SoSe 2014
Prof. Dr. Margarita Esponda
ALP2
11. Übungsblatt (Abgabe am 7. Juli)

Ziel: Auseinandersetzung mit Dynamischen Datenstrukturen, Datenabstraktion, Innere Klassen und Iteratoren in Java.

1. Aufgabe (30 Punkte)

a) Vervollständigen Sie die **BinLinkedTree**-Klasse aus der Vorlesung mit den in der Vorlesung besprochenen Operationen (Methoden). Die innere **TreeNode** Klasse muss dabei durch die **DTreeNode** Klasse ersetzt werden, damit die Nachfolger, Vorgänger und Löscht Operationen implementiert werden können.

```
public class BinLinkedTree <T extends Comparable<T>, D>
                                        implements Iterable<T> {
    . . .
    public boolean empty(){...}

    /* Das DTreeNode-Objekt mit dem kleinsten Schlüssel wird zurückgegeben */
    public DTreeNode minNode(DTreeNode node) {...}

    /* Das DTreeNode -Objekt mit dem größten Schlüssel wird zurückgegeben */
    public DTreeNode maxNode(DTreeNode node) {...}

    /* Der Nachfolger des im node gespeicherten Schlüssels wird berechnet und
       seine DTreeNode-Position zurückgegeben */
    public DTreeNode succ(DTreeNode node) {...}

    /* Der Vorgänger des im node gespeicherten Schlüssels wird berechnet und seine
       DTreeNode-Position zurückgegeben */
    public DTreeNode pred(DTreeNode node) {...}

    /* Das DTreeNode-Objekt, welches das key beinhaltet, wird gelöscht, falls
       dieses existiert */
    public boolean delete(T key) {...}

    /* Berechnet die Tiefe des Baumes */
    public int deep() {...}

    /* Ein sortiertes Array mit allen Elemente des Baumes nach Schlüssel sortiert wird
       zurückgegeben */
    public D[] toArray() {...}

    /* Der Baum wird in Textform ausgegeben */
    public String toString() {...}
```

b) Ist Ihre **delete**-Operation kommutativ (in dem Sinne, dass die hinterlassenen Bäume gleich aussehen, unabhängig von der Reihenfolge der durchgeführten **delete**-Operationen)? Erläutern Sie Ihre Antwort.

Ein Binärbaum heisst perfekt balanciert, wenn an jedem Knoten die Anzahl der Knoten im linken und im rechten Unterbaum sich um höchstens 1 unterscheidet.

c) Definieren Sie eine Methode, die überprüft, ob ein binärer Suchbaum perfekt balanciert ist.

```
public boolean perfectBalanced(DTreeNode node){...}
```

d) Definieren Sie eine zweite **Iterator**-Klasse in Ihrer **BinLinkedTree** Klasse, die die Elemente des Baumes in **Postorder**-Reihenfolge durchläuft.

e) Alle Methoden sollen ausführlich getestet werden.

d) Analysieren Sie die Komplexität aller Ihrer Baumoperationen.

Wichtige Hinweise:

1) Verwenden Sie **selbsterklärende Namen** von Variablen und Methoden.
2) Für die Namen aller Bezeichner müssen Sie die **Java-Konventionen** verwenden.
3) Verwenden Sie **vorgegebene Klassen und Methodennamen**.
4) **Methoden sollten klein gehalten werden**, sodass auf den ersten Blick ersichtlich ist, was diese Methode leistet.
5) Methoden sollten möglichst **wenige Argumente** haben.
6) Methoden sollten entweder den Zustand der Eingabeargumente ändern oder einen Rückgabewert liefern.
7) Verwenden Sie **geeignete Hilfsvariablen** und definieren Sie **sinnvolle Hilfsmethoden** in Ihren Klassendefinitionen.
8) **Zahlen** sollten **durch Konstanten** ersetzt werden.
9) **Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.**