**Instructions**

* You need to solve **one programming problem** in this challenge.
* The duration of this challenge is **120 Minutes**.
* Programming questions have a **Compile and Run** option where you can run your solution against sample test cases before submitting it.
* Click **Evaluate**button only if your code compiles successfully.
* This challenge covers the following topic(s).
  + Entity Framework Fluent API
  + One to Many relationship between entities
  + CRUD Operations

**Scenario:**

WebFlix Company approaches you to create an application where admin of the company can register movies and the genre (like Action/Horror etc) which the movies belong to. Also the admin can view list of movie genres and list of movies. Help the company to create the application.

Using **C# and Entity Framework Fluent API** approach, establish a **one to many relationships** between Movie Genre and Movie Entities.

**Functionalities:**

* Add Movie Genre details to database
* Add Movie details to database with the Movie Category Id from Genre Table as Foreign Key
* Retrieve all Genre details
* Retrieve all Movie details

1. Create a Model class called **MovieGenere** with below public properties:

|  |  |
| --- | --- |
| **Property Name** | **Datatype** |
| MovieGenereId | integer |
| MovieCategory | string |
| MovieCollection | ICollection<MovieInformation> |

2. Create a Model class called **MovieInformation** with below public properties:

|  |  |
| --- | --- |
| **Property Name** | **Datatype** |
| MovieId | integer |
| MovieCategoryId | integer |
| Genere | MovieGenere |
| MovieName | string |
| ReleaseYear | integer |
| AudienceFeedback | string |

**Note:**

Include navigation property at both ends of the above 2 models:

a) MovieGenere entity includes a collection navigation property of type ICollection<MovieInformation>

b) MovieInformation entity includes a reference navigation property of MovieGenere type with the

MovieCategoryId as Foreign key.

***Use*** ***virtual keyword while declaring both the navigation properties.***

3. Create a class **MovieContext** which has the DbContext to establish connection with the SQLSERVER database.

**Additional Implementations in class MovieContext:**

* Implement property for ‘MovieGeneres’ and ‘Movies’ with required 'DbSet' declaration.

**Note:** Use public scope and virtual keyword while implementing DbSet property.

* ***Below should be implemented using Fluent API:***
* MovieGenere Entity should be mapped to tblMovieGenere Table.
* MovieInformation Entity should be mapped to tblMovie Table.
* MovieGenereId should be the Primary Key to tblMovieGenere Table and its values **should not be auto-generated** in database through identity property.
* MovieId should be the Primary Key to tblMovie Table and its values **should not be auto-generated** in database through identity property.
* Create One-to-Many relationship between MovieGenere and MovieInformation. One Movie Genre can have many movies.

4. Create a class called **MovieRepository** with the below methods:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Name** | **Argument** | **Return Type** | **Access Specifier** | **Responsibilities** |
| AddNewGenere | MovieGenere category | MovieGenere | public | This method should insert the given Movie Genre details to the ' tblMovieGenere ' table.  Return the stored MovieGenere object. |
| DisplayMovieGeneres | No Argument | IList<MovieGenere> | public | This method will display all the Movie Genre information present in the ' tblMovieGenere ' table. |
| AddNewMovie | MovieInformation movie | MovieInformation | public | This method should insert the given Movie details to the ' tblMovie' table.  Return the stored Movie object. |
| DisplayMovieLibrary | No Argument | IList<MovieInformation> | public | This method will display all the Movie information present in the ' tblMovie' table. |

5. Create a class called **RatingFeedback** which contains the below method:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Name** | **Argument** | **Return Type** | **Access Specifier** | **Responsibilities** |
| FeedbackInfo | int rating | string | public | This method accepts an integer rating argument and returns a string based on below logic:  (i) If rating<2 return “Bad”  (ii) If rating >=2 and rating< 4 then return “Average”  (iii) If rating>=4 then return “Good”. |

5. Class **Program**

Implement the Main method based on the Sample Input / Output given below:







