COMP 303 Winter 2021

Assignment 3

Belle Pan 260839939

1st March, 2021

`Library` Class

- An additional two fields were created to represent the name and email ID of the `Library` object: `aName` and `aEmailID` respectively.
    - As an email ID is optional, `aEmailID` is of type `Optional<String>`. Its getter method unwraps the value of `aEmailID` if it is present and returns it, and its setter method wraps the input `String` as an `Optional` value.
    - A name is mandatory for a `Library` object, so its representation is simply a `String`, and its getter and setter functions do not need to wrap or unwrap any values.
- Another field was added to ensure that only one `Library` object is created in the application: `aSingleInstance` of type `Library`.
    - This field stores the only existing `Library` in the application.
    - Using the **singleton design pattern** ensures that there will only be one `Library` in the application. Following the **singleton design pattern**, the constructor of the class is set to `private`, and a `Library` object may only be created using the `getInstanceOfLibrary(String pName)` method. This method ensures that the name that the client specified as input is not `null`, then checks if `aSingleInstance` already contains a `Library` object; if a `Library` already exists then it is returned, and if a `Library` does not already exist, a new `Library` is created and then returned by calling the constructor and passing in the input value as its name.

`Movie/TVShow` Classes

- These classes implement the **flyweight design pattern** in which new objects are only created if they do not already exist in the application.
- Fields `TVShowCache` and `MovieCache` in their respective classes represent a collection of `TVShows` or `Movies` as a `Map<String, Watchable>`, where the key is the name of the object and the object itself is stored as the value.
    - Each time a `Movie` or `TVShow` is to be created, the user must call their respective methods: `getTVShow(String pTitle, Language pLanguage, String pStudio)` and `getMovie(File pPath, String pTitle, Language pLanguage, String pStudio)`. There is no other way to create an object of type `Movie` or `TVShow` as their constructors are private.
    - To ensure that only one object with a specific name is created, the methods mentioned above first check to see if there is already an object with the same name in their respective caches; if there exists such an object it is simply returned, but if it does not already exist the constructor is called and a new object is created and then inserted into the cache.

`WatchList` Class

- Two additional methods are added in order to compare `WatchLists`: `equals(Object o)` and `hashCode()`. These two methods override the built-in functions of the same name in Java.
    - The `hashCode()` method returns an `int` value for a `WatchList` by summing up the hash code of all its elements. This means that two `WatchLists` with different elements will have a different hash code, and two `WatchLists` with the same elements (regardless of their order) will have the same hash code.
    - The `equals()` method returns a `Boolean` value indicating whether or not two `WatchLists` are equal. It does this by first ensuring that the `WatchList` passed in as an argument is not `null`, then by checking if the number of elements in the two `WatchLists` are the same and if they have the same hash code. If they have the same hash code, then we compare the elements in the `WatchLists` one by one using Java's built-in `hashCode()` and `equal(Object o)` method.

The diagram below shows the relationships between the interfaces and classes: