# Grundlagen der Programmkonstruktion
# Übungsblatt 4

## 1 IntTreeSet

### 1.1 Erstellen einer Klasse `IntTreeSet` (1)

Erstellen Sie eine Klasse `IntTreeSet`. Diese Klasse repräsentiert eine Menge von Integer-Werten. Die Menge soll intern als binärer Baum nach Vorbild der Klassen `IntTree` und `IntTreeNode` dargestellt werden. Im Baum selbst muss ein Wert, der in einem Knoten gespeichert ist, größer sein als alle Werte im linken Teilbaum und kleiner als die des rechten Teilbaums sein. Ein leerer Teilbaum wird durch `null` repräsentiert.

Erstellen Sie die nötigen Klassen mit Objektvariablen und Konstruktoren, aber noch ohne Methoden. Die Klassen `IntTree` und `IntTreeNode` stehen dabei *nicht* zur Verfügung. Implementieren Sie nur Methoden, die Sie für diese Aufgabe benötigen.

Bei der Präsentation Ihrer Lösung sollen Sie auch beschreiben können, wie die Datenstruktur aussieht!

### 1.2 Grundlegende Methoden (1)

#### 1.2.1 `boolean isEmpty()`

Liefert `true`, wenn die Menge leer ist, ansonsten `false`.

#### 1.2.2 `boolean contains(int elem)`

Liefert `true`, wenn die Menge `elem` enthält. Diese Methode darf die Datenstruktur nicht ändern.

#### 1.2.3 `int count()`

Liefert die Anzahl aller Elemente in der Menge zurück.

#### 1.2.4 `boolean add(int elem)`

Fügt `elem` der Menge hinzu. Sollte das Element bereits vorhanden sein, soll die Datenstruktur nicht geändert werden und `false` zurückgegeben werden. Ansonsten soll `true` zurückgegeben werden und die Menge danach `elem` enthalten, d.h., `contains(elem)` muss danach `true` zurückliefern.

#### 1.2.5 `boolean remove(int elem)`

Entfernt `elem` aus der Menge. Sollte `elem` in der Menge vorhanden sein, so soll `true` zurückgegeben werden, ansonsten `false` (in diesem Fall darf die Datenstruktur nicht geändert werden). Ein Aufruf von `contains(elem)` muss nach einem Aufruf von `remove(elem)` auf jeden Fall `false` zurückliefern.

## 2 `boolean equals(Object o)` (2)

Erstellen Sie in der Klasse `IntTreeSet` die Methode `equals` unter Beachtung der Vorgaben in `Object`. Lesen Sie dazu die Java-Dokumentation von `Object` sorgfältig: `http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#equals(java.lang.Object)` Sie sollen bei der Präsentation dieser Aufgabe die wesentlichen Vorgaben beschreiben können und erklären können, warum diese bei ihrer Lösung erfüllt sind.

Zwei Instanzen der Klasse `IntTreeSet` sind gleich, wenn Sie dieselben Elemente enthalten. Beachen Sie, dass die interne Baumstruktur sich davon wesentlich unterscheiden kann. Wenn somit in beiden Bäumen dieselben Elemente enthalten sind, soll `equals true` liefern, auch wenn die Bäume unterschiedliche Strukturen haben.

Beachten Sie, dass das Verwenden der Methoden `toString()` oder `hashCode()` zu falschen Ergebnissen führen kann!

### Vorgehensweise

Sie können zur Hilfe beliebige Methoden erstellen, z.B., eine Methode `boolean isSubset(IntTreeSet set)` um festzustellen, ob eine Menge eine Untermenge ist (Beachten Sie, dass für Mengen gilt $A \subseteq B \wedge B \subseteq A \Rightarrow A = B$).

## 3 `String toString()` (1)

Erstellen Sie eine Methode `toString()` zur Ausgabe der Menge. Die Elemente sollen dabei *aufsteigend*, durch Beistriche getrennt, ausgegeben werden. Die gesamte Menge soll dabei in geschwungenen Klammern stehen.

## 4 Beispiel

Ihre Lösung sollten zu folgenden Ausgaben führen:

```
IntTreeSet a = new IntTreeSet();

a.isEmpty(); // ergibt true
a.add(1); // ergibt true
a.isEmpty(); // ergibt false
a.add(1); // ergibt false
a.count(); // ergibt 1
a.add(2); // ergibt true
a.add(3); // ergibt true
a.contains(1); // ergibt true
a.remove(1); // ergibt true
a.contains(1); // ergibt false

a.equals(a); // ergibt true

IntTreeSet b = new IntTreeSet();
b.add(3); // ergibt true;
```

```
b.add(2); // ergibt true;

a.equals(b); // ergibt true
b.equals(a); // ergibt true;

a.toString(); // ergibt "{2, 3}"
b.toString(); // ergibt "{2, 3}"
```