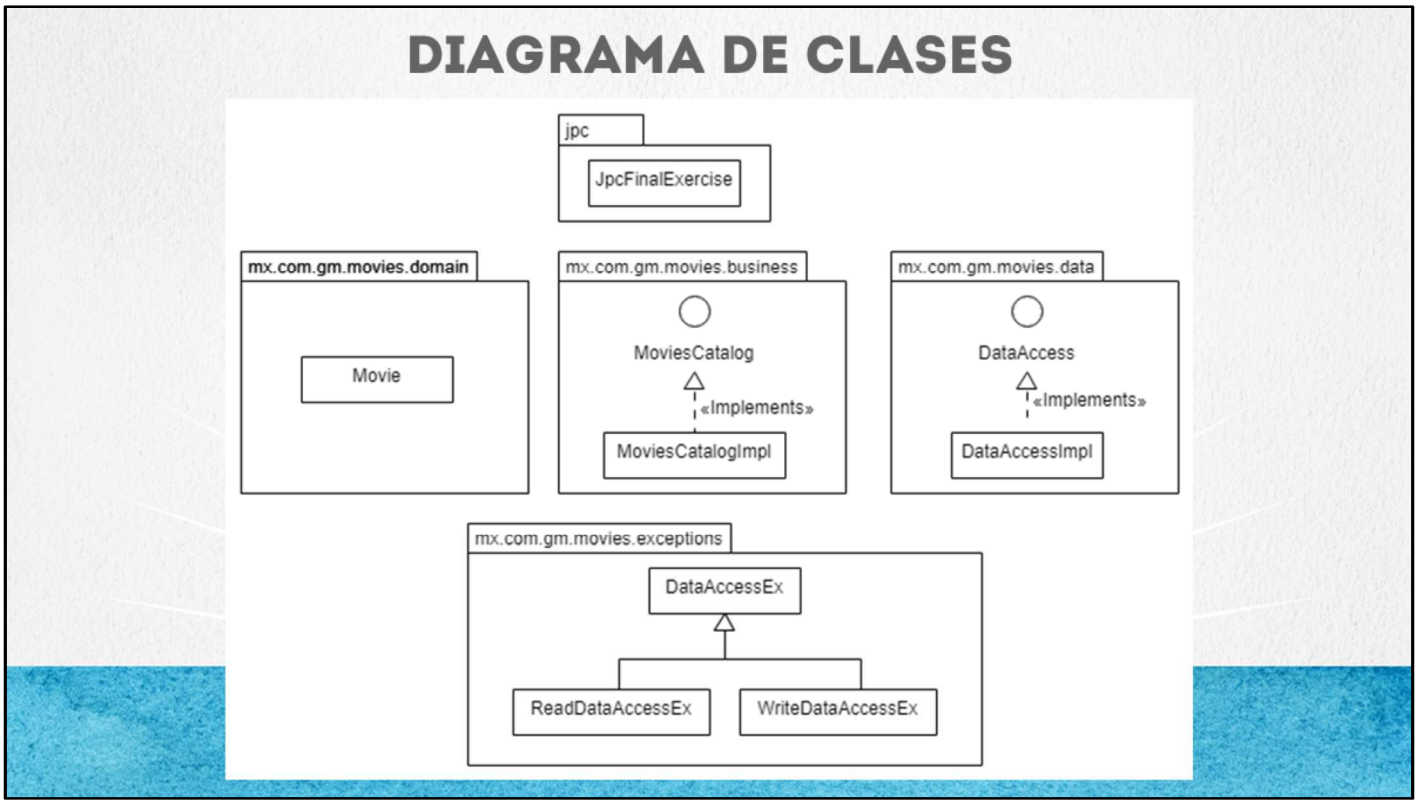
Hello, Ubaldo Acosta greets you. Welcome again to this Java Programming Course.

This is our final lesson, so we are going to do an exercise where we will put into practice what is seen in the Java Programming Course.

Let's start immediately.



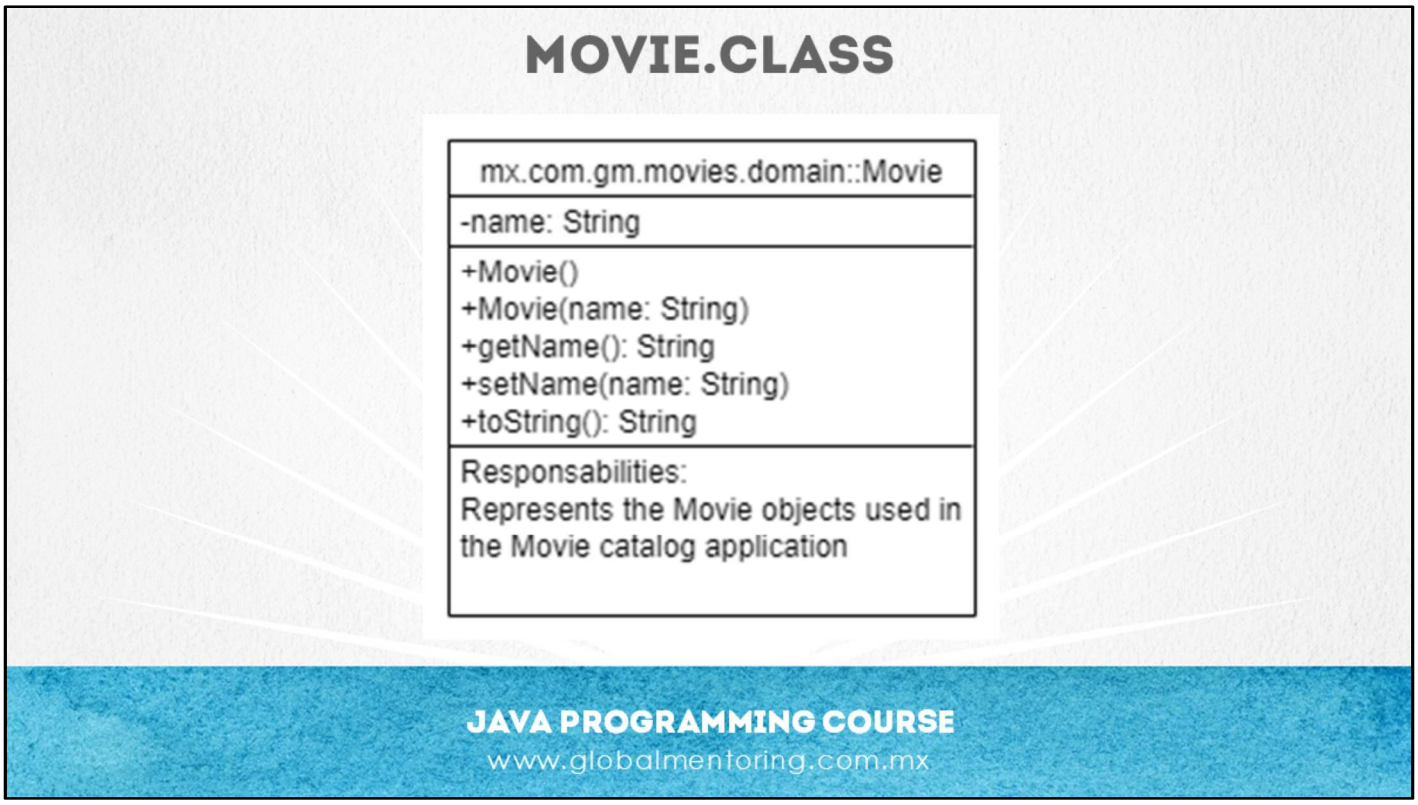
In this final laboratory we will create an exercise that simulates a movie catalog. For this exercise, we will store the information of the movie catalog in a text file, in a folder defined by us, for example in the folder: c:\moviesCatalog\movies.txt

We will create several packages, each with specific tasks to develop in the program.

Packages:

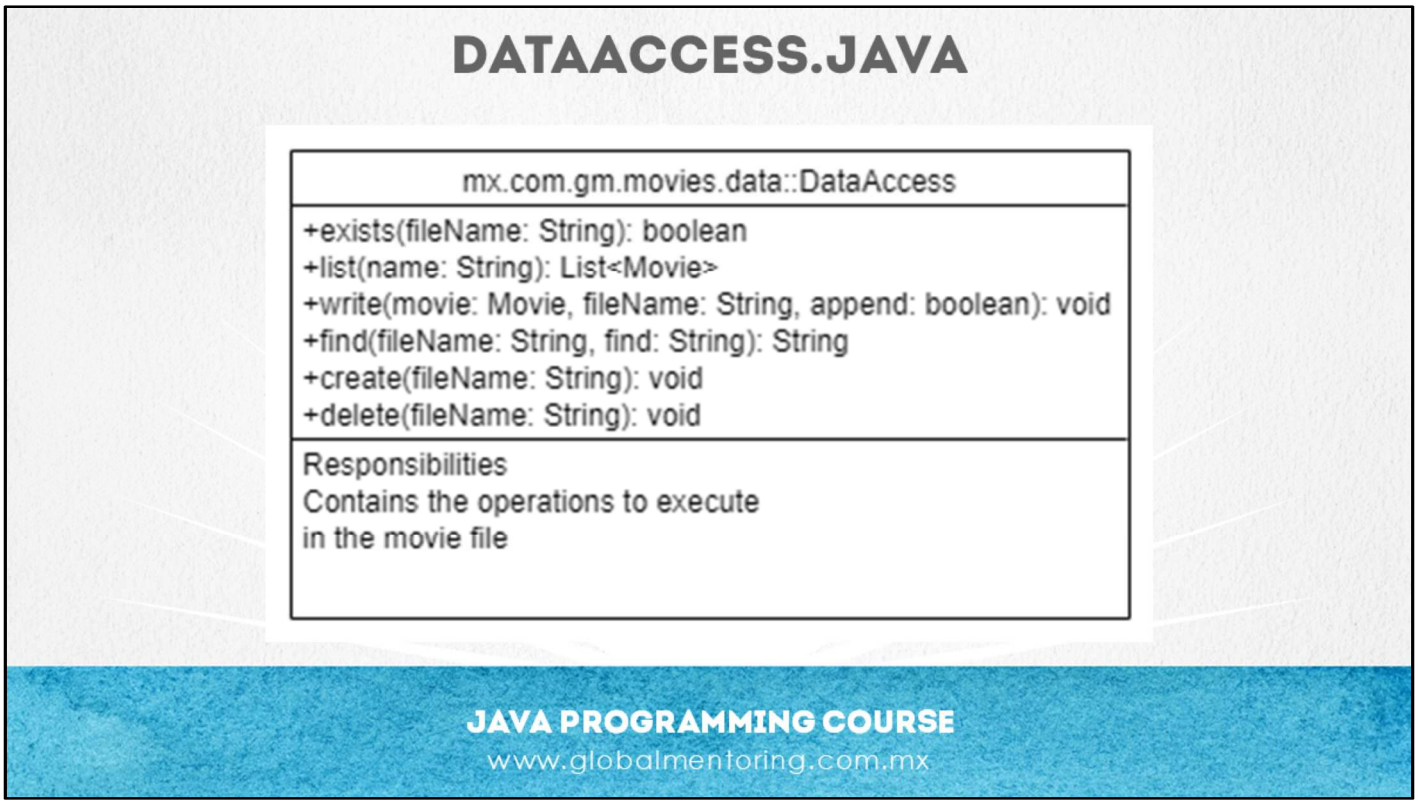
1. Create an exception package, similar to the exception lesson seen in the course, as shown in the class diagram.
2. Create a package mx.com.gm.movies.domain that includes a class called Movie. The classes stored in this package are known as the classes in the problem domain.
3. Create a package called mx.com.gm.movies.data which includes an interface and a class that implements this interface. The objective of these classes is to add the functionality to interact with the file where the movie catalog information will be stored.
4. Create the package called mx.com.gm.movies.business which includes the classes to define the functionality of our application, also known as the business rules of the application.
5. Finally we will create the class JPCFinalExercise, which includes a menu of options to choose the option to process in the program.

Then we will see in more detail each of the packages of the Movies Catalog application to be developed.



This class will help us to represent the Movie object. In this application we leave the definition of this object simple, however it could have more attributes and make this application more complex, but the objective is that we can develop this application with the proposed structure, and not add more complexity than necessary to apply the concepts studied throughout this course.

Therefore, this object must be coded with what is indicated in the class diagram shown.

