```
public class Datenelement {

    private String bezeichner;

    public Datenelement(String bez){
        bezeichner = bez;
    }

    public String datenGeben(){
        return bezeichner;
    }
}
```

```
public class Liste {
    private Listenelement erster;

    public Liste(){
        erster = new Abschluss();
    }

    public void hintenEinfuegen
                                    (Datenelement knoteninhalt){
        erster = erster.hintenEinfuegen(knoteninhalt);
    }

}
```

```
public class Datenknoten extends Listenelement {
   private Datenelement inhalt;
   private Listenelement naechster;

   public Datenknoten(Listenelement n, Datenelement i){
      naechster = n;
      inhalt = i;
   }

   public Datenknoten hintenEinfuegen
                                 (Datenelement knoteninhalt){
      naechster = naechster.hintenEinfuegen(knoteninhalt);
      return this;
   }
}
```

```
public class Abschluss extends Listenelement{

   public Datenknoten hintenEinfuegen
                                   (Datenelement knoteninhalt){
      Datenknoten neuerKnoten =
                           new Datenknoten(this, knoteninhalt);
      return neuerKnoten;
   }
}
```

Wichtig:
Da jeder Abschluss und jeder Datenknoten ein Listenelement ist (Vererbung!), müssen Methoden in beiden Klassen die gleiche, in Listenelement vorgegebene Signatur einhalten.
(gleiche Fähigkeiten!)

Signatur:
public Datenknoten hintenEinfuegen(Datenelement ki)
(alles außer {…Rumpf…})

Da sich Abschluss und Datenknoten unterschiedlich verhalten, unterscheiden sich die Methoden im Rumpf (body).
(unterschiedliches Verhalten!)

Teste dein Programm mit Hilfe einer Testklasse, die du sukzessive erweiterst:

```
public class Testablauf {
  private Liste polonaise = new Liste();

   public void ablaufen(){
      polonaise.hintenEinfuegen(new Datenelement("Daniel"));
      polonaise.hintenEinfuegen(new Datenelement("Luis"));
      polonaise.hintenEinfuegen(new Datenelement("Johannes"));
      polonaise.hintenEinfuegen(new Datenelement("Anja"));
   }

}
```

Implementiere die Methode anzahlElementeGeben() der Klasse Liste (gibt zurück wie viele Datenknoten sich in der Liste befinden).

Liste: anzahlElementeGeben()
Listenelement: anzahlDatenknotenGeben()

Liste:

```
public int anzahlElementeGeben(){
    return erster.anzahlDatenknotenGeben();
}
```

Listenelement:

```
public abstract int anzahlDatenknotenGeben();
```

Datenknoten:

```
public int anzahlDatenknotenGeben(){
    return naechster.anzahlDatenknotenGeben() + 1;
}
```

Abschluss:

```
public int anzahlDatenknotenGeben(){
    return 0;
}
```

Ergänze die Testklasse:

```
public void ablaufen(){
    polonaise.hintenEinfuegen(new Datenelement("Daniel"));
    polonaise.hintenEinfuegen(new Datenelement("Luis"));
    polonaise.hintenEinfuegen(new Datenelement("Johannes"));
    polonaise.hintenEinfuegen(new Datenelement("Anja"));

    System.out.println("Bei der Polonaise machen "
            +polonaise.anzahlElementeGeben()+ " Leute mit!");
  }
```

Implementiere die Methode alleDatenAusgeben() der Klasse Liste (gibt eine Liste aller Daten der Liste aus).

Liste: alleDatenAusgeben()
Listenelement: listendatenAusgeben()

Liste:
```
public void alleDatenAusgeben(){
    System.out.println(erster.listendatenAusgeben());
}
```

Listenelement:
```
public abstract String listenDatenAusgeben();
```

Datenknoten:
```
public void listendatenAusgeben(){
    return inhalt.datenGeben() + "\n"
        + naechster.listendatenAusgeben();
}
```

Abschluss:
```
public String listendatenAusgeben(){
    return "";
}
```

Ergänze außerdem folgende Methoden der Klasse Liste:

