Prof. Dr. Gerd Stumme, Folke Eisterlehner, Dominik Benz
Fachgebiet Wissensverarbeitung

08.06.2009

8. Hausübung „Algorithmen und Datenstrukturen“

Sommersemester 2009

Abgabetermin: Montag, 15.06.2009, 10:00 Uhr

1 Queue als Liste

In der Vorlesung wurde der ADT *Queue* vorgestellt, der eine First-in-First-Out Warteschlange für eine dynamisch veränderliche Anzahl von Daten beschreibt. Dieser besitzt die zwei Methoden

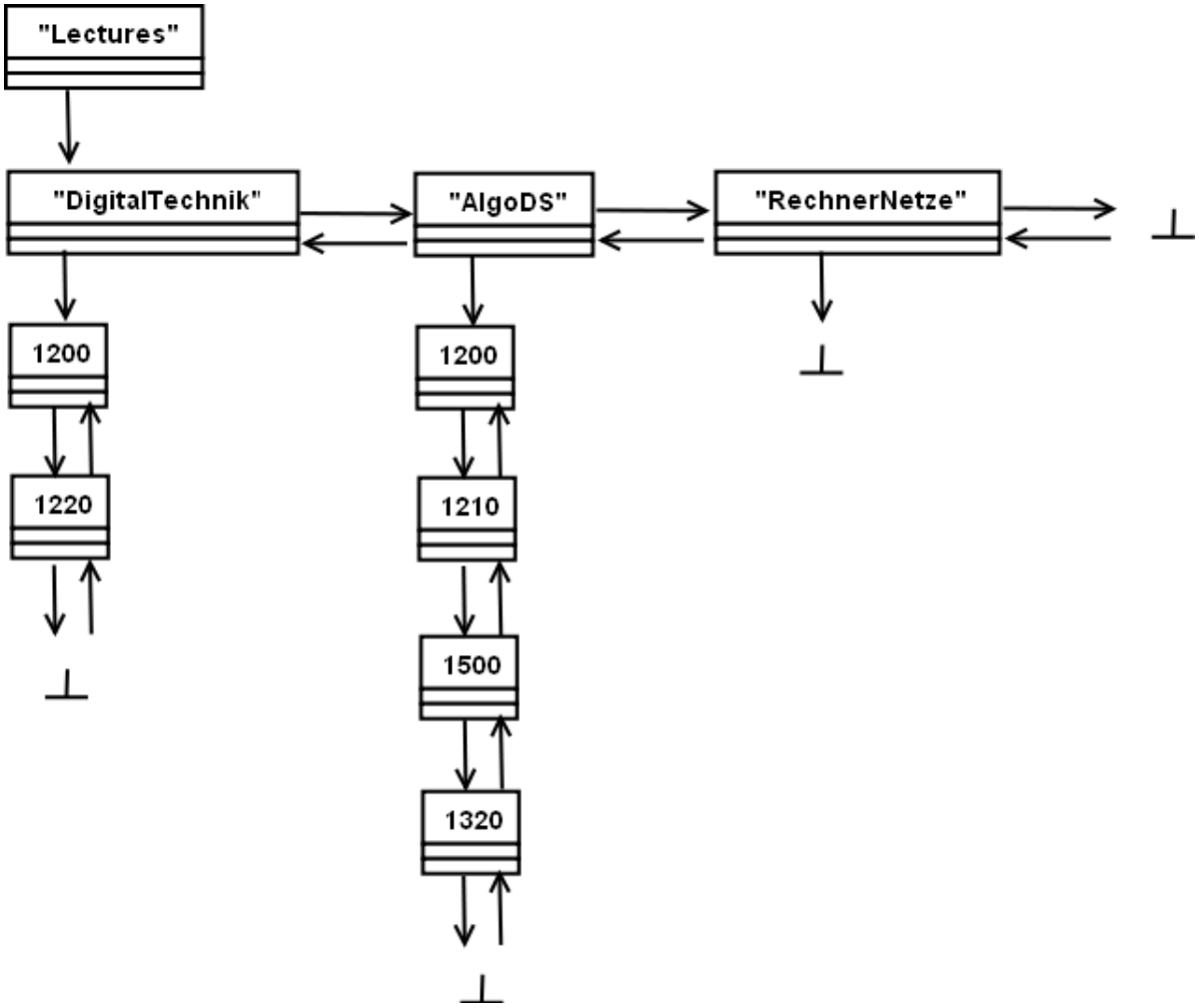
- **`enqueue`** zum Einfügen eines Elements am Ende der Warteschlange und
- **`dequeue`** zum Auslesen und Entfernen des ersten Elements.

