1 Objektorientierung

Ein Getränkelieferservice verwaltet die Bestellungen verschiedener Kunden. Die folgenden Teilaufgaben sind in einer objektorientierten Programmiersprache zu lösen (die verwendete Sprache ist vorab anzugeben.).

(a) Implementieren Sie eine Klasse `Kasten` zur Beschreibung eines Getränkekastens mit den folgenden Eigenschaften. Entscheiden Sie dabei jeweils ob eine Realisierung als Objekt- oder Klassenfeld sinnvoll ist.

- Es existiert ein einheitliches Kastenpfand in Höhe von 1,50 Euro.
- Für alle Flaschen in einem Kasten gelte ein einheitliches Flaschenpfand, das jedoch von Kasten zu Kasten verschieden sein kann.
- Während das Flaschenpfand für alle Flaschen eines Kastens gleich ist, sind die Einzelpreise der Flaschen je nach Inhalt unterschiedlich. Die Einzelpreise (ohne Flaschenpfand) der im Kasten enthaltenen Flaschen sollen in einem 2-dimensionalen Array abgelegt werden.

Geben Sie für die Klasse `Kasten` einen geeigneten Konstruktor an. Ergänzen Sie in der Klasse `Kasten` eine Objektmethode zur Berechnung des Gesamtpreises des Getränkekastens inklusive Kasten- und Flaschenpfand.

(b) Schreiben Sie eine Klasse `Bestellung`. Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet werden, die über den Konstruktoraufruf erstellt wird. Außerdem soll zu jeder Bestellung der Name des Kunden gespeichert werden, sowie eine einfach verkettete Liste der bestellten Getränkekästen. Die Klasse Bestellung soll weiterhin eine Methode beinhalten, die den Gesamtpreis der Bestellung ermittelt.

(c) Schreiben Sie ein kleines Testprogramm, das eine Bestellung erstellt, die zwei Getränkekästen umfasst. Der erste Kasten soll ein 1 x 1 Getränkekasten mit einer Flasche zu 0,75 Euro sein, der zweite Kasten soll - wie in Abbildung 1 dargestellt - ein 3 x 3 Getränkekasten mit 3 Flaschen zu 0,7 Euro auf der Diagonalen und 3 weiteren Flaschen zu je 1 Euro sein. Das Flaschenpfand beider Kästen beträgt 0,15 Euro pro Flasche, das Kastenpfand 1,50 Euro. Anschließend soll der Preis der Bestellung berechnet und auf der Standardausgabe ausgegeben werden.

