Praktische Informatik II SS 2009

Übungsblatt 7

Abgabe: 25.05.2009

Aufgabe 1 Balancierte Bäume

In der Vorlesung habt ihr AVL-Bäume als Möglichkeit kennengelernt, degenerierte Bäume zu vermeiden und so effiziente Suchbäume aufzubauen.

Implementiert eine Klasse

public class AVLTree <T **extends** Comparable<T> >

in Java, welche intern AVL-Bäume zur sortierten Speicherung von generischen Datentypen verwendet.

Stellt in eurer Klasse mindestens die folgenden drei Methoden zur Verfügung:

- **void** insert(T t)

  Fügt Eintrag t vom Typ T sortiert in den AVL-Baum ein und rebalanciert ihn, wenn nötig.

- **void** remove(T t)

  Löscht t aus dem AVL-Baum und rebalanciert ihn, wenn nötig.

- **boolean** contains(T t)

  Gibt genau dann **true** zurück, wenn t in dem sortierten Baum vorkommt.

Für eine halbwegs vernünftige Darstellung des Baums soll ausserdem die Methode *toString()* überschrieben werden.

- String toString()

  Gibt die Struktur des AVL-Baumes in geklammerter Form als String zurück, d.h. zum Beispiel ein Baum mit Inhalten „`Hinz`“ und „`Kunz`“ vom Typ String führt zu der Ausgabe `((-, Hinz, -), Kunz, -)`, falls „`Kunz`“ in der Wurzel gespeichert ist („`-`“ steht für den leeren Baum).

Die interne Datenstruktur eurer AVL-Bäume soll dabei nach außen nicht sichtbar, d.h. vollständig gekapselt sein. Legt für die Knoten eine Klasse Node an, die ebenfalls vollständig gekapselt ist. Um bei den Baumoperationen einige umständliche Abfragen auf die Existenz von Nachfolgeknoten einzusparen, soll analog zur in der Vorlesung vorgestellten Baumimplementierung eine spezielle Knoten-Klasse NullNode definiert werden. Hat ein Baumknoten keinen linken bzw. rechten Nachfolger soll der entsprechende Zeiger auf den Nachfolger keine **null**-Referenz sein, sondern stattdessen auf eine Instanz von NullNode verweisen.