Assignment 2 - Requirements

In this assessment, you will create a program that stores information about a DVD collection.

The program must do the following:

1. Allow the user
   * + - 1. to add a DVD to the collection

Instantiate a dvd object

Add to ArrayList

* + - * 1. remove a DVD from the collection

Removing from the Array/ ArrayList

* + - * 1. edit the information for an existing DVD in the collection

Use the setter if needed

* + - * 1. to list the DVDs in the collection

Print the entire Array/ArrayList

For loop / foreach

* + - * 1. display the information for a particular DVD

Find by ID

* + - * 1. to search for a DVD by title

Find by Title

1. Load the DVD library from a file
2. Save the DVD library back to the file when the program completes
3. Allow the user to add, edit, or delete many DVDs in one session

Additionally, the program must follow the MVC design pattern and use dependency injection as shown in the course material.

You should follow the process outlined in the *Agile Approach Checklist for Console Applications* document elsewhere in this course.

Your DVD data transfer object should have the following fields:

|  |  |
| --- | --- |
| **Variable** | **Datatype** |
| Title | String |
| Release date | Date / String |
| MPAA rating | Int / String / char |
| Director's name | String |
| Studio | String |
| User rating or note | String |
| DVDS | Static Arraylist |

Marking Criteria

|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Specifications**: Apprentice applies the specifications to the application, including the use of custom classes, using multiple objects effectively, and dependency injection. | Meets Expectations | Needs Improvement | No Credit |
| 15points | 8points | 0points |
| 1. **MVC**: The application uses the MVC pattern appropriately. | Meets Expectations | Needs Improvement | No Credit |
| 10points | 5points | 0points |
| 1. **I/O Operations**: The application can perform I/O operations to a file to store and retrieve data. | Meets Expectations | Needs Improvement | No Credit |
| 10points | 5points | 0points |
| 1. **List/Map**: The application uses a List or Map to hold data in memory. | Meets Expectations | Needs Improvement | No Credit |
| 10points | 5points | 0points |
| 1. **Java Syntax**: The application uses proper Java syntax and constructs. | Meets Expectations | Needs Improvement | No Credit |
| 10points | 5points | 0points |
| 1. **Dependency Injection**: Apprentice can explain the relationship between dependency injection and loosely-coupled code. | Meets Expectations | Needs Improvement | No Credit |
| 5points | 3points | 0points |
| 1. **OOP Concepts**: Apprentice can explain object-oriented programming, including classes and objects. | Meets Expectations | Needs Improvement | No Credit |
| 5points | 3points | 0points |
| 1. **Interfaces**: Apprentice can explain what an interface is using examples from the code. | Meets Expectations | Needs Improvement | No Credit |
| 5points | 3points | 0points |
| 1. **Inheritance**: Apprentice can explain inheritance using examples from the code. | Meets Expectations | Needs Improvement | No Credit |
| 5points | 3points | 0points |
| 1. **Composition**: Apprentice can describe the use of composition, using examples from the code. | Meets Expectations | Needs Improvement | No Credit |
| 5points | 3points | 0points |
| 1. **Agile**: Apprentice can explain the use of Agile as an approach to software development. | Meets Expectations | Needs Improvement | No Credit |
| 5points | 3points | 0points |
| 1. **Data Marshaling:** Apprentice can explain data marshaling and unmarshaling. | Meets Expectations | Needs Improvement | No Credit |
| 5points | 3points | 0points |
| 1. **Code Style**: Code is written with appropriate indents, naming conventions, and comments so that other developers can read the code easily. | Meets Expectations | Needs Improvement | No Credit |
| 10points | 5points | 0po |