Report 4

Maryam Soleimani Shaldehi

At first I will be explaining the “runMenu” and then I will explain every classes with their methods individually.

For options add, search , show , add to favorite , remove from favorite, search in favorite, watch , watching history and recommendation the choices for movie or tv-show will be operated individually not together.(for example for each of them again you will have a choice between movie and tv-show):

```
public static void movieOrTVshowMenu() {


    System.out.println("press 1 for TVshow");


    System.out.println("press 2 for movie");


    movieOrTVshowChoice = input.nextInt();
  }
```

More ever for search options again there are three choices (by Title / Genre / Release Year)

Before login you need to create an account.

In creating account or login if the username or password you enter already exists in the related arraylist, the system will ask you to change them and retry.

In the “runMenu” , options 1 , 2 and 3 can be reached without the need of login , but to access options 6 to 11 you must login first (if you don’t , the system will ask you to do)

By pressing “12” you won’t get out of the program totally , it is for logout. So you will only have access to options 1 to 3 (also 4 and 5) and if you want to get back your access to options 6 to 11 you need to login again.

By pressing “0” the program will stop totally.

Now the explanation about classes:

TVshow CLASS:

Here first I defined the private variables that I needed and then operated getter and setter methods for each of them (Encapsulation)

Then I defined an arraylist for the casts.

The means that I used to store the casts in the “cast” arraylist is as follows:

I defined a void method called “casts” and in that as long as the user does not enter ‘o’(in the while true) it will get the names of actors and actresses and store them in “cast”:

```
cast.add(tempCast);
```

And finally in the method TVshow I ask the user to enter the wanted information of the tv-show and then get them as inputs by using setter methods:

Example:

```
System.out.println("Enter Rating:");
setRating(input.nextLine());
```

and to get the casts as input I only need to call the “casts” method.

Movie CLASS:

This class inherited from the “TVshow” class so for defining that we use “extends”.

The only extra variable that it has comparing to “TVshow” is “length” so I defined the variable length and created the getter and setter methods for that and then in the method “Movie” I called its father by super(); and then I ask the user to enter the length and by the setter method for length I can get the length as input:

```
System.out.println("Enter Length:");
setLength(input.nextLine());
```

User CLASS:

Here we have variables “username” and “password”. They are string type and private. So then I defined the getter and setter methods and in the User method I get them as inputs.(it is the same as previous classes so there is no need for more explanation)

NetflixService CLASS:

Here first I defined two arraylists “theTVshows” and “theMovies” to store the movies and tv-shows individually. The type of “theTVshows” must be the TVshow class and for “theMovies” it must be Movie class.

In the compareTVshows method it gets two parameters with type TVshow and in the if it checks if the titles of these two parameters are the same or not (ignoring upper or lower case) and if the titles are the same it will return 0.

Then in the addTVshow method it gets variable tvshow with the type of TVshow then it loops through thTVshows arraylist with variable i with the type of TVshow and in every cells of our theTVshows arraylist which i goes , it uses the compare method to compare i and variable tvshow and if their titles were repeated it won’t let that tv-show with repeated title get added and when it happens that the compare method returns 0 so

```
if (this.compareTVshows( tvShow ,i) == 0)
```

here when it happens the message is shown that this tv-show with repeated title can not be added.

And if that doesn't happen it means that this tvshow is new and can be added to our arraylist (theTVshows) so it will be added like this:

```
theTVshows.add(tvShow);
```

compareMovie method works the same as campareTVshows.

addMovie method works the same as addTVshows.

TVshowSearchByTitle (works the same as MovieSearchByTitle):

Here I defined a string variable called title and get it as input from the user. Then I loop through theTVshow arraylist to find out if there is an item that its title (i.getTitle) is the same as variable title or not. If the answer is yes so it shows the whole information about that tv-show:

```
System.out.println(i.getTitle()
+ "\t\t" + i.getGenre() +
"\t\t" + i.getReleaseYear() +
"\t\t" + i.getDuration() +
"\t\t" + i.getRating() + "\t\t"
+ i.cast);
```

Then here I defined the flag variable with the type of int. I set its default to 0 and every time the previous condition is met it will be added by one. If it remains 0 it only has one meaning and that is that the condition hasn't met and such tv-show with such title doesn't exist.

