Programmieraufgabe "Artikelverwaltung"

Programm zur Verwaltung von Artikeln eines Online-Shops.

1. Abstrakte Klasse *Artikel*

Die abstrakte Klasse *Artikel* dient zum Speichern von Informationen über Artikel. Dabei sollen u.a. folgende Daten als private Instanzvariablen erfasst und notwendige öffentliche Zugriffsmethoden (*set…* und *get…*) erstellt werden:

- *Id* (Artikelnummer - Zahl, eindeutig, aber nicht notwendigerweise fortlaufend)
- *Titel*
- *Erscheinungsdatum*
- *Verlag*
- *Grundpreis (Euro)*

Wählen Sie passende Datentypen.

Es soll ein Konstruktor implementiert werden, der es ermöglicht, die entsprechenden Instanzvariablen direkt zu setzen. Prüfen Sie, ob es sich um plausible Werte handelt. (z.B., das *Erscheinungsjahr* darf nicht in der Zukunft liegen, etc.)

Sollten Bedingungen nicht erfüllt sein, werfen Sie eine *IllegalArgumentException* mit einer vorgegebenen Fehlermeldung (siehe Punkt 7).

Die Methode *alter()* soll das Alter des Artikels (in ganzen Jahren) berechnen.

Zusätzlich soll es eine Methode *preis()* geben, die den Preis eines Artikels auf Basis des Grundpreises abzüglich eines Rabatts berechnet (siehe Punkt 2). Zu diesem Zweck soll eine abstrakte Methode *rabatt()* zur Klasse Artikel hinzugefügt und innerhalb von *preis()* verwendet werden.

2. Klassen Buch und DVD

Es sollen zwei konkrete Unterklassen *Buch* und *DVD* von der abstrakten Klasse *Artikel* abgeleitet werden.

Die Klasse *Buch* hat eine zusätzliche private Instanzvariable und eine entsprechende Zugriffsmethode für die Seitenanzahl. Die Klasse *DVD* hat zusätzliche Instanzvariablen sowie entsprechende Zugriffsmethode für die Spieldauer (in Minuten) und die Altersfreigabe. Mögliche Altersfreigaben sind: 0 (keine Altersbeschränkung), 6 (ab sechs Jahren), 12 (ab zwölf Jahren), 16 (ab sechzehn Jahren), 18 (ab achtzehn Jahren).

Für ein Buch erhält man einen Rabatt, der vom Alter abhängt: pro Jahr einen Rabatt von 5% (maximal 30%); für Bücher mit mehr als 1000 Seiten einen zusätzlichen Rabatt von 3%. Für eine DVD ist der Rabatt abhängig von der Altersfreigabe: keine Altersbeschränkung - 20%, ab 6 Jahren - 15%, ab 12 Jahren - 10%, ab 16 Jahren - 5%, ab 18 Jahren - 0%.

Die Methode *toString()* soll überschrieben werden, sodass gemäß dem vorgegebenen Format (siehe Punkt 6) alle Artikeldaten als String zurückgegeben werden.

3. Interface *ArtikelDAO*

Dieses Interface definiert Methoden zum Zugriff auf die Artikeldaten unabhängig von der konkreten Implementierung der persistenten Datenspeicherung. (vgl. *Data Access Object*).

Das Interface *ArtikelDAO* enthält abstrakte Methoden zum Einlesen und Speichern von Artikelobjekten.

- Die Methode *getArtikel()* gibt alle persistent gespeicherten Artikelobjekte als *java.util.List* zurück.

- Die Methode *getArtikel(int ...)* gibt anhand der Artikelnummer ein Artikelobjekt zurück. Falls der Artikel nicht gefunden wird, soll *null* zurückgeben werden.

- Die Methode *saveArtikel(Artikel …)* soll ein Artikelobjekt persistent speichern. Stellen Sie sicher, dass beim Speichern eines neuen Artikels nicht die *Id* eines bereits gespeicherten Artikels verwendet wird. Werfen Sie in diesem Fall eine *IllegalArgumentException* mit einer entsprechenden Fehlermeldung (siehe Punkt 7).

- Die Methode *deleteArtikel(int …)* soll einen Artikel aus der persistenten Datenquelle löschen. Falls es den Artikel nicht gibt, soll eine *IllegalArgumentException* mit entsprechender Fehlermeldung (siehe Punkt 7) geworfen werden.

Anmerkung: das Interface *ArtikelDAO.java* ist vorgegeben und darf/kann nicht geändert werden.

4. Klasse *SerializationArtikelDAO*

Die Klasse *SerializationArtikelDAO* implementiert das Interface ArtikelDAO und soll als Singleton-Klasse realisiert werden.

Implementieren Sie die persistente Speicherung der Artikeldaten in einer Datei mittels Java *Object Serialization.*

Gewährleisten Sie, dass die Deserialisierung der Daten von der Datei nur einmal erfolgt.

Bei Fehlern aus Dateioperationen soll eine einzeilige Fehlermeldung

- "Fehler bei Serialisierung." oder
- "Fehler bei Deserialisierung."

ausgegeben werden und das Programm mit *System.exit(1)* beendet werden.

5. Klasse *Artikelverwaltung*

Die Klasse *Artikelverwaltung* soll die Business-Logik implementieren. Die Klasse soll eine private Instanzvariable vom Typ *ArtikelDAO* besitzen, um auf die Artikeldaten zugreifen zu können.

Implementieren Sie Methoden, die folgende Funktionalität realisieren.

- Alle Daten aller Artikel bereitstellen
- Alle Daten eines Artikels bereitstellen
- Einen neuen Artikel hinzufügen
- Einen bestehenden Artikel löschen
- Gesamtanzahl aller Artikel ermitteln
- Durchschnittlichen Preis aller Artikel ermitteln

- Id(s) des/der ältesten Artikel ermitteln

6. Frontend

Schreiben Sie ein Java-Programm *ArtikelverwaltungClient* das unter Verwendung der Klasse *Artikelverwaltung* die nachfolgend beschriebene Kommandozeilenschnittstelle bereitstellt. Achten Sie darauf, dass die Programmausgaben den unten angeführten Beispielen **exakt** entsprechen!

Das Programm soll Aufrufe in folgendem Format unterstützen:

```
java ArtikelverwaltungClient <Datei> <Parameter>
```

<Datei>: Name der Datei für die Serialisierung. Falls die Datei nicht existiert, soll sie erstellt werden.

<Parameter>: add, list, delete, count, meanprice, oldest. Pro Aufruf kann jeweils nur einer dieser Parameter angegeben werden.

- Parameter 'add buch <id> <titel> <verlag> <erscheinungsdatum> <grundpreis> <seiten>'
  - Buch persistent hinzufügen

Beispielaufruf:

```
java ArtikelverwaltungClient <Datei> add buch 1 "Ein ganzes
    Leben" Hanser 2014 17.99 767
```

Ausgabe:

```
Info: Artikel 1 added.
```

- Parameter 'add dvd <id> <titel> <verlag> <erscheinungsdatum> <grundpreis> <spieldauer> <altersfreigabe>'
  - DVD persistent hinzufügen

Beispielaufruf:

```
java ArtikelverwaltungClient <Datenquelle> add dvd 2 "Die
    Biene Maja - Box 1" Studio100 2013 14.99 190 0
```

Ausgabe:

```
Info: Artikel 2 added.
```

- Parameter 'list'
  - Alle Daten aller Artikeln ausgeben

Beispielaufruf:

```
java ArtikelverwaltungClient <Datei> list
```

Ausgabe:

```
Typ:         Buch
Id:          1
Titel:       Ein ganzes Leben
Jahr:        2014
Verlag:      Hanser
Grundpreis:  17.99
Preis:       13.49
Seiten:      767

Typ:         DVD
Id:          2
```

```
Titel:      Die Biene Maja - Box 1
Jahr:       2013
Verlag:     Studio100
Grundpreis: 14.99
Preis:      11.99
Dauer:      180
Freigabe:   0
```

