Assignment "Article Management"

Implement an application for managing media articles of an online shop.

1. Abstract Class `Article`

The abstract class **`Article`** is used to store information about articles. The following information should be captured using private instance variables and necessary public access methods (`set… and get…`) should be created:

- *Id* (Article number - a number, unique but not necessarily consecutive)
- *Title*
- *Release year*
- *Publisher*
- *Base price (in Euro)*

Choose appropriate data types.

A constructor should be implemented to allow direct setting of the corresponding instance variables. Check whether the values are plausible (e.g., the release year should not be in the future, etc.).

If conditions are not met, throw an **`IllegalArgumentException`** with a predefined error message (see Section 7).

The method **`getAge()`** should calculate the age of the article (in whole years).

Additionally, there should be a method **`getPrice()`** that calculates the price of an article based on the base price minus a discount (see Section 2). For this purpose, an abstract method **`getDiscount()`** should be added to the Article class and used within **`getPrice()`**.

2. Classes `Book` and `DVD`

Two concrete subclasses, **`Book`** and **`DVD`** should be derived from the abstract class **`Article`**.

The **`Book`** class should have an additional private instance variable for the number of **pages** and a corresponding accessor method.

The **`DVD`** class should have additional instance variables for the **length** (in minutes) and the **age rating**, along with corresponding accessor methods. Possible age ratings are: 0 (no age restriction), 6 (for ages six and up), 12 (for ages twelve and up), 16 (for ages sixteen and up), 18 (for ages eighteen and up).

For a **book**, you receive a discount depending on its age: a 5% discount per year (maximum 30%); for books with more than 1000 pages, an additional 3% discount is applied. For a **DVD**, the discount depends on the age rating: no age restriction - 20% discount, for ages 6 and up - 15% discount, for ages 12 and up - 10% discount, for ages 16 and up - 5% discount, for ages 18 and up - 0% discount.

The **`toString()`** method should be overridden so that it returns all article data as a string according to the specified format (see Section 6).

3. Interface `ArticleDAO`

This interface specifies abstract methods for storing, retrieving, and deleting articles independently of how persistent storage is implemented. (cf. Data Access Object).

The interface **`ArticleDAO`** provides the following abstract methods:

- **`getArticleList()`** return all stored articles as a **`java.util.List`**.
- **`getArticle(int …)`** return an article object based on the article number. If the article is not found, it should return **`null`**.
- **`saveArticle(Article …)`** persistently store an article object. Ensure that when saving a new article, the ID of an already saved article is not used. If this happens, throw an **`IllegalArgumentException`** with an appropriate error message (see Section 7).
- **`deleteArticle(int …)`** delete an article from the persistent storage. If the article doesn't exist, it should throw an **`IllegalArgumentException`** with an appropriate error message (see Section 7).

Note: the interface **`ArticleDAO.java`** is provided and should not be modified.

4. Class `SerializedArticleDAO`

The class **`SerializedArticleDAO`** implements the **`ArticleDAO`** interface. The constructor of the class shall have an argument of type String for passing the name of the file.

Implement persistent storage of article data in a file using **Java Object Serialization**. In case of errors during file operations, a one-line error message should be displayed:

- "Error during serialization." or
- "Error during deserialization."

The program should then be terminated with **`System.exit(1).`**

5. Class `ArticleManagement`

The class **`ArticleManagement`** should implement the business logic. By means of *dependency injection* the class shall obtain an object of type **`ArticleDAO`** to manage article data.

Implement methods that achieve the following functionality:

- Return all data of all articles.
- Return all data of a specific article.
- Add a new article.
- Delete an article.
- Determine the total number of all articles.
- Determine the total number of all books.
- Determine the total number of all dvds.
- Determine the mean price of all articles.
- Determine the ID(s) of the oldest article(s).

6. Class `ArticleCLI`

Implement a Java program **`ArticleCLI`**, which realizes a command line interface (CLI) as described below. Ensure that the output of your program corresponds **exactly** to the examples shown below.

The program **`ArticleCLI`** is to be executed as follows:

```
java ArticleCLI <file> <command>
```

`<file>:` Name of the file used for persistent storage. If the file doesn't exist, it shall be created.

`<command>: add, list, delete, count, meanprice, oldest`. Only one of these parameters can be specified for each invocation.

- Command 'add book <id> <title> <publisher> <release year> <base price> <pages>'
  - Persistently add the book.

