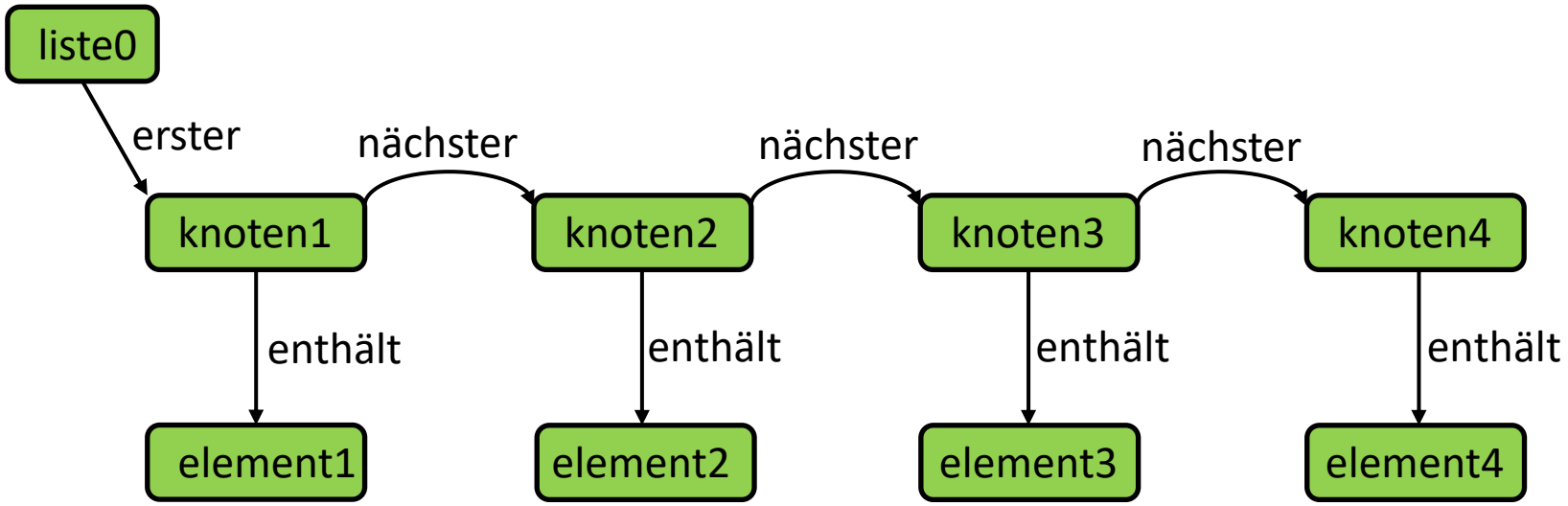
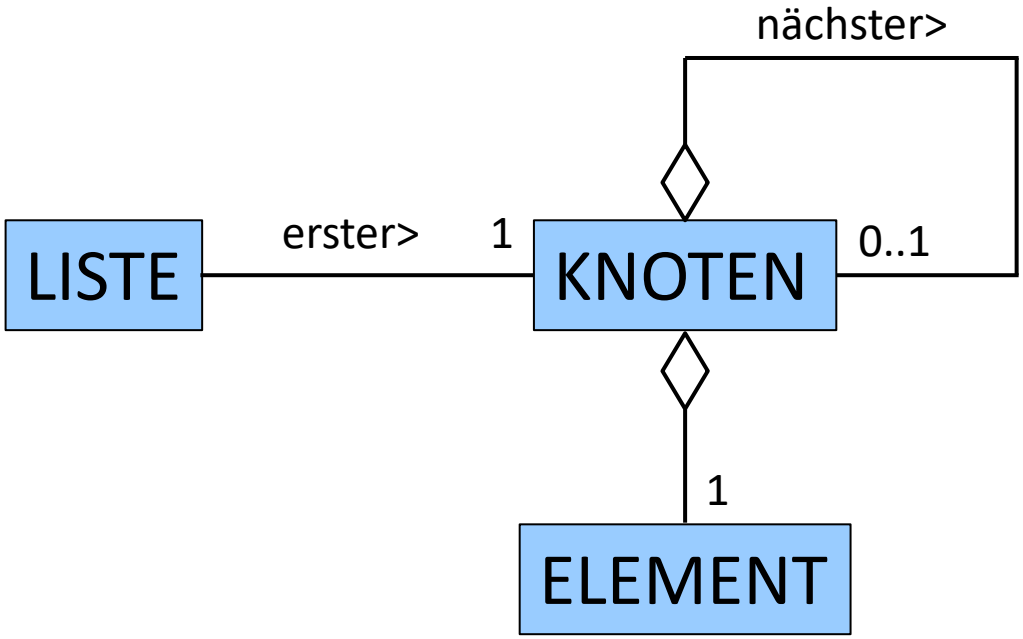
