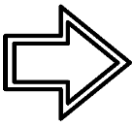
Assignment

Overview

The Retro Rewind, LLC is a movie rental start up company in the metro area. They want a program that can view the movies in their collection via top rated, most recent, or even filtering out in different years. The movies are stored in a text file and will want to continue to add to as needed. I have started the program but need help on setting up the filters and stats of the database of movies so far.

You are tasked with creating the filters and other functionality using functional interfaces and stream methods.



Directions

To Get Started:

1. Create a new java project in IntelliJ(or your preferred IDE) called YOUR_MCC_Name-Assignment7.

2. Download and extract the files needed for the assignment: **1531-Assignment7Starter.zip**
   - Place the java files in your 'src' folder
   - Place the movies.txt file in your project folder

3. Quick breakdown of the Java files

   - Movie is the object holding the title, date and rating
   - MovieCollection is the database for reading/writing the data, holding a list of the movies. You will be adding code in this file.
   - MovieSystem is the runner file that provides a menu to add, view top rated, view most recent, and see an overview of all movie ratings. You will be adding code in this file.
   - movies.txt contains the text data separated by tabs (\t) per each property. This is already setup.

Part 1: Filtering Movies

We need to be able to filter our movies out based on a set parameter. On the MovieCollection.java file created a method called filterMovies() that should use the Predicate function parameter to filter the movies list of movies, and it should return a list of Movie objects that meet the condition in the filter.

Next we have two methods in the MovieSystem.java file that needs to use the filterMovies() method. Complete the viewTopMovies() method so that it gets a list of movies with a rating of 4.0 and higher. That method should have a header and then print out the list all in the method.

```
** Movies rated 4.0 or higher **
The Sixth Sense (1999) Rating: 4.1
Jurassic Park (1993) Rating: 4.05
Rear Window (1954) Rating: 4.25
Rocky (1976) Rating: 4.0
Ford v Ferrari (2019) Rating: 4.0
Logan (2017) Rating: 4.0
The Matrix (1999) Rating: 4.35
Inception (2010) Rating: 4.4
Schindler's List (1993) Rating: 4.5
The Dark Knigh (2008) Rating: 4.5
Avengers: Endgame (2019) Rating: 4.2
Spider-Man: Into the Spider-Verse (2018) Rating: 4.2
Top Gun: Maverick (2022) Rating: 4.1
Joker (2019) Rating: 4.15
To Kill a Mockingbird (1962) Rating: 4.2
Casablanca (1942) Rating: 4.25
1917 (2019) Rating: 4.1
```

Second complete the viewRecentMovies() method that so that it gets a list of movies made within the last 5 years*(I'll leave it up to you if you want to include an extra year especially if we are in the middle of a year- for example 2023 would be 2018 BUT we could show 2017).* Again that method should print out a header and the movies. You should use a lambda expression along with the Predicate function defined above to complete this.

```
** Newer Movies (in the last 5 years) **
Ford v Ferrari (2019) Rating: 4.0
Avengers: Endgame (2019) Rating: 4.2
Spider-Man: Into the Spider-Verse (2018) Rating: 4.2
Top Gun: Maverick (2022) Rating: 4.1
Joker (2019) Rating: 4.15
The Tomorrow War (2021) Rating: 3.25
Don't Look Up (2021) Rating: 3.6
Fantastic Beasts: The Secrets of Dumbledore (2022) Rating: 3.1
A Man Called Otto (2022) Rating: 3.7
The Whale (2022) Rating: 3.65
Spiderhead (2022) Rating: 2.7
```

```
1917 (2019) Rating: 4.1
Once Upon a Time in Hollywood (2019) Rating: 3.8
```

Part 2: Movie Rating Summaries

We want to see the lowest, highest and get the average of our movie ratings in the system. On the MovieCollection.java file we have the getLowestRating(), getHighestRating(), and getAverageRating() methods which should use a map and reduce operation to get the movie ratings. Note that they all return double values so they are just getting the values and sending them back.

