Catálogo de películas

> *Prueba Intra-semestral de Programación - Curso 2022*

NOTA: *Antes de comenzar asegúrese de descompactar el archivo `movie-catalog.zip` y abrir la solución `movie-catalog.sln` en su editor. Asegúrese también de que su código compila, y la aplicación de consola ejecuta (debe lanzar una excepción). Recuerde que todo el código a evaluar debe ir en el archivo `Exam.cs` de la aplicación de consola `exam`.*

En este ejercicio vamos a implementar un sistema sencillo de un catálogo de películas de una plataforma de Streaming. De cada película tendremos su nombre y su calificación según la crítica que podrá ser aumentada o disminuida en el rango de `0` a `100`.

Además, cada película pertenece a un género, que a su vez puede pertenecer a otro género más general, creándose así un árbol de géneros, donde en cualquier nivel puede haber películas concretas y/o otros géneros.

Por ejemplo:

```
- Comedia
    - Romántica
        - Todos los días de mi vida (85)
        - La propuesta (95)
        - Crazy, Stupid, Love (80)
    - Negra
        - El gran Lebowski (93)
        - Malditos vecinos (71)
        - Pineapple Express (75)
- Terror
    - Psicológico
        - El silencio de los corderos (98)
        - Memento (95)
        - El club de la pelea (92)
    - Sobrenatural
        - El conjuro (84)
        - Insidious (77)
        - Poltergeist (72)
```

La funcionalidad principal del sistema de catálogo de películas se encuentra en la interfaz `IMovieCatalog`.

```
interface IMovieCatalog
{
    // Género raíz
    IGenre Root { get; }

    // Navegar por los géneros y películas
    IGenre GetGenre(params string[] genres);
    IMovie GetMovie(string movie, params string[] genres);

    // Buscar las películas que cumplen con una condición
    IEnumerable<IMovie> FindAll(Filter<IMovie> filter);
}
```

Como es usual, usted devolverá una instancia de su implementación de esta interfaz en el método estático `Exam.GetMovieCatalog` de la clase `Exam` en el archivo `Exam.cs` de la aplicación de consola.

Veremos a continuación cada uno de los métodos que usted debe implementar.

Géneros

Un género se define mediante la interfaz `IGenre` . La interfaz `IGenre` se define así (veremos los métodos uno a uno).

```
interface IGenre
{
    string Name { get; }

    // Crear subgéneros
    IGenre CreateSubgenre(string name);

    // Crear o actualizar la calificación de una película
    void UpdateRating(string name, int change);

    // Enumerar todos los subgéneros (en este nivel)
    IEnumerable<IGenre> Subgenres { get; }

    // Enumerar todas los películas (en este nivel)
    IEnumerable<IMovie> Movies { get; }

    // Género padre
    IGenre Parent { get; }
}
```

Todo catálogo de películas se crea con un género raíz, cuyo nombre es el `string` vacío. Este es el género que se devuelve en la propiedad `Root` de la interfaz `IMovieCatalog`.

Para obtener un género arbitrario, se puede usar el método `GetGenre` de la interfaz `IMovieCatalog` que recibe un array de `string` con los nombres de los géneros intermedios.

Por ejemplo, para obtener el género `"Comedia"` se invocaría a este método de la siguiente forma:

```
IMovieCatalog catalog = Exam.GetMovieCatalog();
IGenre comedia = catalog.GetGenre("Comedia");
```

En caso de no existir el género pedido usted debe lanzar una excepción de tipo `ArgumentException`.

Una vez que tenemos una referencia a un grupo táxonimco, es posible utilizarlo para crear nuevos subgéneros. Por ejemplo para crear un subgrupo taxonómico `Absurda` dentro del género `Comedia`:

```
IGenre absurda = comedia.CreateSubgenre("Absurda");
```

Por supuesto, una vez que este género ha sido creado, desde el catálogo de películas original es posible obtener exactamente la misma referencia:

```
Debug.Assert(absurda == catalog.GetGenre("Comedia", "Absurda"));
```

La propiedad `Parent` en `IGenre` devuelve una referencia al género padre. En el caso de ser el género raíz, devuelve una referencia a sí mismo (nunca `null`).

Películas

En cualquier género, el método `UpdateRating` aumenta (o disminuye) la calificación de cualquier película.

Si una película existe, su calificación se modifica en el valor `change` que puede ser positivo o negativo. Si una película no existe en ese género, se crea automáticamente cuando se invoque este método con la calificación pasada. Si se tiene una referencia a una película, se debe mantener la misma referencia luego de cambiar su calificación.

```
IGenre negra = catalog.GetGenre("Comedia", "Negra")

IMovie malditosVecinos = catalog.GetMovie("Malditos vecinos", "Comedia", "Negra");

// Disminuye en 10 la calificación de Malditos vecinos
negra.UpdateRating("Malditos vecinos", -10);

// Crea una nueva película
negra.UpdateRating("El gran golpe", 80);

Debug.Assert(malditosVecinos == catalog.GetMovie("Malditos vecinos", "Comedia", "Negra"
```

Por supuesto ninguna película puede bajar su calificación de `0`, ni sobrepasar `100`, ni crearse con una calificación negativa o mayor que `100`. En cualquiera de estos casos usted debe lanzar una excepción de tipo `ArgumentException`.

La propiedad `Subgenres` enumera todos los subgéneros que son hijos immediatos de este género.

La propiedad `Movie` enumera todas las películas immediatamente en este género. Esta propiedad devuelve instancias de la interfaz `IMovie` que simplemente almacena el nombre y calificación, así como una referencia al género al que pertenece:

```
interface IMovie
{
    string Name { get; }
    int Rating { get; set; }
    IGenre Parent { get; }
}
```

En la interfaz `IMovieCatalog` el método `GetMovie`, muy similar a `GetGenre`, devuelve directamente la película correspondiente, dado su nombre y los géneros a los que pertenece. Por ejemplo:

```
IMovie poltergeist = catalog.GetMovie("Poltergeist", "Terror", "Sobrenatural");
Debug.Assert(poltergeist.Rating == 72);
```

Filtrado de películas

El método `FindAll` de la interfaz `IMovieCatalog` enumera todas las películas que cumplen con cierta condición, definida por el delegado `Filter`:

```
delegate bool Filter<T>(T item);
```

Por ejemplo, para encontrar todas las películas que tienen menos de 85 de calificación:

```
foreach(var movie in catalog.FindAll(s => s.Rating < 85))
{
    // Verificando que efectivamente tiene menos de 85
    Debug.Assert(movie.Rating < 85);
}
```

Ejemplos de prueba

En la aplicación de consola encontrará un ejemplo de prueba muy similar a lo que hemos visto hasta ahora, que le permitirá verificar que los métodos básicos funcionan.

NOTA: El ejemplo de prueba es insuficiente para garantizar que su código está 100% correcto. En particular, los métodos de iteradores no se verifican. Es su responsabilidad adicionar tantos casos de prueba como considere necesario para garantizar la correctitud de su solución.

¡Éxitos a todos!