CS2043 Assignment 2 Programming Review

Due: WED JAN-26 2022 by 4PM

Try this assignment mostly on your own. If you want help, go to the lab on Fri JAN-21. Feel free to talk amongst each other about your solutions, but don't share code with anyone. A quick look at a screen during a conversation is fine. Anything beyond that is cheating.

Even if you don't finish, submit what you get done by the due date. You won't get credit for submitting just anything, but if its worth assessing, you will get credit, even if you don't pass. Note that solutions which are unreasonably similar to other submissions will be reported as an academic offense. Also, solutions which are barely complete, or approached poorly will not be assessed.

System Description:

Two classes that work to provide a simple way to to keep track of movie favorites: `Movie`, and `MovieStore`.

The `Movie` class is defined by a movie title (eg. "Star Wars") which cannot be changed once it is specified, and must be specified on instantiation. Provide a way to add ratings to movies, so that `Movies` can provide an average rating when called upon to do so (a movie can be rated multiple times). Valid ratings are integers between `0` and `5` inclusive, and `Movies` should only accept valid ratings. `Movies` with no ratings should return `null` when asked for their average rating. Also provide a way to clear ratings (but don't bother with functionality required to remove individual ratings).

The `MovieStore` should store movies. This inventory is allowed to be empty. `MovieStore` must provide a way to add `Movies`. `Movies` in the `MovieStore` must be unique, based solely on their titles (ie two `Movies` with the same title cannot exist in the list). `MovieStore` must also provide a way to return 1) all the `Movies` in a list (but protect the `MovieStore` inventory by returning a copy of the list), and 2) the top N rated `Movies` (N should be caller specified). For N larger than the list, all `Movies` in the list should be returned. `Movies` that have no ratings are assumed to have the lowest ratings. For now, when ties increase the length of the list beyond N, any choice to complete the list will do (this is an *ambiguity*, but we won't deal with this now).

Remember
No comments, No CLI, Use whatever Java has available, and follow our Course Coding Standards

Question 1: Design and implement `Movie` and `MovieStore`. To demonstrate the functionality of your classes, create a `MovieStoreDriver`. The driver should do the following in order:

→ create a MovieStore.

→ create a Movie with the name "Star Wars"
→ add "Star Wars" to the MovieStore
→ add a rating of 4 to the "Star Wars" movie, then add another rating of 4, and then another of 5

→ create a Movie with the name "Star Trek"
→ add "Star Trek" to the MovieStore
→ add a rating of 2 to the "Star Trek" movie

→ create a Movie with the name "A Few Good Men", while adding it to the MovieStore

→ create a Movie with the name "Enders Game"
→ add a rating of 5 to the "Enders Game" movie
→ add "Ender's Game" to the MovieStore

→ create a second Movie with the name "Start Trek" while trying to add it to the MovieStore

→ Get a list of all the Movies and display them formatted according to the example

→ Get a list of all the top 2 rated Movies and display them formatted according to the example

```
All Movies -  4 in total:
Star Wars, Ave Rating:  4.3
Star Trek, Ave Rating:  2.0
A Few Good Men
Ender's Game, Ave Rating:  5.0

Top 2 Rated Movies:
Ender's Game, Ave Rating:  5.0
Star Wars, Ave Rating:  4.3
```

HINTS: To make it easier to check for movies in a list, consider overriding equals(). To make it easier to order movies by rating, consider implementing Comparable, or adding a Comparator

continued…

Question 2: Add some unit tests to your solution. For all your tests, be sure to use meaningful test names. Consider both happy paths and alternate paths.

- ☑ For the `Movie` class, include tests that aim to expose failures in functionality related to ratings. Be sure to include a test case for adding ratings above the valid rating range, and one for adding ratings below (you can do this in one Parameterized test if you want), but include at least 3 tests that do something other than testing for inputs out of bounds.
- ☑ For the `MovieStore` class, include tests that aim to expose failures in functionality related to adding `Movies` to the store, and providing the top rated `Movie` List. Be sure to include a test case for adding a duplicate movies, but also consider empty stores, stores with less than the number of top rated requested, ordering movies with no ratings etc. You should include at least 8 independent tests.

Question 3: If you haven't already done so, add an `AllTests` class that collects your tests for `Movie`, and `MovieStore` so you can run them all at once.

Be sure to submit each of the .java files (not the .class files):

- ☐ **Movie.java**
- ☐ **MovieStore.java**
- ☐ **MovieStoreDriver.java**
- ☐ **MovieTest.java**
- ☐ **MovieStoreTest.java**

Submission Instructions

We will be marking this assignment against the course coding standards which are available in D2L under the quick links. We will also assume that you are using the Eclipse IDE to code your assignments, since that is the IDE you will be required to use during the lab exam. For subsequent assignments we will be using GitHub to submit our assignments. For this first assignment however, we will simply submit our code through D2L. Follow the instructions below to submit:

Only solutions submitted in the correct format will be assessed. Your Implementation should be saved in a package labeled **assn2.xyz** (where xyz refer to your initials - if you don't have a middle initial, include it as 'y'). If you set up packaging properly in Eclipse, the package declaration will be automatically generated at the top of each of your java files. If you didn't use Eclipse, be sure to manually declare the package as the first line in each of your files (eg. `package assn2.xyz;`) before you submit.

Submit your files to the folder for Assignment 1 in D2L. **Submit each .java file separately (DO NOT ZIP YOUR PACKAGE)**. You can submit more than once until the assignment deadline.