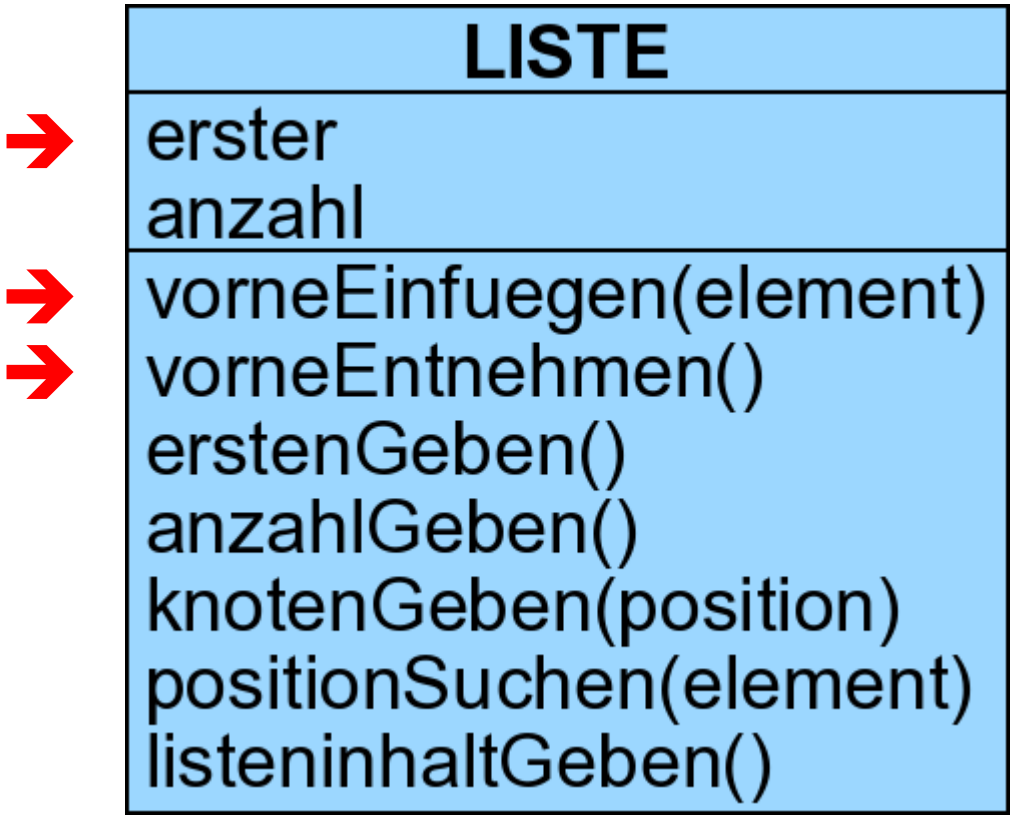
Objektdiagramm:



