P1 - Serie 06

Implementationsaufgaben

(01) Buchbestellung Klasse `Order` erstellen:

- Laden Sie die Datei `Book.java` von ILIAS (→ Serie 6 Vorlagen → Aufgabe 1). Verwenden Sie nicht die Datei aus Serie 3.
- Erstellen Sie eine Klasse `Order` für Buchbestellungen, bestehend aus einer id, Kundennamen (`customerName`), Kundenadresse (`customerAddress`) und beliebig vielen `Book` -Objekten.
- Die Klasse soll `toString()` und `addBook(...)` Methoden enthalten.
- Implementieren Sie einen Konstruktor `Order()` , der die `id` automatisch vergibt (1 für das erste Objekt, 2 für das zweite, usw.), mittels einer `static` -Variablen.
- Testen Sie die Klasse `Order` mit der bereitgestellten Klasse `Test` von ILIAS (Übungen → Serie 6 → Serie 6 Vorlagen → Aufgabe 1). Die Ausgabe sollte exakt wie im Beispiel aussehen.

Beispiel Ausgabe:

```
Order id: 1, Customer: Sophie Muster, Mittelstrasse 10, 3011 Bern
1, Homo Faber, Max Frisch, 01.01.1957
2, Harry Potter, J.K. Rowling, 25.07.2000
...
```

Hinweise:

- Implementieren Sie nur die tatsächlich verwendeten Getter und Setter.
- Die Klasse `Test` darf nicht verändert werden.

```java
public class Order {
    private static int idNumber = 1;

    private int id = idNumber;
    private String customerName;
    private String customerAdress;
    private ArrayList<Book> books = new ArrayList<Book>();

    public Order() {
        this.id = idNumber;
        Order.idNumber += 1;
    }

    public void addBook(Book book) {
        this.books.add(book);
    }

    public String toString() {
        String string = "Order id: " + this.getId() + " Customer: " + this.getCustomerName() +
                        ", " + this.getCustomerAdress() + "\n";
        for(Book b : this.books) {
            string += "" + b.toString() + "\n";
        }
        return string;
    }
}
```

```java
public String toString() {
    String string = "Order id: " + this.getId() + " Customer: " + this.getCustomerName() +
                    ", " + this.getCustomerAdress() + "\n";
    for(Book b : books) {
        string += "" + b.toString() + "\n";
    }
    return string;
}
```

(02) Implementierung der Klasse `Store`

- Laden Sie `Store.java` und `Book.java` (nicht dieselbe Datei wie in Teilaufgabe 1) von ILIAS herunter.
- Das Programm `Store` verfügt über ein Menü zur Erfassung neuer Bestellungen (Bücher, DVDs, CDs).
- Ihre Aufgabe ist es, dafür zu sorgen, dass das Programm `Store` einwandfrei funktioniert, ohne `Store` selbst zu verändern.
- Sie müssen folgende Klassen und Schnittstellen programmieren:
  (a) Schnittstelle `IArticle` mit den Methoden `getId()`, `getPrice()` und `getDescription()`. Passen Sie die Klasse `Book` so an, dass sie `IArticle` implementiert.
  (b) Erstellen Sie die Klassen `DVD` und `CD`, die beide `IArticle` implementieren. `CD` soll einen Interpreten statt eines Autors haben, DVD kein Feld `author`.
  (c) Passen Sie die Klasse aus Aufgabe 1 an. Die erforderlichen Methoden für `Order` können Sie aus der Klasse `Store` entnehmen.

Beispiel Menüausgabe `Store`:

```
============================================================
| 1. Create a new order 2. Show all registered articles |
| 3. Show all orders 9. Exit |
============================================================
What do you want to do? 1
1 (Book) Die Blechtrommel, by Guenter Grass, 1959, 29 CHF
...
Enter the customer's name: Susi Meier
Enter the customer's address: Mittelstrasse 10, 3011 Bern
```

Insbesondere muss Order eine Methode `getOrderedArticles()` besitzen. Definieren sie dessen Rückgabetyp als `Iterable<IArticle>`.
(d) Zeichnen Sie ein UML-Klassendiagramm aller involvierten Klassen und Schnittstellen.