On the MovieSystem.java there the method viewRatings() that you need to complete. This should print a header, number of movies, lowest, highest and average rating. Your output should look like the following:

```
** Movie Rating Data **
Number of Movies: 29
Lowest Rating: 2.7
Highest Rating: 4.5
Average Rating: 3.91
```

Extra Credit:

- If you would like to simplify some of the code you could potentially combine the Predicate with a Consumer function for printing everything all at once instead of returning back a list. This does save on memory having to create a list over and over.
- The viewTopMovies() only goes from 4.0 and higher. What if we wanted the user to define the rating? You can rework the code in the method to get user input for a rating to start at instead of 4.0. Or you could even have then enter in a range, say you want movies 2.3 to 3.75, etc.
- The viewRecentMovies() does movies from 5 years back. Again, what if we want the user to define how far back to go? You can rework this method to get input for how many years back to view. Or a range? Then print out results.

Example Output(user input bolded):

```
Welcome to the Movie Ratings application
1. Enter a Movie
2. View Top Rated Movies
3. View Most Recent Movies
4. View All Movies
5. View Ratings
6. Quit Application
Choose menu option: 2

** Movies rated 4.0 or higher **
The Sixth Sense (1999) Rating: 4.1
Jurassic Park (1993) Rating: 4.05
Rear Window (1954) Rating: 4.25
Rocky (1976) Rating: 4.0
Ford v Ferrari (2019) Rating: 4.0
Logan (2017) Rating: 4.0
The Matrix (1999) Rating: 4.35
Inception (2010) Rating: 4.4
Schindler's List (1993) Rating: 4.5
The Dark Knigh (2008) Rating: 4.5
Avengers: Endgame (2019) Rating: 4.2
Spider-Man: Into the Spider-Verse (2018) Rating: 4.2
Top Gun: Maverick (2022) Rating: 4.1
Joker (2019) Rating: 4.15
To Kill a Mockingbird (1962) Rating: 4.2
Casablanca (1942) Rating: 4.25
1917 (2019) Rating: 4.1

Welcome to the Movie Ratings application
1. Enter a Movie
2. View Top Rated Movies
3. View Most Recent Movies
4. View All Movies
5. View Ratings
6. Quit Application
Choose menu option: 3

** Newer Movies (in the last 5 years) **
Ford v Ferrari (2019) Rating: 4.0
Avengers: Endgame (2019) Rating: 4.2
Spider-Man: Into the Spider-Verse (2018) Rating: 4.2
Top Gun: Maverick (2022) Rating: 4.1
Joker (2019) Rating: 4.15
The Tomorrow War (2021) Rating: 3.25
Don't Look Up (2021) Rating: 3.6
Fantastic Beasts: The Secrets of Dumbledore (2022) Rating: 3.1
A Man Called Otto (2022) Rating: 3.7
The Whale (2022) Rating: 3.65
Spiderhead (2022) Rating: 2.7
1917 (2019) Rating: 4.1
Once Upon a Time in Hollywood (2019) Rating: 3.8
```

```
Welcome to the Movie Ratings application
1. Enter a Movie
2. View Top Rated Movies
3. View Most Recent Movies
4. View All Movies
5. View Ratings
6. Quit Application
Choose menu option: 5

** Movie Rating Data **
Number of Movies: 29
Lowest Rating: 2.7
Highest Rating: 4.5
Average Rating: 3.91

Welcome to the Movie Ratings application
1. Enter a Movie
2. View Top Rated Movies
3. View Most Recent Movies
4. View All Movies
5. View Ratings
6. Quit Application
Choose menu option: 6
System exiting, movies saved out.
```

Submission

1. Compress the IntelliJ project folder and submit it to this assignment.

   to compress: On Windows, right click -> send to -> compressed .zip file

   On Mac, right-click -> Compress

Hints/Tips (Before Submitting):

- *While it might feel like streams are the way to go, you do need to show using functional interfaces with the predicate and streams in the map/reduce.*
- *Don't forget to have a header at the top of your file and include a **Resource statement**.*
- *Use comments throughout for full points.*
- *Follow all Java Styling Guides as covered*