Klassendiagramm:



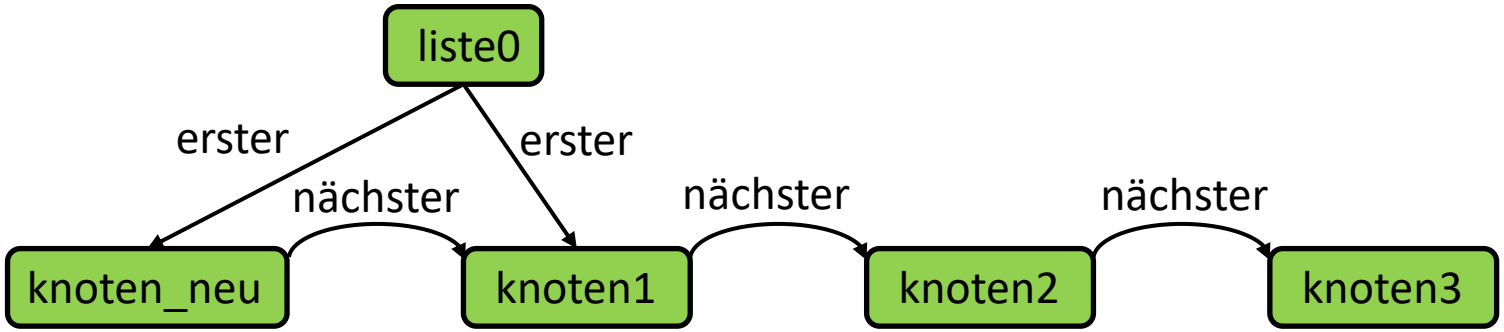
Klassenkarte:



Implementiere die Klasse Liste (zunächst nur Attributdeklaration und Konstruktor).

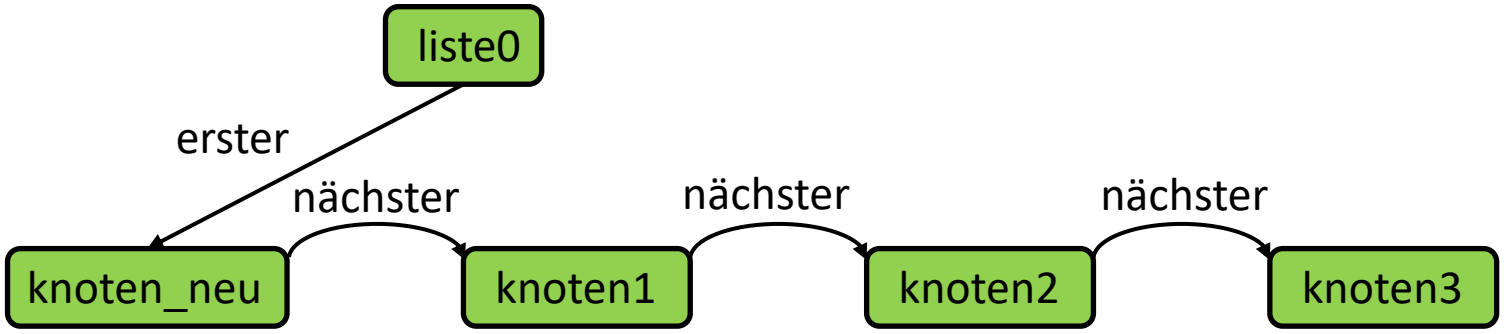
```
public class Liste {

    private Knoten erster;
    private int anzahl;

    public Liste() {
        erster = null;
        anzahl = 0;
    }
```

vorneEinfügen(element)




```
public void vorneEinfuegen(Element elem){
      Knoten neuerkn = new Knoten(elem);
      neuerkn.naechsterSetzen(erster);
      erster = neuerkn;
      anzahl ++;
   }
```

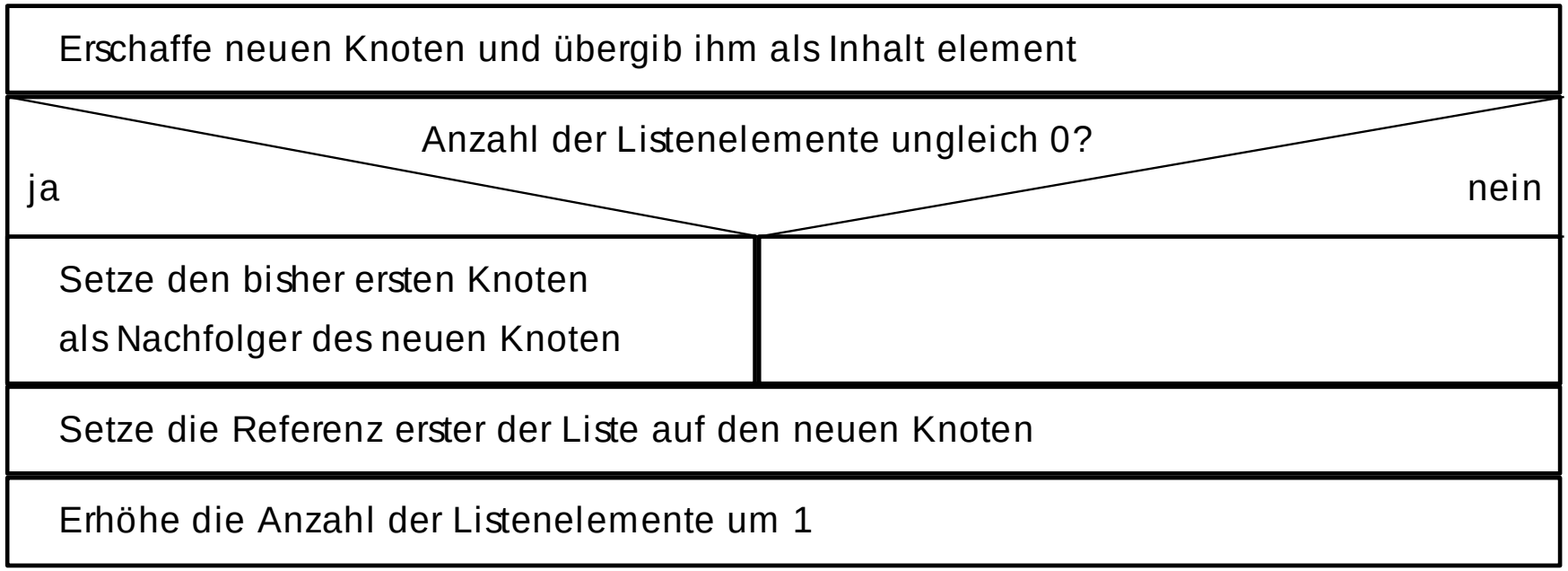
vorneEinfügen(element)



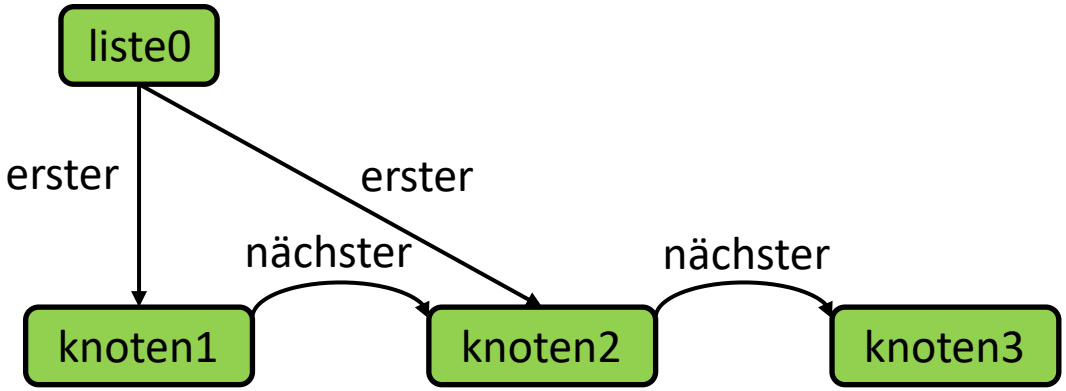

```
public void vorneEinfuegen(Element elem){
    Knoten neuerkn = new Knoten(elem);
    if (anzahl!=0) {
       neuerkn.naechsterSetzen(erster);
    }
    erster = neuerkn;
    anzahl = anzahl +1;
  }
```

