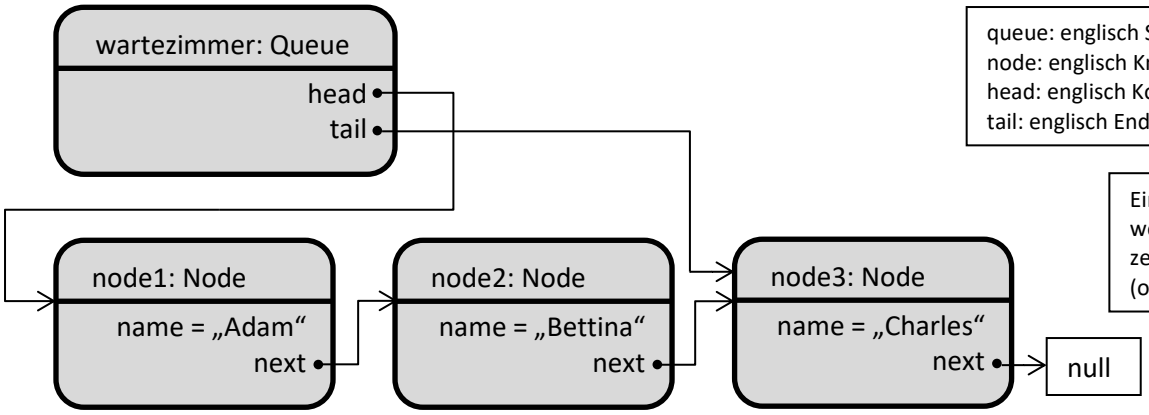
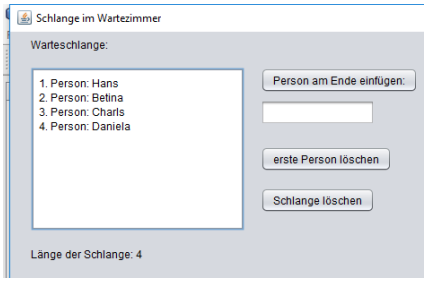
Projekt 7 Die Wartezimmerschlange – mit der Queue unterwegs

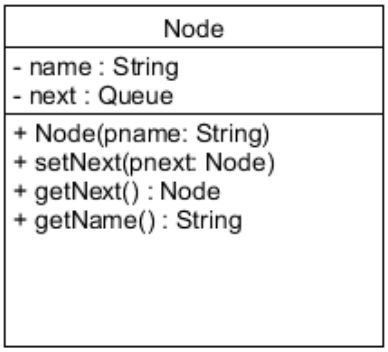
In einem Wartezimmer warten Adam, Betina und Charls auf ihren Arzttermin. Sie kamen in der angegeben Reienfolge. Wenig später kommt Daniela hinzu und Adam wird vom Arzt aufgerufen, er verlässt das Wartezimmer: Das typische an einer Schlange im Wartezimmer ist, das ständig neue Leute hinzukommen. Derjenige, der zuerst kam, darf als erster die Wartezimmerschlange verlassen.
Typisch ist auch, dass man im Voraus nicht weiß, wie lang die Schlange werden wird. Eine Solche Schlange wird in Java von einer sogenannten Queue (Englisch Schlange, sprich „kiuk“) realisiert. Man spricht vom First-In-First-Out-Prinzip (Fifo-Prinzip): Wer zuerst kommt, darf als erster gehen.

Jede Person wird ein einem sogenannten Node (englisch Knoten) gespeichert. Der Knoten beinhaltet sowohl den Namen der Person, als auch die Information, wer nach dieser Person dran ist. Die Queue selbst merkt sich zwei besondere Knoten: den ersten Knoten (head) und den letzten Knoten (tail).