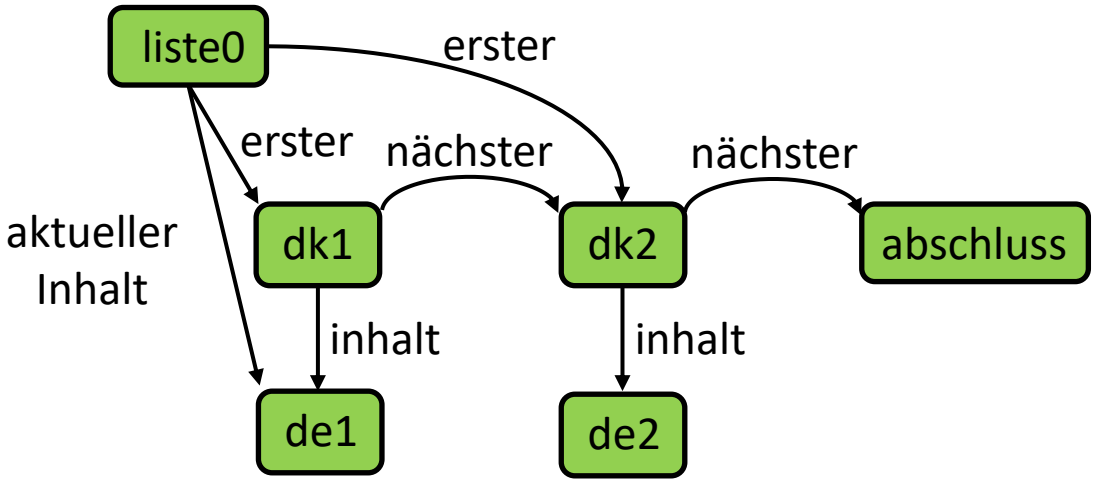
vorneEinfuegen(datenelement)
vorneEntnehmen()
hintenEntnehmen()
istLeer()

- vorneEinfuegen(datenElement) nur in Klasse Liste

```
public void vorneEinfuegen(Datenelement knoteninhalt){
        Datenknoten neuerKnoten =
                        new Datenknoten(erster, knoteninhalt);
        erster = neuerKnoten;
   }
```

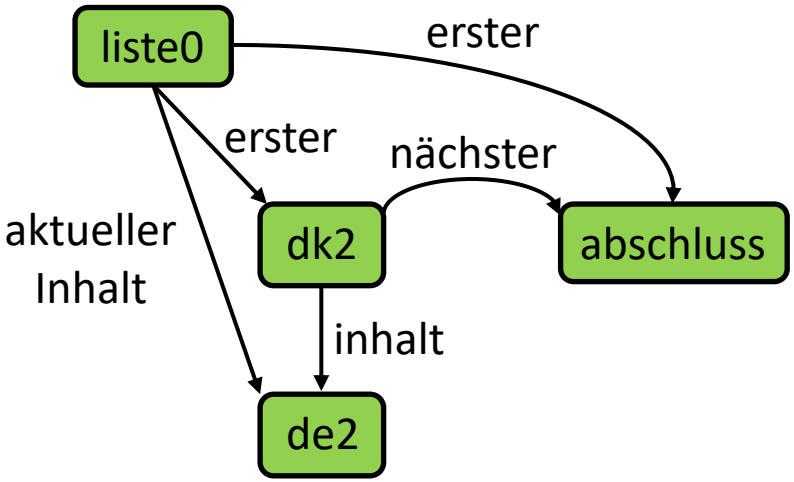
- public Datenelement vorneEntnehmen()

```
public Datenelement vorneEntnehmen(){
        Datenelement aktuellerInhalt = erster.inhaltGeben();
        erster = erster.naechsterGeben();
        return aktuellerInhalt;
   }
```



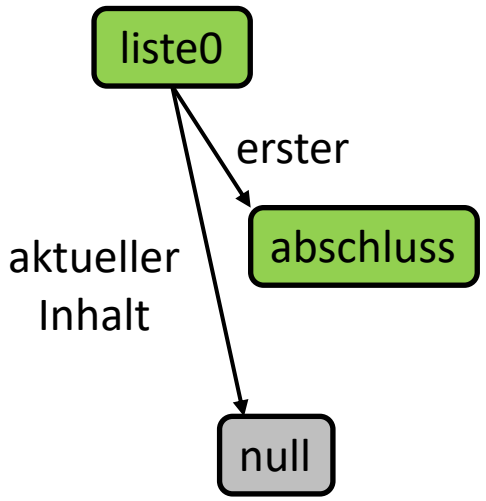
- public Datenelement vorneEntnehmen()

```
public Datenelement vorneEntnehmen(){
        Datenelement aktuellerInhalt = erster.inhaltGeben();
        erster = erster.naechsterGeben();
        return aktuellerInhalt;
   }
```



- public Datenelement vorneEntnehmen()