Interface

```java
public interface IArticle {
        public int getPrice();
        public int getId();
        public String getDescription();
}
```

Bsp. Book

```java
public class Book implements IArticle {
    private int id;
    private String title;
    private String author;
    private int year;
    private int price; // CHF

    /** constructor */
    public Book(int id, String title, String author, int year, int price) {
        this.id = id;
        this.title = title;
        this.author = author;
        this.year = year;
        this.price = price;
    }

    public String getDescription() {
        return this.id + " (Book) " + this.title + ", by " + this.author +
               ", " + this.year + ", " + this.price + " CHF";
    }

    public int getPrice() {
        return this.price;
    }

    public int getId() {
        return this.id;
    }
}
```

Klasse Order

```java
public class Order {
    private static int idNumber = 1;
    private int id = idNumber;
    private String customerName;
    private String customerAddress;
    private ArrayList<IArticle> articles = new ArrayList<IArticle>();

    public Order() {
        this.id = idNumber;
```

```
        Order.idNumber += 1;
    }

    public void add(IArticle a) {
        this.articles.add(a);
    }

    public String getTotalPrice() {
        int sum = 0;
        for(IArticle a : this.articles) {
            sum += a.getPrice();
        }
        return String.valueOf(sum);
    }

    public Iterable<IArticle> getOrderedArticles() {
        return this.articles;
    }

    public String toString() {
    String string = "Order id: " + this.id + " Customer: " + this.customerName + ", " +
                    this.customerAddress + "\n";
    for (IArticle a : this.articles) {
        string += a.getDescription() + "\n";
    }
    return string;
        }
}
```

(03) Preisberechnungssystem einer Möbelfirma

Sie sollen für eine Firma, welche Möbel herstellt, ein System entwickeln. Die Firma will sehen, ob sich dieses System bewährt, deshalb soll zunächst nur die Preisberechnung für die Tische implementiert werden. Falls die Firmenleitung zufrieden ist, sollen mehr Möbelstücke und mehr Funktionalitäten integriert werden. Implementieren Sie für dieses System `Furniture.java`, `Material.java` und `Table.java` nach dem folgenden UML-Diagramm (FurnitureTest.java ist auf Ilias verfügbar):

Anweisungen:
(a) Die Klassen sollen nach diesem UML-Diagramm implementiert werden ohne zusätzliche Variablen oder Methoden zu verwenden.
(b) Vergeben Sie jedem Objekt in der Aufzählung einen anderen Preis pro Quadratmeter (speichern Sie diese Information in einer Instanzvariablen)
(c) Erstellen Sie einen passenden Konstruktor für `Furniture` und `Table`. Der Konstruktor von `Table` soll dabei den Konstruktor von `Furniture` verwenden.
(d) Die Variable `pricePerHour` gibt die Kosten pro Stunde an für die Anfertigung des Möbelstückes. `workedHours` gibt die Anzahl Stunden an, welche nötig waren, um das Möbelstück fertigzustellen. Die Methode `calculateEffort()` soll nun den Aufwand für die Herstellung eines Möbelstückes berechnen.
(e) Da in der Aufwandsberechnung noch nicht der Materialpreis inbegriffen ist, soll in der Methode `totalPrice()` zunächst mithilfe der Methode aus `Furniture` den Preis für den Aufwand berechnet werden. Danach wird der Materialpreis berechnet (gegeben durch multipliziert mit dem Quadratmeterpreis des Materials) und zum Aufwand dazu addiert.

Klasse `Furniture`

```
import java.text.DecimalFormat;

public class Furniture {
    public Material material;
    protected double pricePerHour;
    protected double workedHours;

    DecimalFormat form = new DecimalFormat("0.## CHF");

    public Furniture(Material material, double pricePerHour, double workedHours) {
        this.material = material;
        this.pricePerHour = pricePerHour;
        this.workedHours = workedHours;
    }

    public double calculateEffort() {
        return this.pricePerHour * this.workedHours;
    }
}
```

Enum `Material`

```
public enum Material {
    Esche(2.5),
    Buche(1.8),
    Eiche(5.5);

    private double materialCost;

    private Material(double materialCost) {
        this.materialCost = materialCost;
    }

    public double materialCost() {
        return this.materialCost;
    }
}
```

Klasse `Table`

```
public class Table extends Furniture {
    private double area;

    public Table(Material mat, double pricePerHour, double workedHours, double area) {
        super(mat, pricePerHour, workedHours);
        this.area = area;
    }

    public double totalPrice() {
        return this.calculateEffort() + this.material.materialCost() * this.area;
    }
}
```