

**Aufgabenbeschreibung: Binärer Suchbaum**

**Hauptaufgabe (1Punkt):** Schreiben Sie eine partielle Implementierung eines binären Suchbaumes. Ihre Implementation sollte eine Klasse für Baumknoten enthalten. Hierfür **müssen** Sie die Klasse Baumknoten nutzen. Diese Implementation hat für einen Knoten nur eine Zahl als Schlüsselwert und keine weiteren Datenfelder. Bearbeiten Sie die folgenden Teilaufgaben:

1. Ihre Implementation soll die folgenden Funktionen bereitstellen.
  - a. Suche nach Elementen, per `search(int k)`: Falls gefunden, soll `true`, ansonsten `false` zurückgegeben werden.
  - b. Einfügen von Elementen, per `insert(int k)`: Die Methode soll `true` zurückgeben, wenn das Einfügen gelungen ist, `false` andernfalls. Das Einfügen von bereits im Baum befindlichen Elemente soll nicht möglich sein.
2. Ergänzen Sie Ihre Implementation um eine Methode `printout()` zur Ausgabe des Bauminhaltes: Wahlweise entweder als Pre- oder In-Order-Darstellung des Baumes, oder als sortierte Sequenz der Elemente im Baum. Für letzteres dürfen Sie auch einen Sortieralgorithmus verwenden (falls die direkte Ausgabe noch nicht sortiert sein sollte).
3. Testen Sie Ihr Programm in einer `main()`-Methode, durch iteratives Einfügen der
  - a. Zahlen 3,7,6,10,12,9, sowie anschließendes Aufrufen der Ausgabe-Methode -- so dass z.B. folgende Ausgaben erfolgen:  

$$5 \{ 3 \{ \text{null} \text{ null} \} 7 \{ 6 \{ \text{null} \text{ null} \} 10 \{ 9 \{ \text{null} \text{ null} \} 12 \{ \text{null} \text{ null} \} \} \}$$

oder

$$[3, 5, 6, 7, 9, 10, 12]$$

**Optionale Zusatzaufgabe 1 (1/2 Punkt):**

Ergänzen Sie Ihre Implementation um eine Methode `delete()` zum Löschen von Elementen. Die Methode soll `true` zurückgeben, wenn gelöscht wurde, `false` andernfalls.

**Optionale Zusatzaufgabe 2 (1/2 Punkt):**

Ergänzen Sie Ihre Implementation und eine Methode `public static boolean depth()` die mit einem Baumknoten `p` aufgerufen werden kann, und die die Höhe des Baumes, der `p` als Wurzel hat, ausgibt.

**Optionale Zusatzaufgabe 3 (1/2 Punkt):**

Ergänzen Sie Ihre Implementierung um eine Methode `public static boolean isBalanced()`, die mit einem Knoten `p` eines Baumes aufgerufen werden kann, und die den Wert `true` genau dann liefert, wenn der Baum, zu dem `p` die Wurzel ist, höhenbalanciert ist, im Sinne der AVL-Baum-Definition.

Hinweis: Sie können die Methode `height()` verwenden.

```
package tree;
public class Treenode {

    Treenode firstChild;
    Treenode secondChild;
    int wert;
    public Treenode (int wert) {
        this.wert = wert;
    }
}
```