Vorgehensweise und Entscheidungsfindung

Wir haben unser Deque an Hand von JUnit-Testfällen erstellt.
Anschließend haben wir, wie in a03 bereits, eine Klasse DequeScenario erstellt.
Hier lassen wir wieder unsere Implementierungen gegen die des JDK antreten.

Die Methode doBenchmark() ist überladen, so dass wir einmal unser selbst entwickeltes Deque testen können und dann das aus dem JDK.

Beispielhafte Ausgabe auf unserem Testsystem:

```
*****************************************
Test für Liste: a04.Deque mit a01.LinkedList
addFirst(Element): Zeit: 15ms
peekFirst(): Zeit: 10ms
removeFirst(): Zeit: 12ms
pollFirst(): Zeit: 11ms
addLast(Element): Zeit: 9ms
peekLast(): Zeit: 7ms
removeLast(): Zeit: 7ms
pollLast(): Zeit: 4ms
*****************************************
Test für Liste: a04.Deque mit a02.ArrayList
addFirst(Element): Zeit: 1793ms
peekFirst(): Zeit: 6ms
removeFirst(): Zeit: 1417ms
pollFirst(): Zeit: 1456ms
addLast(Element): Zeit: 9ms
peekLast(): Zeit: 7ms
removeLast(): Zeit: 8ms
pollLast(): Zeit: 1367ms
*****************************************
Test für Liste: LinkedBlockingDeque
addFirst(Element): Zeit: 29ms
peekFirst(): Zeit: 8ms
removeFirst(): Zeit: 14ms
pollFirst(): Zeit: 5ms
addLast(Element): Zeit: 9ms
peekLast(): Zeit: 8ms
removeLast(): Zeit: 15ms
pollLast(): Zeit: 9ms
*****************************************
Test für Liste: ArrayDeque
addFirst(Element): Zeit: 6ms
peekFirst(): Zeit: 5ms
```

Martin Slowikowski
Matrikelnummer: 199 91 66

Jan-Tristan Rudat
Matrikelnummer: 200 78 52

removeFirst(): Zeit: 6ms
pollFirst(): Zeit: 4ms
addLast(Element): Zeit: 7ms
peekLast(): Zeit: 5ms
removeLast(): Zeit: 7ms
pollLast(): Zeit: 5ms

Erklärungen

Die Klasse Deque kapselt eine Liste und bietet alle Methoden aus Liste als Indirektion an, sowie alle weiteren die für die Deque-Funktionalität benötigt werden.

- Auffällig ist, dass die ArrayDeque bei removeFirst, addFirst und pollFirst wesentlich schneller ist als unsere ArrayList-Implementierung. Der Grund hierfür ist, dass sich ArrayDeque head und tail der Elemente im Daten-Array merkt und so nicht bei jeder Größenänderung (wie z.B. das Entfernen oder hinzufügen des ersten Elements) teure Kopieroperationen durchführen muss.

- Die Zeitunterschiede bei der Linked-Varianten sind wesentlich geringer. Zu beobachten ist, dass LinkedBlockingDeque bei addFirst() geringfügig länger benötigt. Dies könnte auf das intern verwendete Locking zurückzuführen sein.

Martin Slowikowski
Matrikelnummer: 199 91 66

Jan-Tristan Rudat
Matrikelnummer: 200 78 52