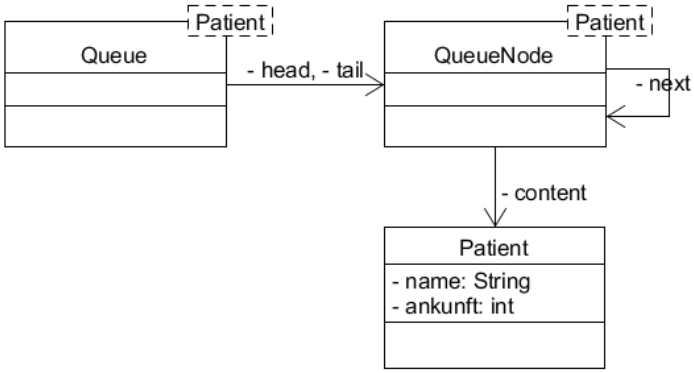
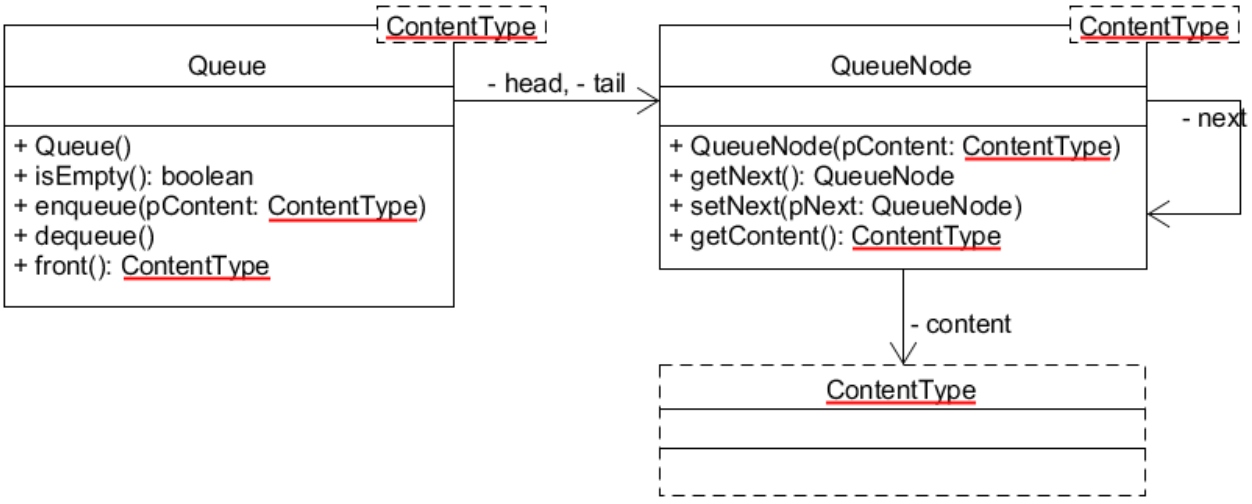
Das Schulministerium stellt für das Zentralabitur Java-Klassen für die Arbeit mit Datenstrukturen (Queue, Stack, Liste und Binärbaum) zur Verfügung.



Das nebenstehende Klassendiagramm stellt eine Warteschlange für Patienten dar:

- Die Warteschlange ist die **Queue** (englisch für Warteschlange)
- Der Anfang heißt **head** (engl. für Kopf), das Ende **tail** (engl. Schwanz).
  Man kann sich die Schlange also bildlich mit Kopf und Schwanz vorstellen.
- Ein einzelner Platz in der Warteschlange heißt **QueueNode** (node ist engl. für „Knoten“).
- Die Referenz eines Knotens auf den nächsten Knoten (seinen Nachfolger) heißt **next**.
- Jeder Knoten hat eine Referenz auf einen Patienten.
  Da man eine Queue generell auch für andere Dinge, z.B. Vokabel-Lernkarten, Netzwerkpakete etc. verwenden kann, wird diese Referenz einfach **content** genannt (engl. für Inhalt).

