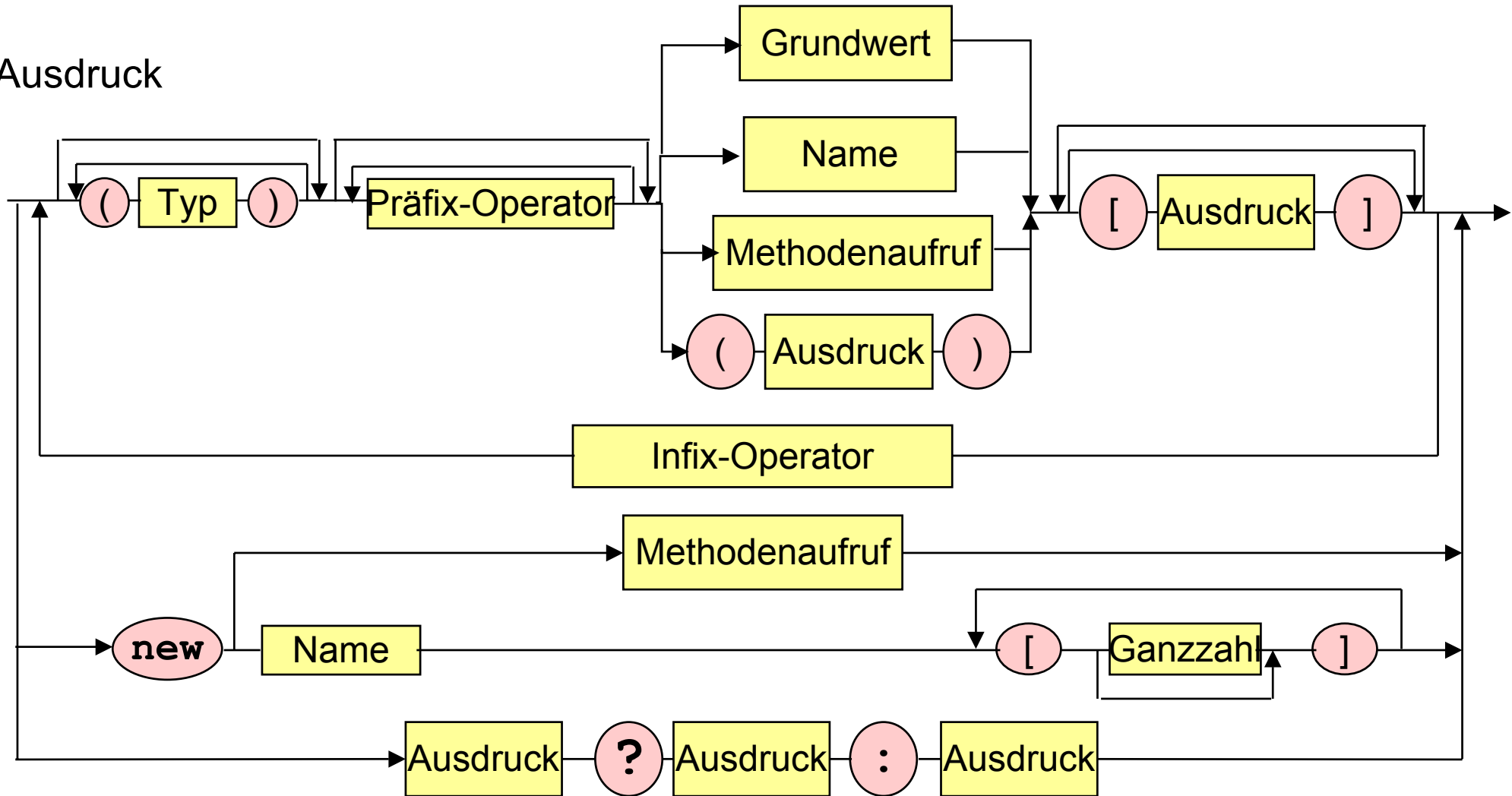

II.3. Rekursion und dynamische Datenstrukturen

- 1. Rekursive Algorithmen
- 2. Rekursive (dynamische) Datenstrukturen

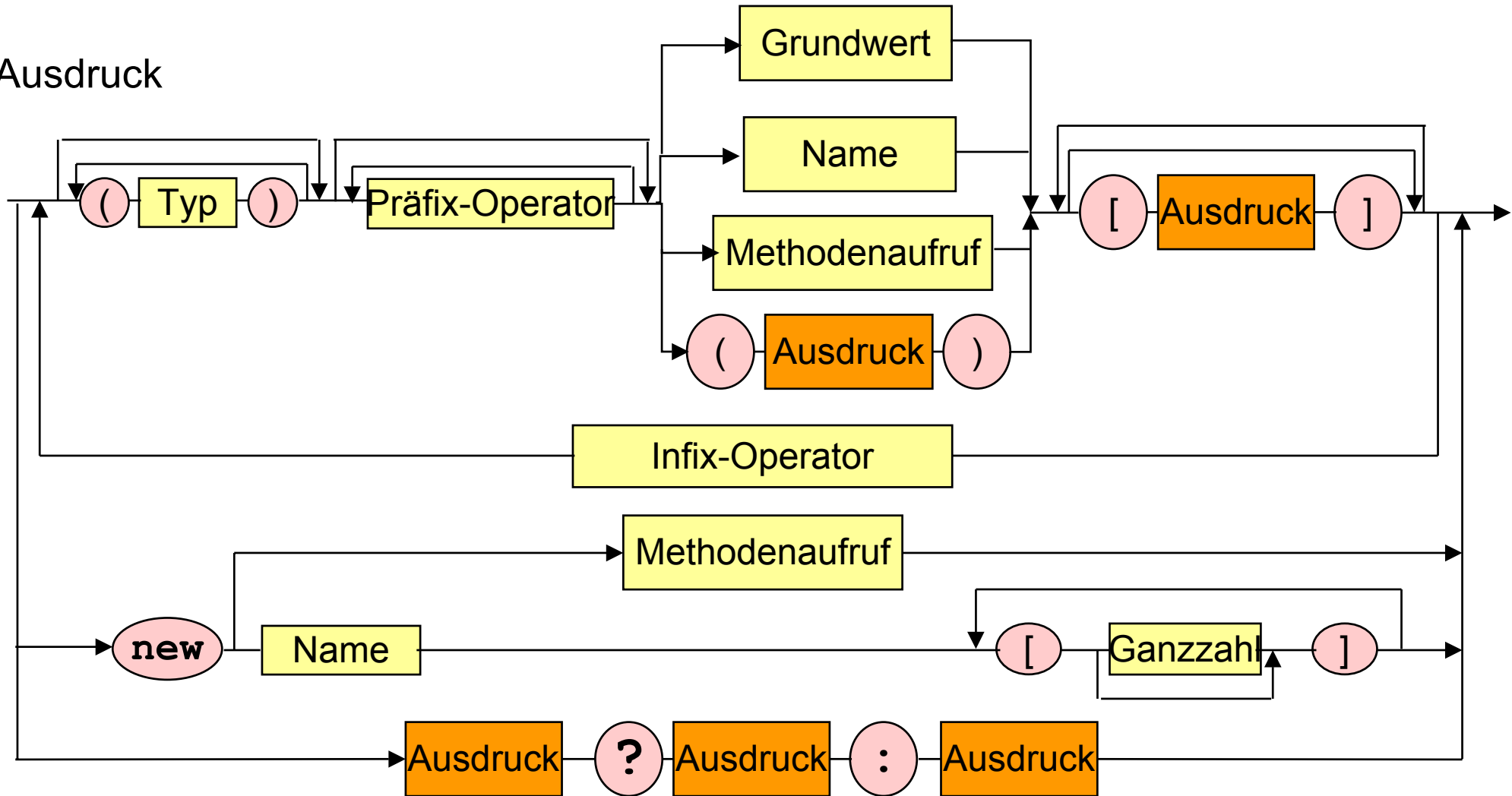
Ausdruck

Ausdruck

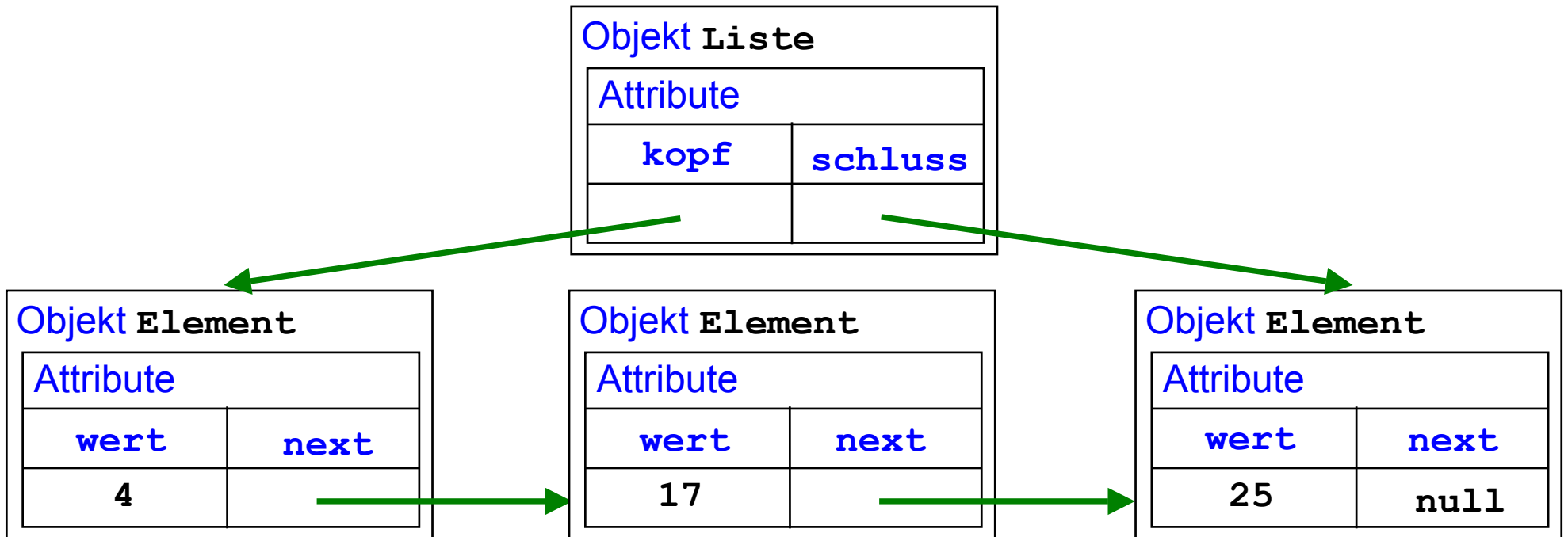


Ausdruck

Ausdruck



Realisierung von Listen



```
public class Element {  
  
    int wert;  
    Element next;  
    ...  
}
```

```
public class Liste {  
  
    private Element kopf;  
    private Element schluss;  
    ...  
}
```

Schnittstellendokumentation