Example:

```
java ArticleCLI <file> add book 1 "Ein ganzes Leben" Hanser 2014
    17.99 767
```

Output**:**

```
Info: Article 1 added.
```

- Command 'add dvd <id> <title> <publisher> <release year> <base price> <length> <age rating>'
  - Persistently add the dvd.

Example:

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

Output:

```
Info: Article 2 added.
```

Example:

```
java ArticleCLI <file> add dvd 23 "Die Biene Maja - Box 2" Studio100
    2013 17.99 195 0
```

Output:

```
Info: Article 23 added.
```

- command 'list'
  - Print data of all stored articles.

Example:

```
java ArticleCLI <file> list
```

Output:

```
Type:        Book
Id:          1
Title:       Ein ganzes Leben
Year:        2014
Publisher:   Hanser
Base price:  17.99
Price:       12.59
Pages:       767

Type:        DVD
```

```
Id:          2
Title:       Die Biene Maja - Box 1
Year:        2013
Publisher:   Studio100
Base price:  14.99
Price:       11.99
Length:      190
Age rating:  0

Type:        DVD
Id:          23
Title:       Die Biene Maja - Box 2
Year:        2013
Publisher:   Studio100
Base price:  17.99
Price:       14.39
Length:      195
Age rating:  0
```

- Command 'list <id>'
  - Print all data of an article.

Example:

```
java ArticleCLI <file> list 2
```

Output:

```
Type:        DVD
Id:          2
Title:       Die Biene Maja - Box 1
Year:        2013
Publisher:   Studio100
Base price:  14.99
Price:       11.99
Length:      190
Age rating:  0
```

- Command 'delete <id>'
  - Delete an article.

Example:

```
java ArticleCLI <file> delete 1
```

Output:

```
Info: Article 1 deleted.
```

- Command 'count'
  - Determine total number of all articles.

Example:

```
java ArticleCLI <file> count
```

Output:

```
2
```

- Command 'count' <type>
  - Determine total number of all books or dvds.

Example:
```
java ArticleCLI <file> count book
```
Output:
```
0
```

Example:
```
java ArticleCLI <file> count dvd
```
Output:
```
2
```

- Command 'meanprice'
  - Determine the average price of all articles.

Example:
```
java ArticleCLI <file> meanprice
```
Output:
```
13.19
```

- Command 'oldest'
  - Determine the oldest article(s).

Example:
```
java ArticleCLI <file> oldest
```
Output:
```
Id: 2
Id: 23
```

7. Error Messages:

All exceptions due to invalid user input have to be caught and the program has to be terminated with one of the following error messages:

- "Error: Invalid parameter."
- "Error: Invalid release year."
- "Error: Invalid age rating."
- "Error: Article already exists. (id=<id>)"
- "Error: Article not found. (id=<id>)"

Example:
```
java ArticleCLI <file> add book 1 "Ein ganzes Leben" Hanser 20.14
    17.99 767
```
Output:
```
Error: Invalid parameter.
```

Example:
```
java ArticleCLI <file> add book 1 "Ein ganzes Leben" Hanser 2014
    17.99
```
Output:
```
Error: Invalid parameter.
```

Example:
```
java ArticleCLI <file> add book 1 "Ein ganzes Leben" Hanser 2104
    17.99 767
```
Output:
```
Error: Invalid release year.
```

Example:
```
java ArticleCLI <file> delete 4711
```
Output:
```
Error: Article not found. (id=4711)
```

Example:
```
java ArticleCLI <file> list 4711
```
Output:
```
Error: Article not found. (id=4711)
```

Example:
```
java ArticleCLI <file> add dvd 2 "Die Biene Maja - Box 1" Studio100
    2013 14.99 190 5
```
Output:
```
Error: Invalid age rating.
```

8. Additional Remarks:

- **Inputs**
  If a parameter contains spaces, quotation marks can be used (see example 'add').
- **Outputs**
  Floating point numbers have to be formatted with the decimal delimiter '.' and exactly two decimal places. For formatting floating point numbers, the provided method **Article.getDecimalFormat()** can be used.
- All exceptions should be caught.

Example:
```
double price = 12.345;
DecimalFormat df = Article.getDecimalFormat();
System.out.println("Price:      " + df.format(price));
```
Output:
```
Price:      12.35
```

9. Submission Guidelines

Deadline: **Wednesday, 8.11.2023 12:00**

For developing your code, the java classes/interfaces provided in **Assignment1PLC23WS** can be used. The names of classes and the interface **ArticleDAO** must not be modified. All classes and interfaces should be left in the default package.

The **program has to be submitted before the deadline on the online platform** after it has passed all checks. Further information is provided in the lectures, tutorials and on Moodle.