Tecnológico de Monterrey

TC1030.328

Object Oriented Programming

Integrative project

Ernesto Miranda Solís

Delivery date: June 15th, 2022

Teacher:
Sergio Ruiz Loza

Tecnológico de Monterrey
Campus de Ciudad de México

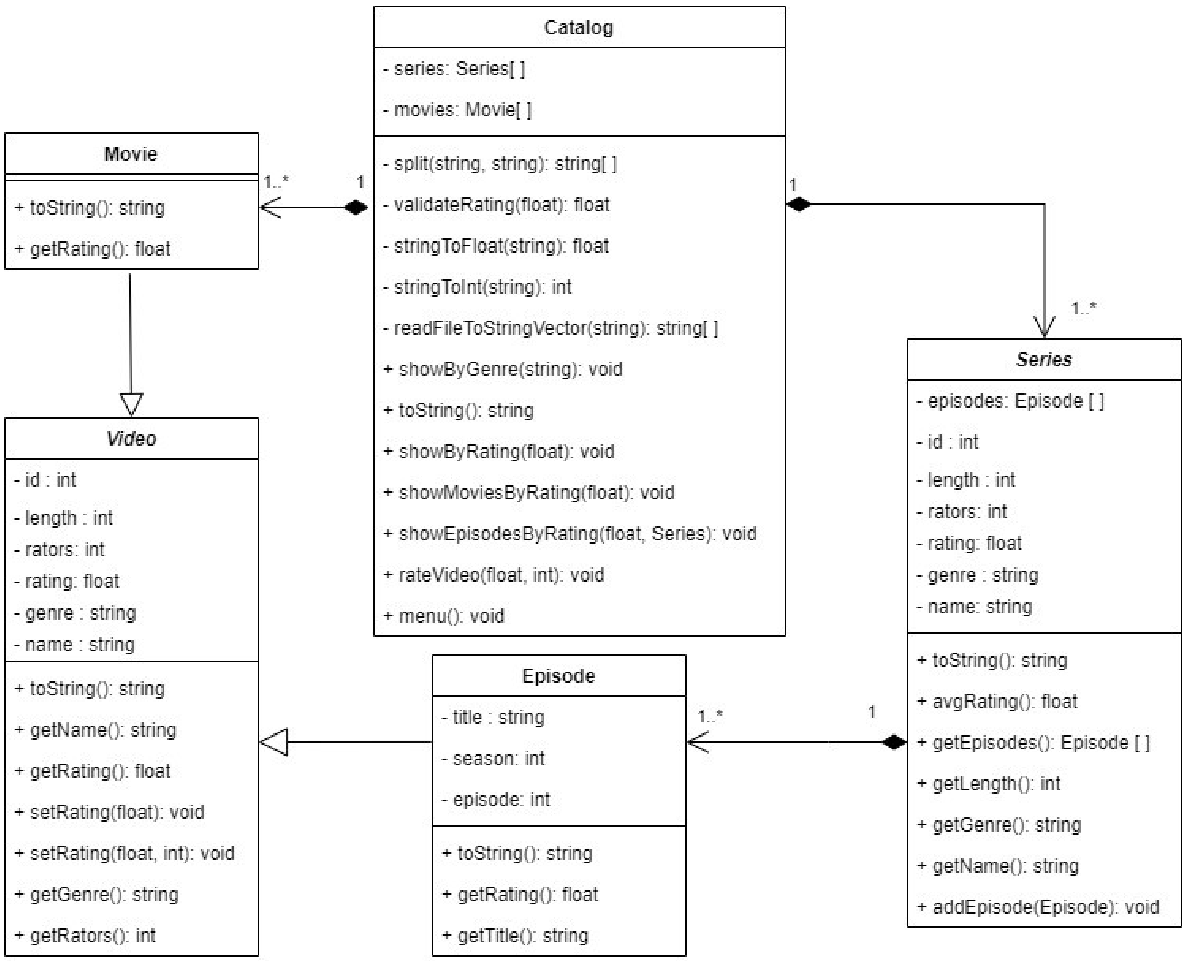
Table of contents

Introduction

This project has the scope of modelling a system that displays catalog of a contemporary streaming platform, such as Netflix, Amazon Prime Video, HBO and so on. The provider works with two main collections. A collection of movies and a collection of series. Each series has also a collection of episodes. Each video stored in the catalog has an ID, name, length and genre. Episodes have also a title and a season to which they belong. They have also a specific rating and a specific number of raters that vote to give that current rating. The rating average value goes from 1 to 5.

In this project the main concern is to show the collection of videos, the episodes of a specific series, and the movies, along with their rankings. The functionality to rank a video is also mandatory, and the user can choose to show a video according to either its ranking or its genre.

UML Diagram

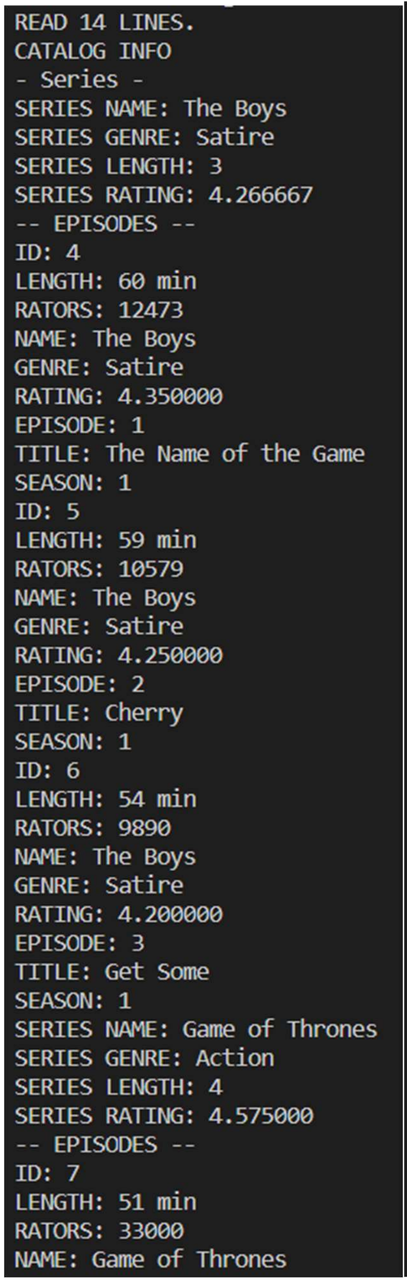


Five important classes were identified. The catalog class oversees unifying all the other ones. It handles the functionality of the program overall through the menu function. In this class are also allocated the collections of series and movies. It can read a file to obtain all the information of a catalog.

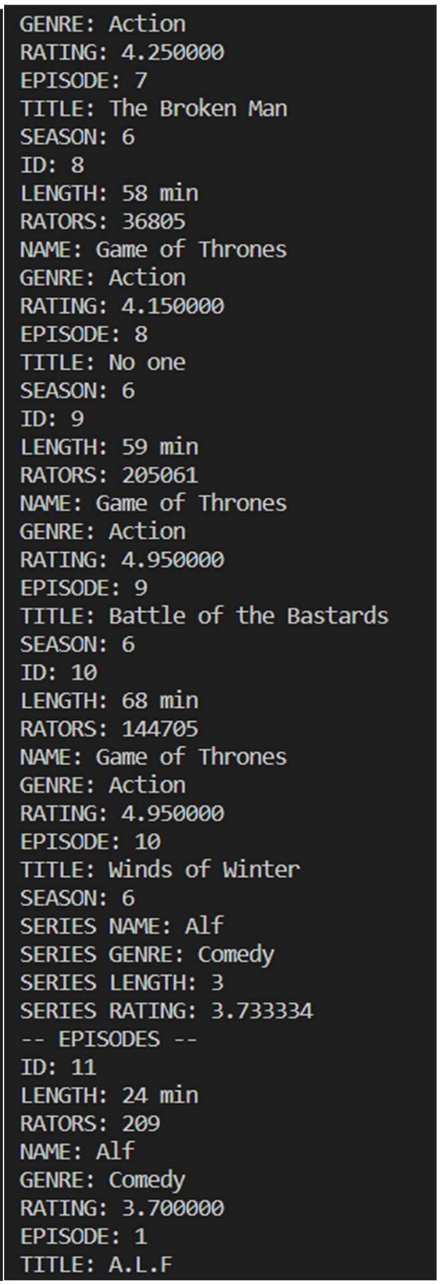
Moving forward the next important class might be the Video class. It is an abstract class from which Movie and Episode will import their attributes and methods. As their attributes are private, it provides the correct encapsulation to make this class safe. The Episode class implements also encapsulation by making its attributes private. The overridden functions from both classes are the “toString” method and the “getRating” method, therefore, episode also implements a “getTitle” function.

Finally, the Series class makes a composition of episodes and has some repeated attributes from the video class. However, a series is not a video, rather it is a collection of episodes with a different behavior as a Video.

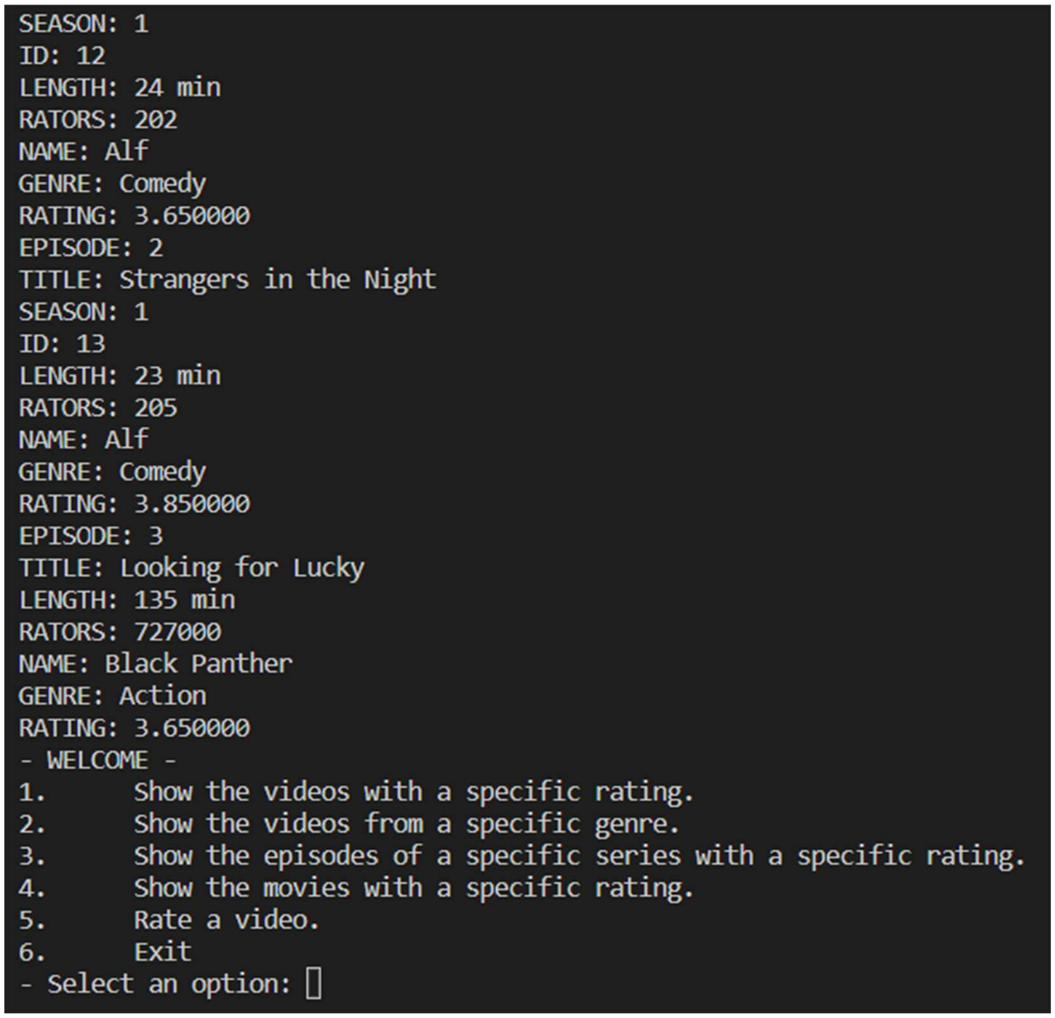
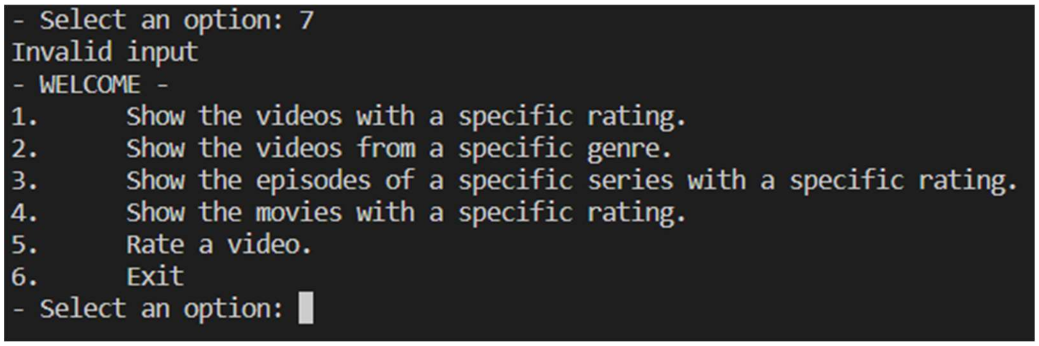
Execution example