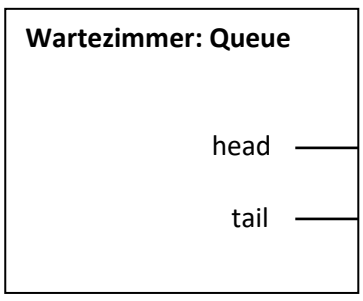
Aufgabe 1: Kopiere das Projekt Wartezimmer-schlange aus dem Schülerordner in deinen persönlichen Ordner und öffne das Projekt in Netbeans. Du siehst in der Klasse Verwaltung eine fertige Oberfläche, mit der man die Wartezimmerschlange in einer Arztpraxis führen kann. Implementiere als erstes die Klasse **Node** nach folgendem Klassendiagramm:



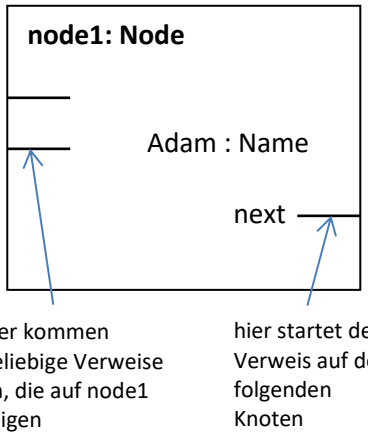


Die Klasse **Node**, in der später eine einzelne Person gespeichert werden kann, hat zwei Attribute: Den Namen der Person und den Verweis auf den nächsten Knoten der Schlange, dieser ist wieder vom Datentyp Node.
Mit dem Konstruktor von **Node** muss man einen Knoten anlegen können, dabei soll der übergebene Namen pname gespeichert werden, außerdem soll der Verweis auf den nächsten Knoten auf keinen weiteren Knoten zeigen, diesen Zustand nennt man **null** (**next = null**). Der Konstruktor der Klase **Node** ist bereits fertig.
Mit einer set-Methode soll man den Nachfolgeknoten zuweisen können, hier reicht die Zeile next = pnext im Code aus.
Mit zwei weiteren get-Methoden soll man den Nachfolgeknoten und den gespeicherten Namen des Patienten abrufen können.

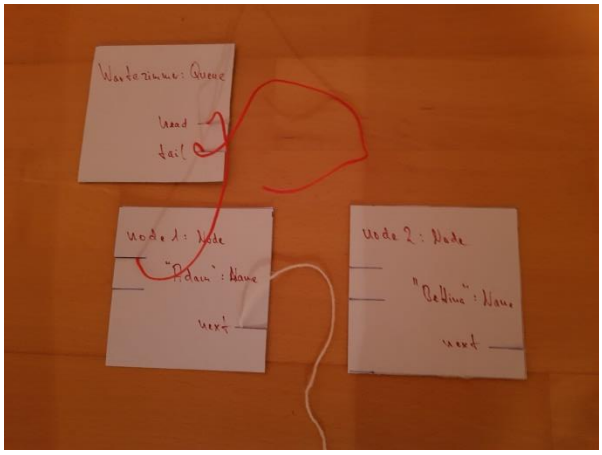
Aufgabe 2: Wenn du das Projekt jetzt startest, sollte der erste Name angezeigt werden, dies ist in der Klasse **Queue**, die die die Schlange verwaltet, bereits implementiert. Jetzt müssen wir uns überlegen, wie man weitere Elemente in eine Schlange einfügen kann. Schneide dazu vier Pappstücke aus den bereitliegenden Pappen aus. Jede Pappe soll quadratisch mit der Seitenlänge 10 cm sein. Die erste Pappe stellt die Queue Wartezimmer dar, beschrifte sie wie rechts angegeben. Die beiden waagerechten Striche rechts von **head** und **tail** sind ca. 2 cm lang. Schneide die Pappe entlang dieser Striche mit der Schere ein, hier werden später Fäden eingezogen, die die Verweise zu den verschiedenen Knoten darstellen.