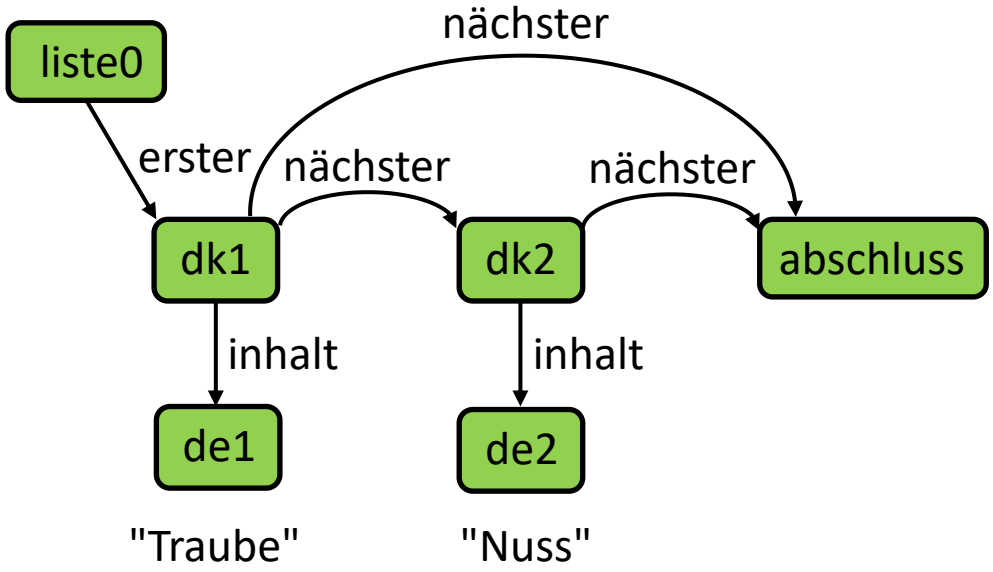
```
public Datenelement vorneEntnehmen(){
        Datenelement aktuellerInhalt = erster.inhaltGeben();
        erster = erster.naechsterGeben();
        return aktuellerInhalt;
    }
```



inhaltGeben() in der Klasse Abschluss liefert null.

naechsterGeben() in der Klasse Abschluss liefert this.

- public void entfernen (String datenwert)




```
liste0.entfernen("Nuss");

erster.entfernen("Nuss");

naechster.entfernen("Nuss");
```

In jedem Datenknoten wird der übergebene Datenwert mit dem entsprechenden Wert des Inhalts verglichen.

Stimmen die Werte nicht überein, wird die Methode an den Nächsten weitergereicht.

Bei Übereinstimmung wird der Datenknoten dk entfernt, d.h. die Referenz naechster des Vorgängerknotens von dk muss auf den Nachfolger von dk gelegt werden.

Rückgabe im Falle der Übereinstimmung:

naechster (Typ: Listenelement)

Andernfalls Rückgabe:

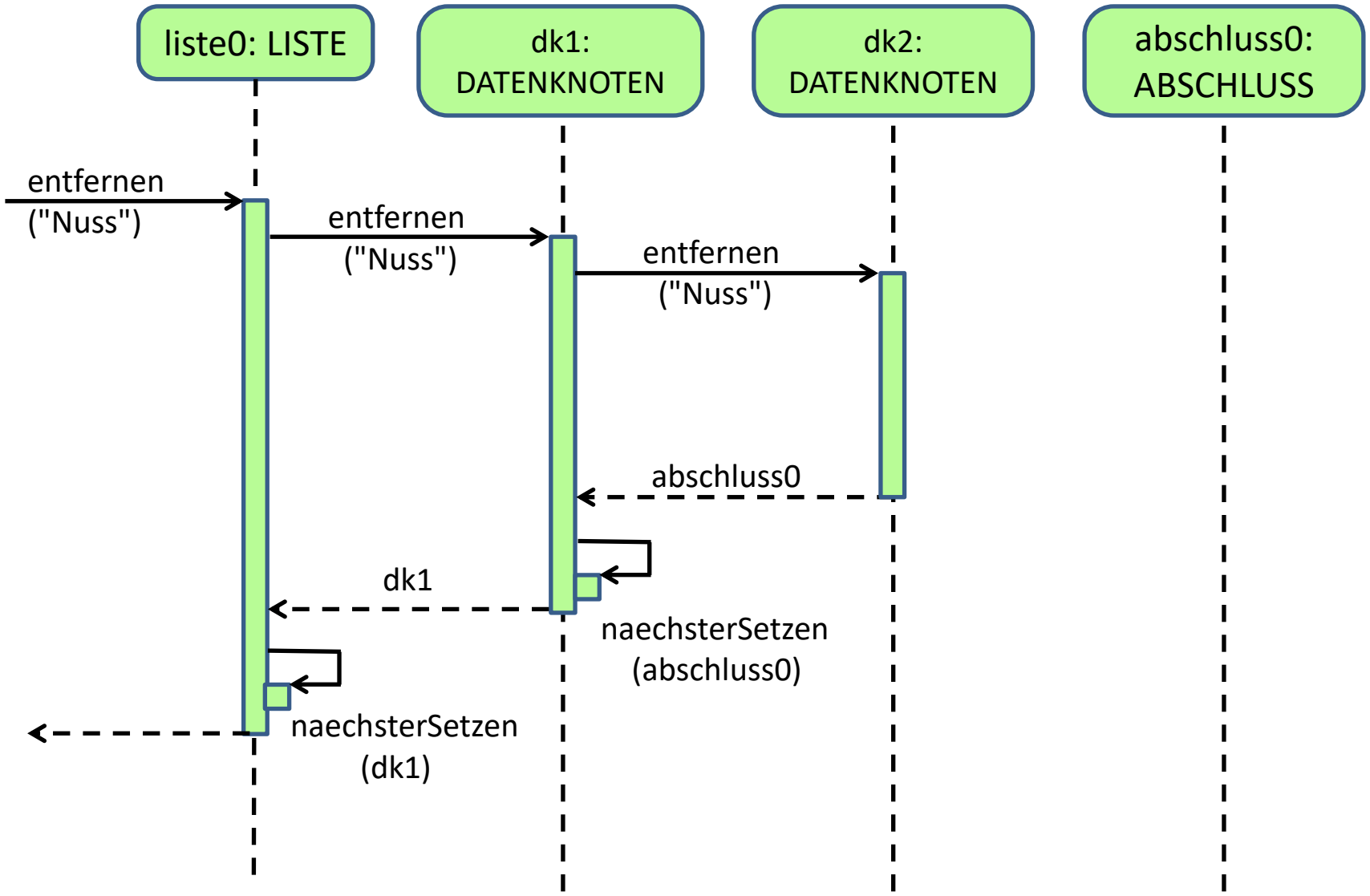
this (Typ: Listenelement)

Im Abschluss:

Wenn die Methode bis zum Abschluss durchgereicht wird, heißt das, dass der übergebene Datenwert nicht gefunden wurde.

Rückgabe: this (+ evtl. Ausgabe einer Fehlermeldung)

- public void entfernen(String datenwert)



Liste:
```
public void entfernen(String datenwert){
    erster=erster.entfernen(datenwert);
}
```

Listenelement:

```
public abstract Listenelement entfernen(String datenwert);
```

Datenknoten:

```
public Listenelement entfernen(String datenwert){
        if (datenwert.equals(inhalt.datenGeben())) {
            return naechster;
        }
        else {
            naechster = naechster.entfernen(datenwert);
            return this;
        }
    }
```

Abschluss:

```
    public Listenelement entfernen(String datenwert){
            return this;
        }
```

Da Strings Objekte sind, werden sie mit .equals verglichen:

```
public boolean stringVergleich(){
        String s1 = "Hallo";
        String s2 = "Hallo";
        return s1==s2;
 }
```

liefert true ☺

Aber:

```
public boolean stringVergleich(){
        String s1 = new String("Hallo");
        String s2 = new String("Hallo");
        return s1==s2;
 }
```

liefert false ☹

s1.equals(s2);

liefert bei gleichen Strings stets true

☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺☺

public Datenknoten datenknotenGeben(String datenwert)

Gibt den (ersten) zu einem Datenwert gehörenden Datenknoten zurück.

Liste:

```
public Datenknoten datenknotenGeben(String datenwert){
      return erster.datenknotenGeben(datenwert);
   }
```

Listenelement:

```
public abstract Datenknoten datenknotenGeben
                                        (String datenwert);
```

