```
1
2   /**
3    * Knoten einer Liste - Implementierung entsprechend dem Buch Informat
    ik Oberstufe 1
4    *                     - Datenstrukturen und Softwareentwicklung
5    *
6    * @author:     Sebastian Funke
7    * @version:    1.0
8    */
9   public class Knoten
10  {
11      private Datenelement daten;
12      private Knoten nachfolger;
13
14      /**
15       * Constructor for objects of class Knoten
16       */
17      public Knoten(Datenelement d, Knoten k)
18      {
19          this.daten = d;
20          this.nachfolger = k;
21      }
22
23      public Knoten knotenEntfernen(Datenelement dvergleich) {
24          if (daten != dvergleich) {
25              if (nachfolger != null) {
26                  nachfolger.knotenEntfernen(dvergleich);
27              }
28              return this;
29          }
30          else {
31              return nachfolger;
32          }
33      }
34
35      public Knoten endeEntfernen() {
36          if (nachfolger != null) {
37              nachfolger = nachfolger.endeEntfernen();
38              return this;
39          }
40          else {
41              return null;
42          }
43      }
44
45      public Datenelement endeGeben() {
46          if (nachfolger != null) {
47              return nachfolger.endeGeben();
48          }
49          else {
50              return daten;
51          }
52      }
```

```
53
54      public Knoten sortiertEinfuegen(Datenelement dneu) {
55          if (daten.istKleinerAls(dneu)) {
56              if (nachfolger != null) {
57                  nachfolger = nachfolger.sortiertEinfuegen(dneu);
58              }
59              else {
60                  hintenEinfuegen(dneu);
61              }
62              return this;
63          }
64          else {
65              Knoten kneu;
66              kneu = new Knoten(dneu, this);
67              return kneu;
68          }
69      }
70
71      public Knoten einfuegenVor(Datenelement dneu, Datenelement dvergle
    ich) {
72          if (daten != dvergleich) {
73              if (nachfolger != null) {
74                  nachfolger = nachfolger.einfuegenVor(dneu, dvergleich)
    ;
75              }
76              else {
77                  hintenEinfuegen(dneu);
78              }
79              return this;
80          }
81          else {
82              Knoten kneu;
83              kneu = new Knoten(dneu, this);
84              return kneu;
85          }
86      }
87
88      public void hintenEinfuegen(Datenelement dneu) {
89          if (nachfolger != null) {
90              nachfolger.hintenEinfuegen(dneu);
91          }
92          else {
93              Knoten kneu;
94              kneu = new Knoten(dneu, null);
95              nachfolger = kneu;
96          }
97      }
98
99      public Datenelement suchen(String vergleichswert) {
100         if (daten.schluesselIstGleich(vergleichswert)) {
101             return daten;
102         }
103         else {
```

```
104             if (nachfolger != null) {
105                 return nachfolger.suchen(vergleichswert);
106             }
107             else {
108                 return null;
109             }
110         }
111     }
112 
113     public int restlaengeGeben() {
114         if (nachfolger == null) {
115             return 1;
116         }
117         else {
118             return nachfolger.restlaengeGeben() + 1;
119         }
120     }
121 
122     public void informationAusgeben() {
123         this.daten.informationAusgeben();
124 
125         if (nachfolger != null) {
126             nachfolger.datenelementGeben().informationAusgeben();
127         }
128     }
129 
130     public Datenelement datenelementGeben() {
131         return daten;
132     }
133 
134     public Knoten nachfolgerGeben() {
135         return nachfolger;
136     }
137 }
138 
```