Anschließend beschriftest du die anderen drei quadratischen Pappen als Knoten wie rechts dargestellt. Die beiden linken Striche dienen zum Aufnehmen der Fäden, die auf diesen Knoten verweisen. Der rechte Strich dient dazu, den Verweis auf den folgenden Knoten zu setzen. Alle drei Striche sind wieder ca. 2 cm lang, schneide dort die Pappe jeweils ein. Besorge dir außerdem verschiedene Fäden: Zwei rote Fäden (ca. 20 cm lang) , die die Verweise von **head** und **tail** der Wartezimmer-schlange darstellen.
Drei weitere weiße Fäden (ca. 15 cm lang) die die Verweise von Knoten zu Knoten darstellen. Lege anschließend auf deinem Tisch die Wartezimmerschlange, in der Adam und Betina im Wartezimmer warten. Adam kam vor Betina. Auf dem Foto siehst du den Verweis von **head** auf den ersten Knoten. Die Verweise von **tail** und der Verweis vom ersten zum zweiten Knoten fehlen noch. Die Fäden werden einfach durch die durchgeschnittenen Linien gesteckt und halten darin (s. Foto rechts neben Aufgabe 3).



Aufgabe 3: Jetzt soll in die Warteschlange der neue Patient Charles am Ende eingefügt werden. Zunächst erzeugst du dazu den dritten Knoten node3. Schreibe danach möglichst konkret auf, welche Verweise neu eingefügt bzw. verändert werden müssen. Man kann immer nur auf **head**, **tail** und den neuen Knoten **node3** zugreifen. Wichtig ist folgende Regel: Zu keinem Zeitpunkt darf ein Knoten in der Luft hängen, d. h. wenn man die Schlange vom Tisch hochheben würde, müssen alle Knoten an der Schlange hängen.



Zusatzaufgabe 1: Die Methode **enqueue (pname: String)** soll eine weitere Person in die Wartezimmerschlange aufnehmen können. Überlege, wie man diese Methode ergänzen muss, damit euer Programm tatsächlich die zweite und dritte Person aufnehmen kann. Bisher kann diese Methode nur die erste Person in die Warteschlange aufnehmen.

Aufgabe 4: Implementiere jetzt die Methode **enqueue (pname: String)** wie besprochen: Die Methode soll eine weitere Person in die Wartezimmerschlange aufnehmen können. **Hinweis:** Die Methode **isEmpty()** prüft, ob die Schlange leer ist und gibt ein Wert vom Typ **boolean** zurück: wenn ein Schlange leer ist, zeigt **head** auf **null**, in diesem Fall liefert **isEmpty()** den Wert **true**, ansonsten **false**.

Aufgabe 5: Die Methode **dequeue()** soll das erste Element von der Schlange entfernen. Dies entspricht dem Aufruf des Patienten im Wartezimmer, der schon am längsten da ist. Erkläre die Methode **dequeue()** und unterscheide dabei die folgenden Fälle: a) Vor dem Entfernen gab es mindestens 2 Patienten, b) vor dem Entfernen gab es genau einen Patienten im Wartezimmer c) man versucht einen Patienten zu entfernen, wenn das Wartezimmer schon leer ist.

Aufgabe 6: Erkläre, was die Methode **front()** aus der Klasse **Queue** bewirkt.

Zusatzaufgabe 2: Die Methode **queueSchreiben (Queue queue1)** soll alle Elemente der Schlange **queue1** der Reihe nach in einer **jTextArea queueListe** ausgeben. Gehe davon aus, dass in der Schlange die Elemente Adam, Bettina, Charles und Daniela in dieser Reihenfolge enthalten sind. In welcher Reihenfolge werden die Elemente in die Schlange **queue2** geschrieben? Warum ist eine weitere while-Schleife notwendig, in der die Elemente der Schlange **queue2** zurück in die Schlange **queue1** geschrieben werden?

Zusatzaufgabe 3: Die neue Methode **schlangenLängeBerechnen(Queue queue1)** soll die Anzahl der Elemente berechnen, die in einer Schlange gespeichert sind. Implementiere die Methode. **Hinweis:** Ähnlich wie in der Methode **queueSchreiben (Queue queue1)** muss die Schlange zum Zählen einmal durchlaufen werden, wobei eine zweite Schlange **queue2** aufgebaut wird. Nach dem Zählen muss die alte Schlange wieder aufgebaut werden.