Aufgabe 2

Implementieren Sie in einer objekt-orientierten Sprache Ihrer Wahl eine Klasse namens `BinBaum`, deren Instanzen binäre Bäume mit ganzzahligen Datenknoten darstellen, nach folgender Spezifikation:

(a) Beginnen Sie zunächst mit der Datenstruktur selbst:

- Mit Hilfe des Konstruktors soll ein neuer Baum erstellt werden, der aus einem einzelnen Knoten besteht, in dem der dem Konstruktor als Parameter übergebene Wert (ganzzahlig, in Java z.B. `int`) gespeichert ist.

```
14    class Knoten {
15      int value;
16
17      Knoten left;
18      Knoten right;
19
20      public Knoten(int value) {
21        this.value = value;
22      }
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

- Die Methoden `setLeft(int value)` bzw. `setRight(int value)` sollen den linken bzw. rechten Teilbaum des aktuellen Knotens durch einen jeweils neuen Teilbaum ersetzen, der seinerseits aus einem einzelnen Knoten besteht, in dem der übergebene Wert `value` gespeichert ist. Sie haben keinen Rückgabewert.

```
24      public void setLeft(Knoten left) {
25        this.left = left;
26      }
27
28      public void setRight(Knoten right) {
29        this.right = right;
30      }
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

- Die Methoden `getLeft()` bzw. `getRight()` sollen den linken bzw. rechten Teilbaum zurückgeben (bzw. `null`, wenn keiner vorhanden ist)

```
32      public Knoten getLeft() {
33        return left;
34      }
35
36      public Knoten getRight() {
37        return right;
38      }
```

Code-Beispiel auf Github ansehen:
src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

- Die Methode `int getValue()` soll den Wert zurückgeben, der im aktuellen Wurzelknoten gespeichert ist.

```java
40      public int getValue() {
41        return value;
42      }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

(b) Implementieren Sie nun die Methoden `preOrder()` bzw. `postOrder()`. Sie sollen die Knoten des Baumes mit Tiefensuche traversieren, die Werte dabei in pre-order bzw. post-order Reihenfolge in eine Liste (z.B. `List<Integer>`) speichern und diese Ergebnisliste zurückgeben. Die Tiefensuche soll dabei zuerst in den linken und dann in den rechten Teilbaum absteigen.

```java
45    void preOrder(Knoten knoten, List<Integer> list) {
46      if (knoten != null) {
47        list.add(knoten.getValue());
48        preOrder(knoten.getLeft(), list);
49        preOrder(knoten.getRight(), list);
50      }
51    }
52
53    List<Integer> preOrder() {
54      List<Integer> list = new ArrayList<>();
55      preOrder(head, list);
56      return list;
57    }
58
59    void postOrder(Knoten knoten, List<Integer> list) {
60      if (knoten != null) {
61        postOrder(knoten.getLeft(), list);
62        postOrder(knoten.getRight(), list);
63        list.add(knoten.getValue());
64      }
65    }
66
67    List<Integer> postOrder() {
68      List<Integer> list = new ArrayList<>();
69      postOrder(head, list);
70      return list;
71    }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

(c) Ergänzen Sie schließlich die Methode `isSearchTree()`. Sie soll überprüfen, ob der Binärbaum die Eigenschaften eines binären Suchbaums erfüllt. Beachten Sie, dass die Laufzeit-Komplexität Ihrer Implementierung linear zur Anzahl der Knoten im Baum sein muss.

```java
73    boolean isSearchTree(Knoten knoten) {
74      if (knoten == null) {
75        return true;
76      }
77
78      if (knoten.getLeft() != null && knoten.getValue() <
        ↪  knoten.getLeft().getValue()) {
79        return false;
80      }
81
```

```
82        if (knoten.getRight() != null && knoten.getValue() >
          ↪  knoten.getRight().getValue()) {
83          return false;
84        }
85
86        return isSearchTree(knoten.getLeft()) &&
          ↪  isSearchTree(knoten.getRight());
87      }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java

Additum

```
3   import java.util.ArrayList;
4   import java.util.List;
5
6   public class BinBaum {
7
8     Knoten head;
9
10    public BinBaum(Knoten head) {
11      this.head = head;
12    }
13
14    class Knoten {
15      int value;
16
17      Knoten left;
18      Knoten right;
19
20      public Knoten(int value) {
21        this.value = value;
22      }
23
24      public void setLeft(Knoten left) {
25        this.left = left;
26      }
27
28      public void setRight(Knoten right) {
29        this.right = right;
30      }
31
32      public Knoten getLeft() {
33        return left;
34      }
35
36      public Knoten getRight() {
37        return right;
38      }
39
40      public int getValue() {
41        return value;
42      }
43    }
44
45    void preOrder(Knoten knoten, List<Integer> list) {
46      if (knoten != null) {
47        list.add(knoten.getValue());
```

```
48        preOrder(knoten.getLeft(), list);
49        preOrder(knoten.getRight(), list);
50      }
51    }
52
53    List<Integer> preOrder() {
54      List<Integer> list = new ArrayList<>();
55      preOrder(head, list);
56      return list;
57    }
58
59    void postOrder(Knoten knoten, List<Integer> list) {
60      if (knoten != null) {
61        postOrder(knoten.getLeft(), list);
62        postOrder(knoten.getRight(), list);
63        list.add(knoten.getValue());
64      }
65    }
66
67    List<Integer> postOrder() {
68      List<Integer> list = new ArrayList<>();
69      postOrder(head, list);
70      return list;
71    }
72
73    boolean isSearchTree(Knoten knoten) {
74      if (knoten == null) {
75        return true;
76      }
77
78      if (knoten.getLeft() != null && knoten.getValue() <
        ↪  knoten.getLeft().getValue()) {
79        return false;
80      }
81
82      if (knoten.getRight() != null && knoten.getValue() >
        ↪  knoten.getRight().getValue()) {
83        return false;
84      }
85
86      return isSearchTree(knoten.getLeft()) &&
        ↪  isSearchTree(knoten.getRight());
87    }
88
89    boolean isSearchTree() {
90      return isSearchTree(head);
91    }
92  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2014/fruehjahr/BinBaum.java`