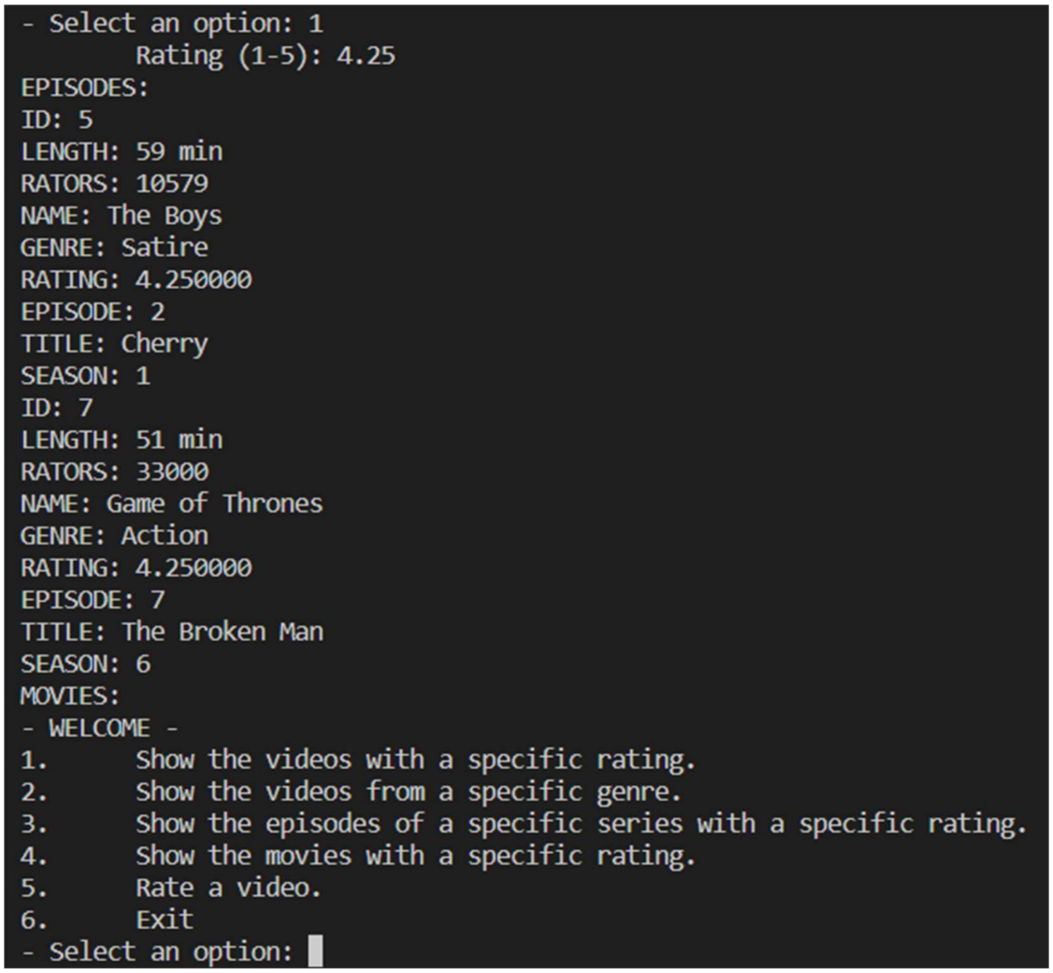


```
READ 14 LINES.
CATALOG INFO
- Series -
SERIES NAME: The Boys
SERIES GENRE: Satire
SERIES LENGTH: 3
SERIES RATING: 4.266667
-- EPISODES --
ID: 4
LENGTH: 60 min
RATORS: 12473
NAME: The Boys
GENRE: Satire
RATING: 4.350000
EPISODE: 1
TITLE: The Name of the Game
SEASON: 1
ID: 5
LENGTH: 59 min
RATORS: 10579
NAME: The Boys
GENRE: Satire
RATING: 4.250000
EPISODE: 2
TITLE: Cherry
SEASON: 1
ID: 6
LENGTH: 54 min
RATORS: 9890
NAME: The Boys
GENRE: Satire
RATING: 4.200000
EPISODE: 3
TITLE: Get Some
SEASON: 1
SERIES NAME: Game of Thrones
SERIES GENRE: Action
SERIES LENGTH: 4
SERIES RATING: 4.575000
-- EPISODES --
ID: 7
LENGTH: 51 min
RATORS: 33000
NAME: Game of Thrones
```

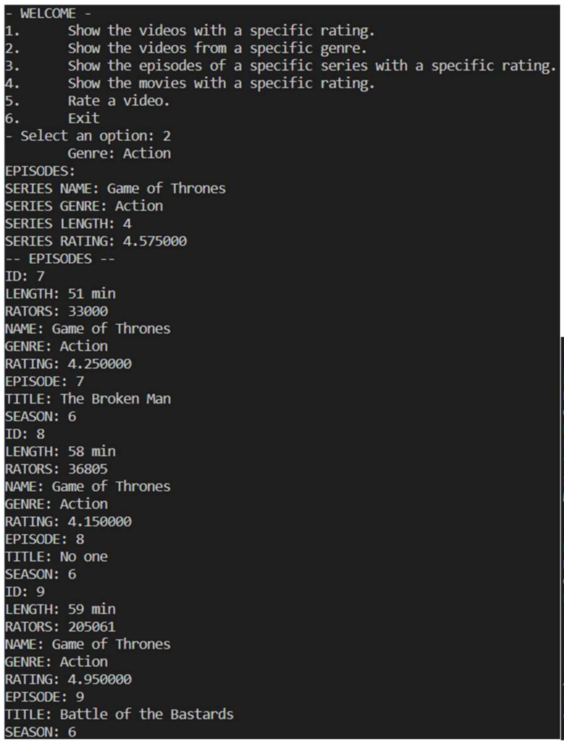


```
GENRE: Action
RATING: 4.250000
EPISODE: 7
TITLE: The Broken Man
SEASON: 6
ID: 8
LENGTH: 58 min
RATORS: 36805
NAME: Game of Thrones
GENRE: Action
RATING: 4.150000
EPISODE: 8
TITLE: No one
SEASON: 6
ID: 9
LENGTH: 59 min
RATORS: 205061
NAME: Game of Thrones
GENRE: Action
RATING: 4.950000
EPISODE: 9
TITLE: Battle of the Bastards
SEASON: 6
ID: 10
LENGTH: 68 min
RATORS: 144705
NAME: Game of Thrones
GENRE: Action
RATING: 4.950000
EPISODE: 10
TITLE: Winds of Winter
SEASON: 6
SERIES NAME: Alf
SERIES GENRE: Comedy
SERIES LENGTH: 3
SERIES RATING: 3.733334
-- EPISODES --
ID: 11
LENGTH: 24 min
RATORS: 209
NAME: Alf
GENRE: Comedy
RATING: 3.700000
EPISODE: 1
TITLE: A.L.F
```