Every other searching related methods in this class for both Movies and TVshows work with the same algorithm and if I explain them again I will just repeat myself!

showAllTVshows: here it will loop through theTvshows arraylist and prints every attributes of each item in the array with TVshow type:

```
for (TVShow i : theTVshows ) {
```

```
System.out.println(i.getTitle()
+ "\t\t" + i.getGenre() +
"\t\t" + i.getReleaseYear() +
"\t\t" + i.getDuration() +
"\t\t" + i.getRating() + "\t\t"
+ i.cast);
 }
```

ShowAllMovies works the same.

Users CLASS:

The following arraylists here are used to:

theUsers: to store info (username and password) of users.

favoriteTVshows : to store the favorite TV-shows of a specific user.

favoriteMovie: the same as favoriteTVshows but for movies.

watchHistoryTVshows: to store the tv-shows that the user already watched.

watchHistoryMovies: the same as watchHistoryTVshows but for movies.

createAccount method:

first define a variable called u from the User class(an object actually). Here it will ask for the username and password.

Then if the password that is entered by the user already exists (

```
if
(u.getPassWord().equals(i.getPassWord())){
```

)

in theUsers arraylist it prints that user with this password already exists:

```
System.out.println("user
with password " +
u.getPassWord() + " is
Already Registered.");
```

So the same as before I defined an int variable called flag and set its default to 0.

Whenever the username or password already exists it will be added by one and if it remains 0 it means that username and password are not repeated and can be added to thUsers arraylist:

```
if (flag == 0){

theUsers.add(u);

System.out.println("Account Created !");

}
```

Login: here it will get a variable with the type of user called u and gets username and password as inputs. And it loops through theUsers arraylist and in each loop it compares the username and the password that was entered by the user to the ones that existing in that item of the array. If both of them are existing at the same time in the same item of the array it means that , that account is already created and login is successful:

if (u.getUserName().equals(i.getUserName()) && u.getPassWord().equals(i.getPassWord())) {

System.out.println("Login Successfully!");

I have used variable flag for the same reason that I have already said (I don't want to repeat myself)

Then here I have Boolean variable called loginChecker. This variable is widely used in the Main class.

Whenever the login is successful it is set to true and whenever login fails it is set to false.

addFavoriteTVshow:

here first all the tv-shows are shown to the user by:

```
TVshows.showAllTVshows();
```

(here first we define TVshows variable with Netflixservice type because method showAllTVshows is in Netflix class and in order to call that we have to define such variable)

Then a title is gotten as input from the user and if such title exists in theTVshows list it can be added to the favoriteTVshows list:

```
String title = input.nextLine();


    for (TVShow i : TVshows.theTVshows ){


        if (i.getTitle().equalsIgnoreCase(title)){


            favoriteTVshows.add(i);
        }


    }


  }
```

viewFavoriteTVshows works the same as showAllTVshows.

removeFromFavoriteTVshows:

here first all favorite tv shows will be shown:

```
viewFavoritesTVshows();
```

then it will get variable title as input and searches if it exists in the favorite tv shows arraylist and if yes ,removes it:

```
for (TVShow i : favoriteTVshows){
        if (i.getTitle().equalsIgnoreCase(title)){


          favoriteTVshows.remove(i);
```

there are related methods for movies and they work the same as the above methods.

The searching methods in this class work the same as the searching methods in the Nettflix class with this difference that here they will search the favoriteTVshows(or movies) instead of the whole movies or TVshows.

getRecommendationBasedOnGenreTVshow: (works the same as getRecommendationBasedOnGenreMovie)

here I defined an arraylist to store all the genres of tv-shows that exist in the favorite tv show arraylist.

So it will look for those genres in theTVshows arraylist (which contains all of the TVshows)

And recommend them to the user.

watchTVshow (the same as watchMovie) : it gets the title of the tv-show that the user wants to watch as input and then it searches for that in theTVshows arraylist and when finds a match add it to the watchHistoryTVshows arraylist.

showWatchingHistoryForTVshows(the same as showWatchingHistoryForMovies):

here It will only loop through the watchHistoryTVshows arraylist and shows its items.

Main CLASS:

Explaining this class was difficult here so I have added some comments in my codes in this section please check there (Main class in github).