Übung 4: Traversierung & Balancieren

Abgabetermin: 04.04.2017

Name: **Matrikelnummer:**

Gruppe: ☐ **G1** Di 10:15-11:00 ☐ **G2** Di 11:00-11:45 ☐ **G3** Di 12:45-13:30

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Punkte
Aufgabe 1	6	☐	Java-Programm	Projekt Archiv	☐	
Aufgabe 2	12	☐	Java-Programm	Projekt Archiv	☐	
Aufgabe 3	6	☐	Zeichnung nach jedem Schritt	Zeichnung nach jedem Schritt	☐	

Aufgabe 1: Traversieren eines Binärbaums (6 Punkte)

Implementieren Sie verschiedene Binärbaum Traversierungen in der Klasse *BinaryTreeUtil*. Die Klasse definiert Methoden zur **PreOrder, InOrder & PostOrder Traversierung** eines Binärbaums. Diese Methoden bekommen als Parameter ein *BinaryTreeSet (*aus der Übung 3) und bauen ein **int[]** Array auf, dass die Werte der Knoten in ihrerer Traversierungsreihenfolge enthält. Testen Sie Ihre Implementierung am Beispiel Baum aus dem Anhang.

Die Methoden liefern für den Beispiel Baum (siehe Anhang) folgende Ergebnisse:

- `InOrder : [0, 1, 3, 4, 5, 6, 7, 8]`
- `PreOrder : [4, 1, 0, 3, 5, 6, 7, 8]`
- `PostOrder : [0, 3, 1, 8, 7, 6, 5, 4]`

Abzugeben ist: Projekt Archiv

Aufgabe 2: Aufbauen eines Binärebaums mit Hilfe gegebener Traversierung (12 Punkte)

Implementieren Sie die Methode ***buildTree()*** in der Klasse *BinaryTreeBuilder* die aus einer gegebenen **InOrder** und **PreOrder Traversierung** einen allgemeinen (nicht zwingendermaßen sortierten) Binärbaum aufbaut. ***buildTree*** bekommt die InOrder- und PreOrder Traversierung als **int[]** Arrays übergeben. Beachten Sie, dass bei gegebener InOrder- und PreOrder Traversierung der ursprüngliche Binärbaum (der die gegebenen Traversierungsreihenfolgen produziert hat) wieder aufgebaut werden kann.

Abzugeben ist: Projekt Archiv

Aufgabe 3: Wurzel-Balancieren eines Binärebaums (6 Punkte)

Simulieren sie das Balancieren des Beispiel Binärbaums (sie Anhang) unter Zuhilfenahme des in der Vorlesung präsentierten Algorithmus. Starten Sie mit dem angegebenen Baum (Anhang) und führen sie zuerst den Umbau des Baums in eine “Rebe” durch (siehe Vorlesung ***treeToVine***). Skizzieren Sie dabei den Baum nach jeder einzelnen Iteration des präsentierten Algorithmus. Am Ende sollte der Baum eine aufsteigend sortierte lineare Liste repräsentieren die über die **right** Pointer der Baumknoten verbunden sind.

Als nächsten Schritt simulieren Sie die eigentliche Balancierung des Baumes durch schrittweise Balancierung (siehe Vorlesung ***vineToTree***). Geben Sie die notwendigen Berechnungen der Parameter des Algorithmus an, die nötig sind um ***vineToTree*** auszuführen. Insbesondere beachten sie, dass der Baum

unter Umständen nicht vollständig gefüllt ist. Skizzieren Sie hier auch wieder die einzelnen Schritte des Algorithmus nach jeder Rotationsoperation. Am Ende muss der Baum balanciert sein.

Abzugeben ist: Baum Skizze nach jedem Schritt

Implementierungshinweise:

- Verwenden Sie das Vorgabeprojekt **PI2_UE04.zip**.
- Fügen Sie Ihre Implementierung in den mit **TODO** markierten Teilen der Klasse *BinaryTreeUtil* ein.
- Ändern sie **keine public Interfaces** vorgegebener Skeleton Klassen (mit Ausnahme der Sichtbarkeiten).
- Halten Sie sich an die **Codierungsrichtlinien** auf der Kurs Website.

Anhang: Vorgabe Baum