Gib das Struktogramm der Methode vorneEinfuegen(Element elem) an.

public void vorneEinfügen(element)



vorneEntnehmen()



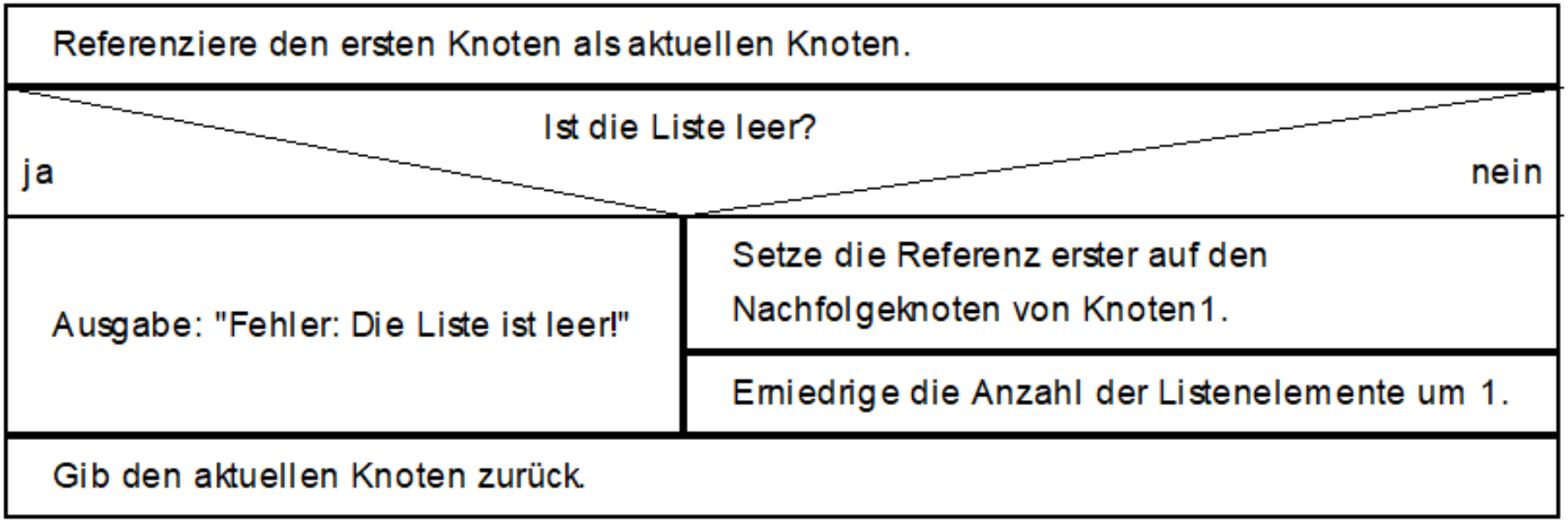
Vorsicht: Nun kein Zugriff mehr auf knoten1!!!

➔ Vorher knoten1 als aktuellerknoten speichern!