1,0	1,0	0,7
1,0	0,7	0
0,7	0	0

```
3  /**
4   * „Implementieren Sie eine Klasse Kasten zur Beschreibung eines
    ↪  Getränkekastens
5   * mit den folgenden Eigenschaften. Entscheiden Sie dabei jeweils ob
    ↪  eine
6   * Realisierung als Objekt- oder Klassenfeld sinnvoll ist."
7   */
8  public class Kasten {
9    /**
```

```
10      * Wir verwenden static, also ein sogenanntes Klassenfeld, da das
    ↪  Kastenpfand
11      * für alle Kästen gleich ist: „Es existiert ein einheitliches
    ↪  Kastenpfand in
12      * Höhe von 1,50 Euro."
13      */
14     static double kastenPfad = 1.5;
15
16     /**
17      * Wir verwenden ein Objektfeld, d.h. ein nicht statisches Feld:
    ↪  „Für alle
18      * Flaschen in einem Kasten gelte ein einheitliches Flaschenpfand,
    ↪  das jedoch
19      * von Kasten zu Kasten verschieden sein kann."
20      */
21     double flaschenPfad;
22
23     /**
24      * „Während das Flaschenpfand für alle Flaschen eines Kastens gleich
    ↪  ist, sind
25      * die Einzelpreise der Flaschen je nach Inhalt unterschiedlich. Die
26      * Einzelpreise (ohne Flaschenpfand) der im Kasten enthaltenen
    ↪  Flaschen sollen
27      * in einem 2-dimensionalen Array abgelegt werden."
28      */
29     double[][] flaschen;
30
31     /**
32      * „sowie eine einfach verkettete Liste der bestellten
    ↪  Getränkekästen. "
33      */
34     Kasten nächsterKasten = null;
35
36     /**
37      * „Geben Sie für die Klasse Kasten einen geeigneten Konstruktor
    ↪  an."
38      *
39      * @param flaschen     Die Belegung des Kasten mit Flaschen als
40      *                     zweidimensionales Feld der Flaschenpreise
    ↪  ohne
41      *                     Flaschenpfand.
42      * @param flaschenPfad Die Höhe des Flaschenpfads, dass für alle
    ↪  Flaschen in
43      *                     diesem Kasten gleich ist.
44      */
45     public Kasten(double[][] flaschen, double flaschenPfad) {
46       this.flaschen = flaschen;
47       this.flaschenPfad = flaschenPfad;
48     }
49
50     /**
51      * „Ergänzen Sie in der Klasse Kasten eine Objektmethode zur
    ↪  Berechnung des
52      * Gesamtpreises des Getränkekastens inklusive Kasten- und
    ↪  Flaschenpfand."
53      *
54      * @return Der Gesamtpreis des Getränkekastens inklusive Kasten- und
55      *         Flaschenpfand.
56      */
57     double berechneGesamtPreis() {
58       double gesamtPreis = kastenPfad;
```

```
59      for (int i = 0; i < flaschen.length; i++) {
60        double[] reihe = flaschen[i];
61        for (int j = 0; j < reihe.length; j++) {
62          double flaschenPreis = flaschen[i][j];
63          // Nur im Kasten vorhandene Flaschen kosten auch
            ↪  Flaschenpfand.
64          if (flaschenPreis > 0)
65            gesamtPreis += flaschenPfad + flaschen[i][j];
66        }
67      }
68      return gesamtPreis;
69    }
70  }
```

```
3   import java.text.DecimalFormat;
4
5   /**
6    * „Schreiben Sie eine Klasse Bestellung"
7    */
8   public class Bestellung {
9     /**
10     * „Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet
    ↪  werden, die
11     * über den Konstruktoraufruf erstellt wird."
12     */
13    int bestellNummer;
14
15    /**
16     * „Außerdem soll zu jeder Bestellung der Name des Kunden
    ↪  gespeichert werden."
17     */
18    String kundenName;
19
20    /**
21     * „sowie eine einfach verkettete Liste der bestellten
    ↪  Getränkekästen. "
22     */
23    Kasten kästen = null;
24
25    /**
26     * „Jeder Bestellung soll eine eindeutige Bestellnummer zugeordnet
    ↪  werden, die
27     * über den Konstruktoraufruf erstellt wird. Außerdem soll zu jeder
    ↪  Bestellung
28     * der Name des Kunden gespeichert werden."
29     *
30     * @param bestellNummer Die Nummer der Getränkebestellung.
31     * @param kundenName    Der Name des/der KundenIn.
32     */
33    public Bestellung(int bestellNummer, String kundenName) {
