Selbsttest 4

1) Information hiding

Ein Student soll einen Suchbaum für Strings implementieren und soll dabei die Prinzipien des Information hiding anwenden. Der Student schlägt folgende Schnittstelle vor:

```
public class Tree {
   Node head;
   public Tree() { … }
   public void insert(String name) { … }
   public boolean search(String name) { … }
   public boolean remove(String name) { … }
}
public class Node {
   String name;
   Node left, right;
   Node(String name) { … }
}
```

Hat der Student seine Arbeit gut gemacht, oder sind ihm Fehler unterlaufen?
Gibt es Fehler? Welche? Korrigieren Sie eventuelle Fehler. Wieviele Zeilen müssen Sie minimal ändern?
Gibt es Möglichkeiten die Sichtbarkeit von Variablen/Methoden weniger restriktiv zu gestalten und trotzdem die Aufgabenstellung zu erfüllen? Wieviele Zeilen können Sie so ändern?

Lösung:
- Statt "Node head" "private Node head". Weil sonst innerhalb desselben Packages auf die Baum-interne Datenstruktur zugegriffen werden könnte. Anzahl zu ändernder Zeilen: 1
- Die Variablen und der Konstruktor der Klasse Node können public sein. Das ändert nichts daran, dass die Knoten innerhalb des Baumes nicht zugreifbar sind. Anzahl Zeilen: 3

2) Bäume

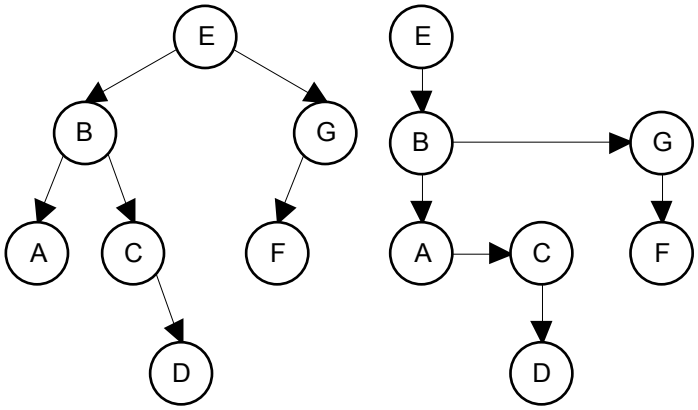
	Ja	Nein
Ein balancierter Baum ist nicht unbedingt vollständig	X	
In manchen Anwendungsfällen können lineare Listen bei der Suche effizienter sein als balancierte Suchbäume	X	
In einem AVL-Baum unterscheidet sich die Anzahl der Knoten im linken Unterbaum maximal um 1 vom rechten Unterbaum		X
Die Auswertung eines arithmetischen Ausdrucks (dargestellt als Syntaxbaum) erfolgt "in order"		X
Gibt man die Knoten beliebiger Suchbäume "in order" aus, so ist die Ausgabe automatisch sortiert	X	
Bei der Umwandlung von Mehrwegbäumen in Bruderbäume bleibt die logarithmische Suchzeit erhalten		X

3) Bruderbaum

Versuchen Sie einen binären Suchbaum (linker Unterbaum < Wurzel <= rechter Unterbaum) für Strings möglichst gut auf einen Bruderbaum abzubilden.
Gibt es wesentliche Unterschiede zwischen Binär- und Bruderbaum, sodass eine Abbildung schwierig werden könnte?

Geht bei Ihrer Lösung die logarithmische Suchzeit verloren?

Lösung:
Eine Möglichkeit ist die Söhne eines Binärbaum-Knotens als Söhne des Bruderbaumes anzusehen:



Die Information "Linker bzw. rechter Sohn vom aktuellen Knoten" geht dabei (auf den ersten Blick) verloren, die direkten Söhne müssen sequentiell durchsucht werden. Aufgrund der Ordnungsrelation (linker Unterbaum < Wurzel <= rechter Unterbaum) ist der binäre Suchbaum jedoch (gedanklich) vollständig erhalten geblieben.
Die logarithmische Suchzeit geht nicht verloren, die Suchzeit kann sich aber in etwa verdoppeln (Die direkten Söhne eines Knotens müssen sequentiell durchsucht werden, bzw. mit dem Vater verglichen werden).