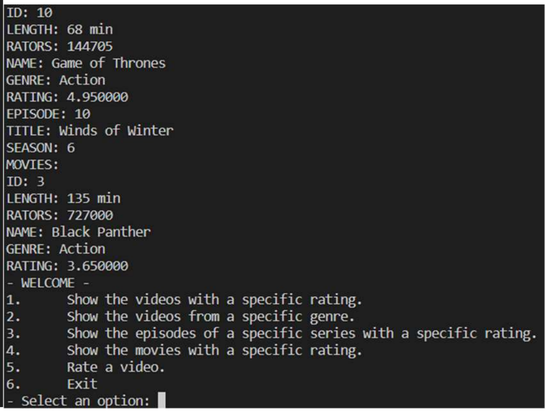

```
SEASON: 1
ID: 12
LENGTH: 24 min
RATORS: 202
NAME: Alf
GENRE: Comedy
RATING: 3.650000
EPISODE: 2
TITLE: Strangers in the Night
SEASON: 1
ID: 13
LENGTH: 23 min
RATORS: 205
NAME: Alf
GENRE: Comedy
RATING: 3.850000
EPISODE: 3
TITLE: Looking for Lucky
LENGTH: 135 min
RATORS: 727000
NAME: Black Panther
GENRE: Action
RATING: 3.650000
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 
```

```
- Select an option: 7
Invalid input
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 
```



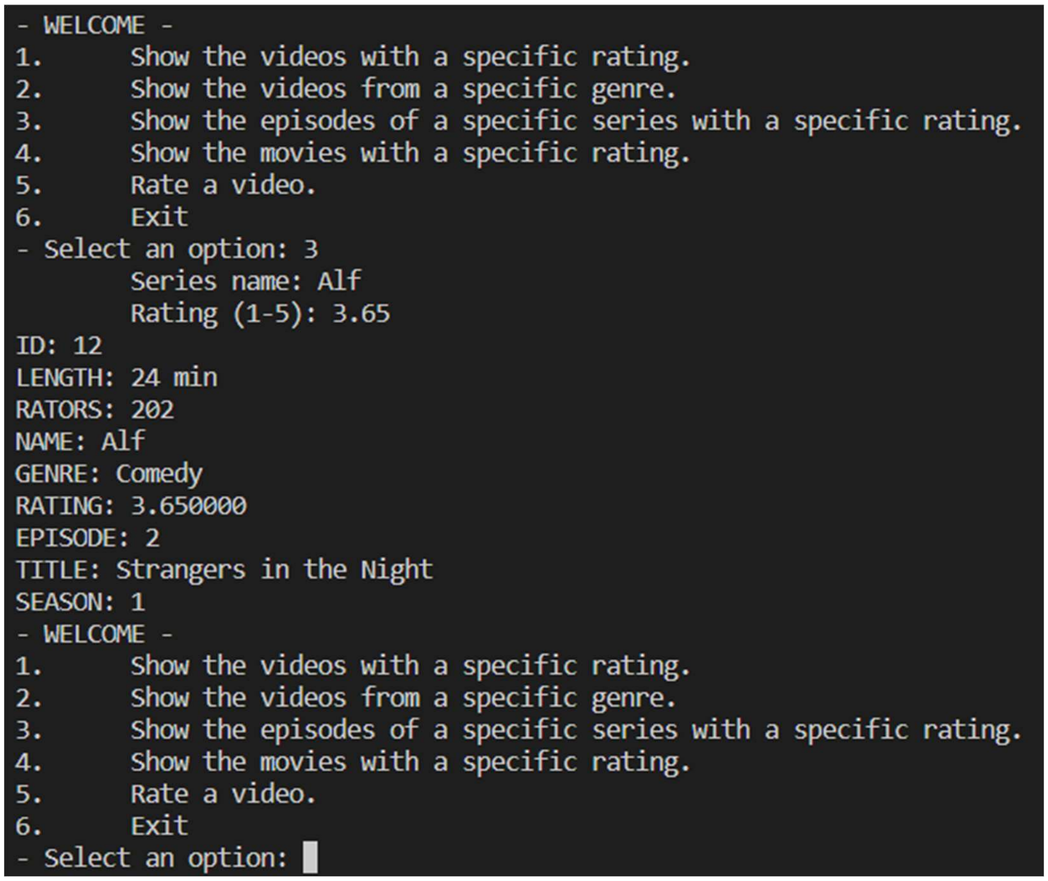
```
- Select an option: 1
        Rating (1-5): 4.25
EPISODES:
ID: 5
LENGTH: 59 min
RATORS: 10579
NAME: The Boys
GENRE: Satire
RATING: 4.250000
EPISODE: 2
TITLE: Cherry
SEASON: 1
ID: 7
LENGTH: 51 min
RATORS: 33000
NAME: Game of Thrones
GENRE: Action
RATING: 4.250000
EPISODE: 7
TITLE: The Broken Man
SEASON: 6
MOVIES:
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 
```



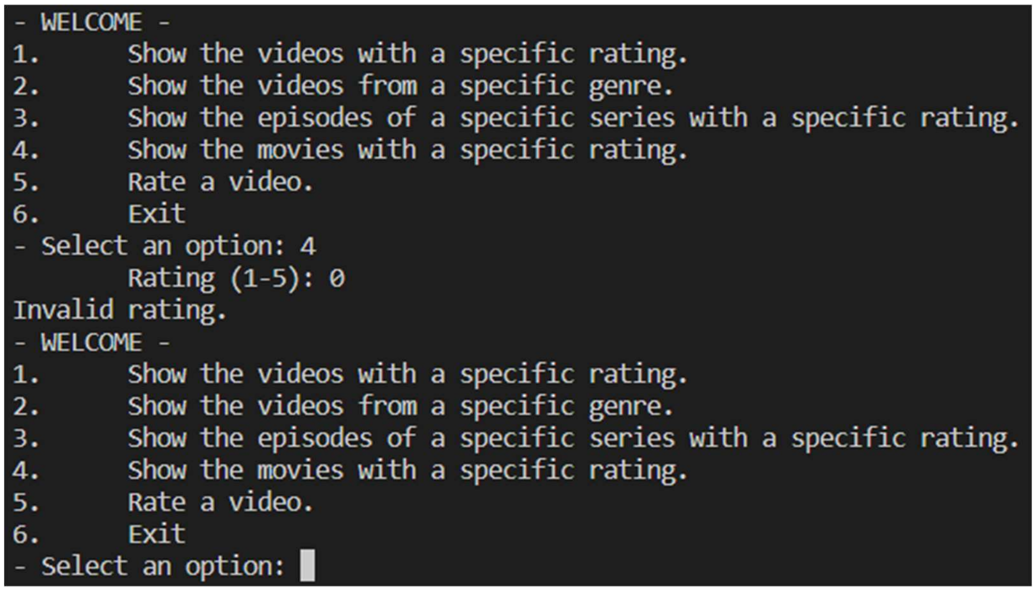
```
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 2
        Genre: Action
EPISODES:
SERIES NAME: Game of Thrones
SERIES GENRE: Action
SERIES LENGTH: 4
SERIES RATING: 4.575000
-- EPISODES --
ID: 7
LENGTH: 51 min
RATORS: 33000
NAME: Game of Thrones
GENRE: Action
RATING: 4.250000
EPISODE: 7
TITLE: The Broken Man
SEASON: 6
ID: 8
LENGTH: 58 min
RATORS: 36805
NAME: Game of Thrones
GENRE: Action
RATING: 4.150000
EPISODE: 8
TITLE: No one
SEASON: 6
ID: 9
LENGTH: 59 min
RATORS: 205061
NAME: Game of Thrones
GENRE: Action
RATING: 4.950000
EPISODE: 9
TITLE: Battle of the Bastards
SEASON: 6
```



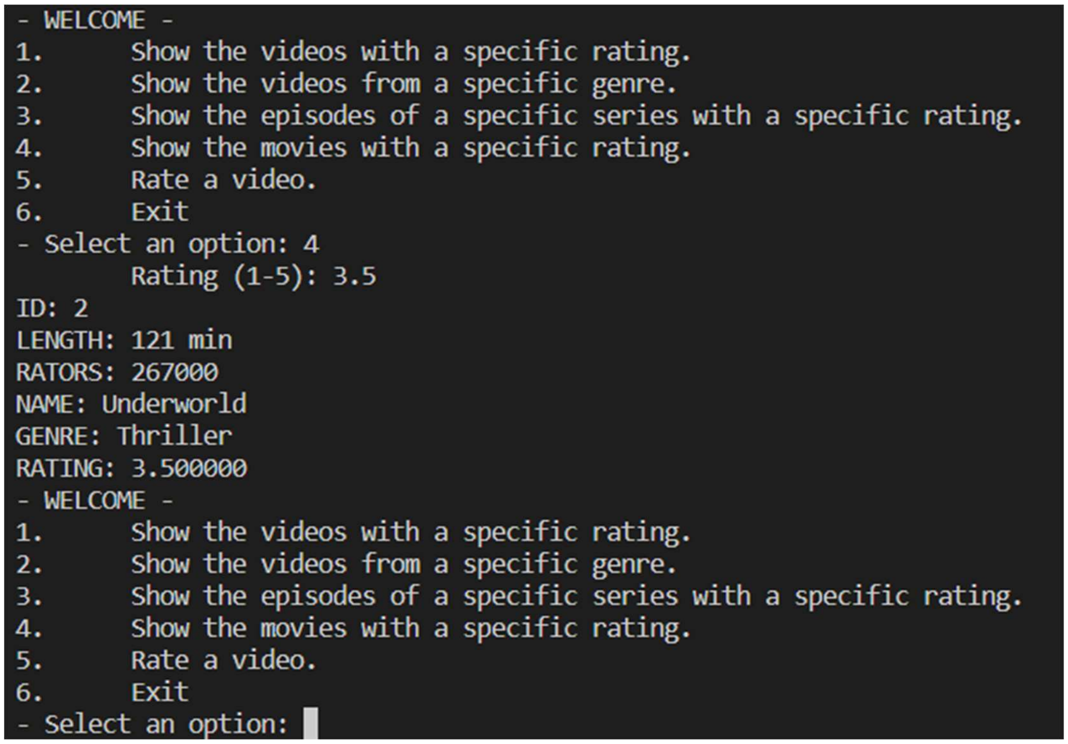
```
ID: 10
LENGTH: 68 min
RATORS: 144705
NAME: Game of Thrones
GENRE: Action
RATING: 4.950000
EPISODE: 10
TITLE: Winds of Winter
SEASON: 6
MOVIES:
ID: 3
LENGTH: 135 min
RATORS: 727000
NAME: Black Panther
GENRE: Action
RATING: 3.650000
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option:
```



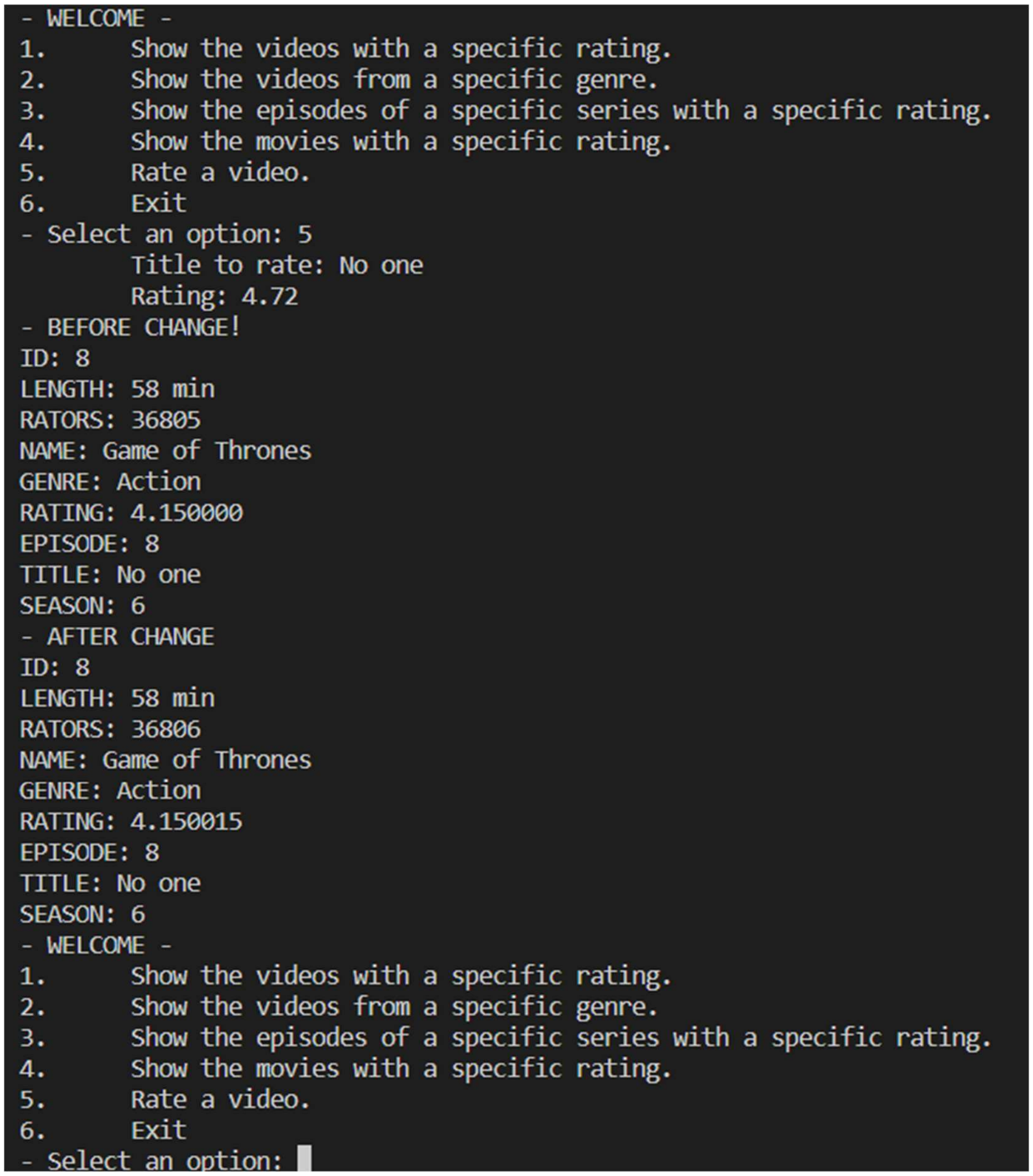
```
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 3
        Series name: Alf
        Rating (1-5): 3.65
ID: 12
LENGTH: 24 min
RATORS: 202
NAME: Alf
GENRE: Comedy
RATING: 3.650000
EPISODE: 2
TITLE: Strangers in the Night
SEASON: 1
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option:
```



