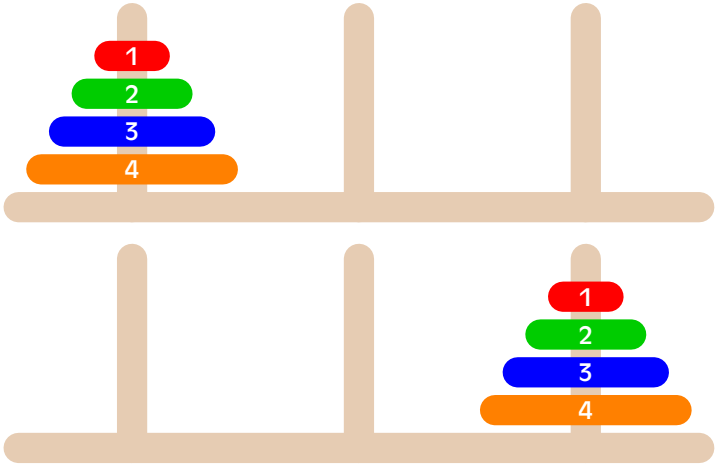
Aufgabe 2: Türme von Hanoi

Betrachten wir das folgende Spiel (Türme von Hanoi), das aus drei Stäben 1, 2 und 3 besteht, die senkrecht im Boden befestigt sind. Weiter gibt es n kreisförmige Scheiben mit einem Loch im Mittelpunkt, so dass man sie auf die Stäbe stecken kann. Dabei haben die Scheiben verschiedene Radien, alle sind unterschiedlich groß. Zu Beginn stecken alle Scheiben auf dem Stab 1, wobei immer eine kleinere auf einer größeren liegt. Das Ziel des Spiels ist es nun, die Scheiben so umzuordnen, dass sie in der gleichen Reihenfolge auf dem Stab 3 liegen. Dabei darf immer nur eine Scheibe bewegt werden und es darf nie eine größere auf einer kleineren Scheibe liegen. Stab 2 darf dabei als Hilfsstab verwendet werden.

Ein Beispiel für 4 Scheiben finden Sie in folgendem Bild:



(a) Ein `Element` hat immer einen Wert (Integer) und kennt das Nachfolgende `Element`, wobei immer nur das jeweilige `Element` auf seinen Wert und seinen Nachfolger zugreifen darf.

```
6  public class Element {
7    private int wert;
8    private Element nächstes;
9
10   public Element(int wert) {
11     this.wert = wert;
12     nächstes = null;
13   }
14
15   public int gibWert() {
16     return wert;
17   }
18
19   public Element gibNächstes() {
20     return nächstes;
21   }
22
23   public void setzeNächstes(Element next) {
24     this.nächstes = next;
25   }
```

```
26  }
```

(b) Ein `Turm` ist einem Stack (Kellerspeicher) nachempfunden und kennt somit nur das erste Element. Hinweis: Beachten Sie, dass nur kleinere Elemente auf den bisherigen Stack gelegt werden können

```
6   public class Turm {
7     private Element oben;
8
9     public Turm() {
10      oben = null;
11    }
12
13    public Element gibOben() {
14      return oben;
15    }
16
17    public void legeDrauf(Element e) {
18      if (oben == null) {
19        oben = e;
20      } else if (oben.gibWert() == e.gibWert()) {
21        System.out.println("Fehler! schieben sind gleich gross");
22      } else if (oben.gibWert() < e.gibWert()) {
23
          ↪  System.out.println("Fehler! Größere Scheiben dürfen nicht auf kleinere");
24      } else {
25        e.setzeNächstes(oben);
26        oben = e;
27      }
28    }
29
30    public Element nimmHerunter() {
31      if (oben == null) {
32        System.out.println("Turm ist bereits leer!");
33        return null;
34      } else {
35        Element merker = new Element(oben.gibWert());
36        oben = oben.gibNächstes();
37        return merker;
38      }
39    }
40
41  }
```