34      this.bestellNummer = bestellNummer;
35      this.kundenName = kundenName;
36    }
37
38    /**
39     * „Die Klasse Bestellung soll weiterhin eine Methode beinhalten,
    ↪  die den
40     * Gesamtpreis der Bestellung ermittelt."
41     *
42     * @return Der Gesamtpreis der Getränkebestellung.
43     */
```

```
44    double berechneGesamtPreis() {
45      double gesamtPreis = 0;
46      Kasten kasten = kästen;
47      while (kasten != null) {
48        gesamtPreis += kasten.berechneGesamtPreis();
49        kasten = kasten.nächsterKasten;
50      }
51      return gesamtPreis;
52    }
53
54    /**
55     * Nicht verlangt. Könnte auch in die Test-Methode geschrieben
      ↪ werden.
56     *
57     * @param flaschen     Die Belegung des Kasten mit Flaschen als
58     *                     zweidimensionales Feld der Flaschenpreise
      ↪ ohne
59     *                     Flaschenpfand.
60     * @param flaschenPfad Die Höhe des Flaschenpfads, dass für alle
      ↪ Flaschen in
61     *                     diesem Kasten gleich ist.
62     */
63    void bestelleKasten(double[][] flaschen, double flaschenPfad) {
64      Kasten bestellterKasten = new Kasten(flaschen, flaschenPfad);
65      if (kästen == null) {
66        kästen = bestellterKasten;
67        return;
68      }
69      Kasten kasten = kästen;
70
71      Kasten letzterKasten = null;
72      while (kasten != null) {
73        letzterKasten = kasten;
74        kasten = kasten.nächsterKasten;
75      }
76      letzterKasten.nächsterKasten = bestellterKasten;
77    }
78
79    /**
80     * Kleines Schmankerl. Nicht verlangt. Damit wir nicht
      ↪ 9.899999999999999 als
81     * Aufgabe bekommen.
82     *
83     * @param preis Ein Preis als Gleitkommazahl.
84     *
85     * @return Der Preis als Text mit zwei Stellen nach dem Komma.
86     */
87    static String runde(double preis) {
88      DecimalFormat df = new DecimalFormat("#.##");
89      df.setMinimumFractionDigits(2);
90      return df.format(preis);
91    }
92
93    /**
94     * Die main-Methode soll hier als Testmethode verwendet werden:
95     *
96     * „Schreiben Sie ein kleines Testprogramm, das eine Bestellung
      ↪ erstellt, die
97     * zwei Getränkekästen umfasst. Der erste Kasten soll ein 1 x 1
      ↪ Getränkekasten
```

```
98      * mit einer Flasche zu 0,75 Euro sein, der zweite Kasten soll - wie
    ↪  in
99      * Abbildung 1 dargestellt - ein 3 x 3 Getränkekasten mit 3 Flaschen
    ↪  zu 0,7 Euro
100     * auf der Diagonalen und 3 weiteren Flaschen zu je 1 Euro sein. Das
101     * Flaschenpfand beider Kästen beträgt 0,15 Euro pro Flasche, das
    ↪  Kastenpfand
102     * 1,50 Euro. Anschließend soll der Preis der Bestellung berechnet
    ↪  und auf der
103     * Standardausgabe ausgegeben werden."
104     *
105     * @param args Kommandozeilenargumente, die uns nicht zu
    ↪  interessieren brauchen.
106     */
107    public static void main(String[] args) {
108      Bestellung bestellung = new Bestellung(1, "Hermine Bschlangaul");
109
110      // Müsste eigentlich nicht mehr gesetzt werden, da wir es schon in
         ↪  der
111      // Klassendefinition gesetzt haben.
112      Kasten.kastenPfad = 1.50;
113
114      bestellung.bestelleKasten(new double[][] { { 0.75 } }, 0.15);
115      bestellung.bestelleKasten(new double[][] { { 1.0, 1.0, 0.7 }, {
         ↪  1.0, 0.7, 0 }, { 0.7, 0, 0 } }, 0.15);
116
117      // Oder kürzer
118      // bestellung.bestelleKasten(new double[][] { { 1, 1, .7 }, { 1,
         ↪  .7 }, { .7 } }, .15);
119
120      // Gegenrechnung:
121      // 1 x 0.75 = 0.75
122      // 3 x 1.00 = 3.00
123      // 3 x 0.70 = 2.10
124      // 7 x 0.15 = 1.05 (Flaschenpfad)
125      // 3 x 1.50 = 3.00 (Kastenpfand)
126      // ----
127      // 9.90
128
         ↪  System.out.println("Der Gesamtpreis der Getränkebestellung beträgt: "
         ↪  + runde(bestellung.berechneGesamtPreis()) + " €");
129    }
130  }
```