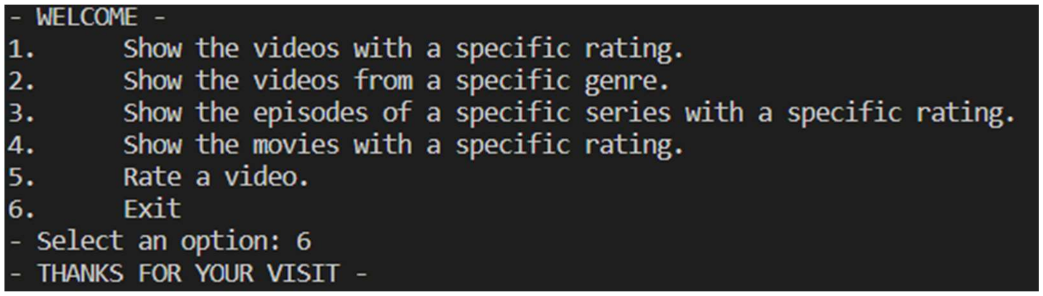
```
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 4
        Rating (1-5): 0
Invalid rating.
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 
```



```
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 4
        Rating (1-5): 3.5
ID: 2
LENGTH: 121 min
RATORS: 267000
NAME: Underworld
GENRE: Thriller
RATING: 3.500000
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 
```



```
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 5
        Title to rate: No one
        Rating: 4.72
- BEFORE CHANGE!
ID: 8
LENGTH: 58 min
RATORS: 36805
NAME: Game of Thrones
GENRE: Action
RATING: 4.150000
EPISODE: 8
TITLE: No one
SEASON: 6
- AFTER CHANGE
ID: 8
LENGTH: 58 min
RATORS: 36806
NAME: Game of Thrones
GENRE: Action
RATING: 4.150015
EPISODE: 8
TITLE: No one
SEASON: 6
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option:
```



```
- WELCOME -
1.      Show the videos with a specific rating.
2.      Show the videos from a specific genre.
3.      Show the episodes of a specific series with a specific rating.
4.      Show the movies with a specific rating.
5.      Rate a video.
6.      Exit
- Select an option: 6
- THANKS FOR YOUR VISIT -
```

Argumentation

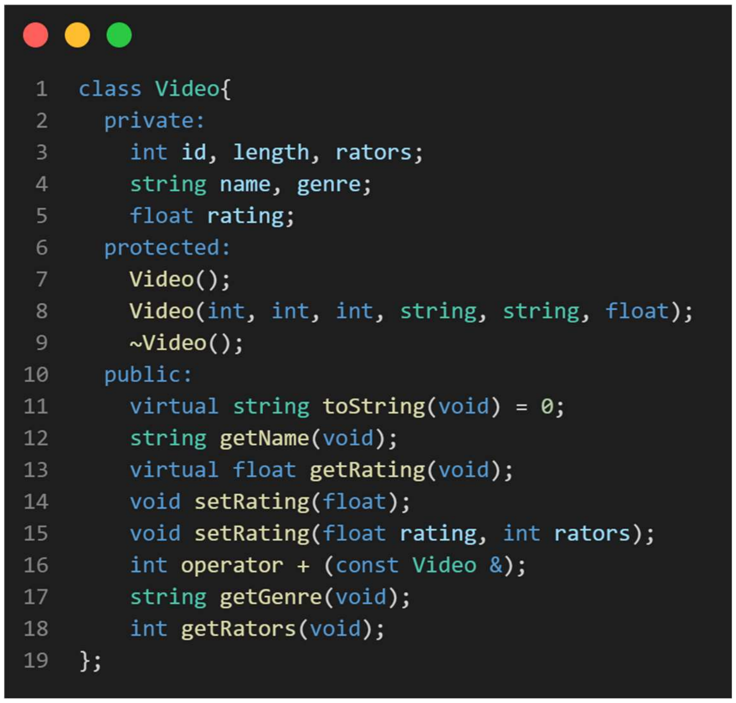
a) The proper class are identified

There was one helping class that was implemented in the code but wasn't expressed in the UML diagram because it can be declared as a struct. This class is the Comparator class, which helps to develop and implement one solution seen in Stack Overflow for finding objects.

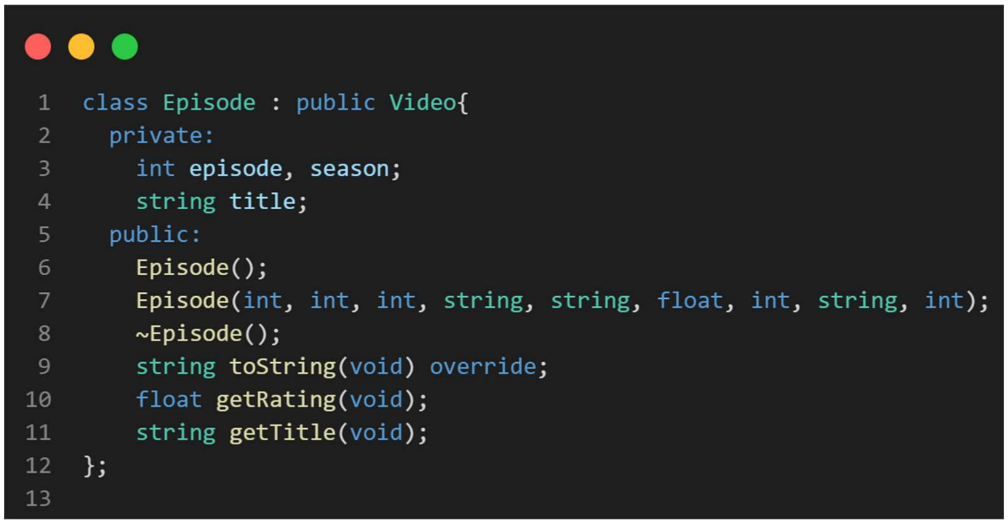
Other than that, the classes identified served not only a functional way, but also a logical implementation of inheritance and polymorphism based on the abstraction of the elements of a streaming catalog.

