**TC1030.328**

Object - Oriented Programming

Integrative Project

| Nombre | Matrícula |
|---|---|
| Karina Ruiz Tron | A01656073 |
| Aylín Millán Cázares | A01655861 |

Fecha de entrega: 15 de junio del 2022

Profesora:

Sergio Ruiz Loza

Tecnológico de Monterrey

Campus de Ciudad de México

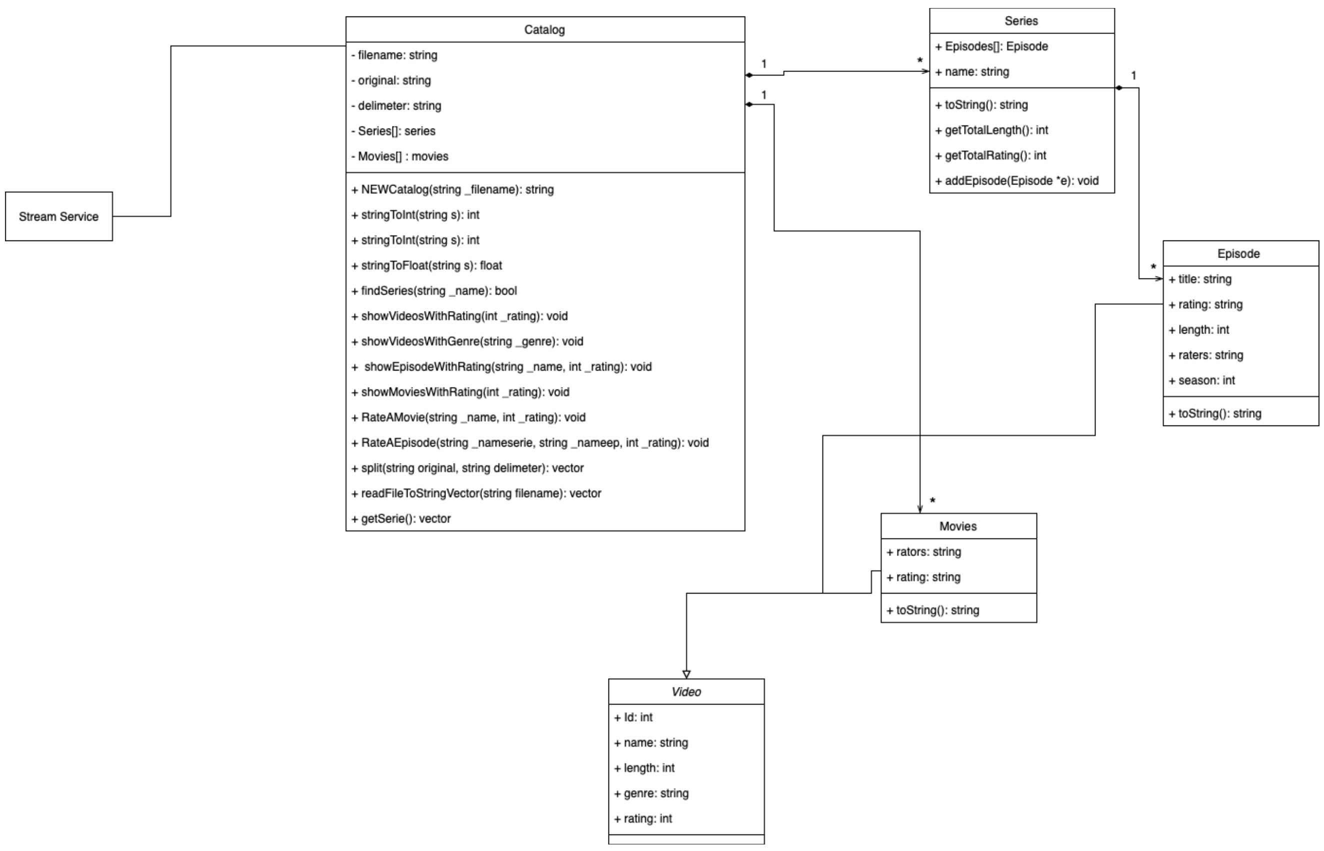
Index

## Introduction

The problem presents us with the goal of making programming similar to streaming services. To make this possible, certain methods and concepts, commonly used in C++, were required including: inheritance (where it's possible to "inherit" attributes from class to class, the class that inherits is called "derived class" and the class that is passing it's attributes is known as "base class"), polymorphism (is related to inheritance, with the difference that here, a class is not only being inherited but also using this methods to do different tasks), another important concept used in the code is abstract classes which are classes created to be used as a base class.

The problem also required the program to have certain thing to demonstrate our knowledge on the matter, such as; access modifiers, something basic in C++ programming because this allows the programmer to access the class members (either public or protected) from one class to another; overwriting, that, just as its name, means to replace a value for a new one; finally, the last thing needed was overloading demonstrated by the usage of, for example, a "+" to join two strings or to sum two integers.

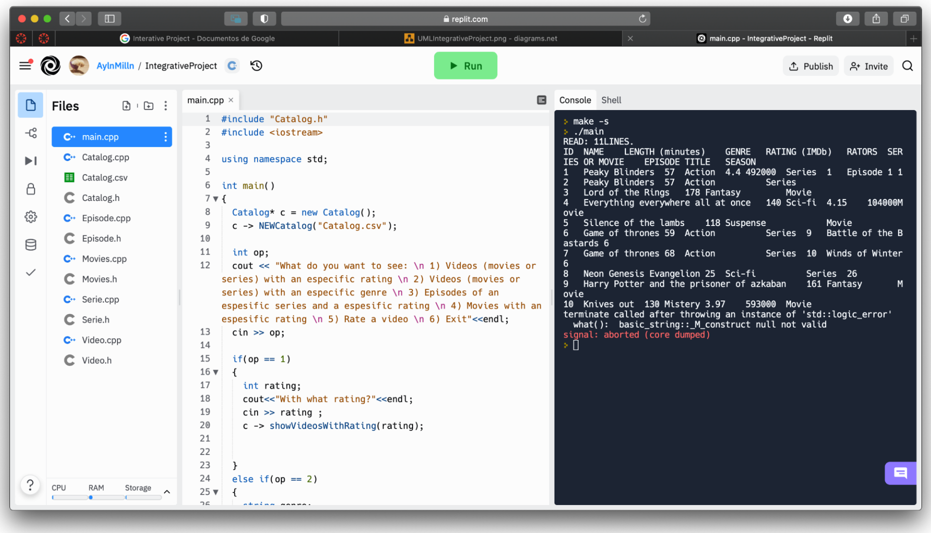
UML Class Diagram


Stream Service
Catalog
- filename: string
- original: string
- delimeter: string
- Series[]: series
- Movies[] : movies
+ NEWCatalog(string _filename): string
+ stringToInt(string s): int
+ stringToInt(string s): int
+ stringToFloat(string s): float
+ findSeries(string _name): bool
+ showVideosWithRating(int _rating): void
+ showVideosWithGenre(string _genre): void
+ showEpisodeWithRating(string _name, int _rating): void
+ showMoviesWithRating(int _rating): void
+ RateAMovie(string _name, int _rating): void
+ RateAEpisode(string _nameserie, string _nameep, int _rating): void
+ split(string original, string delimeter): vector
+ readFileToStringVector(string filename): vector
+ getSerie(): vector
1
1
*
Series
+ Episodes[]: Episode
+ name: string
+ toString(): string
+ getTotalLength(): int
+ getTotalRating(): int
+ addEpisode(Episode *e): void
1
*
Episode
+ title: string
+ rating: string
+ length: int
+ raters: string
+ season: int
+ toString(): string
*
Movies
+ rators: string
+ rating: string
+ toString(): string
Video
+ Id: int
+ name: string
+ length: int
+ genre: string
+ rating: int

Justification:

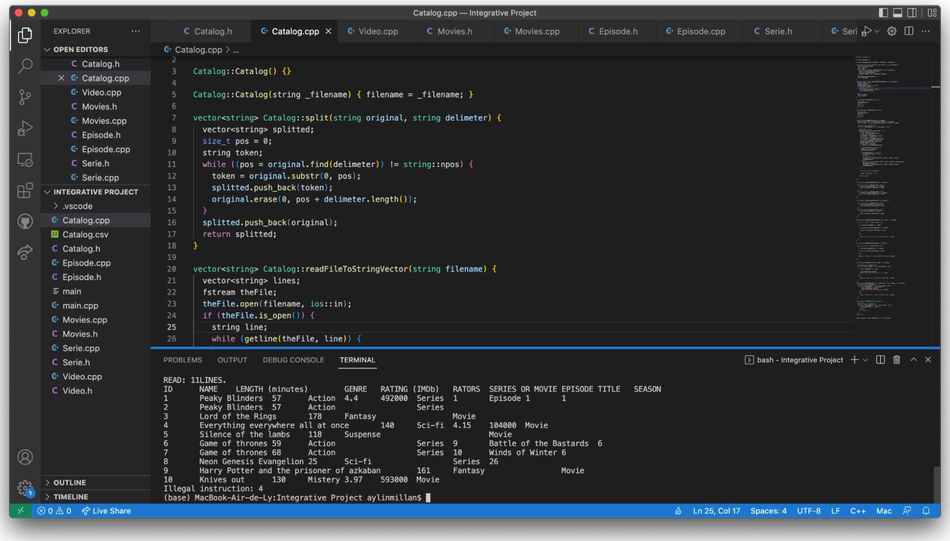
In this diagram we have 5 classes, of which 1 of them is the bigger class, Catalog, where all the methods of printing and searching the vectors are happening. We use this design because we think it is the easiest way of  doing inheritance and polymorphism. The class Video is an abstract class, because we are only initializing all the characteristics that Movies and Episodes have, like they all have a length, an id, a name, etc, so we don’t have to duplicate them. Each of them have more characteristics but they are only used in that class or for that constructor. Also in this design it's easier to understand where the different elements of the rubric are in the part of programming.

Execution example



```cpp
#include "Catalog.h"
#include <iostream>

using namespace std;

int main()
{
  Catalog* c = new Catalog();
  c -> NEWCatalog("Catalog.csv");

  int op;
  cout << "What do you want to see: \n 1) Videos (movies or series) with an especific rating \n 2) Videos (movies or series) with an especific genre \n 3) Episodes of an espesific series and a espesific rating \n 4) Movies with an espesific rating \n 5) Rate a video \n 6) Exit"<<endl;
  cin >> op;

  if(op == 1)
  {
    int rating;
    cout<<"With what rating?"<<endl;
    cin >> rating ;
    c -> showVideosWithRating(rating);


  }
  else if(op == 2)
  {
```

```
> make -s
> ./main
READ: 11LINES.
ID  NAME    LENGTH (minutes)    GENRE   RATING (IMDb)   RATORS  SER
IES OR MOVIE    EPISODE TITLE   SEASON
1   Peaky Blinders  57  Action  4.4 492000  Series  1   Episode 1 1
2   Peaky Blinders  57  Action          Series
3   Lord of the Rings   178 Fantasy         Movie
4   Everything everywhere all at once   140 Sci-fi  4.15    104000M
ovie
5   Silence of the lambs    118 Suspense            Movie
6   Game of thrones 59  Action          Series  9   Battle of the B
astards 6
7   Game of thrones 68  Action          Series  10  Winds of Winter
6
8   Neon Genesis Evangelion 25  Sci-fi          Series  26
9   Harry Potter and the prisoner of azkaban    161 Fantasy        M
ovie
10  Knives out  130 Mistery 3.97    593000  Movie
terminate called after throwing an instance of 'std::logic_error'
  what():  basic_string::_M_construct null not valid
signal: aborted (core dumped)
> 
```



```cpp
Catalog::Catalog() {}

Catalog::Catalog(string _filename) { filename = _filename; }

vector<string> Catalog::split(string original, string delimeter) {
  vector<string> splitted;
  size_t pos = 0;
  string token;
  while ((pos = original.find(delimeter)) != string::npos) {
    token = original.substr(0, pos);
    splitted.push_back(token);
    original.erase(0, pos + delimeter.length());
  }
  splitted.push_back(original);
  return splitted;
}

vector<string> Catalog::readFileToStringVector(string filename) {
  vector<string> lines;
  fstream theFile;
  theFile.open(filename, ios::in);
  if (theFile.is_open()) {
    string line;
    while (getline(theFile, line)) {
```

```
READ: 11LINES.
ID      NAME    LENGTH (minutes)        GENRE   RATING (IMDb)   RATORS  SERIES OR MOVIE EPISODE TITLE   SEASON
1       Peaky Blinders  57      Action  4.4     492000  Series  1       Episode 1       1
2       Peaky Blinders  57      Action                  Series
3       Lord of the Rings       178     Fantasy                 Movie
4       Everything everywhere all at once       140     Sci-fi  4.15    104000  Movie
5       Silence of the lambs    118     Suspense                        Movie
6       Game of thrones 59      Action                  Series  9       Battle of the Bastards  6
7       Game of thrones 68      Action                  Series  10      Winds of Winter 6
8       Neon Genesis Evangelion 25      Sci-fi                  Series  26
9       Harry Potter and the prisoner of azkaban        161     Fantasy                 Movie
10      Knives out      130     Mistery 3.97    593000  Movie
Illegal instruction: 4
(base) MacBook-Air-de-Ly:Integrative Project aylinmillan$ 
```

