Aufgabe 8

(a) Implementieren Sie in einer objektorientierten Sprache einen binären Suchbaum für ganze Zahlen! Dazu gehören Methoden zum Setzen und Ausgeben der Attribute `zahl`, `linker_teilbaum` und `rechter_teilbaum`. Design: eine Klasse `Knoten` und eine Klasse `BinBaum`. Ein Knoten hat einen linken und einen rechten Nachfolger. Ein Baum verwaltet die Wurzel. Er hängt neue Knoten an und löscht Knoten.

```java
public class BinBaum {

  private Knoten wurzel = null;

  public void setzeWurzel(Knoten knoten) {
    wurzel = knoten;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

```java
public class Knoten {
  private int zahl;
  private Knoten links = null;
  private Knoten rechts = null;

  public Knoten() {
  }

  public Knoten(int zahl) {
    this.zahl = zahl;
  }

  public void setzeZahl(int zahl) {
    this.zahl = zahl;
  }

  public int gibZahl() {
    return zahl;
  }

  public void setzeLinks(Knoten k) {
    links = k;
  }

  public Knoten gibLinks() {
    return links;
  }

  public void setzeRechts(Knoten k) {
    rechts = k;
  }

  public Knoten gibRechts() {
    return rechts;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java`

(b) Schreiben Sie eine Methode `fügeEin(...)`, die eine Zahl in den Baum ein-

fügt!

```java
public void fügeEin(int zahl) {
  Knoten aktueller = wurzel;
  Knoten neuerKnoten = new Knoten(zahl);
  if (wurzel == null) {
    wurzel = neuerKnoten;
    return;
  }
  while (aktueller != null) {
    // suche links
    if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
      aktueller = aktueller.gibLinks();
      // fuege ein
    } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() == null) {
      aktueller.setzeLinks(neuerKnoten);
      break;
    }
    // suche rechts
    if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
      aktueller = aktueller.gibRechts();
    // fuege ein
    } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() == null) {
      aktueller.setzeRechts(neuerKnoten);
      break;
    }
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java

(c) Schreiben Sie eine Methode `void besuchePostOrder(...)`, die die Zahlen in der Reihenfolge postorder ausgibt!

```java
public static void besuchePostOrder(Knoten knoten) {
  // Sonderfall leerer (Teil-)Baum
  if (knoten == null) {
    System.out.println("Leerer Baum");
  } else {
    // Linker
    if (knoten.gibLinks() != null) {
      besuchePostOrder(knoten.gibLinks());
    }
    // Rechter
    if (knoten.gibRechts() != null) {
      besuchePostOrder(knoten.gibRechts());
    }
    System.out.println(knoten.gibZahl());
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java

(d) Ergänzen Sie Ihr Programm um die rekursiv implementierte Methode `int berechneSumme(...)`, die die Summe der Zahlen des Unterbaums, des-

sen Wurzel der Knoten ist, zurückgibt! Falls der Unterbaum leer ist, ist der Rückgabewert 0!

```
1  int summe (Knoten x)...
```

```
55   public int berechneSumme(Knoten knoten) {
56     int ergebnis = 0;
57
58     // Sonderfall: leerer Unterbaum
59     if (knoten == null) {
60       return 0;
61     }
62     // linker
63     if (knoten.gibLinks() != null) {
64       ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
65     }
66     // rechter
67     if (knoten.gibRechts() != null) {
68       ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
69     }
70     // Wurzel
71     ergebnis = ergebnis + knoten.gibZahl();
72     return ergebnis;
73   }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java`

(e) Schreiben Sie eine Folge von Anweisungen, die einen Baum mit Namen BinBaum erzeugt und nacheinander die Zahlen 5 und 7 einfügt! In den binären Suchbaum werden noch die Zahlen 4, 11, 6 und 2 eingefügt. Zeichnen Sie den Baum, den Sie danach erhalten haben, und schreiben Sie die eingefügten Zahlen in der Reihenfolge der Traversierungsmöglichkeit `postorder` auf!

(f) Implementieren Sie eine Operation `isSorted(...)`, die für einen (Teil-)baum feststellt, ob er sortiert ist.

