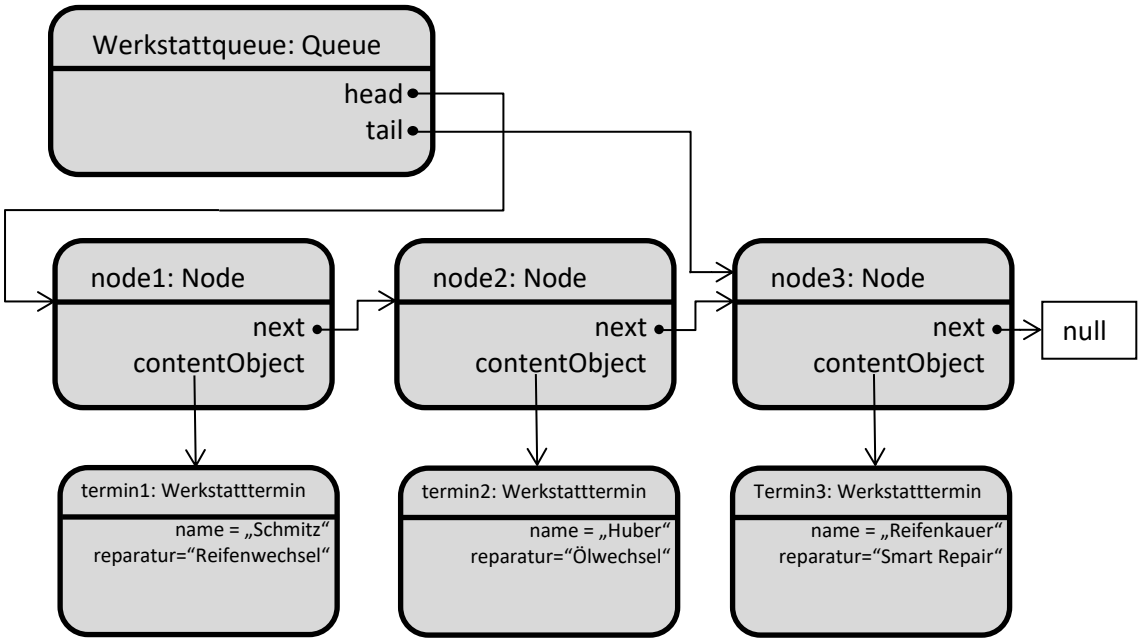
Projekt 8 Die Werkstattschlange – mit mehr Daten unterwegs

In einr Werkstatt werden die Kudenaufträge nacheinander abgearbeitet. Für jeden Kundenauftrag soll neben dem Namen auch die Art des Auftrags gespeichert werden, z. B. Reifenwechsel, Ölwechsel usw. Für diese Anwendung müssen also pro Termin jetzt zwei Strings gespeichert werden (Name und Reparaturart). Es wäre viel zu aufwendig, wenn man jetzt für jede neue Anwendung mit anderen Daten, die gespeichert werden müssen, jeweils eine neue Klasse Queue und Node schreiben müsste. Statt dessen benutzt man sogenannte generische Klassen: Die generische Klasse **Queue <ContentType>** funktioniert mit jedem beliebigen Datentyp, dieser Datentyp wird **ContentType** genannt: Dies kann ein String, eine Integervariable oder eine Kombination aus mehrern Variablen sein. In unserem Fall benutzen wir eine neue Klasse Werkstatttermin, die die beiden Attribute **name** und **reparatur** als Attribute für jeden Termin verwaltet. Die Schlange **Queue <ContentType>** arbeitet dann mit dem **ContentType Werkstatttermin**. Jeder Knoten hat dann einen Verweis auf ein Objekt vom Typ **Werkstatttermin**:



Aufgabe 1: Kopiere das Projekt Werkstatttermin aus dem Schülerordner in deinen persönlichen Ordner und öffen das Projekt in Netbeans. Implementiere als erstes die Klasse **Werkstatttermin** fertig mit ihren beiden Attributen **name** und **reparatur** vom Datentyp **String**. Man muss mit dem Konstruktor einen Werkstatttermin anlegen können, außerdem sollte man mit zwei get-Methoden die beiden Attribute abfragen können.

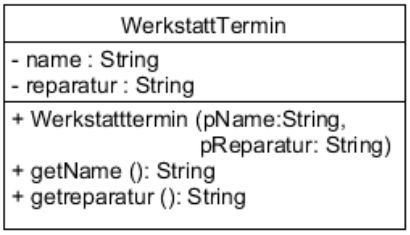


Abbildung 1

Aufgabe 2: Die Klasse **Queue <ContentType>** ist bereits ferig vorgegeben. Sie enthält die private innere Klasse **Queue**, mit der die Knoten verwaltet werden, eine separate Klasse Queue wie im letzten Projekt ist somit nicht mehr notwendig. Die Klasse **Queue** stellt die auf der folgenden Seite angebeben Methoden (Abb. 3) zur Verfügung. Ergänze die Klasse Verwaltung so, dass beim Start des Programms vier Werkstatttermine wie angegeben angezeigt werden. Dazu sind zunächst in der Klasse Verwaltung folgende Deklarationen notwendig und bereits vorhanden:

```
Queue<WerkstattTermin> queue1;
WerkstattTermin termin1, termin2, termin3, termin4;
```