Klasse `Element`

- `Element (int wert)`
- `Element (int wert, Element next)`
- `int getWert ()`
- `void setWert (int wert)`
- `Element getNext()`
- `void setNext (Element next)`
- `String toString ()`

Klasse `Liste`

- `Liste ()`
- `Element suche (int wert)`
- `String toString ()`
- `void drucke ()`
- `void druckeRueckwaerts ()`
- `void fuegeVorneEin (int wert)`
- `void fuegeHintenEin (int wert)`
- `void fuegeSortiertEin (int wert)`
- `void loesche (int wert)`
- `void loesche ()`

Verwendung von Listen

```
Liste l = new Liste ();  
l.fuegeHintenEin (25); l.fuegeVorneEin (17);  
l.fuegeHintenEin (30); l.fuegeVorneEin (4);  
l.drucke (); l.druckeRueckwaerts ();  
l.fuegeSortiertEin (28); l.fuegeSortiertEin (12);  
l.fuegeSortiertEin (45); l.fuegeSortiertEin (2); l.drucke ();  
if (l.suche (17) != null) System.out.println (l.suche(17));  
l.loesche (28); l.loesche (10); l.loesche (17); l.drucke ();  
l.loesche (); l.drucke ();
```

```
( 4 17 25 30 )  
( 30 25 17 4 )  
( 2 4 12 17 25 28 30 45 )  
17  
( 2 4 12 25 30 45 )  
( )
```

