Algorithmen und Datenstrukturen
Prof. Claudia Fohry
M.Sc. Marco Bungart
Paul Plock

Universität Kassel
Sommersemester 2017
Abgabe: 03.07.2017 8:00 Uhr

Aufgabenblatt 9

Im Nachfolgenden werden Sie zwei Datenstrukturen, einen abstrakten Datentyp und zwei Realisierungen dieses abstrakten Datentyps implementieren. In moodle stehen die Dateien `DoubleLinkedListTest.java`, `RingBufferTest.java` und `DequeTest.java` bereit, welche die Datenstrukturen sowie die Realisierungen des abstrakten Datentyps rudimentär testen. Sie finden zu den jeweiligen Tests auch die erwarteten Ausgaben in moodle.

a) `DoubleLinkedList.java`, `DoubleLinkedNode.java`: Sie haben in der Vorlesung die Liste als Datenstruktur kennengelernt. Implementieren Sie eine unsortierte, doppelt verkettete Liste. Hierbei soll die Liste `double`-Werte speichern und folgende Operationen unterstützen:

- `public void addFirst(double value)`, `public void addLast(double value)`: fügt einen `double`-Wert am Anfang/Ende der Liste hinzu.
- `public double removeFirst() throws EmptyListException`, `public double removeLast() throws EmptyListException`: Entfernt das erste/letzte Element aus der Liste und gibt dieses zurück. Werden diese Methoden auf einer leeren Liste ausgeführt, wird eine `EmptyListException` geworfen.
- `public int size()`: Gibt die Anzahl der in der Liste gespeicherten Elemente zurück.
- `public boolean isEmpty()`: Gibt `true` zurück, wenn die Liste leer ist und sonst `false`.

Es bleibt Ihnen überlassen, ob Sie die Liste mit oder ohne Sentinel implementieren. Verwenden Sie bei der Implementierung und Verwendung der `DoubleLinkedNode`-Klasse Datenkapselung.

Hinweis: Es bietet sich an, neben dem Verweis `head` auf das erste Element auch einen Verweis `tail` auf das letzte Element anzulegen.

Sie dürfen neben den geforderten Methoden beliebige Helfermethoden zur Klasse `DoubleLinkedList` hinzufügen. Es ist Ihnen überlassen, wie Sie die Klasse `DoubleLinkedNode` aufbauen, solange diese eine doppelte Verkettung der Knoten realisiert.

In moodle finden Sie eine Vorgabe für die `EmptyListException`, welche von der Klasse `RuntimeException` erbt. Durch diese Vererbungsbeziehung werden Sie nicht gezwungen, jeden Aufruf der `remove...(...)`-Methoden mit einem `try-catch`-Block zu umstellen.

(10 Punkte)

b) `RingBuffer.java`: Sie haben neben der Liste auch den Ringspeicher als Datenstruktur kennen gelernt. Implementieren Sie einen Ringspeicher. Verwenden Sie zum Speichern der Daten ein `double`-Array. Der Ringspeicher soll dieselben Operationen unterstützen wie die Liste aus Aufgabe a), anstelle einer `EmptyListException` soll jedoch eine `EmptyRingBufferException` geworfen werden. Die Datei `EmptyRingBufferException.java` finden Sie in moodle zum Download.

Wenn das `double`-Array voll ist, so soll die Größe des Arrays automatisch verdoppelt werden. Hierzu legen Sie ein neues Array doppelter Größe an und kopieren die Daten vom alten in das neue Array. Es bleibt Ihnen überlassen, ob sie vergrößern, wenn das letzte freie Element belegt wurde oder erst, wenn ein Element in einen vollen Ringspeicher hinzugefügt werden soll.

Hinweis: Achten Sie beim Vergrößern darauf, dass wenn die Daten im `double`-Array „über den Rand des Arrays hinauslaufen“ (Folie 99, Beispiel unten rechts), Sie die Werte ggf. verschieben müssen, damit die Reihenfolge der gespeicherten Elemente beibehalten bleibt.

(15 Punkte)

c) `Deque.java`, `DoubleLinkedListDeque.java`, `RingBufferDeque.java`: Der abstrakte Datentyp der Deque (Kurzform für „double-ended Queue“, ausgesprochen „Deck“) ist eine Erweiterung des abstrakten Datentyps Queue, welche das Einfügen und Entfernen von Elementen an beiden Enden der Queue zulässt. Für diese Aufgabe soll eine Deque folgende Operationen unterstützen:

- `public void addFirst(double value)`, `public void addLast(double value)`: Fügt einen `double`-Wert am Anfang/Ende der Deque ein.
- `public void addAllFirst(double[] values)`, `public void addAllLast(double[] value)`: Fügt alle Werte des Arrays am Anfang/Ende der Deque ein. Hierbei bleibt die Reihenfolge der Elemente im Array `values` erhalten.
- `public double removeFirst()`, `public double removeLast()`: Entfernt das erste/letzte Element aus der Deque und gibt dieses zurück.
- `public int size()`: Gibt die Anzahl der in der Deque gespeicherten Elemente zurück.
- `public boolean isEmpty()`: Gibt `true` zurück, wenn die Deque leer ist und sonst `false`.

Erstellen Sie das Interface `Deque`, sodass dieses die oben beschriebene Deque abbildet. Schreiben Sie dann die Realisierungen `DoubleLinkedListDeque` und `RingBufferDeque` dieses abstrakten Datentyps. Nutzen Sie hierzu die in Aufgabe a) und b) entwickelten Datenstrukturen.

(15 Punkte)

Abgabetermin: Die Lösungen sind bis spätestens 03.07.2017 um 8:00 Uhr über das elektronische Abgabesystem einzureichen. Form und Anforderungen sind die gleichen wie bisher (s. Blatt 1).