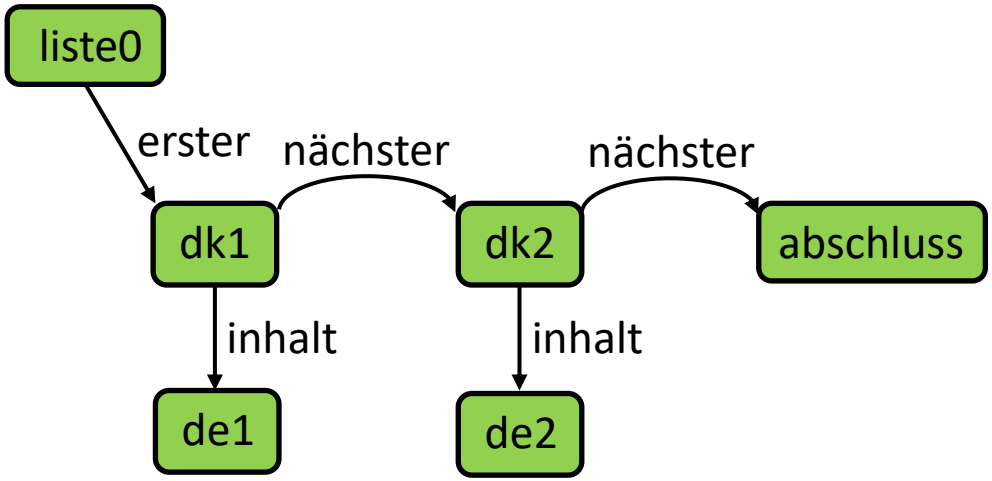
Datenknoten:

```
public Datenknoten datenknotenGeben(String datenwert){
        if (datenwert.equals(inhalt.datenGeben())){
            return this;
        }
        else {
            return naechster.datenknotenGeben(datenwert);
        }
    }
```

Abschluss:

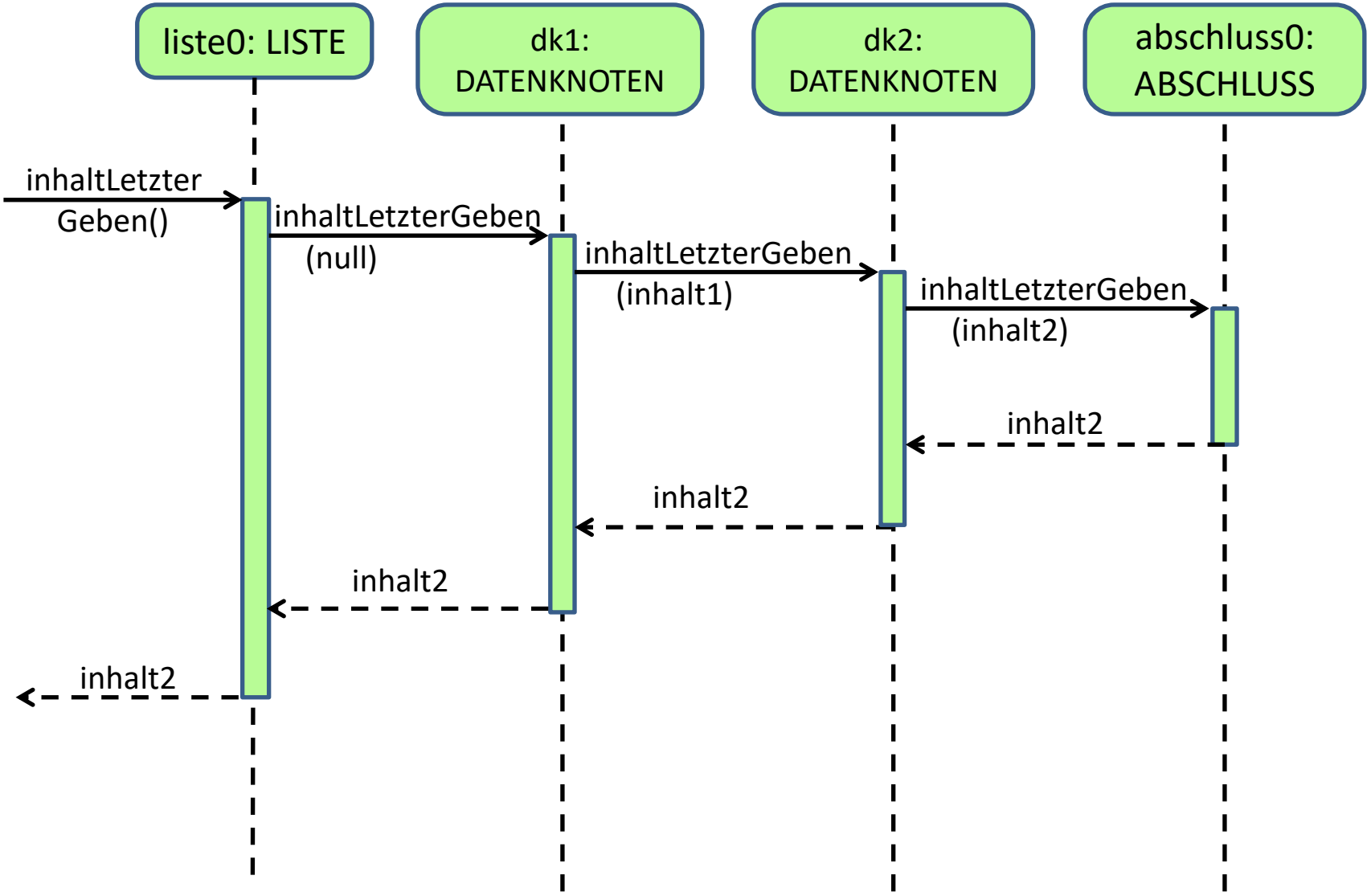
```
public Datenknoten datenknotenGeben(String datenwert){
        return null;
}
```

