1 Allgemeine Aufgabenbeschreibung

In dieser Übung sollen Sie die Grundlagen der objektorientierten Denkweise und Programmierung erlernen. Sie sollen dazu ein einfaches Shop-System entwickeln und auf diesem beispielhafte Einkaufsvorgänge modellieren.

In dem zu entwickelnden Shop-System werden Klassen für `Kunden`, `Artikel` und `Warenkörbe` modelliert und benutzt. Wie in einem „ganz modernen“ Markt benötigen wir keinen expliziten Verkäufer.

2 Erstellen der Projektstruktur

Erstellen und Implementieren Sie die folgenden Packages und Klassen:

- Packages `main` und `shop`
- Klasse `main.Main` implementiert die `main()`-Methode
- Klassen `shop.Artikel`, `shop.Kunde`, `shop.Warenkorb`

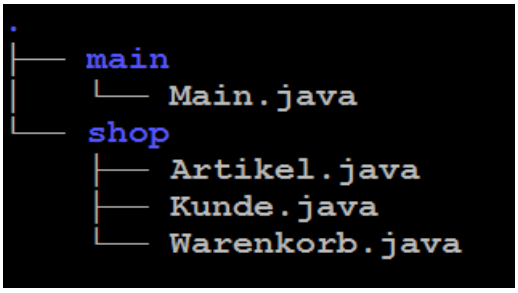

Abbildung 1: Ordner- und Dateistruktur im src-Verzeichnis

3 Klasse `shop.Kunde` 6P.

Erstellen und vervollständigen Sie die Klasse `shop.Kunde` um die Variablen und Methoden, welche in den folgenden Unterabschnitten beschrieben sind!

3.1 Variablen

In der Klasse sollen folgende Objekt-Variablen definiert werden:

- `private String name`
- `private Warenkorb wkorb`

  Die Variable `wkorb` soll bei Erzeugung eines neuen `Kunde`-Objekts mit einem neuen Objekt vom Typ `shop.Warenkorb` initialisiert werden, welches keine Artikel enthält.

3.2 Konstruktor `Kunde()`

Implementieren Sie den Konstruktor `public Kunde(String name)`

Parameter: Name des Kunden `String name`

3.3 Methode `getWkorb()` 1P.

Implementieren Sie die Methode `public Warenkorb getWkorb()`.

Rückgabe: interner Warenkorb

Beschreibung: Die Methode soll den Warenkorb des Kunden, also das Attribut `wkorb` als Rückgabewert liefern.

3.4 Methode `getName()` 1P.

Implementieren Sie die Methode `public String getName()`.

Rückgabe: Name des Kunden

Beschreibung: Die Methode soll den Namen des Kunden, also das Attribut `name` als Rückgabewert liefern.

3.5 Methode `kauft()` 1P.

Implementieren Sie die Methode `public void kauft(Artikel a)`.

Parameter: den zu kaufenden Artikel

Beschreibung: Die Methode soll den übergebenen Artikel `a` in den internen Warenkorb `wkorb` des Kunden einfügen.

3.6 Methode `bezahlt()` 3P.

Implementieren Sie die Methode `public double bezahlt()`.

Rückgabe: Gesamtpreis des Einkaufs inklusive Versandkosten

Beschreibung: Die Methode rechnet den zu zahlenden Gesamtpreis für den Einkauf des Kunden aus und gibt diesen zurück.

Der zu zahlende Gesamtpreis soll aus der Summe aller Artikelpreise im Warenkorb des Kunden zzgl. einer gestaffelten Versandpauschale nach folgender Tabelle berechnet werden:

Warenwert	Versandkosten	Bewertung
0.01-9.99 EUR	6 EUR	**0.5P**
10-29.99 EUR	4 EUR	**0.5P**
Die Klasse soll folgende Attribute besitzen: 30-99.99 EUR	3% des Warenwerts	**0.5P**
ab 100 EUR	frei	**0.5P**

Der Gesamtpreis soll nach Addition der Versandkosten gerundet auf zwei Nachkommastellen zurückgegeben werden. **1P.**

4 Klasse `shop.Artikel` 2P.

Erstellen und vervollständigen Sie die Klasse `shop.Artikel` um die Variablen und Methoden, welche in den folgenden Unterabschnitten beschrieben sind!

4.1 Variablen

In der Klasse sollen folgende Objekt-Variablen definiert werden:

- `String bezeichnung`
- `double preis`

4.2 Konstruktor `Artikel()`

Implementieren Sie den Konstruktor `public Artikel(String bezeichnung, double preis)`

Der Konstruktor `public Artikel(String bezeichnung, double preis)` soll als Parameter Artikelbezeichnung und -preis entgegennehmen und in den zugehörigen internen Variablen speichern.

4.3 Methode `getBezeichnung()` 1P.

Implementieren Sie die Methode `public String getBezeichnung()`.

Rückgabe: Artikelbezeichnung

Beschreibung: Die Methode soll die gespeicherte Artikelbezeichnung als Rückgabewert liefern.

4.4 Methode `getPreis()` 1P.

Implementieren Sie die Methode `public double getPreis()`.

Rückgabe: Artikelpreis

Beschreibung: Die Methode soll den gespeicherten Artikelpreis als Rückgabewert liefern.

5 Klasse `shop.Warenkorb` 6P.

Erstellen und vervollständigen Sie die Klasse `shop.Warenkorb` um die Variablen und Methoden, welche in den folgenden Unterabschnitten beschrieben sind!

5.1 Variablen

In der Klasse sollen folgende Objekt-Variablen definiert werden:

- `private int anzahl = 0`
- `private Artikel artFeld[]`

  Das Array `artFeld[]` speichert die sich im Warenkorb befindlichen Objekte vom Typ `Artikel`. Dieses soll maximal 100 `Artikel` enthalten können.