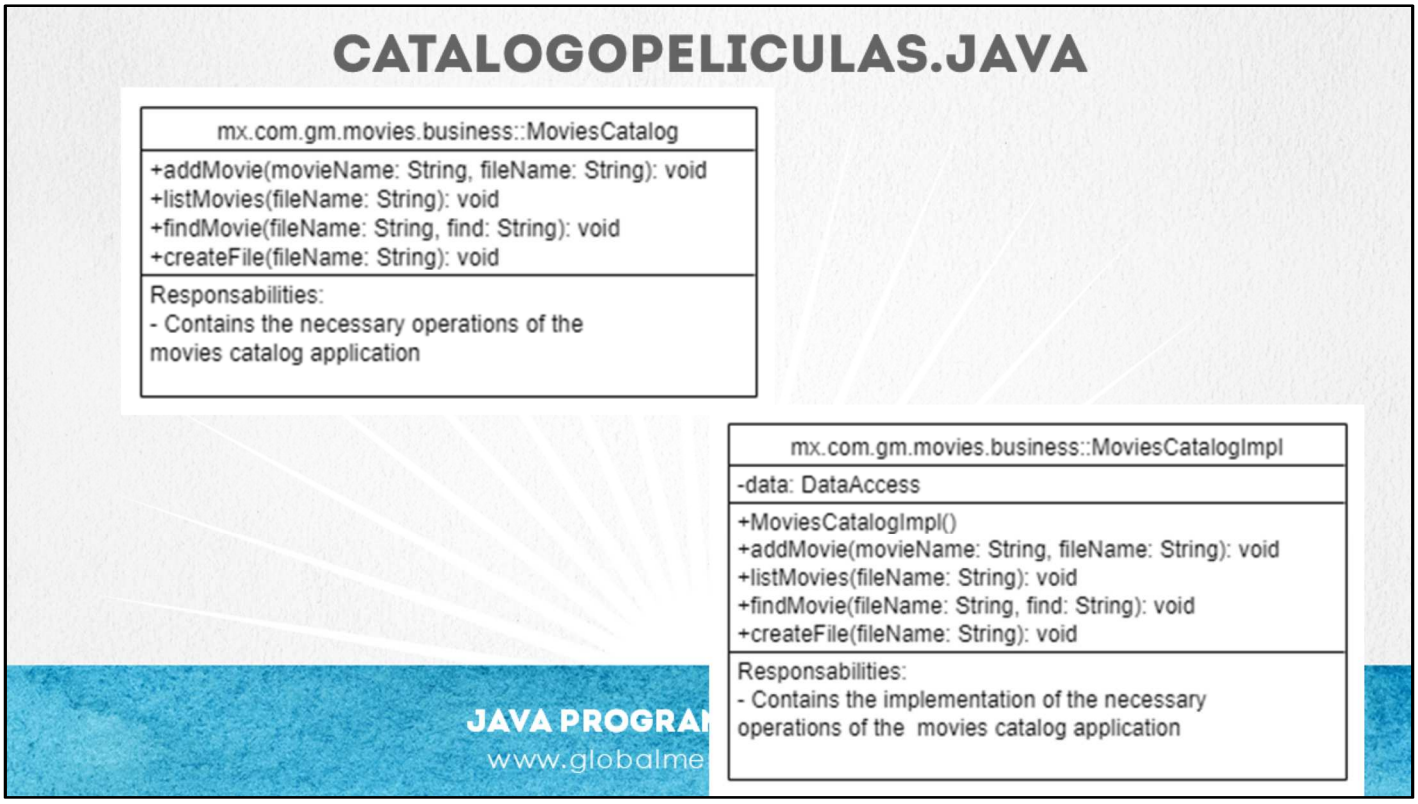


This interface defines the methods to interact with the file that will manage the movie catalog. The class DataAccessImpl is the one that will finally be responsible for adding the behavior of each of the described methods.

From this project we will begin to separate packages and classes according to the functionality of each of them. In the case of the data access package, its function is to define the way in which it interacts with the file that is where the information will be stored.

However, the use of interfaces and their implementation begins to take importance, because in the course of Java with JDBC, instead of working with the implementation for saving the information in a file, we will add the implementation but writing to a base of data. So for this it is important to define the interface that is the one that contains the methods that will be used by other classes in the application, but the implementation we can define which one is going to be used, if for example one that stores the information in a file or some another implementation that stores the information in database.