Implementiere die Methode:

- Welche Fälle werden unterschieden?
- Wie kommt man auf knoten2?
- Was muss mit anzahl passieren?

public Knoten vorneEntnehmen()



```
public Knoten vorneEntnehmen(){
    Knoten aktuellerKnoten = erster;
    if(          ) {
        System.out.println("Fehler: Die Liste ist leer!");
    }
    else {
        erster =                                    ;
        anzahl =          
    }
                                    
}
```

```
public Knoten vorneEntnehmen(){
    Knoten aktuellerKnoten = erster;
    if(anzahl==0) {
        System.out.println("Fehler: Die Liste ist leer!");
    }
    else {
        erster = erster.naechsterGeben();
        anzahl = anzahl -1;
    }
    return aktuellerKnoten;
  }
```

```
public Knoten erstenGeben(){
    return erster;
}

public int anzahlGeben(){
    return anzahl;
}
```

knotenGeben(position)

- Welche Fallunterscheidung muss getroffen werden?
- Welche Kontrollstruktur braucht man, um an die entsprechende Position zu kommen?
- Spiele bei der for-Schleife in Gedanken die Fälle position = 1 bzw. position = 2 durch.

```
public Knoten knotenGeben(int position){
      Knoten aktuellerKnoten = null
      if ((position<1)||(position>anzahl)){
         System.out.println("Fehler: An dieser Position
                                    befindet sich kein Knoten!");
        }
      else {
         aktuellerKnoten = erster;
         for (int i=1; i<position;i++) {
                  aktuellerKnoten =
                           aktuellerKnoten.naechsterGeben();
         }
         return aktuellerKnoten;
      }
   }
```

positionSuchen(element)

```
public int positionSuchen(Element ele){
        int position
        Knoten k =
        for (int i=1; i        ; i++){
            if (ele ==                ) {position     }
            k=
        }
        if (position == 0) {
            
        }
        return position;
    }
```

positionSuchen(element)

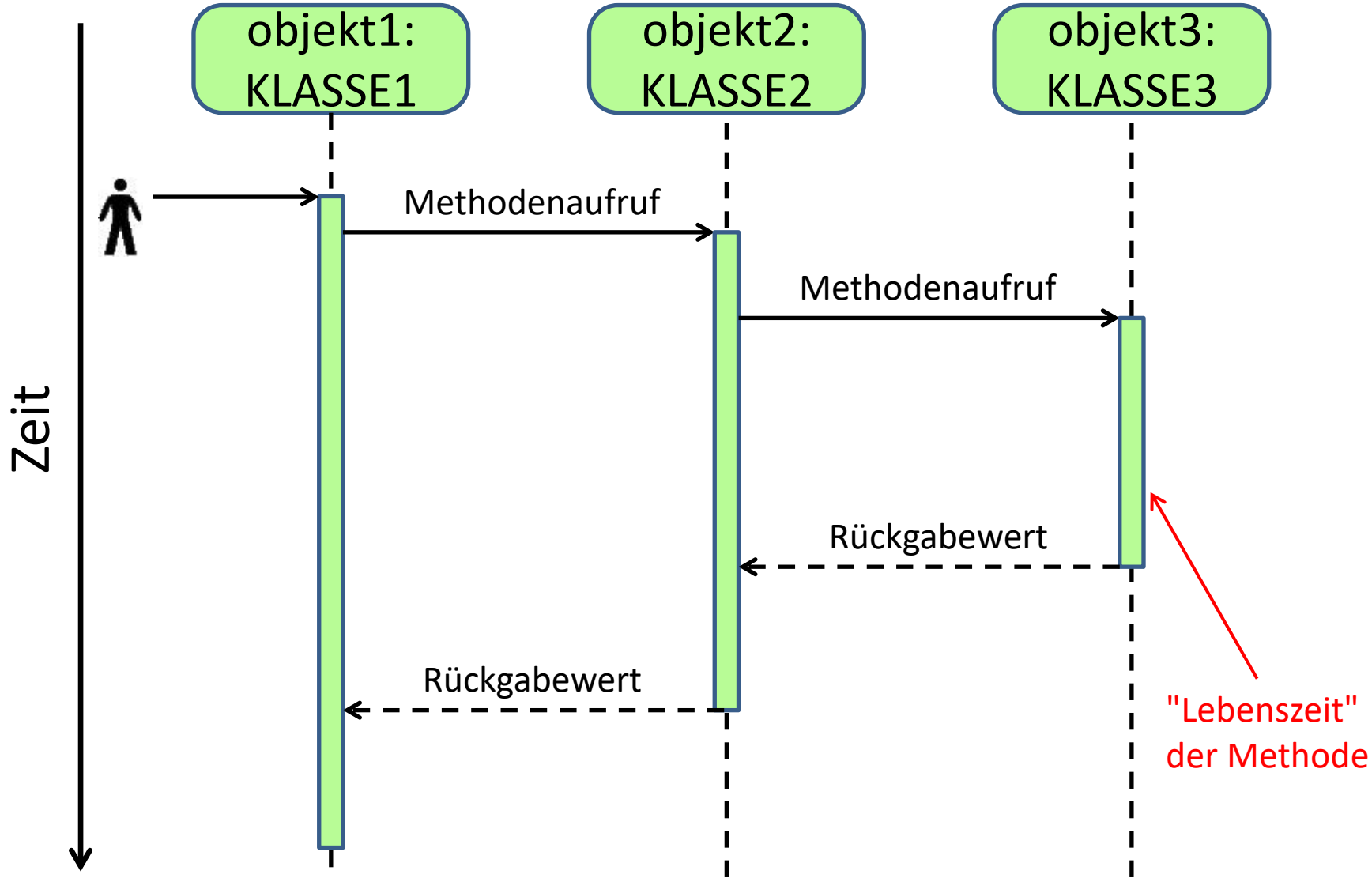
```
public int positionSuchen(Element ele){
        int position = 0;
        Knoten k = erster;
        for (int i=1; i<=anzahl; i++){
            if (ele == k.inhaltGeben()) {position = i;}
            k=k.naechsterGeben();
        }
        if (position == 0) {
            System.out.println("Das gesuchte Element befindet
                                        sich nicht in der Liste!");
        }
        return position;
    }
```