Formulieren Sie in Pseudocode eine Implementierung dieses ADT, basierend auf einer einfach verketteten Liste wie sie auf den Folien 4-26ff beschrieben ist. Ihre Implementierung soll den Konstruktor der Queue beinhalten, der eine leere Warteschlange anlegt, sowie die beiden Methoden **`void enqueue(Object o)`** und **`Object dequeue()`**. Im Falle einer leeren Liste soll **`enqueue`** *null* zurückliefern.

(7 Punkte)

2 Listen

Datenstruktur “Lectures” Schreiben Sie ein Java-Programm, das eine Datenstruktur der folgenden Form abspeichert:



Die Skizze zeigt drei "Vorlesungen" mit ihren Teilnehmern:

- Die Vorlesung Digitaltechnik wird von den Studenten mit den Matrikelnummern 1200 und 1220 besucht.
- Die Vorlesung AlgoDS wird von den Studenten mit den Matrikelnummern 1200, 1210, 1500 und 1320 besucht.
- Die Vorlesung Rechnernetze hat keine Teilnehmer.

Die Gesamt-Datenstruktur **`Lectures`** fasst die Vorlesungen zusammen. Programmieren Sie für Ihre Lösung drei Klassen:

- **`LectureNode`**: Knoten, in denen der Name der Vorlesung, ein Verweis auf die Teilnehmerliste, ein **`next`**- und ein **`previous`**-Verweis abgespeichert sind,
- **`StudentNode`**: Knoten, in denen der Name eines Studenten, ein **`next`**- und ein **`previous`**-Verweis abgespeichert sind, und
- **`Lectures`**: die Gesamt-Datenstruktur.

Die Klasse **Lectures** soll drei Operationen unterstützen:

- **void insertLecture(String lecture)**: Anlegen eines neuen Objekts der Klasse **Lecture** mit Namen *lecture* und Einfügen in die Datenstruktur.
- **void insertStudent(Integer student, String lecture)**: Anlegen eines neuen Objekts der Klasse **StudentNode** mit Namen *student* und Einfügen in die Liste der Teilnehmer von *lecture*.
- **void listParticipantOverlap(String lecture1, String lecture2)**: Berechnen der Schnittmenge der Teilnehmer zweier Vorlesungen *lecture1* und *lecture2*. Es sollen die Matrikelnummern der Studenten auf dem Bildschirm ausgegeben werden, die an beiden Kursen teilnehmen.
- **void listAllCourses()**: Ausgabe aller Vorlesungen sowie ihrer Teilnehmer auf dem Bildschirm (siehe unten).

Zur Implementierung der Methoden müssen Sie Konstrukturen, **get**/**set**- und weitere Hilfsmethoden in die Klassen **LectureNode** und **StudentNode** einfügen. Sie müssen nur diejenigen Methoden und Konstruktoren programmieren, die in Ihrem Programm tatsächlich verwendet werden. Die Behandlung von Fehlern ist hierbei nicht gefordert, d.h. Sie dürfen ohne Überprüfung davon ausgehen, dass es beim Einfügen noch keine gleichnamigen Vorlesungen oder Studenten gibt. Ob das Einfügen am Anfang oder am Ende der Liste erfolgt, bleibt Ihnen überlassen. Zum Vergleich von Zeichenketten können Sie die Methode **equals** verwenden.

Nutzen Sie zur Implementierung verschachtelte Klassen. Das Paket **java.util** darf *nicht* verwendet werden. Testen Sie ihr Programm mit Hilfe der Klasse **LectureTest.java** von der Vorlesungswebseite. Die Ausgabe des Aufrufs der Methode **listAllCourses** Ihres Programms soll wie folgt aussehen (die Ausgabereihenfolge der Vorlesungen / Teilnehmer ist hierbei egal):

```
Teilnehmer der Vorlesung Digitaltechnik:
  * 1200
  * 1220
Teilnehmer der Vorlesung AlgoDS:
  * 1200
  * 1210
  * 1500
  * 1320
Teilnehmer der Vorlesung Rechnernetze:
  * (keine Teilnehmer vorhanden)
```

Der Aufruf der Methode **listParticipantOverlap('DigitalTechnik', 'AlgoDS')** soll folgende Ausgabe erzeugen:

```
Teilnehmer der Vorlesungen Digitaltechnik und AlgoDS:
  * 1200
```

(25 Punkte)

Aufbau der Datenstrukur: Beschreiben Sie den inneren Aufbau folgender Datenstruktur:

```
LinkedList<Integer>[] testList;
```

Sie können hierzu eine ähnliche Skizze wie in voriger Aufgabe verwenden.

(5 Punkte)

Viel Spaß und viel Erfolg!