```
75   public boolean istSortiert(Knoten knoten) {
76     // Baum leer
77     if (knoten == null) {
78       return true;
79     }
80
81     // linker Nachfolger nicht okay
82     if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() >
       ↪  knoten.gibZahl()) {
83       return false;
84     }
85
86     // rechter Nachfolger nicht okay
87     if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <=
       ↪  knoten.gibZahl()) {
88       return false;
89     }
90     // sonst prüfe Teilbaeume
91     return (istSortiert(knoten.gibRechts()) &&
       ↪  istSortiert(knoten.gibLinks()));
92   }
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java

```java
public class BinBaum {

  private Knoten wurzel = null;

  public void setzeWurzel(Knoten knoten) {
    wurzel = knoten;
  }

  public void fügeEin(int zahl) {
    Knoten aktueller = wurzel;
    Knoten neuerKnoten = new Knoten(zahl);
    if (wurzel == null) {
      wurzel = neuerKnoten;
      return;
    }
    while (aktueller != null) {
      // suche links
      if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() != null) {
        aktueller = aktueller.gibLinks();
        // fuege ein
      } else if (zahl <= aktueller.gibZahl() && aktueller.gibLinks() ==
       ↪  null) {
        aktueller.setzeLinks(neuerKnoten);
        break;
      }
      // suche rechts
      if (zahl > aktueller.gibZahl() && aktueller.gibRechts() != null) {
        aktueller = aktueller.gibRechts();
      // fuege ein
      } else if (zahl > aktueller.gibZahl() && aktueller.gibRechts() ==
       ↪  null) {
        aktueller.setzeRechts(neuerKnoten);
        break;
      }
    }
  }

  public static void besuchePostOrder(Knoten knoten) {
    // Sonderfall leerer (Teil-)Baum
    if (knoten == null) {
      System.out.println("Leerer Baum");
    } else {
      // Linker
      if (knoten.gibLinks() != null) {
        besuchePostOrder(knoten.gibLinks());
      }
      // Rechter
      if (knoten.gibRechts() != null) {
        besuchePostOrder(knoten.gibRechts());
      }
      System.out.println(knoten.gibZahl());
    }
  }

  public int berechneSumme(Knoten knoten) {
    int ergebnis = 0;

    // Sonderfall: leerer Unterbaum
```

```java
    if (knoten == null) {
      return 0;
    }
    // linker
    if (knoten.gibLinks() != null) {
      ergebnis = ergebnis + berechneSumme(knoten.gibLinks());
    }
    // rechter
    if (knoten.gibRechts() != null) {
      ergebnis = ergebnis + berechneSumme(knoten.gibRechts());
    }
    // Wurzel
    ergebnis = ergebnis + knoten.gibZahl();
    return ergebnis;
  }

  public boolean istSortiert(Knoten knoten) {
    // Baum leer
    if (knoten == null) {
      return true;
    }

    // linker Nachfolger nicht okay
    if (knoten.gibLinks() != null && knoten.gibLinks().gibZahl() >
    ↪  knoten.gibZahl()) {
      return false;
    }

    // rechter Nachfolger nicht okay
    if (knoten.gibRechts() != null && knoten.gibRechts().gibZahl() <=
    ↪  knoten.gibZahl()) {
      return false;
    }
    // sonst prüfe Teilbaeume
    return (istSortiert(knoten.gibRechts()) &&
    ↪  istSortiert(knoten.gibLinks()));
  }

  public static void main(String[] args) {
    BinBaum baum = new BinBaum();

    baum.fügeEin(5);
    baum.fügeEin(7);
    baum.fügeEin(4);
    baum.fügeEin(11);
    baum.fügeEin(6);
    baum.fügeEin(2);

    besuchePostOrder(baum.wurzel);
  }
}
```

Code-Beispiel auf Github ansehen: src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/BinBaum.java

```java
public class Knoten {
  private int zahl;
  private Knoten links = null;
  private Knoten rechts = null;

  public Knoten() {
  }
```

```
10
11    public Knoten(int zahl) {
12      this.zahl = zahl;
13    }
14
15    public void setzeZahl(int zahl) {
16      this.zahl = zahl;
17    }
18
19    public int gibZahl() {
20      return zahl;
21    }
22
23    public void setzeLinks(Knoten k) {
24      links = k;
25    }
26
27    public Knoten gibLinks() {
28      return links;
29    }
30
31    public void setzeRechts(Knoten k) {
32      rechts = k;
33    }
34
35    public Knoten gibRechts() {
36      return rechts;
37    }
38  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66112/jahr_2003/herbst/Knoten.java`