5.2 Methode `add()` 3P.

Implementieren Sie die Methode `public void add(Artikel a)`.

Parameter: in den Warenkorb zu legenden Artikel `a`

Beschreibung: Diese Methode fügt den als Parameter übergebenen Artikel `a` dem Warenkorb hinzu. Der Artikel soll intern im Array `artFeld` gespeichert werden und danach die Anzahl `anzahl` der im Warenkorb befindlichen Artikel inkrementiert werden. **1P.**

Sollte kein Platz im Warenkorb sein, d.h. die maximale Anzahl `Artikel` ist erreicht, wird das Hinzufügen ignoriert. **1P.**

Außerdem sollen keine `null`-Referenzen (Parameter `a = null`) hinzugefügt werden. **1P.**

5.3 Methode `getSumme()` 2P.

Implementieren Sie die Methode `public double getSumme()`.

Rückgabe: Gesamtbetrag aller Artikelpreise

Beschreibung: Die Methode gibt die Summe aller Artikelpreise im Warenkorb zurück.

5.4 Methode `getAnzahl()` 1P.

Implementieren Sie die Methode `public int getAnzahl()`.

Rückgabe: Anzahl der Artikel im Warenkorb

Beschreibung: Die Methode gibt die Anzahl der Artikel im Warenkorb zurück.

6 Zusatzaufgabe `Warenkorb.remove()` 6P.

Implementieren Sie die Methode `public void remove(Artikel a)`.

Parameter: der zu entfernende Artikel `a`

Beschreibung: Mittels der Methode sollen alle `Artikel` im Warenkorb die identisch bzw. gleich `a` sind entfernt werden. Die Erhaltung der Reihenfolge der verbleibenden Artikel ist nicht notwendig. **3P.**

Zwei `Artikel` sollen als identisch/gleich betrachtet werden, wenn sowohl die Artikelnamen als auch die Preise (beide Attribute müssen übereinstimmen) beider `Artikel` identisch sind. **2P.**

Die Verwendung der Methode soll als ausnahmesicher gelten, d. h. die Verwendung soll keine Ausnahmen (`Exceptions`) auslösen. **1P.**

Hinweis: Führen Sie beim Entfernen von Artikeln eine Reorganisation des Arrays im Warenkorb durch, sodass keine Leerplätze entstehen. Eine mögliche Variante ist, gelöschte Plätze sofort durch den aktuell letzten Artikel im Array wieder zu belegen (Verdichtungsstrategie).

7 Bewertung

Aufgabe	Voraussetzung	Punkte
Programm compilierbar	X	-
Klassen, Packages, Methoden, Signaturen richtig benannt/umgesetzt	X	-
Klasse `shop.Kunde`		6P.
Klasse `shop.Artikel`		2P.
Klasse `shop.Warenkorb`		6P.
Zusatzaufgabe Methode `Warenkorb.remove()`		6P.
Gesamtpunkte 100%:		14P.
maximale Gesamtpunkte 143%:		20P.

8 Anwendungsbeispiel Einkaufsvorgänge

Mit den Klassen kann z.B. folgendes Einkaufsszenario abgebildet werden. Die Implementierung des Hauptprogramms ist angezeigt:

```
1 public class Main {
2   public static void main(String [] args){
```

Zwei neue Kunden mit den Namen Peter und Ina werden angelegt. Ina kauft sofort ein Kleid für 199,99 Euro.

```
3       Kunde p = new Kunde ("Peter");
4       Kunde i = new Kunde ("Ina");
5       i.kauft(new Artikel("Kleid",199.99));
```

Für den Artikel wird hier keine extra Variable vom Typ Artikel angelegt, was man aber auch machen könnte, wie man am Deo für 46 Euro, das Peter kauft, sehen kann.

```
6       Artikel d = new Artikel("Deo",46.0);
7       p.kauft(d);
```

Der Unterschied der beiden Vorgehensweisen ist, dass wir weitergehend eine Referenz auf den Artikel mit der Bezeichnung Deo über die Variable `d` haben und ihn somit auch anderweitig verwenden könnten, während Inas Kleid nur über deren Warenkorb erreichbar ist.

Nun kauft Ina noch eine Kette für 499,00 Euro und Peter ein Smartphone für 600 Euro.

```
1       i.kauft(new Artikel("Kette",499.));
2       p.kauft(new Artikel("Smartphone",600.));
```

Das könnte lange so weitergehen und irgendwann wird bezahlt. Dafür werden die Summen der Preise der beiden Kunden ermittelt, ebenso die Anzahl der entsprechenden Artikel und die summarischen Rechnungstexte werden erstellt und gedruckt.

```
3       String petersRechnungsText;
4       String inasRechnungsText;
5       double petersPreis = p.bezahlt();
6       double inasPreis = i.bezahlt();
7       int petersAnzahl = p.getWkorb().getAnzahl();
8       int inasAnzahl = i.getWkorb().getAnzahl();

10      petersRechnungsText = p.name + " bezahlt " + petersPreis + "
           ➥fuer " + petersAnzahl + " Artikel.";
11      inasRechnungsText   = i.name + " bezahlt " + inasPreis + "
           ➥fuer " + inasAnzahl + " Artikel.";

13      System.out.println(petersRechnungsText);
14      System.out.println(inasRechnungsText);
15    }
16  }
```

Als Ausgabe erhalten wir folgende Texte:

Peter bezahlt 646,00 Euro fuer 2 Artikel.

Ina bezahlt 698,99 Euro fuer 2 Artikel.