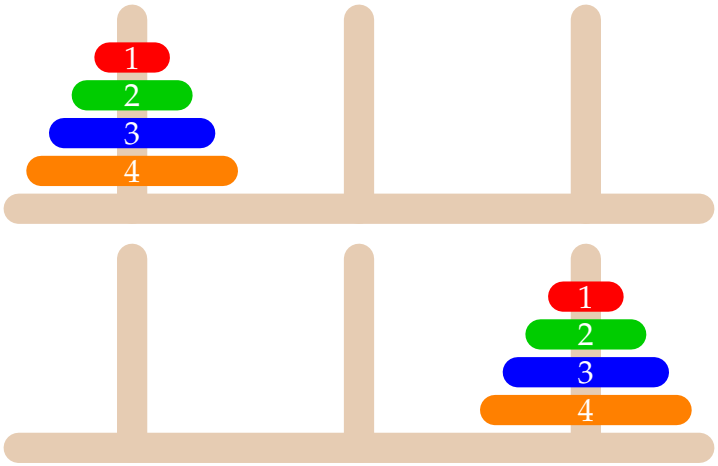
Kellerspeicher: Türme von Hanoi (*Hanoi*)

Stichwörter: Stapel (Stack), Teile-und-Herrsche (Divide-and-Conquer), Rekursion

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.

Lösungsvorschlag

```
public class Element {
  private int wert;
  private Element nächstes;

  public Element(int wert) {
    this.wert = wert;
    nächstes = null;
  }

  public int gibWert() {
    return wert;
  }

  public Element gibNächstes() {
    return nächstes;
  }

  public void setzeNächstes(Element next) {
```

```
    this.nächstes = next;
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/hanoi/Element.java`

(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

Lösungsvorschlag

```
public class Turm {
  private Element oben;

  public Turm() {
    oben = null;
  }

  public Element gibOben() {
    return oben;
  }

  public void legeDrauf(Element e) {
    if (oben == null) {
      oben = e;
    } else if (oben.gibWert() == e.gibWert()) {
      System.out.println("Fehler! schieben sind gleich gross");
    } else if (oben.gibWert() < e.gibWert()) {
      System.out.println("Fehler! Größere Scheiben dürfen nicht auf kleinere");
    } else {
      e.setzeNächstes(oben);
      oben = e;
    }
  }

  public Element nimmHerunter() {
    if (oben == null) {
      System.out.println("Turm ist bereits leer!");
      return null;
    } else {
      Element merker = new Element(oben.gibWert());
      oben = oben.gibNächstes();
      return merker;
    }
  }

}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/hanoi/Turm.java`

(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.

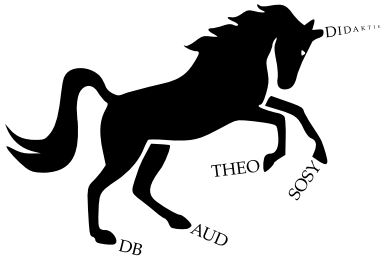
Lösungsvorschlag

```
public class Hanoi {
  private int anzahlScheiben = 3;
  private int[] turm0helper, turm1helper, turm2helper;
  Turm turm0 = new Turm();
  Turm turm1 = new Turm();
  Turm turm2 = new Turm();

  public Hanoi(int anzahlScheiben) {
    if (anzahlScheiben <= 0) {

↪   System.out.println("Zu wenige Scheiben. Defaulfall (anzahlScheiben = 3) wird angewendet!");
    } else {
      this.anzahlScheiben = anzahlScheiben;
    }
    for (int i = this.anzahlScheiben; i > 0; i--) {
      turm0.legeDrauf(new Element(i));
    }
    zeigeTürme();
  }

  /**
   * @param n      Höhe des Turms (Anzahl der Scheiben).
   * @param quelle Der Ausgangs-Turm.
   * @param ziel   Der Ziel-Turm.
   * @param hilfe  Der Hilfs-Turm in der Mitte.
   */
  public void hanoi(int n, Turm quelle, Turm ziel, Turm hilfe) {
    if (n == 1) {
      verschiebeScheibe(quelle, ziel);
      System.out.println("Fertig!");
    } else {
      hanoi(n - 1, quelle, hilfe, ziel);
      verschiebeScheibe(quelle, ziel);
      hanoi(n - 1, hilfe, ziel, quelle);
    }
  }

  public void verschiebeScheibe(Turm quelle, Turm ziel) {
    // Bereinigt die Konsole
    System.out.print('\u000C');

    if (quelle == ziel) {
      System.out.println("Quelle ist gleich dem Ziel!");
      return;
    }
    ziel.legeDrauf(quelle.nimmHerunter());
    zeigeTürme();
  }

  /**
   * Zeige die drei Türme in der Konsole
```

```
   */
  public void zeigeTürme() {
    Element merker = turm0.gibOben();
    int zeiger0 = 0;
    int zeiger1 = 0;
    int zeiger2 = 0;
    turm0helper = new int[this.anzahlScheiben];
    turm1helper = new int[this.anzahlScheiben];
    turm2helper = new int[this.anzahlScheiben];
    int i = 0;
    while (merker != null) {
      turm0helper[i++] = merker.gibWert();
      merker = merker.gibNächstes();
      zeiger0++;
    }
    merker = turm1.gibOben();
    i = 0;
    while (merker != null) {
      turm1helper[i++] = merker.gibWert();
      merker = merker.gibNächstes();
      zeiger1++;
    }
    merker = turm2.gibOben();
    i = 0;
    while (merker != null) {
      turm2helper[i++] = merker.gibWert();
      merker = merker.gibNächstes();
      zeiger2++;
    }

    int help0 = zeiger0 % this.anzahlScheiben;
    int help1 = zeiger1 % this.anzahlScheiben;
    int help2 = zeiger2 % this.anzahlScheiben;

    for (int j = 0; j < turm0helper.length; j++) {
      System.out.print(turm0helper[help0++] + "  " + turm1helper[help1++] + "  " +
↪   turm2helper[help2++]);
      System.out.println();
      help0 = help0 % this.anzahlScheiben;
      help1 = help1 % this.anzahlScheiben;
      help2 = help2 % this.anzahlScheiben;
    }

    try {
      Thread.sleep(500);
    } catch (InterruptedException e) {
      // ignore
    }
  }

  public static void main(String[] args) {
    Hanoi h = new Hanoi(5);
    h.hanoi(5, h.turm0, h.turm2, h.turm1);
  }
```

```
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/aud/listen/hanoi/Hanoi.java`



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieser Aufgabe kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben-tex/blob/main/Module/30_AUD/70_Listen/30_Stapel/Aufgabe_Hanoi.tex`