Die Schlange **queue1** sowie vier Werkstatttermine werden deklariert. a) Im Konstruktor der Klasse Verwaltung musst du als nächstes die Schlange sowie die vier Termine erzeugen, wie rechts angegeben:

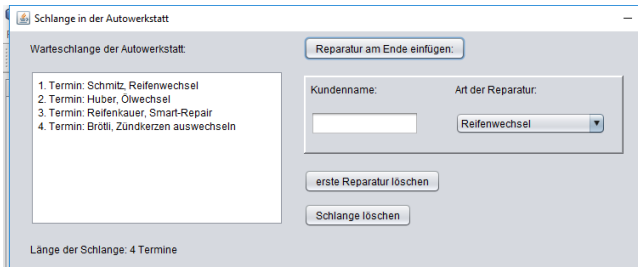


Abbildung 2

Instanzierung der zunächst leeren Schlange und der WerkstattTermine im Konstruktor von Verwaltung:

```
termin1 = new WerkstattTermin ("Schmitz","Reifenwechsel");
queue1 = new Queue<WerkstattTermin>();
```

Die anderen drei Termine musst du analog im Konstruktor von Verwaltung erzeugen.

b) Anschließend musst du die vier Werksttermine in die Schlange nacheinander am Ende einfügen. Auf der rechten Seite (Abb. 3) sind alle Methoden aufgelistet, die du aus der Klasse Queue<ConentType> benutzen kannst. Benutze viermal die entsprechende Methode **enqueue**, um die vier Termine am Ende jeweils anzuhängen. c) Am Ende des Konstruktors kannst du die Methode `queueSchreiben(queue1);` aufrufen. Die Methode ist bereits implementiert und stellt die gesamte Schlange in der **JTextarea queueArea** dar. Damit die Methode funktioniert muss im Konstruktor der Klasse Verwaltung zusätzlich die Schlange **queue2** als **Queue<WerkstattTermin>** analog zu **queue1** instanziert werden. Nach dem Start des Projektes sollte der Bildschirm jetzt aussehen, wie in der Abbildung auf der letzten Seite rechts dargestellt.	**Zur Verfügung stehende Methoden der generischen Klasse Queue<ContentType>:** Konstruktor **Queue()** Eine leere Schlange wird erzeugt. Objekte, die in dieser Schlange verwaltet werden, müssen vom Typ **ContentType** sein. Anfrage **boolean isEmpty()** Die Anfrage liefert den Wert true, wenn die Schlange keine Objekte enthält, sonst liefert sie den Wert false. Auftrag **void enqueue(ContentType pContent)** Das Objekt pContent wird an die Schlange angehängt. Falls pContent gleich null ist, bleibt die Schlange unverändert. Auftrag **void dequeue()** Das erste Objekt wird aus der Schlange entfernt. Falls die Schlange leer ist, wird sie nicht verändert. Anfrage **ContentType front()** Die Anfrage liefert das erste Objekt der Schlange. Die Schlange bleibt unverändert. Falls die Schlange leer ist, wird null zurückgegeben. Abbildung 3

Aufgabe 3: Implementiere die folgenden drei Methoden:

Reparartur am Ende einfügen	Innerhalb der schon existierenden Event-Methode **personEinfügenButMouseClicked** soll eine weitere Reparatur an das Ende der Schlange eingefügt werden können. Über die Variable **namenEingabe** wird der Name bereitgestellt, der in der **JTextField namenBox** eingegeben wurde. Über die Variable **aktuelleReparatur** wird ebenfalls als String der Reparaturauftrag bereitgestellt, dieser wurde bereits über die Event-Methode der **JComboBox reparaturCombo** ausgewertet.
erste Reparatur löschen	Innerhalb der schon existierenden Event-Methode **personLöschenButMouseClicked** soll der erste anstehende Reparaturauftrag aus der Schlange gelöscht werden können.
Schlange löschen	Innerhalb der schon existierenden Event-Methode **schlangeLöschenButMouseClicked** soll die gesamte Schlange gelöscht werden können. **Tipp:** Benutze eine while-Schleife : **while (! queue1.isEmpty())** …

Aufgabe 4: Schau dir die Methode **queueSchreiben (Queue<WerkstattTermin> queue1)** an und übersetze diesen innerhalb eines Worddokumentes „queueSchreiben“ in einen Pseudocode mit möglichst verständlicher Sprache. Erkläre zum Schluss, warum die einzelnen Knoten aus der Schlange beim Schreiben gelöscht werden müssen und wie die Schlange zum Schluss wieder hergestellt wird.

Zusatzaufgabe 1: Implementiere eine Methode **queueLänge (Queue<WerkstattTermin> queue1**), die die Länge einer Schlange berechnet, d. h. die Zahl der Termine in der Schlange. Die Methode soll am Ende der Methode **queueSchreiben** aufgerufen werden, das Ergebnis in einem Label angezeigt werden.

Zusatzaufgabe 2: In zwei weiteren Labels soll angezeigt werden, wie viele Termine mit Ölwechsel bzw. Reifenwechsel stattfinden.