- public Datenelement inhaltLetzterGeben()



- Es muss de2 zurückgegeben werden.
- Der abschluss muss für die Rückgabe sorgen.
- Da der abschluss keinen direkten Zugriff auf de2 hat, muss ihm dieser beim Methodenaufruf übergeben werden.
- Was der abschluss zurückgibt, wird von den Datenknoten bis zur Liste zurückgereicht.

- public Datenelement inhaltLetzterGeben()



Liste:

```
public Datenelement inhaltLetzterGeben(){
    return erster.inhaltLetzterGeben(null);
}
```

Listenelement:

```
public abstract Datenelement inhaltLetzterGeben
                                (Datenelement aktuellerInhalt);
```

Datenknoten:

```
public Datenelement inhaltLetzterGeben
                                (Datenelement aktuellerInhalt){
    return naechster.inhaltLetzterGeben(inhalt);
}
```

Abschluss:

```
public Datenelement inhaltLetzterGeben
                                (Datenelement aktuellerInhalt){
    return aktuellerInhalt;
}
```

public Datenelement entnehmen(String datenwert)

Teste das Programm auch für Datenwerte, die in der Liste nicht vorhanden sind!

Nur in Liste:

```
public Datenelement entnehmen(String datenwert){
    Datenknoten gesuchterDk = datenknotenGeben(datenwert);
    if (gesuchterDk != null) {
        Datenelement gesuchtesDe = gesuchterDk.inhaltGeben();
        entfernen(datenwert);
        return gesuchtesDe;
    }
    else return null;
 }
```

public Datenelement hintenEntnehmen()

Teste das Programm auch für die leere Liste!

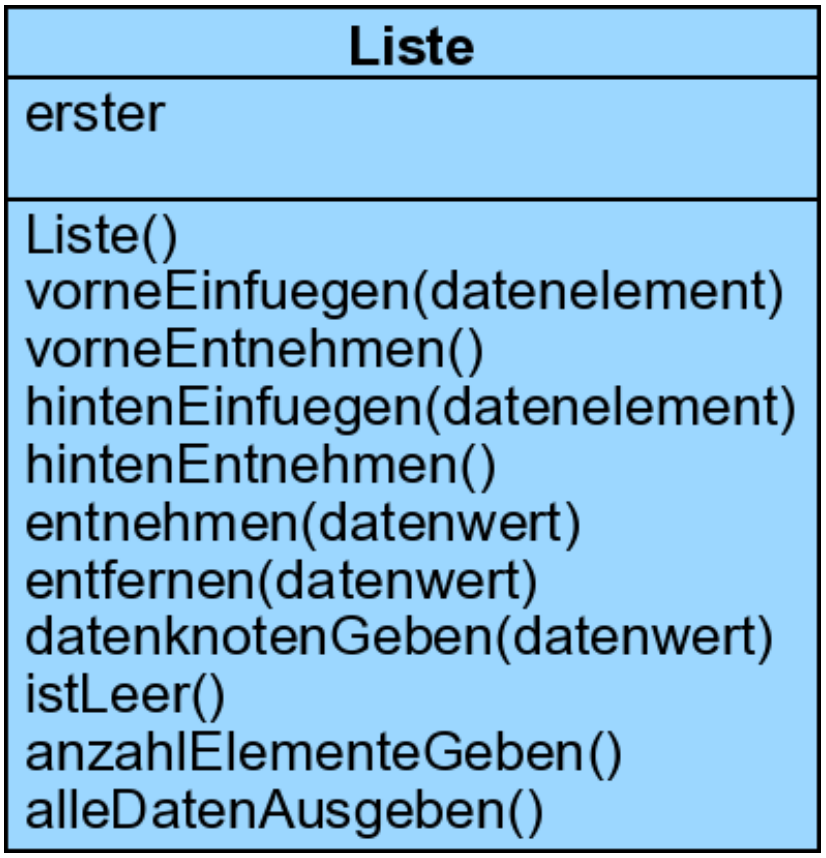
Nur in Liste:

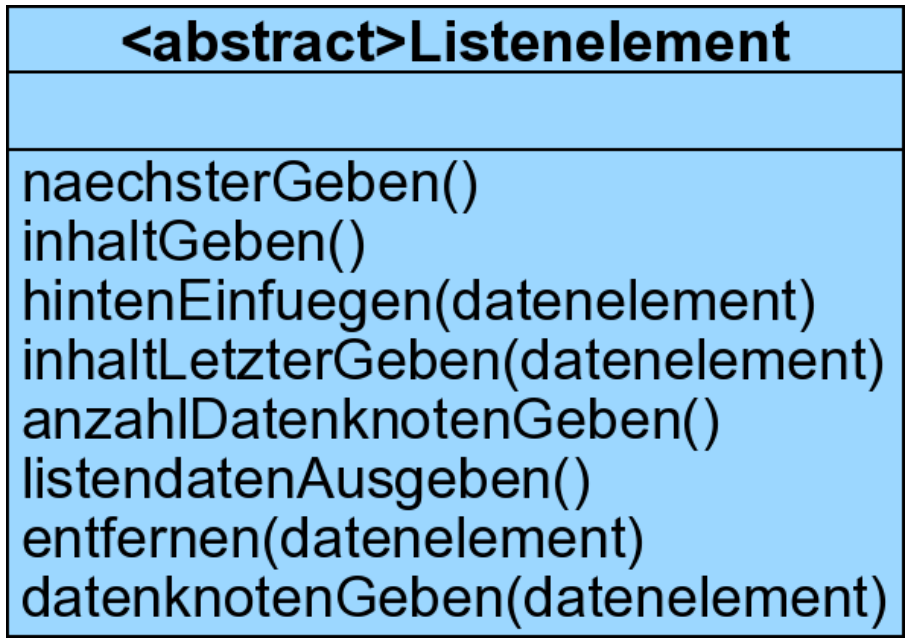
```
public Datenelement hintenEntnehmen(){
    Datenelement gesuchtesDe = inhaltLetzterGeben();
    if (gesuchtesDe != null) {
        return entnehmen(gesuchtesDe.datenGeben());
    }
    else return null;
}
```

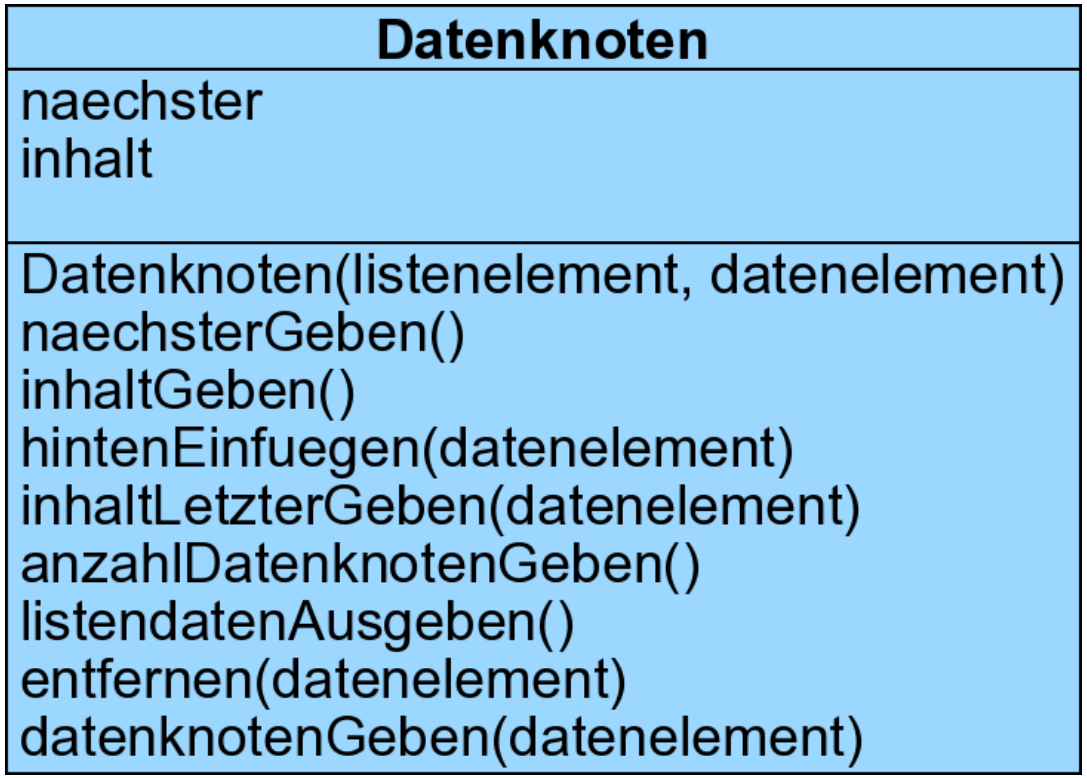
Funktioniert allerdings nur, wenn es sich bei den Datenwerten um Schlüssel handelt.

