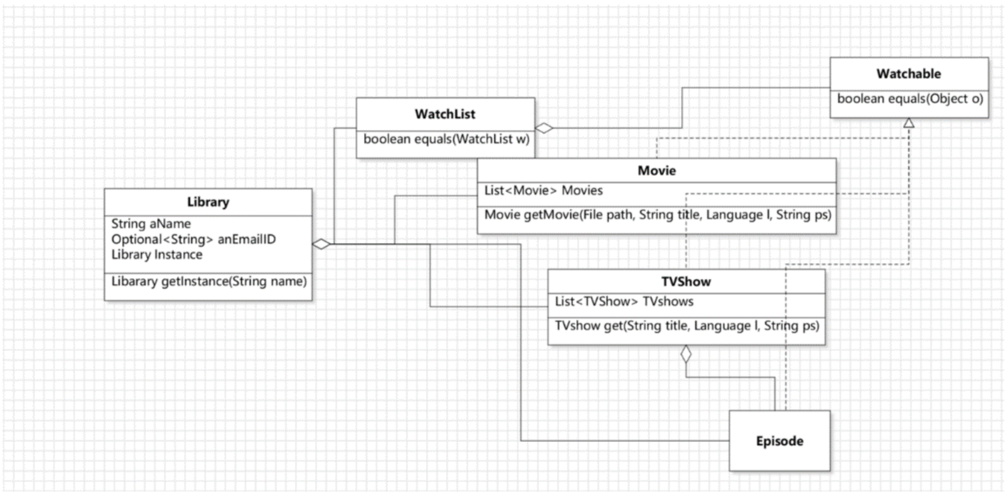
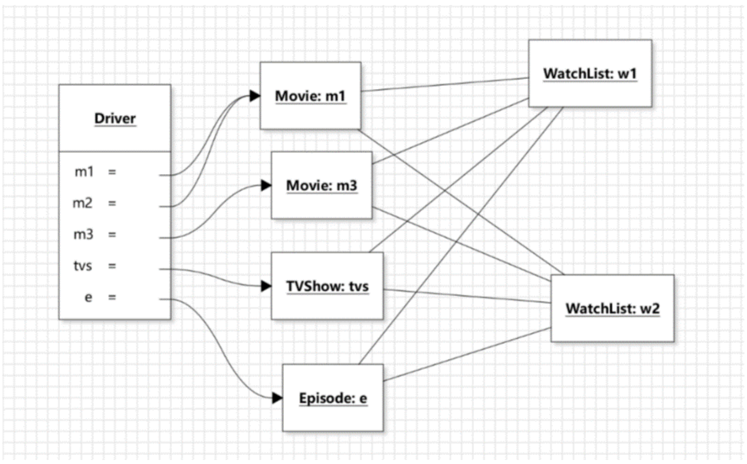
Comp 303 Assignment3

1. I used the singleton design pattern so that the client can only create one library object. When creating a library, the client calls getInstance() method and passes a String name as a parameter, so the field aName would never be null. The field anEmailID is of type Optional<String>. It prevents the client from calling getEmailID when emailID is null.
2. The flyweight design pattern is applied in this part. Since we don't know how many movies or TVshows will be created, I used lazy initialization and an arraylist is used to store all the instances. The client should call get() method to create an instance. This method will first check if there is an object in the arraylist with the given title. If so, it returns the object, if not, the method would call the private constructor to create a new instance and store it in the arraylist.
3. Since two watchlists are compared by the watchable objects they contained, I add a method equals() in the interface Watchable. Each watchable object will need to implement its own way to compare two objects. For Movie and TVShow, since they are implemented by flyweight design pattern, two equal objects must be the same object. For two Episodes, I compare their corresponding TVshow and episode_num to check if they are equal.


Watchable
boolean equals(Object o)
WatchList
boolean equals(WatchList w)
Movie
List<Movie> Movies
Movie getMovie(File path, String title, Language l, String ps)
Library
String aName
Optional<String> anEmailID
Library Instance
Libarary getInstance(String name)
TVShow
List<TVShow> TVshows
TVshow get(String title, Language l, String ps)
Episode


Driver
m1 =
m2 =
m3 =
tvs =
e =
Movie: m1
Movie: m3
TVShow: tvs
Episode: e
WatchList: w1
WatchList: w2