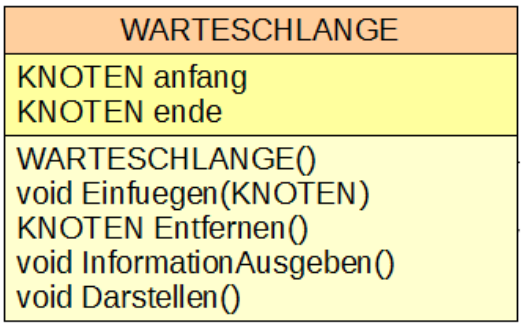
1.4 Rekursive Methoden

In der dynamischen Struktur der Liste, die wir nun implementiert haben, können wir nicht mehr ohne weiteres auf einzelne Elemente zugreifen. Am Beispiel der Darstellen() Methode sehen wir eine andere Art und Weise, alle Elemente einer Liste durchgehen zu können. Wir betrachten den Code der Darstellen Methoden:

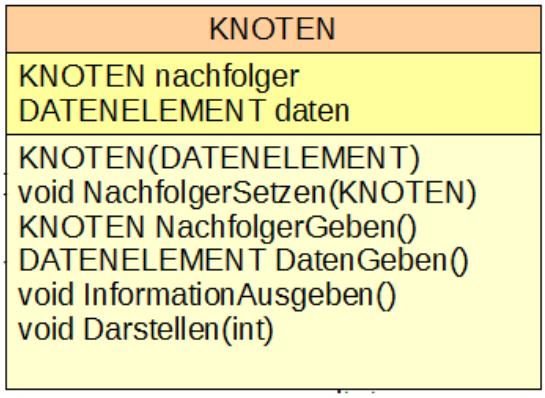
WARTESCHLANGE:

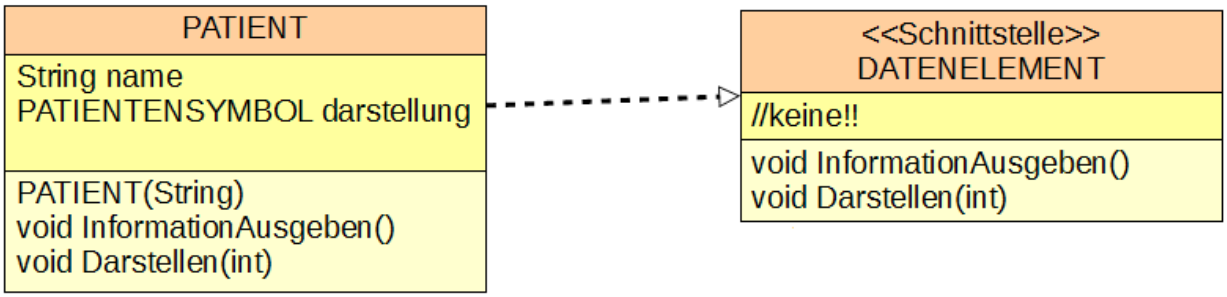
```
public void Darstellen()
    {
        if(anfang==null)
        {
            PATIENTENSYMBOL.LeerZeigen();
        }
        else
        {
            anfang.Darstellen(1);
        }
    }
```

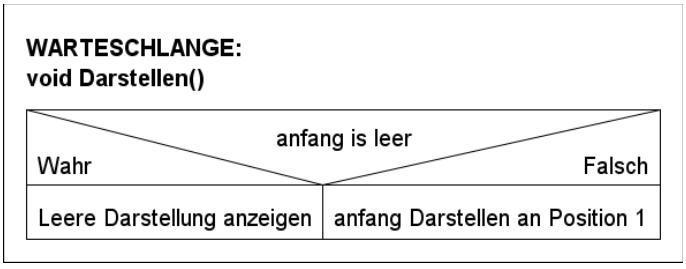


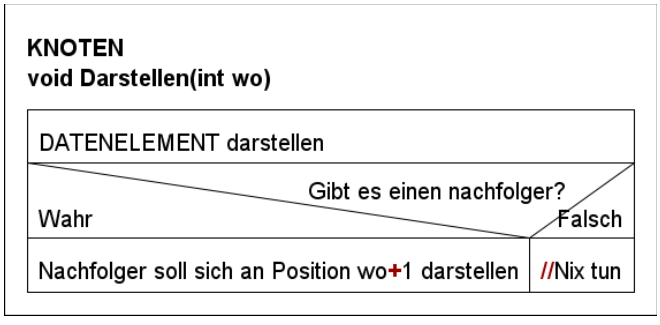
KNOTEN:

```
public void Darstellen(int wo)
    {
        daten.Darstellen(wo);
        if(nachfolger!=null)
        {
            nachfolger.Darstellen(wo+1);
        }
    }
```