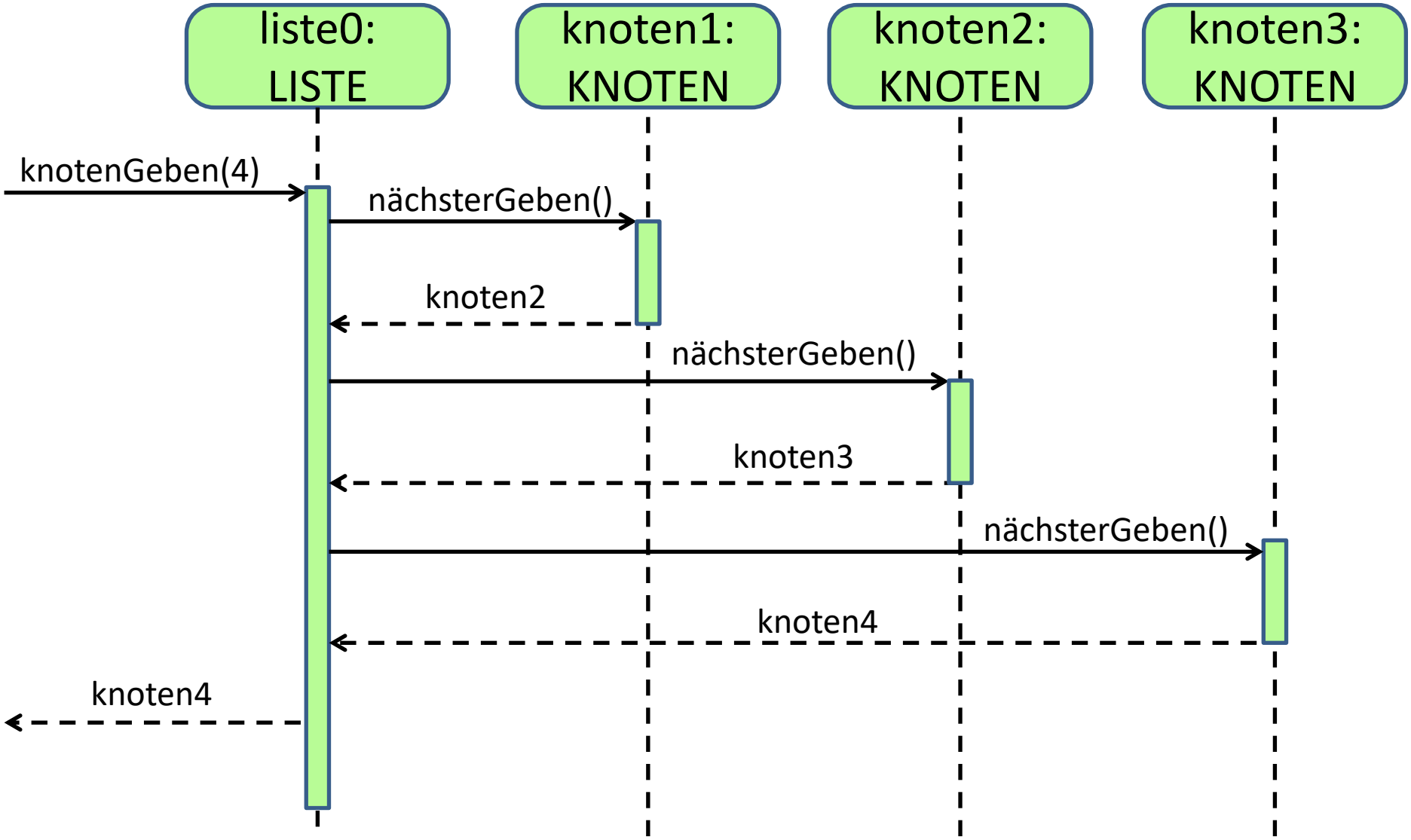
listeninhaltGeben()

```
public String listeninhaltGeben(){
        String inhalt="Listeninhalt: ";
        Knoten k = erster;
        for (int i=1;i<=anzahl;i++){
                inhalt = inhalt +k.inhaltGeben().datenGeben()+"\n";
                k=k.naechsterGeben();
        }
        return inhalt;
    }
```

```
public Knoten knotenGeben(int position){
        …
        Knoten k = erster;
        for (int i=1; i<position;i++) {
                k=k.naechsterGeben();
        }
        return k;
    }
}
```

Gib das Sequenzdiagramm für knotenGeben(4) an!


objekt1:
KLASSE1
objekt2:
KLASSE2
objekt3:
KLASSE3
Methodenaufruf
Methodenaufruf
Rückgabewert
Rückgabewert
Zeit
"Lebenszeit"
der Methode


liste0:
LISTE
knoten1:
KNOTEN
knoten2:
KNOTEN
knoten3:
KNOTEN
knotenGeben(4)
nächsterGeben()
knoten2
nächsterGeben()
knoten3
nächsterGeben()
knoten4
knoten4

Iteration

S. 19 / 1, 2

Lies dazu den Lehrtext und passe die Implementierung unserer Liste entsprechend an (oder implementiere neu). Teste die Implementierung mit Hilfe einer geeigneten Testklasse!

Stativ entspricht Liste
Geändert werden muss nichts, allenfalls die Methodennamen (obenEntnehmen statt vorneEntnehmen).

2) dazwischenEntnehmen

```
public Knoten dazwischenEntnehmen(int position){
        Knoten knotenAlt = knotenGeben(position);
        if (knotenAlt != null) {
            if (position==1) {
                erster=erster.naechsterGeben();
            }
            else {
                knotenGeben(position-1).
                                naechsterSetzen(knotenGeben(position+1));
            }
            anzahlKnoten=anzahlKnoten-1;
        }
        return knotenAlt;
```