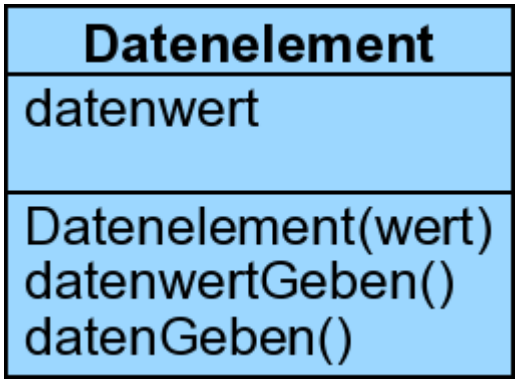
istLeer() in Liste:

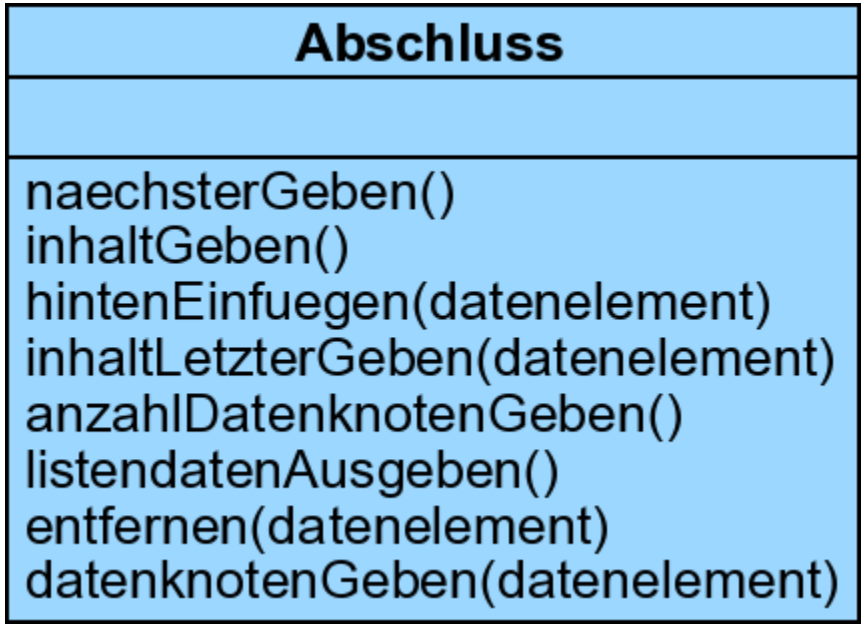
```
public boolean istLeer(){
        return (anzahlElementeGeben()==0);
    }


public boolean istLeer2(){
        return (erster instanceof Abschluss);
    }
```


Liste
erster
Liste()
vorneEinfuegen(datenelement)
vorneEntnehmen()
hintenEinfuegen(datenelement)
hintenEntnehmen()
entnehmen(datenwert)
entfernen(datenwert)
datenknotenGeben(datenwert)
istLeer()
anzahlElementeGeben()
alleDatenAusgeben()


<abstract>Listenelement
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
anzahlDatenknotenGeben()
listendatenAusgeben()
entfernen(datenelement)
datenknotenGeben(datenelement)


Datenknoten
naechster
inhalt
Datenknoten(listenelement, datenelement)
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
anzahlDatenknotenGeben()
listendatenAusgeben()
entfernen(datenelement)
datenknotenGeben(datenelement)


Datenelement
datenwert
Datenelement(wert)
datenwertGeben()
datenGeben()


Abschluss
naechsterGeben()
inhaltGeben()
hintenEinfuegen(datenelement)
inhaltLetzterGeben(datenelement)
anzahlDatenknotenGeben()
listendatenAusgeben()
entfernen(datenelement)
datenknotenGeben(datenelement)