Argument relating each project part to the points a) through h) stated above. Why is that your best solution?

**a) The proper classes are identified:** After having read the problem situation we managed to understand what was asked, so we decided to do 5 classes. The most important is Catalog, since the whole process of reading the document and making the vectors is carried out there. As the problem mentioned, there is a collection of videos, movies or series, so we created two more classes. Except that the series is a collection of episodes so you must also create an episode class. Video, being the collection of movies and series, only contains the variables needed by the two classes (series and movies).

**b) Inheritance is implemented properly:** Inheritance is seen in the Video, Series and Movies class. We observe which video inherits all its public variables to series and movies. As mentioned above, this class (video) only has a few variables that are repeated in this class. That is to say, it works as a class of characteristics that belong to those two classes (movies and serie).

**c) Access modifiers are implemented properly:** Most of the access modifiers in all classes are public, because these modifiers are used in several classes. As in Video all its modifiers are public because Series and Movies use them to create a new series or movies, or as in Episode the class Would be is a vector of episodes so it needs all the episode attributes to add them to the vector.

**d) Method overwriting is implemented properly:** Method overwriting is implemented in three classes, with public attributes and methods of other classes. For example, for the movie class, since video has all its characteristics, we pass all the public attributes so that it can work correctly. As with series, we pass the attributes of Episode.

**e) Polymorphism is implemented properly and f) Abstract classes are implemented properly:** As we know the abstract classes are obtained by creating a virtual class, but if this virtual class is equal to zero there we have the polymorphism. In this case, we use the video class for these two incises. As mentioned before this class is not necessary to implement it in the main code, it is only necessary that its attributes are inherited to the other classes.

**g) At least one operator is overloaded properly:** In this program there are 2 operators, one for the duration of the whole series and one for the rating of the whole series. This is because we know that the series is a vector of episodes and each episode has a duration and a rating.

Knowing this we can use the + and * to obtain this data. All this is first implemented in video and then put in different gets in serie.

**h) Pre defined and user-defined exceptions are handled properly:** We don’t use any exceptions

Identification of cases that would prevent the project from functioning properly:

Unfortunately, when the program is run it presents an error (visual: Illegal instruction: 4 and replit (signal: aborted (core dumped)). As far as I know I investigated and it mentions that it is something due to memory. We don't exactly know how to solve this problem. However, the code compiles correctly and prints the catalog we have in the .csv document.

Personal Conclusions

**Aylín:**

Personally, I think that our way of solving the problem was the easiest one. Because we were able to make the UML diagram without problems. The only problem was that we had a hard time coding the code because we did not know or did not understand the concepts of some of the necessary topics. In this activity I realized that you need a lot of knowledge about this class to be able to do a project like this. It is a great activity that can help us demonstrate the knowledge we acquired in class.

**Karina:**

I liked this activity because it represented a challenge to implement everything we saw during classes, though I also learned how difficult it can be to program with someone else. I found it especially uncomfortable to adapt to my partner's type of programming, everything was different, from simple things such as the design to the syntax. Overall, this was a really helpful experience to polish certain things in C++ and learn some other new stuff.

## References

https://www.w3schools.com/CPP/cpp_inheritance.asp

https://www.w3schools.com/CPP/cpp_polymorphism.asp

https://www.geeksforgeeks.org/access-modifiers-in-c/