(c) In der Klasse `Hanoi` müssen Sie nur die Methode `public void hanoi (int n, Turm quelle, Turm ziel, Turm hilfe)` implementieren. Die anderen Methoden sind zur Veranschaulichung des Spiels! Entwerfen Sie eine rekursive Methode die einen Turm der Höhe n vom Stab `quelle` auf den Stab `ziel` transportiert und den Stab `hilfe` als Hilfsstab verwendet.

```
6   public class Hanoi {
7     private int anzahlScheiben = 3;
8     private int[] turm0helper, turm1helper, turm2helper;
9     Turm turm0 = new Turm();
10    Turm turm1 = new Turm();
11    Turm turm2 = new Turm();
12
13    public Hanoi(int anzahlScheiben) {
```

```
14      if (anzahlScheiben <= 0) {
15
          ↪  System.out.println("Zu wenige Scheiben. Defaulfall (anzahlScheiben = 3) wird angewendet!");
16      } else {
17        this.anzahlScheiben = anzahlScheiben;
18      }
19      for (int i = this.anzahlScheiben; i > 0; i--) {
20        turm0.legeDrauf(new Element(i));
21      }
22      zeigeTürme();
23    }
24
25    /**
26     * @param n Höhe des Turms (Anzahl der Scheiben).
27     * @param quelle Der Ausgangs-Turm.
28     * @param ziel Der Ziel-Turm.
29     * @param hilfe Der Hilfs-Turm in der Mitte.
30     */
31    public void hanoi(int n, Turm quelle, Turm ziel, Turm hilfe) {
32      if (n == 1) {
33        verschiebeScheibe(quelle, ziel);
34        System.out.println("Fertig!");
35      } else {
36        hanoi(n - 1, quelle, hilfe, ziel);
37        verschiebeScheibe(quelle, ziel);
38        hanoi(n - 1, hilfe, ziel, quelle);
39      }
40    }
41
42    public void verschiebeScheibe(Turm quelle, Turm ziel) {
43      // Bereinigt die Konsole
44      System.out.print('\u000C');
45
46      if (quelle == ziel) {
47        System.out.println("Quelle ist gleich dem Ziel!");
48        return;
49      }
50      ziel.legeDrauf(quelle.nimmHerunter());
51      zeigeTürme();
52    }
53
54    /**
55     * Zeige die drei Türme in der Konsole
56     */
57    public void zeigeTürme() {
58      Element merker = turm0.gibOben();
59      int zeiger0 = 0;
60      int zeiger1 = 0;
61      int zeiger2 = 0;
62      turm0helper = new int[this.anzahlScheiben];
63      turm1helper = new int[this.anzahlScheiben];
64      turm2helper = new int[this.anzahlScheiben];
65      int i = 0;
66      while (merker != null) {
67        turm0helper[i++] = merker.gibWert();
68        merker = merker.gibNächstes();
69        zeiger0++;
70      }
71      merker = turm1.gibOben();
72      i = 0;
73      while (merker != null) {
```

```
74          turm1helper[i++] = merker.gibWert();
75          merker = merker.gibNächstes();
76          zeiger1++;
77        }
78        merker = turm2.gibOben();
79        i = 0;
80        while (merker != null) {
81          turm2helper[i++] = merker.gibWert();
82          merker = merker.gibNächstes();
83          zeiger2++;
84        }
85
86        int help0 = zeiger0 % this.anzahlScheiben;
87        int help1 = zeiger1 % this.anzahlScheiben;
88        int help2 = zeiger2 % this.anzahlScheiben;
89
90        for (int j = 0; j < turm0helper.length; j++) {
91          System.out.print(turm0helper[help0++] + "  " +
            ↪  turm1helper[help1++] + "  " + turm2helper[help2++]);
92          System.out.println();
93          help0 = help0 % this.anzahlScheiben;
94          help1 = help1 % this.anzahlScheiben;
95          help2 = help2 % this.anzahlScheiben;
96        }
97
98        try {
99          Thread.sleep(500);
100       } catch (InterruptedException e) {
101         // ignore
102       }
103     }
104
105     public static void main(String[] args) {
106       Hanoi h = new Hanoi(5);
107       h.hanoi(5, h.turm0, h.turm2, h.turm1);
108     }
109   }
```