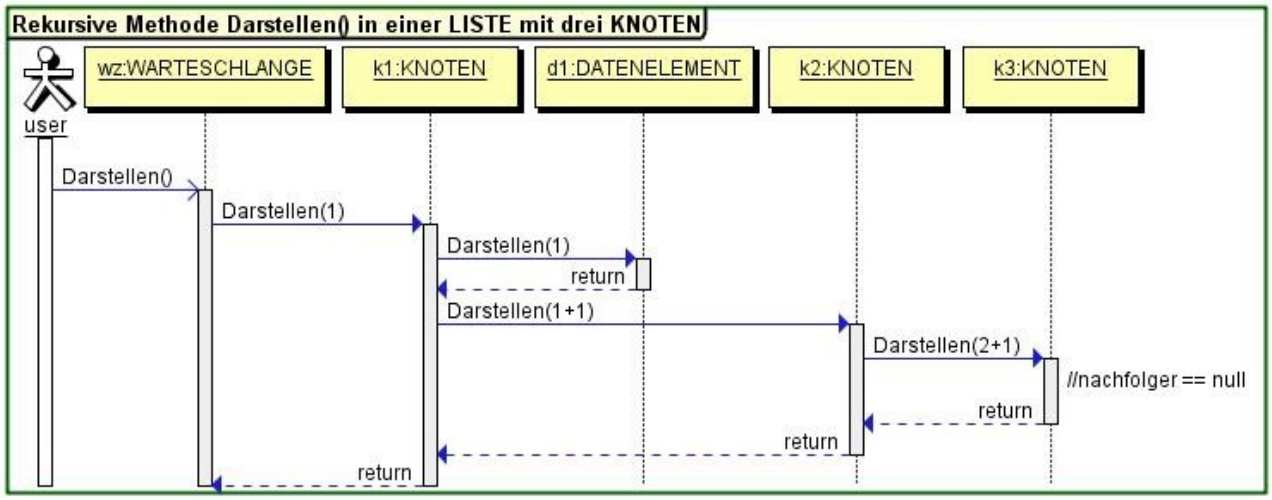






Es fällt auf, dass jeder Knoten zuerst auf seinem DATENELEMENT die Methode ausführt und dann den Auftrag mit neuem Parameter an den nächsten KNOTEN weitergibt. Das bedeutet die Methode Darstellen(int) in KNOTEN ruft sich selber wieder in einem anderen KNOTEN auf. Eine solche „sich-selbst-wieder-aufrufende“ Methode nennt man ***rekursive Methode***.

Man sieht den Ablauf deutlich im Sequenzdiagramm:



Aufgaben:

1) Kopiere die Vorlage aus dem Klassenordner (Swing_Arztzimmer_04). Wenn du auf deiner Version weiterarbeiten willst gehe sicher, dass deine Liste einwandfrei Struktur und Daten teilt. Falls nicht geschehen, ergänze die Darstellen() Methoden der Klassen durch den Code oben.

2) Überlege dir anhand geeigneter Struktogramme und eines Sequenzdiagrammes, wie die Methode InformationAusgeben() in den Klassen WARTESCHLANGE, KNOTEN und PATIENT umgesetzt werden muss. Nutze die Anweisung `System.out.println(„Text“)` um Text auf der Konsole ausgeben zu lassen.

3) Erweitere deine Klasse WARTESCHLANGE um die Methode int LaengeGeben(), die rekursiv die Länge der Liste ausgibt. Überlege zuerst, welche Klassen hier eine neue Methode „LaengeGeben“ brauchen und welche nicht. Überlege dir wieder anhand von Struktogrammen und eines Sequenzdiagrammes, wie die Struktur der Methode abläuft! (HINWEIS: Denke auch daran, dass gegebenenfalls in manchen Klassen die Methode einen Parameter braucht und in anderen nicht.)

4) Schreibe eine rekursive Methode, die an beliebiger Stelle einfügt oder entfernt.

5) Wir haben uns mit Schnittstellen, als Möglichkeit Kommunikatioswege zwischen Klassen sicherzustellen, beschäftigt. Ein sehr wichtiges Konzept, besonders bei Listen und Reihenfolgen, ist hierbei die „Vergleichbarkeit“ von Elementen mit einem bestimmten Element. Java hat hierfür die „Standard-Schnittstelle“ „Comparable“ definiert. Eine Klasse, die die Schnittstelle Comparable implementiert muss die Methode „`boolean equals(...)`“ überschreiben. Die Methode gibt „`true`“ zurück, wenn die Elemente gleich sind und sonst „`false`“. Lasse DATENELEMENT Comparable implementieren und die Methode `equals(DATENELEMENT)` überschreiben um den Nutzer nach Elementen in der WARTESCHLANGE suchen zu lassen und ihn auszugeben.

6) Nutze „equals“ um einen KNOTEN gezielt zu suchen und zu enternen.

7) (SCHWER!) Schreibe eine Methode, die deine PATIENTEN alphabetisch sortiert.