b) Inheritance is implemented properly

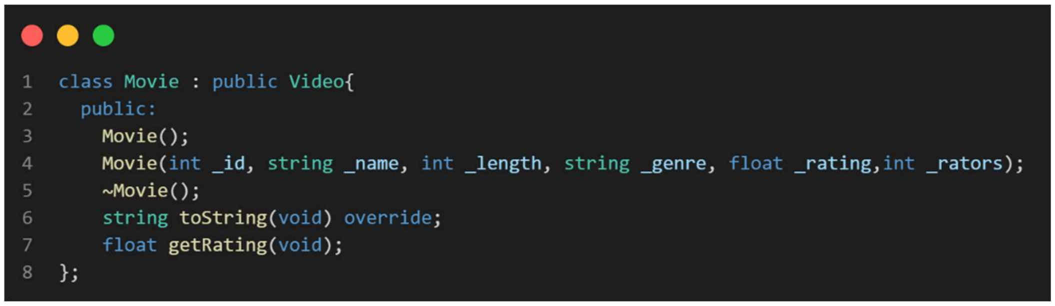
As it was already discussed, the project serves as a logical abstraction. That's why it is better to say that both episode and movie extend video rather than saying that series extend video, because series is not a type of video, rather it has multiple videos, which in this case are episodes. It helped to reuse code that otherwise will create redundancies. Although, not all the attributes from the series class were used in the functionality, they were needed according to the description of the problem.



```cpp
class Video{
  private:
    int id, length, rators;
    string name, genre;
    float rating;
  protected:
    Video();
    Video(int, int, int, string, string, float);
    ~Video();
  public:
    virtual string toString(void) = 0;
    string getName(void);
    virtual float getRating(void);
    void setRating(float);
    void setRating(float rating, int rators);
    int operator + (const Video &);
    string getGenre(void);
    int getRators(void);
};
```



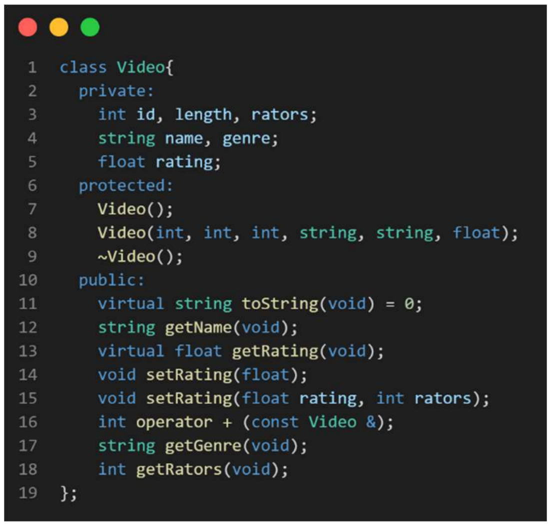
```cpp
class Episode : public Video{
  private:
    int episode, season;
    string title;
  public:
    Episode();
    Episode(int, int, int, string, string, float, int, string, int);
    ~Episode();
    string toString(void) override;
    float getRating(void);
    string getTitle(void);
};

```



```cpp
class Movie : public Video{
  public:
    Movie();
    Movie(int _id, string _name, int _length, string _genre, float _rating,int _rators);
    ~Movie();
    string toString(void) override;
    float getRating(void);
};
```

c) Access modifiers are implemented properly

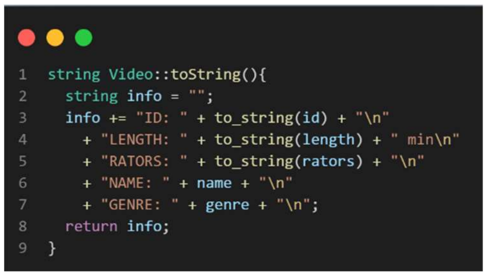
Throughout the code encapsulation played a huge roll in deciding which attributes were going to be public or private. Neither functions nor attributes were protected because they were implemented through inheritance in most cases when needed, and if it wasn't needed, in order to access or modify an attribute, the getters and setters were implemented. The only protected elements from a class were the video constructors because they were only needed for the inherited classes.



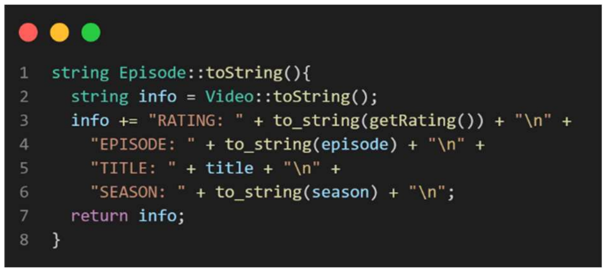
```cpp
class Video{
  private:
    int id, length, rators;
    string name, genre;
    float rating;
  protected:
    Video();
    Video(int, int, int, string, string, float);
    ~Video();
  public:
    virtual string toString(void) = 0;
    string getName(void);
    virtual float getRating(void);
    void setRating(float);
    void setRating(float rating, int rators);
    int operator + (const Video &);
    string getGenre(void);
    int getRators(void);
};
```

d) Method overwriting is implemented properly

Because video was an abstract class, many of the methods were declared as virtual and although they might have already some implemented codes in the source file, most of them must be overridden/overwritten. Particularly the methods to string and get rating must be overwritten in the inherited classes.



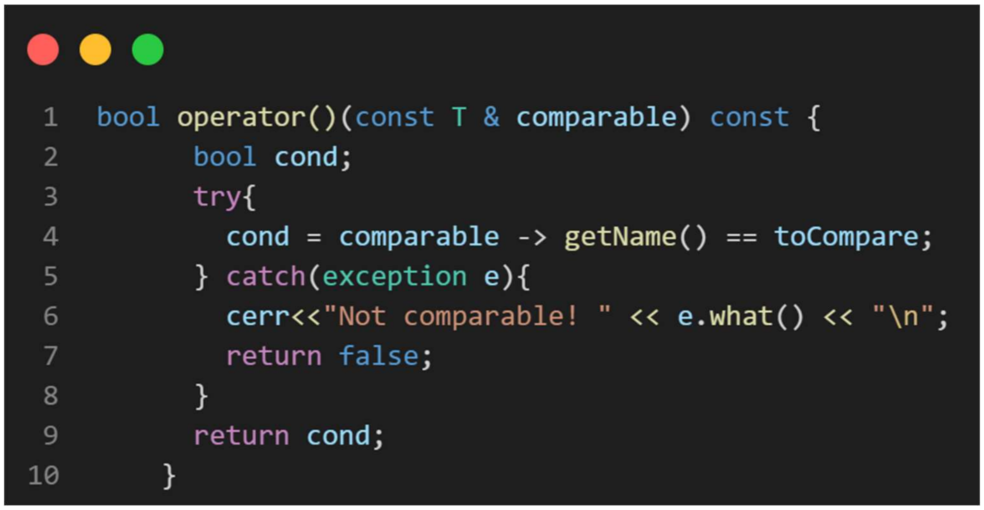
```cpp
string Video::toString(){
  string info = "";
  info += "ID: " + to_string(id) + "\n"
    + "LENGTH: " + to_string(length) + " min\n"
    + "RATORS: " + to_string(rators) + "\n"
    + "NAME: " + name + "\n"
    + "GENRE: " + genre + "\n";
  return info;
}
```



```cpp
string Episode::toString(){
  string info = Video::toString();
  info += "RATING: " + to_string(getRating()) + "\n" +
    "EPISODE: " + to_string(episode) + "\n" +
    "TITLE: " + title + "\n" +
    "SEASON: " + to_string(season) + "\n";
  return info;
}
```

e) Polymorphism is implemented properly

Both movie and episode share the methods from video, which was key using the comparator because it used exclusively the method get name. Additionally, if it was needed, a vector of video pointers can be populated with instances of both movies and episodes.