Element-Klasse

```
public class Element {
    int wert;
    Element next;

    public Element (int wert) { this.wert = wert; next = null; }

    public Element (int wert, Element next) {
        this.wert = wert; this.next = next; }

    public int getWert () { return wert; }
    public void setWert (int wert) { this.wert = wert; }

    public Element getNext () { return next; }
    public void setNext (Element next) { this.next = next; }

    public String toString () {
        return new Integer(wert).toString(); }
}
```

Liste-Klasse: Erzeugung und Suche

```
public class Liste {  
  
    private Element kopf;  
    private Element schluss;  
  
    public Liste () {  
        kopf = schluss = null;  
    }  
  
    public Element suche (int wert) {  
        return suche (wert, kopf);  
    }  
  
    private static Element suche (int wert, Element kopf) {  
        if      (kopf == null)      return null;  
        else if (kopf.wert == wert) return kopf;  
        else      return suche (wert, kopf.next);  
    }  
}
```


Liste-Klasse: Ausgabe

```
public String toString () {  
    return "( " + durchlaufe(kopf) + ")"; }  
}
```

```
private static String durchlaufe (Element kopf) {  
    if (kopf != null)  
        return kopf.wert + " " + durchlaufe(kopf.next);  
    else return "";  
}
```

```
public void drucke() { System.out.println (this); }
```

```
public String toStringRueckwaerts () {  
    return "(" + durchlaufeRueckwaerts(kopf) + " )"; }  
}
```

```
private static String durchlaufeRueckwaerts (Element kopf) {  
    if (kopf != null)  
        return durchlaufeRueckwaerts(kopf.next) + " " + kopf.wert;  
    else return "";  
}
```

```
public void druckeRueckwaerts () {  
    System.out.println (this.toStringRueckwaerts ()); }  
}
```

Liste-Klasse: Einfügen

```
public void fuegeVorneEin (int wert) {

    if    (kopf == null)    kopf = schluss = new Element (wert);
    else                                kopf = new Element (wert, kopf);

}

public void fuegeHintenEin (int wert) {

    if    (kopf == null)    kopf = schluss = new Element (wert);
    else                                {schluss.next = new Element (wert);
                                        schluss = schluss.next;                                }

}
```

Liste-Klasse: Einfügen

```
public void fuegeSortiertEin (int wert) {  
    kopf = fuegeSortiertEin (wert, kopf); }  
  
private Element fuegeSortiertEin (int wert, Element element) {  
    if (element == null) {  
        schluss = new Element (wert);  
        return schluss; }  
    else if (wert < element.wert)  
        return new Element (wert, element);  
    else {  
        element.next = fuegeSortiertEin (wert, element.next);  
        return element; } }  
}
```

Liste-Klasse: Einfügen

```
public void fuegeSortiertEin (int wert) {
    fuegeSortiertEin (wert, kopf); }

private void fuegeSortiertEin (int wert, Element element) {
    if (element == null) {
        schluss = new Element (wert);
    }

    else if (wert < element.wert)
        new Element (wert, element);

    else
        fuegeSortiertEin (wert, element.next);
    } }
```

Liste-Klasse: Einfügen

```
public void fuegeSortiertEin (int wert) {
    kopf = fuegeSortiertEin (wert, kopf); }

private Element fuegeSortiertEin (int wert, Element element) {
    if      (element == null)           {
        schluss = new Element (wert);
        return schluss;              }
    else if (wert < element.wert)
        return new Element (wert, element);
    else
        {
            element.next = fuegeSortiertEin (wert, element.next);
            return element;           } }

public void fuegeSortiertEin (int wert) {
    Element element = kopf;
    if (kopf == null || wert < kopf.wert) fuegeVorneEin(wert);
    else {while (element.next != null && wert > element.next.wert)
        element = element.next;
        element.next = new Element (wert, element.next); } }
```