- Parameter 'list <id>'
  - Alle Daten eines Artikels ausgeben

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> list 2
```
Ausgabe:
```
Typ:        DVD
Id:         2
Titel:      Die Biene Maja - Box 1
Jahr:       2013
Verlag:     Studio100
Grundpreis: 14.99
Preis:      11.99
Dauer:      180
Freigabe:   0
```

- Parameter 'delete <id>'
  - Artikel löschen

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> delete 2
```
Ausgabe:
```
Info: Artikel 2 deleted.
```

- Parameter 'count'
  - Gesamtzahl der erfassten Artikel berechnen

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> count
```
Ausgabe:
```
1
```

- Parameter 'meanprice'
  - Durchschnittlichen Preis aller Artikel berechnen

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> meanprice
```
Ausgabe:
```
12.74
```

- Parameter 'oldest'
  - Älteste(n) Artikel suchen

Beispielaufruf:

```
java ArtikelverwaltungClient <Datei> oldest
```
Ausgabe:
```
Id: 2
Id: 6
```

7. Fehlermeldungen

Alle auf Grund ungültiger oder fehlerhaften Eingaben geworfenen *Exceptions* müssen abgefangen und das Programm mit einer Fehlermeldung beendet werden. Es darf dabei immer **nur eine der folgenden Fehlermeldungen** ausgegeben werden:

- "Error: Parameter ungueltig."
- "Error: Erscheinungsjahr ungueltig."
- "Error: Altersfreigabe ungueltig."
- "Error: Artikel bereits vorhanden. (id=<id>)"
- "Error: Artikel nicht vorhanden. (id=<id>)"

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> add buch 1 "Ein ganzes
    Leben" Hanser 20.14 17.99 767
```
Ausgabe:
```
Error: Parameter ungueltig.
```

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> add buch 1 "Ein ganzes
    Leben" Hanser 2014 17.99
```
Ausgabe:
```
Error: Parameter ungueltig.
```

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> add buch 1 "Ein ganzes
    Leben" Hanser 2104 17.99 767
```
Ausgabe:
```
Error: Erscheinungsjahr ungueltig.
```

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> delete 4711
```
Ausgabe:
```
Error: Artikel nicht vorhanden. (id=4711)
```

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> list 4711
```
Ausgabe:
```
Error: Artikel nicht vorhanden. (id=4711)
```

Beispielaufruf:
```
java ArtikelverwaltungClient <Datei> add dvd 2 "Die Biene Maja
    - Box 1" Studio100 2013 14.99 190 5
```
Ausgabe:
```
Error: Altersfreigabe ungueltig.
```

8. Achten Sie auch auf folgende Punkte:

- Eingaben
  Falls Parameter Leerzeichen enthalten, können Sie " " verwenden (siehe Beispielaufruf 'add').
- Ausgaben
  Gleitkommazahlen immer mit '.' (Punkt) als Dezimaltrenner und genau 2 Stellen hinter dem Komma ausgeben. z.B.: 12.35
  Benutzen Sie dafür z.B. die Methode `Artikel.getDecimalFormat()`
- Sämtliche Exceptions müssen abgefangen werden.

Beispiel:

```
double grundpreis = 12.345;
DecimalFormat df = Artikel.getDecimalFormat();
System.out.println "Grundpreis: " + df.format(grundpreis));
```

Ausgabe:

```
Flaeche:        12.35
```

9. Abgabemodalitäten

Abgabetermin: **Mittwoch, 13.11.2019 12:00** auf der Online-Abgabeplattform

Verwenden Sie als Basis zur Entwicklung Ihres Programms die im Archiv ***ArtikelverwaltungPLC19WS.zip*** enthaltenen Java Klassen/Interfaces. Ändern Sie keinesfalls die Klassennamen oder das Interface *ArtikelDAO* und belassen Sie alle Files im Default-Package.

Für eine positive Abgabe ist das lauffähige (Java 1.8) Programm mit allen hier genannten Anforderungen **rechtzeitig vor dem Abgabetermin** auf der Online-Plattform erfolgreich abzugeben. Weitere Informationen dazu in den VU-Einheiten und auf Moodle.