Das vollständige Klassendiagramm der Queue

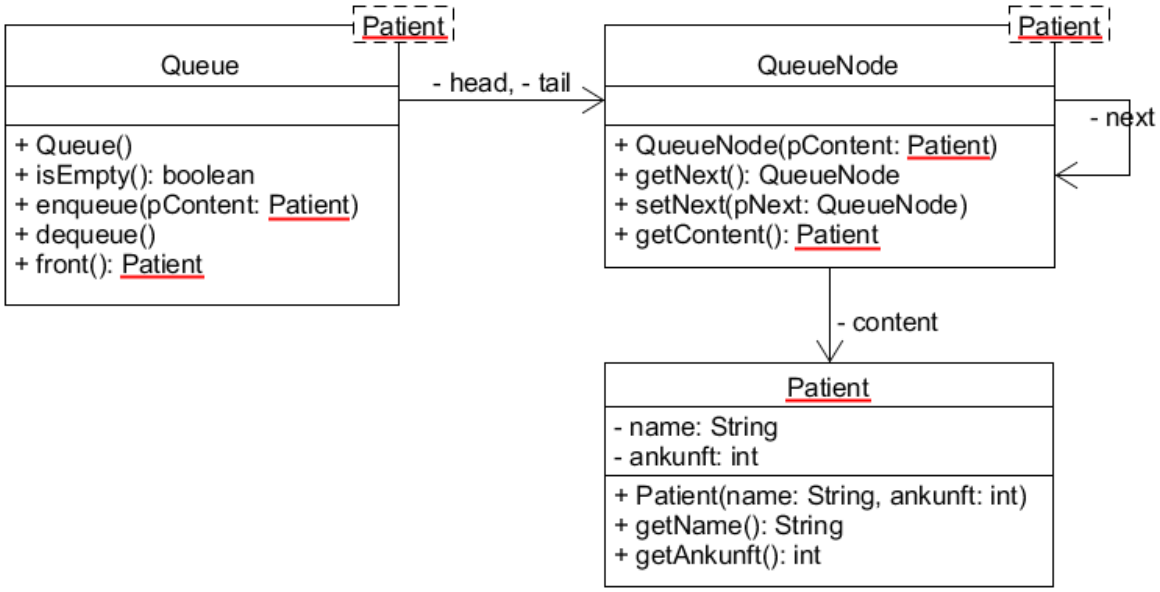


Die NRW-Abitur-Klasse für die Queue ist „**generisch**“. Das bedeutet, dass man sie für beliebige Inhalts-Objekte verwenden kann (z.B. Patienten, Netzwerkpakete, Vokabel-Lernkarten, usw.)

„Generisch“ bedeutet, dass man die Klasse „ContentType“ durch jede beliebige Klasse ersetzen kann. Man deklariert im Quellcode z.B. ein Objekt `Queue<Patient> q1`, und kann `q1` für Patienten-Objekte verwenden. Ein Objekt `Queue<Netzwerkpaket> q2` kann man entsprechend für Netzwerkpakete verwenden.

ContentType ist also ein Platzhalter. Durch die Deklaration `Queue<Patient>` wird „ContentType“ an allen Stellen (im Diagramm rot unterstrichen) durch die Klasse „Patient“ ersetzt.

So sieht das Diagramm aus, wenn man durch die Deklaration `Queue<Patient>` die Klasse „ContentType“ durch die Klasse „Patient“ ersetzt hat: Auch in den Methoden der Queue, z.B. enqueue(pContent: Patient) wird ContentType durch Patient ersetzt.



Die Methoden der Klasse „Queue“ (Beispiel: ContentType = Patient)

Für die Programmierung mit der Queue benötigen wir nur die Methoden der Klasse „Queue“. Die Methoden der Klasse „QueueNode“ sind für uns nicht wichtig, sie werden nur für die Implementierung der Klasse Queue benötigt.

- **`Queue()`** (Konstruktor)
  Erzeugt eine leere Warteschlange. head und tail sind null.
- **`boolean isEmpty()`**
  Gibt true zurück, falls die Queue leer ist.
  Die Queue ist nicht leer, falls sie mindestens ein Inhaltsobjekt (hier: einen Patienten) enthält.
- **`void enqueue(Patient pContent)`**
  Ein Patient-Objekt wird ans Ende der Queue eingefügt. Dabei wird ein neues QueueNode-Objekt erzeugt und an den bisher letzten QueueNode angehängt.
  Die Referenz „tail“ wird auf den gerade neu eingefügten QueueNode verschoben.
  (enqueue ist englisch für „anstellen“)
- **`void dequeue()`**
  Das Patient-Objekt am Anfang der Schlange wird entfernt.
  Die Referen „head“ wird auf den nächsten QueueNode in der Reihe verschoben
  (falls es nur diesen einen Patienten in der Schlange gab, sind head und tail anschließend null).
  Falls es keine Referenz mehr auf den entfernten Patienten gibt, wird das Objekt gelöscht.
- **`Patient front()`**
  Gibt eine Referenz auf das Patienten-Objekt am Anfang der Schlange zurück.
  D.h. man kann die Queue jederzeit „fragen“, wer gerade am Anfang steht
  (bevor man diesen Patient dann mit dequeue aus der Queue entfernt).

Aufgabe

Michel Platini, Jogi Löw und Franz Beckenbauer gehen (in dieser Reihenfolge) zum Arzt.
Die Situation wird durch ein Programm simuliert, in dem eine Queue verwendet wird.

```
01  public class WartezimmerSimulation
02  {
03     private Queue<Patient> queue;
04
05     public WartezimmerSimulation()
06     {
07        queue = new Queue();                          // 1. Diagramm
08     }
09
10     public void main()
11     {
12        Patient p;
13
14        // Zwei neue Patienten stellen sich an
15        p = new Patient("Michel Platini", 1);
16        queue.enqueue(p);
17        p = new Patient("Jogi Löw", 2);
18        queue.enqueue(p);                             // 2. Diagramm
19
20        // Der Patient am Anfang der Schlange wird behandelt
21        if (!queue.isEmpty())
22        {
23           p = queue.front();
24           queue.dequeue();
25           Console.println(p.getName() + " wird jetzt behandelt!");
26        }
27
28        // Ein weiterer Patient stellt sich an
29        p = new Patient("Franz Beckenbauer", 3);
30        queue.enqueue(p);
31
32        // Der Patient am Anfang der Schlange wird behandelt
33        if (!queue.isEmpty())
34        {
35           p = queue.front();
36           queue.dequeue();                           // 3. Diagramm
37           Console.println(p.getName() + " wird jetzt behandelt!");
38        }
39     }
40  }
```

a) Zeichne jeweils ein Objektdiagramm (also insgesamt 3 Stück) aller Objekte, und zwar
1. nach Zeile 07, 2. nach Zeile 18, 3. nach Zeile 36

b) Nach dem Aufruf von dequeue() in Zeile 24 wird das Objekt für den ersten Patienten nicht gleich gelöscht. Warum?

c) Gib an, welche Ausgabe das Programm auf der Konsole macht.