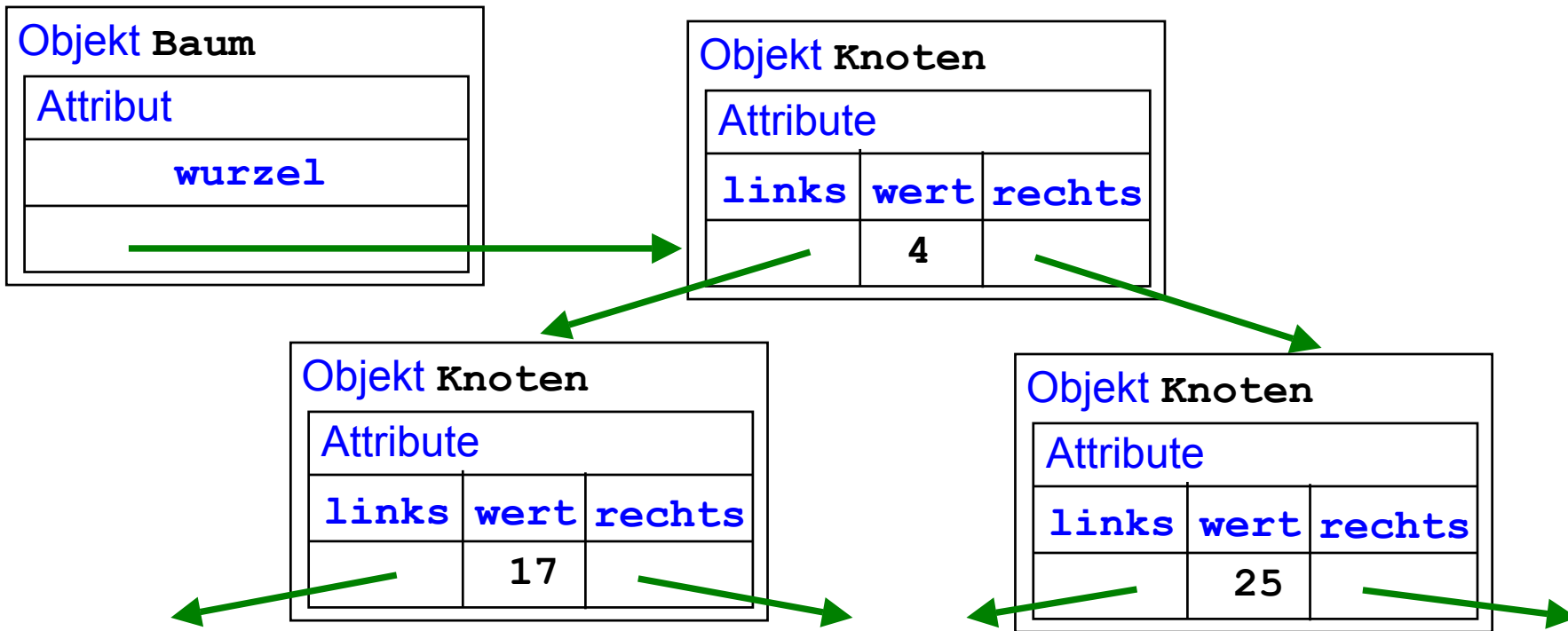
Liste-Klasse: Löschen

```
public void loesche () {  
    kopf = schluss = null;  
}
```

```
public void loesche (int wert) {  
    kopf = loesche (wert, kopf);  
}
```

```
private static Element loesche (int wert, Element element) {  
  
    if      (element == null)      return null;  
    else if (wert == element.wert) return element.next;  
    else                               {  
        element.next = loesche (wert, element.next);  
        return element;  
    }  
  
}}
```

Realisierung von binären Bäumen



```
public class Knoten {  
    int wert;  
    Knoten links, rechts;  
    ...  
}
```

```
public class Baum {  
    private Knoten wurzel;  
    ...  
}
```