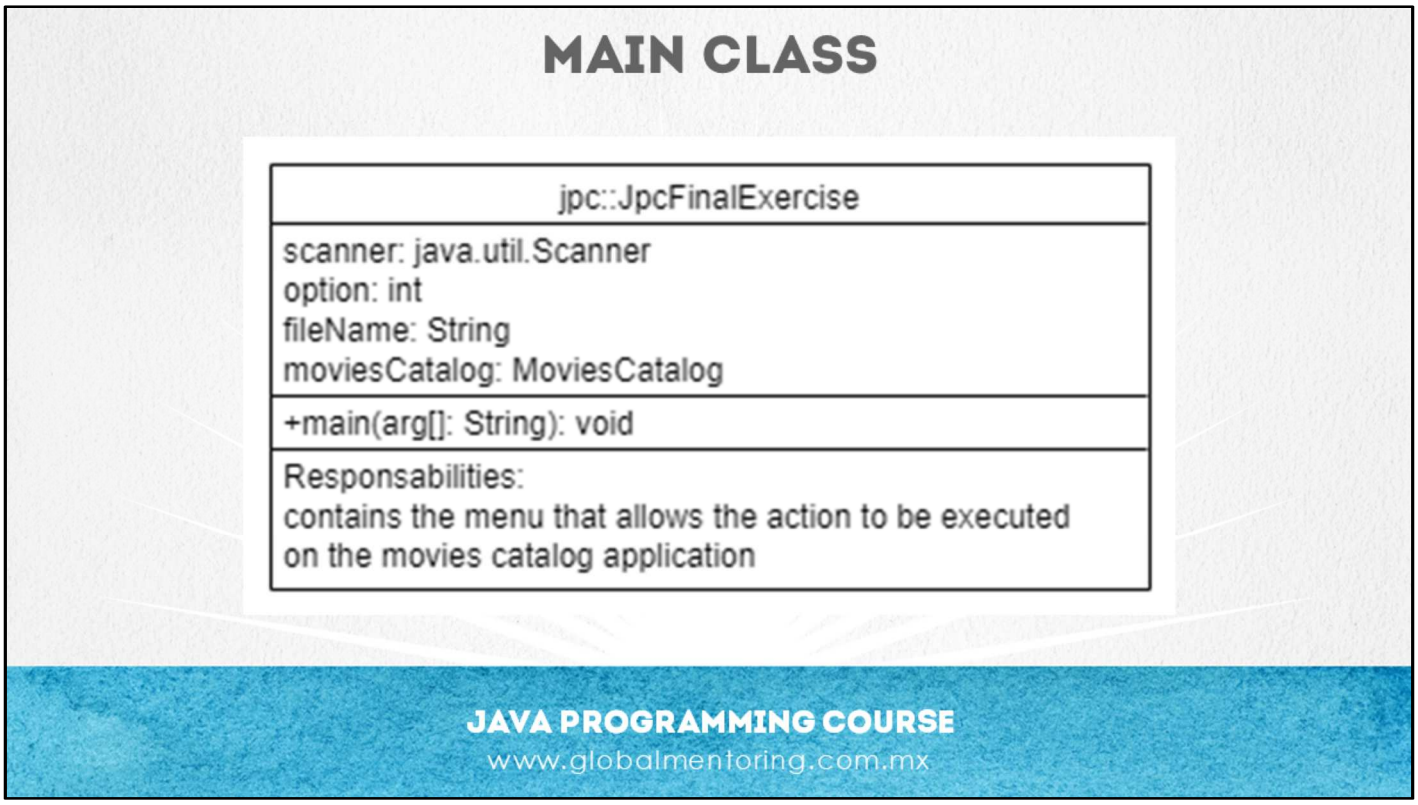
Finally, it should be mentioned that the implementation of this interface is completely technical regarding the use of file management, so unlike the business package that we will see next, the data package, performs the most technical tasks that have to do with the direct manipulation of the file, on the other hand, the business package is more in charge of defining the operations and functionality from the point of view of what the application needs to do, but somehow it is not interested if the information is stored or comes from of a file, since that will be precisely the data package.



The mx.com.gm.movies.business package contains the classes that will implement the business rules of our application. And in turn this is the package that will be the intermediary between the interface with the user (main method) and the storage of the information (data package).

Similar to the data package, this package has also added an interface and an implementation. In this way the user interface (main method) will use the interface, and it will be the implementation of CatalagoPeliculasImpl.java who will finally execute the tasks that are necessary for this application.

At the same time, the implementation of the MoviesCatalog will be the one that uses the DataAccess.java interface, so when using interfaces it is possible to change the implementation at any time and affect as little as possible the code that we have already programmed. This feature is known as low coupling, since a change in one component or class affects as little as possible to another class, so it is only one of the many advantages of programming using interfaces and not directly using the implementation of functionality what do we need.



This class represents the interface with which the user interacts. This class requests via the console the action required by the user, and requests options such as:

1.- Start catalog movies

2.- Add movie

3.- List Movies

4.- Search Movie

0.- Exit

And depending on the option chosen by the user is the action that is executed.

This class is supported by the MoviesCatalog interface to perform the operations that this application needs.

In turn, the option variable is the one that will be responsible for storing the option selected by the user.

