# Aufgabenblatt #7

Wenn Sie es nicht bereits gemacht haben, lesen Sie bitte die relevanten Teile des Skriptums und erarbeiten Sie gemeinsam die bereits durchgegangen Fragen am Ende des Kapitels.

Siehe Skriptum bis Seite 286, aber mit Iteratoren.

## Binärbaum

Bei der gegebenen Datenstruktur handelt es sich um einen binären Baum, der Integerwerte speichert. Erweitern Sie diesen Baum um die unten stehenden Methoden. Achten Sie aber darauf, dass Sie keine weiteren Objektvariablen als diejenigen, die vorgegeben sind, verwenden werden dürfen.

Listing 1: Binärbaum

```
public class BinaryTree {

    private Node root;

    private class Node {
        private int value;
        private Node left;
        private Node right;

        // TODO: implement methods here
    }

    // TODO: implement methods here
}
```

Vergessen Sie nicht ihren Code verständlich zu dokumentieren. Schreiben Sie zur jeder Implementierungsaufgabe ein Testprogramm.

### Aufgabe 1: Abfragen und Hinzufügen

Implementieren Sie die Methode void `add(int val)`. Des Weiteren schreiben Sie eine Methode `boolean contains(int val)`, die angibt, ob sich ein bestimmter Wert im Baum befindet.

### Aufgabe 2: Größe

Erweitern Sie die Klasse um die Methode `int size()`, die die Anzahl der Elemente im Baum zurückgibt und die Methode `boolean empty()`, welche wahr zurückliefert wenn der Baum leer ist.

### Aufgabe 3: Traversieren

Ein Binärbaum kann auf drei verschiedene Arten durchlaufen werden:

**Inorder:** Zuerst wird der linke Teilbaum untersucht, der Inhalt des aktuellen Knoten ausgegeben und anschließend wird der rechte Teilbaum untersucht.

**Preorder:** aktueller Knoten, linker Teilbaum, rechter Teilbaum.

**Postorder:** linker Teilbaum, rechter Teilbaum, aktueller Knoten.

Jeder dieser Durchlaufvarianten wird so lange rekursiv ausgeführt, bis der gesamte Baum durchlaufen wurde.

Schreiben Sie drei Methoden welche den gesamten Inhalt des Baumes zurückgeben und zwar auf alle drei Arten: Inorder, Pre- oder Postorder.

### Aufgabe 4: Aufwand abschätzen

Schätzen Sie den durchschnittlichen und maximalen Aufwand für alle Methoden des Baumes hinsichtlich Laufzeit und Speicherverbrauch ab.

## Hashtabelle

### Aufgabe 5: Generische Hashtabelle

In der Vorlesung wurde eine Hashtabelle `Hashtable` für `int` präsentiert. Schreiben Sie diese Klasse so um, dass Sie mit beliebigen gleichartigen Objekten umgehen kann. Bei Kollisionen sollen die Elemente in einer linearen Liste verkettet werden.

Über den selben Typparameter sichergestellt.

Schreiben Sie ein Testprogramm, welche die `Hashtable` mit `Integer`, `String` und auch eine selber geschriebene Klasse (z.b. `Ladybird`) verwendet. Stellen Sie dabei aber auch sicher, dass zwei Elemente mit gleichem `hashCode()` eingefügt werden um Kollisionen zu testen.

# Theorie

Beantworten Sie die folgenden Fragen.

## Aufgabe 6: Algorithmen und Datenstrukturen

Was versteht man unter Algorithmen und Datenstrukturen, und wie hängen diese beiden Begriffe zusammen? Unter welchen Bedingungen sind zwei Algorithmen bzw. Datenstrukturen gleich? Wann sind sie es nicht? Nennen Sie fünf unterschiedliche Datenstrukturen und charakterisieren Sie sie.

## Aufgabe 7: Aufwandsabschätzung

Wie kann man den Aufwand eines Algorithmus abschätzen? Wofür stehen $O(1)$, $O(\log(n))$, $O(n)$, $O(n \cdot \log(n))$, $O(n^2)$ und $O(2^n)$? Wie wirkt sich eine Verdopplung oder Verhundertfachung von $n$ aus? Wieso kann man konstante Faktoren bei der Aufwandsabschätzung einfach ignorieren?

## Aufgabe 8: Such- und Sortieralgorithmen

Was ist eine binäre Suche? Wie funktionieren Bubblesort, Mergesort und Quicksort? Wie hoch ist der Aufwand dafür im Durchschnitt und im schlechtesten Fall?

## Aufgabe 9: Generizität

Was unterscheidet generische von nicht-generischen Klassen? Was unterscheidet einen Typ von einem Typparameter? Kann man Typen und Typparameter gleich verwenden? Wie kann man in Java primitive Typen wie `int` als Elementtypen in generischen Containern verwenden?

## Aufgabe 10: Iteratoren

Was sind und warum verwendet man Iteratoren? Welche Schwierigkeiten treten bei der Implementierung von Iteratoren im Zusammenhang mit Rekursion auf? Wie löst man sie? Durch welches spezielle Sprachkonstrukt unterstützt Java die Verwendung von Iteratoren?