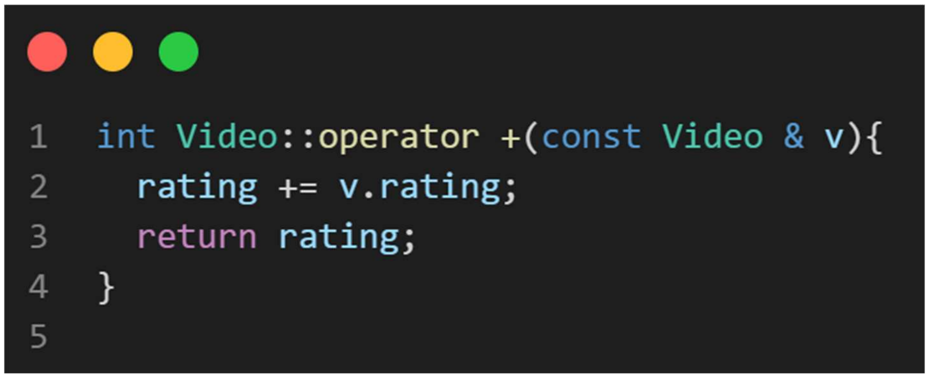
```cpp
bool operator()(const T & comparable) const {
      bool cond;
      try{
        cond = comparable -> getName() == toCompare;
      } catch(exception e){
        cerr<<"Not comparable! " << e.what() << "\n";
        return false;
      }
      return cond;
    }
```

f) Abstract classes are implemented properly

As mentioned previously, video is a pure virtual class. This means that it is a well-defined abstract class implemented in both episode and movie classes.

g) At least one operator is overloaded properly

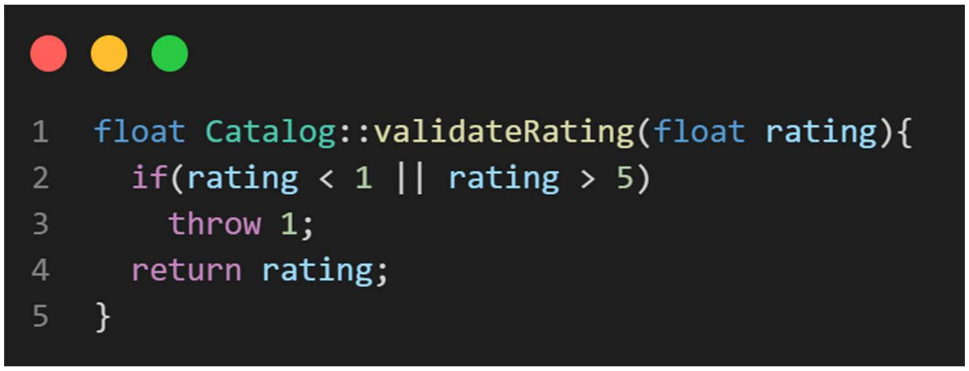
In the development of this project, two operators were overloaded. The first one was in the comparator class, while the operator + was overloaded in the video class which helped calculate the average rating from a series.



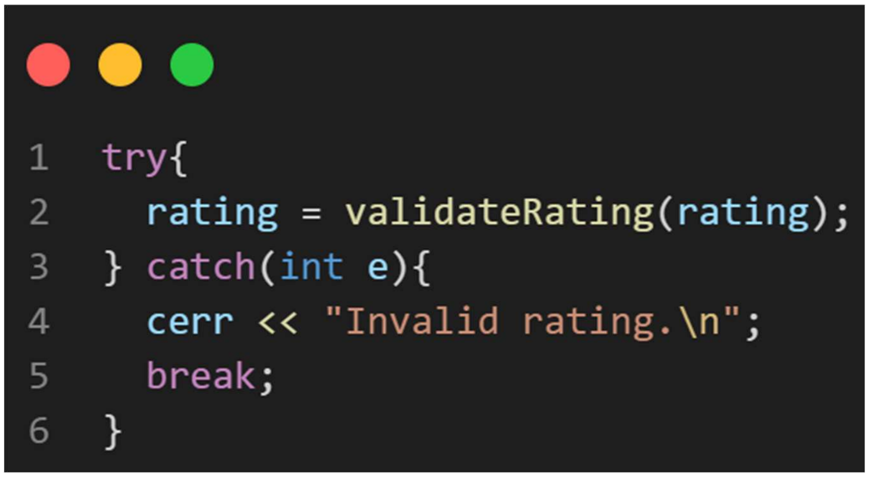
```cpp
int Video::operator +(const Video & v){
  rating += v.rating;
  return rating;
}

```

h) Pre-defined and user-defined exceptions are handled properly

The exceptions handle some user introduced data, like the validation of the rating. However, another exception handling is seen in the comparator class when it checks if the classes are comparable.



```cpp
float Catalog::validateRating(float rating){
  if(rating < 1 || rating > 5)
    throw 1;
  return rating;
}
```



```cpp
try{
  rating = validateRating(rating);
} catch(int e){
  cerr << "Invalid rating.\n";
  break;
}
```

Cases that would prevent the program from working

This program doesn't handle incorrect datatype for input streams. So, if a user introduces a string instead of an integer some errors might be produced. It also depends on the order of the columns on the csv file. That means that it doesn't handle cases where the rows are changed.

Conclusion

Throughout this project's development many object-oriented design competencies were developed. It helped understand a little bit more the structure of different streaming catalogs relying on abstraction and code implementation. For further development it might be interesting to optimize the rating calculation, I'm particularly unsure if the solution implemented was mathematically optimized. Hence, documentation shall be added in the code and not only in an external pdf file, particularly in the source files for further development.

References

IMDb. (n.d.). IMDb: Ratings, Reviews, and Where to Watch the Best Movies & TV Shows. https://www.imdb.com

Sevilla, D. (2011, May 26). Searching for string in vector of pointers. Stack Overflow. https://stackoverflow.com/questions/6145670/searching-for-string-in-vector-of-pointers

cplusplus. (n.d.). Input/output with files - C++ Tutorials. https://m.cplusplus.com/doc/tutorial/files/

cplusplus, (n.d.). Reference. https://cplusplus.com/reference/