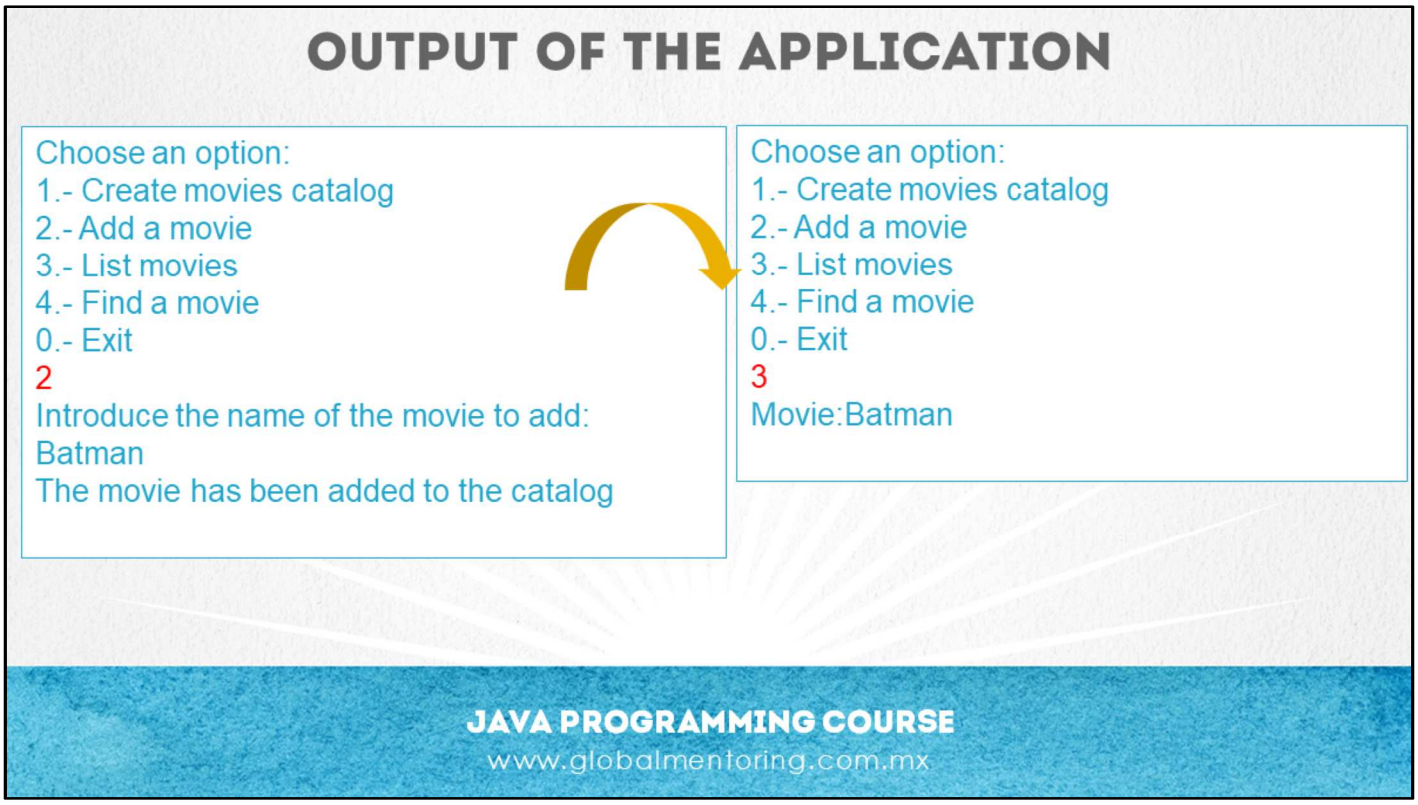
And the name of the file indicates the path where the file will be stored. For example, the Java chain that can be used if we are in Windows is:

"c:\\moviesCatalog\\movies.txt"

However, if it is another operating system, such as MacOS, the route could be:

"/Volumes/gm/moviesCatalog/movies.txt"

We must remember that the folder on which we are going to work must already be created and if necessary must have write permissions, this is particular to the requirements of each operating system.



As we have said, the options shown in the options menu for the user are:

1.- Create catalog movies

2.- Add movie

3.- List Movies

4.- Find a Movie

0.- Exit

And as we see, depending on the option chosen by the user is the action that is executed. So this is the menu that we must program so that the user can interact with the application.

So we have everything ready to start making the Movie Catalog application, hands on and we wish you the best success in the realization of your final practice.