Übungsblatt 7

Aufgabe 1 - Ringpuffer

Stelle Dir die folgende Situation vor: Du bist Tutorin oder Tutor und organisierst eine Klausureinsicht für DAP 1. In den Raum passt nur eine begrenzte Anzahl an Studierenden. Den Studierenden wird in der Reihenfolge ihrer Ankunft ihre Klausur ausgehändigt. Es muss also immer darauf geachtet werden, dass frei werdende Plätze immer wieder neu belegt werden. Diese Organisationsform soll nun durch eine Datenstruktur unterstützt werdenwerden.

Programmiere dafür die Klasse `RingBuffer`, die einen Ringpuffer umsetzen soll. Ein Ringpuffer ist eine Datenstruktur mit einer festen Größe. Die Elemente werden nach dem Prinzip *first-in-first-out* verwaltet.

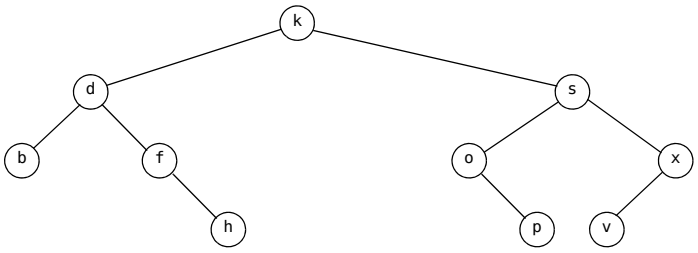
Implementiere dazu folgende Methoden:

- `RingBuffer( int n )` – Der Konstruktor bekommt die Größe `n` übergeben und initialisiert die Datenstruktur.
- `public Student pop()` – Gibt das älteste Element des Ringpuffers zurück und entfernt es.
- `public Student peek()` – Gibt das älteste Element des Ringpuffers nur zurück, ohne es zu entfernen.
- `public void push( Student s )` – Fügt das `Student`-Objekt am Ende des Ringpuffers ein. Ist dieser voll, so soll eine `RuntimeException` geworfen werden.
- `public int size()` – Gibt die Größe des Ringpuffers zurück.
- `public int countElements()` – Gibt die Anzahl der im Ringpuffer befindlichen Objekte zurück.

Aufgabe 2 - Tiefe eines CharacterSearchTrees

Erweitere die Klasse `CharacterSearchTree` um die Methode `public int height()`, die die Höhe des Baums liefert (siehe auch Praktikumsblatt 5). Die Höhe des Baums ist die Anzahl der Knoten auf dem längsten möglichen Weg von der Wurzel zu einem Blatt. Ein leerer Baum hat die Höhe `0`. Ein Baum, der nur aus der Wurzel besteht, hat die Höhe `1`.

Aufgabe 3 - Binärer Suchbaum – Konzeption



- Gebe die Wurzel an:
- Gebe alle Blätter an:
- Füge genau in der gegeben Reihenfolge die folgenden Werte in den Baum ein: `z, c, n, m, e, y`

Aufgabe 4 - Breite eines CharacterSearchTrees

Erweitere die Klasse `CharacterSearchTree` folgende Methoden:

- `public int[] levelWidths()` – Legt ein ausreichend großes neues Array an und ruft die folgende Methode auf. `private void levelWidths( int[] widths, int level)` – Bestimmt rekursiv die Breiten aller Ebenen des Baumes und fügt diese in das Array so ein, dass der Index des Arrays der Tiefe der jeweiligen Ebene entspricht. Die Wurzel liegt auf Ebene `0`.
- `public int maxLevelWidth()` – Gibt die Anzahl der Knoten der Ebene zurück, die die meisten Knoten enthält.

Aufgabe 5 - Breitendurchlauf

Passe die Klasse `RingBuffer` aus Aufgabe 1 so an, dass diese `CharacterSearchTree`-Objekte verwalten kann. Implementiere anschließend die Methode `public void breadthFirstTraversal()` in der Klasse `CharacterSearchTree`. Diese Methode durchläuft den Baum ebenenweise und gibt die besuchten Knoten in dieser Reihenfolge aus. Verwende in der Lösung die Klasse `RingBuffer`. Überlege Dir, wie hierfür eine geeignete minimale Größe bestimmt werden kann.

Beispiel:
Der Breitendurchlauf durchläuft die Knoten des abgebildeten Baumes in folgender Reihenfolge:
`m, f, p, c, k, x, a, h, s`