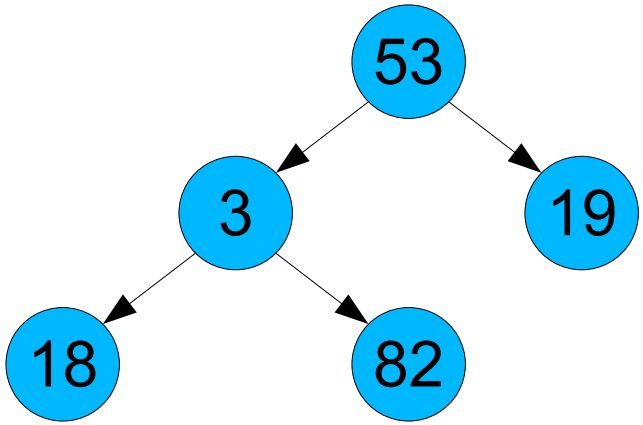
Eine Implementation als Java-Klasse könnte so aussehen:



Durch Überladen des Konstruktors wird Polymorphie des Konstruktors erzwungen um verschiedene Knoten erzeugen zu können.

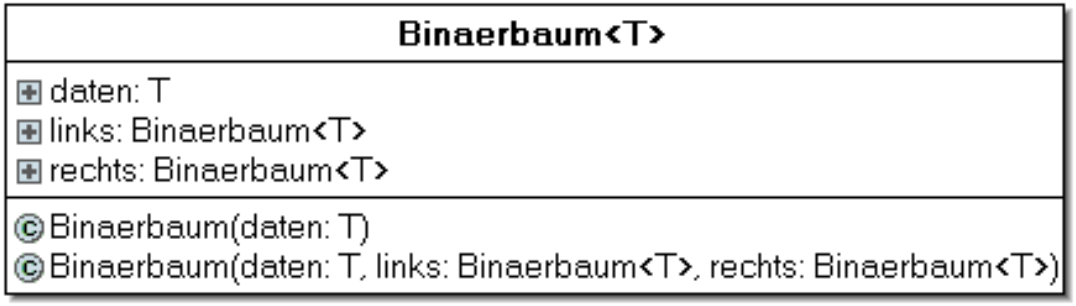
Binärbäume: Konstruktion

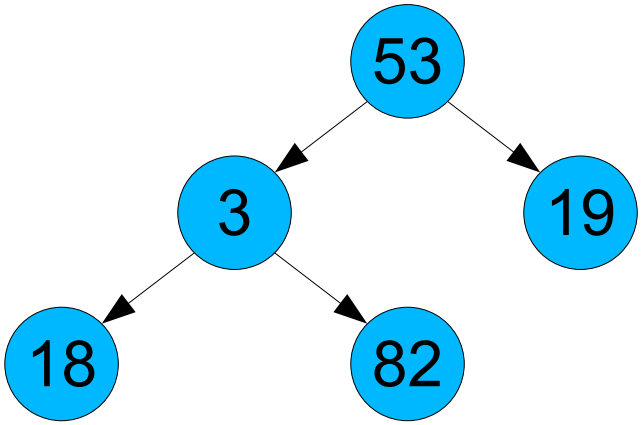
- **Top-down**: Man beginnt mit der Wurzel und setzt ihre Kindknoten, dann deren Kindknoten u.s.w.



- **Bottom-up**: Man beginnt mit den Blättern und verbindet sie jeweils mit ihren gemeinsamen Elternknoten.

Binärbäume



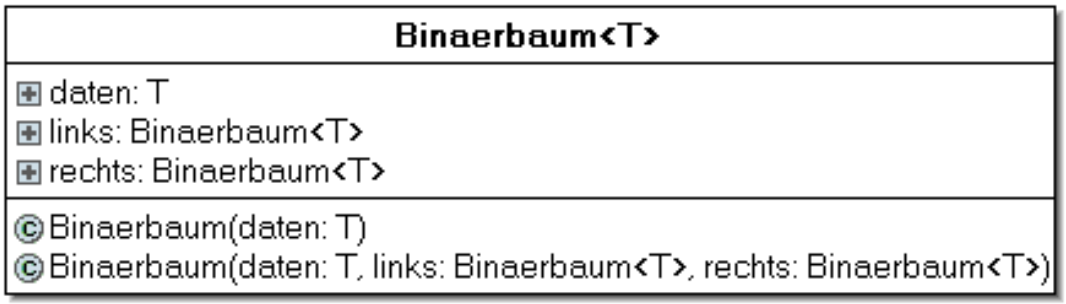


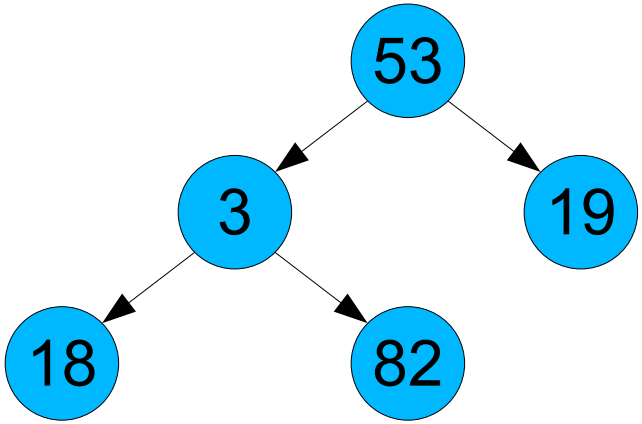
Top-down:

```
Binaerbaum<Integer> wurzel = new Binaerbaum<Integer>(53);
wurzel.links  = new Binaerbaum<Integer>(3);
wurzel.links.links  = new Binaerbaum<Integer>(18);
wurzel.links.rechts = new Binaerbaum<Integer>(82);
wurzel.rechts = new Binaerbaum<Integer>(19);
```

Und in echt natürlich mit Settern...

Binärbäume





Bottom-up:

```
Binaerbaum<Integer> k1 = new Binaerbaum<Integer>(18);
Binaerbaum<Integer> k2 = new Binaerbaum<Integer>(82);
Binaerbaum<Integer> k3 = new Binaerbaum<Integer>(3, k1, k2);
Binaerbaum<Integer> k4 = new Binaerbaum<Integer>(19);
Binaerbaum<Integer> wurzel = new Binaerbaum<Integer>(53, k3, k4);
```

Und in echt natürlich mit Settern...

Allgemeine Bäume

Für einen allgemeinen Baumknoten ist die Anzahl der Kindknoten variabel. Dies könnte z.B. mit einem Array umgesetzt werden, das die Verweise auf die Kinder enthält.