

```
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) { ... }
}
```

- Gibt es Verletzungen bzgl. Information hiding? Wenn ja, wo?
- Können einige Variablen/Methoden weniger restriktiv deklariert werden, ohne das Information hiding aufzuweichen?

## Selbsttest – Lineare Liste vs. Binärer Suchbaum

Es liegen eine sortierte einfach verkettete lineare Liste und ein vollständiger binärer Suchbaum vor. Beide Datenstrukturen haben dieselbe Anzahl von Stringelementen ( $n$  Stück) gespeichert.

- Wie viele Stringvergleiche sind bei beiden Datenstrukturen maximal, minimal und durchschnittlich nötig um ein bestimmtes Element zu finden?
- Angenommen der binäre Suchbaum ist stark (d.h. maximal) entartet, auf welche Anzahl (minimal, maximal, durchschnittlich) von Vergleichen ändert sich die Suche?
- Welche Tiefe hat der Suchbaum im entarteten Fall?

## Selbsttest – Einfügen in binären Suchbaum

Ein binärer Suchbaum für Integer-Werte soll aufgebaut werden. Zeichnen Sie jeweils die Bäume auf die sich zwischen den einzelnen Schritten ergeben (Die Suchbaum-Ordnung ist: Elemente des linken Unterbaums < Wurzelement < Elemente des rechten Unterbaums )

- Einfügen von 6, 12, 4, 13, 9, 7, 8, 5, 3, 1, 20
- Welche max. Tiefe hat der Baum?
- Fügen Sie das Element 7 ein. Geht das? Welche Möglichkeiten gibt es dann?

Welche der folgenden Behauptungen stimmen (nicht)? Und warum?

- Der Grad eines Knoten ist die Anzahl der Knoten, die auf ihn verweisen.
- Die Anzahl der Blätter eines vollständigen Baumes wächst exponentiell mit dessen Höhe.
- Die Höhe eines beliebigen Baumes mit  $n$  Knoten ist ungefähr  $\lg(n)$ .
- In einem Binärbaum hat jeder Knoten Grad 2.
- Binäre Suchbäume sind entweder aufsteigend oder absteigend sortiert.
- Von der Wurzel zu einem beliebigen Blatt gibt es immer genau einen Pfad.