Overview

The purpose of this assessment is to demonstrate your proficiency in basic Java syntax involving console input and output, basic file input and output, basic string/text manipulation, flow of control statements, expressions, and basic data structures such as arrays, Lists, and Maps. Additionally, you will demonstrate your proficiency in implementing the MVC design pattern and dependency injection.

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 to **add** a DVD to the collection
2. Allow the user to **remove** a DVD from the collection
3. Allow the user to **edit** the information for an existing **DVD** in the collection
4. Allow the user to **list the DVDs** in the collection
5. Allow the user to **display** the information for a particular **DVD**
6. Allow the user to **search for a DVD by title**
7. **Load the DVD library** from a file
8. **Save the DVD library back to the file** when the program completes
9. 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:

1. **Title**
2. **Release date**
3. **MPAA rating**
4. **Director's name**
5. **Studio**
6. **User rating or note (allows the user to enter additional information, e.g., "Good family movie")**

Ideas for implementation

Controller, View, Model (DVD), and intermediate classes.

The intermediate classes will handle extracting DVD objects and writing DVD objects to the file - DVDReader, DVDWriter. The DVDWriter is used after any changes (creating, editing, removing DVDs) to rewrite the file.

The View (CollectionView) will be just for displaying content to the user. It has many methods to just print out statements to the user - the idea being that if it were expanded to a UI, code in the Controller would not have to be changed - just the View code.

The Controller (CollectionController) will coordinate and process requests.

The DVD class defines the DVD object.

Five custom exceptions are used to describe different problems the user could face.

The system **searches by exact movie title** as they are stored as keys in the hashtable - easy and fast access. Because of this hashtable, if a user enters a new movie with the same name, it will overwrite the previous entry.

It seems strange that after creation you would need to edit anything except the user note / review so my edit functionality allows just that.

The collection can be loaded from any file specified by the user. There is error checking for missing data but only to the extent that at least six pieces of data on each line exists and that the mpaaRating is written as an integer / floating point number.

The CollectionView implements the CollectionViewInterface. The CollectionController implements the CollectionControllerInterface. The CollectionController has a CollectionViewInterface which is dependency injection? This is so that it can take any kind of view as long as it implements CollectionViewInterface.