Übungsblatt 6

Ausgabe: 4.6.2020
Abgabe: 17.6.2020 (24:00 Uhr)
Testat: 19.6.2020 (9:45–13:00 Uhr)

Aufgabe 1: AVL-Bäume — 100 Punkte

Im dritten Übungsblatt haben Sie den ADT `BinaryTree` als binären Suchbaum implementiert.

Implementieren sie nun in der Klasse `AVLTreeI` den ADT `AVLTree` (die Eigenschaften kennen sie aus der Vorlesung), mit (fast) denselben Operatoren, die der `BinaryTree` auch zur Verfügung stellt:

`void insert (Comparable val)`	Fügt val in den Baum ein.
`void insert (String filename)`	Fügt die Comparable-Objekte, die in der Datei stehen in den Baum ein.
`boolean contains(Comparable val)`	Testet, ob val im Baum vorhanden ist.
`int size()`	Ermittelt die Anzahl der Knoten im Baum.
`int height()`	Ermittelt die Höhe des Baums.
`Comparable getMax()`	Liefert das größte Element im Baum.
`Comparable getMin()`	Liefert das kleinste Element im Baum.
`void remove (Comparable val)`	Entfernt val aus dem Baum.
`booelan isEmpty()`	true genau dann, wenn der Baum leer ist.
`void clear()`	Entfernt alle Elemente aus dem Baum.
`void printInorder()`	Gibt Baum in Inorder aus.
`void printPostorder()`	Gibt Baum in Postorder aus.
`void printPreorder()`	Gibt Baum in Preorder aus.
`void printLevelorder()`	Gibt Baum in Levelorder aus.
`void saveToFile (String filename)`	Speichert die int-Werte des Baums in der Datei.
`void visualize ()`	Gibt den Baum graphisch aufbereitet aus.

Denken sie daran, dass sie sowohl für das Einfügen als auch für das Löschen von Elementen geeignete Rebalancierungsmaßnahmen (Rotationen) zur Verfügung stellen müssen.

Hinweise:
Einfüge- und Löschoperationen müssen rekursiv arbeiten, damit die Rebalancierung des AVL-Baumes korrekt durchgeführt werden kann.

Beim Löschen eines Elements im AVL-Baum wird das zu löschende Element durch das kleinste Element im rechten Teilbaum des zu löschenden Elements ersetzt.

Zur Rebalancierung: Es gibt vier Rebalancierungsmaßnahmen (L-, LR-, R- und RL-Rotation). Wenn sie die komplizierten Rotationsvarianten (LR und RL) aufteilen in zwei aufeinander folgende Rotationen (LR = L und dann R bzw. RL = R und dann L), müssen sie nur die beiden Rotationen L (Linksrotation) und R (Rechtsrotation) implementieren.

Schwieriger als die Implementierung der Rotation ist es, die Kriterien für die jeweilige Rotationsart zu bestimmen. Überlegen sie genau, wie die Balancefaktoren der einzelnen Knoten eine Rebalancierungsmaßnahme bestimmen und wie sich die Balancefaktoren wiederum durch die Rebalancierung verändern. Beim Einfügen und Löschen sieht das durchaus unterschiedlich aus.

Schreiben sie einen Programmrahmen mit Benutzerschnittstelle. Dort sollen sie verschiedene AVL-Bäume verwalten, für welche sie die gewünschten Operationen aufrufen. Für das Testat soll eine möglichst einfach bzw. intuitiv zu bedienende Schnittstelle bereitgestellt werden.

Die JUnit-Tests werden mit `Song`-Objekten noch hochgeladen.
Beachten Sie auch: Der `AVLTree` bzw. auch seine Operationen werden noch in einem zweiten Schritt erweitert. Das wird auf eine Art geschehen, ohne dass die Programmstruktur verändert werden muss.