Práctico II
23/08/2021
Programación II

Biblioteca de música

Para trabajar un poco más con los conceptos de la Programación Orientada a Objetos, crearemos una biblioteca de música. Crearemos bandas y artistas, agregaremos sus canciones y tendremos la oportunidad de listar sus canciones, integrantes, y más información sobre éstos.

- Crear nuestro proyecto

  Dentro nuestra carpeta de proyectos (ej.: C:/progprojects), abriremos la consola y seguiremos uno de los siguientes caminos:

  Camino 1:

  mkdir biblioteca-musica → con esto creamos el directorio donde estará contenido el proyecto. El comando **mkdir** nos permite crear directorios; "biblioteca-musica" será el nombre de este nuevo directorio.

  dotnet new console → "**dotnet**" es el comando que nos permite utilizar el SDK de .NET. El argumento "**new**" indica que queremos crear un nuevo proyecto, y "**console**" nuestro tipo de aplicación (consola).

  Camino 2:

  dotnet new console --name "biblioteca-musica" → el directorio "biblioteca-musica" se crea automáticamente, y se generan todos los archivos del proyecto. "**--name**" es quien asigna el nombre al nuevo proyecto.

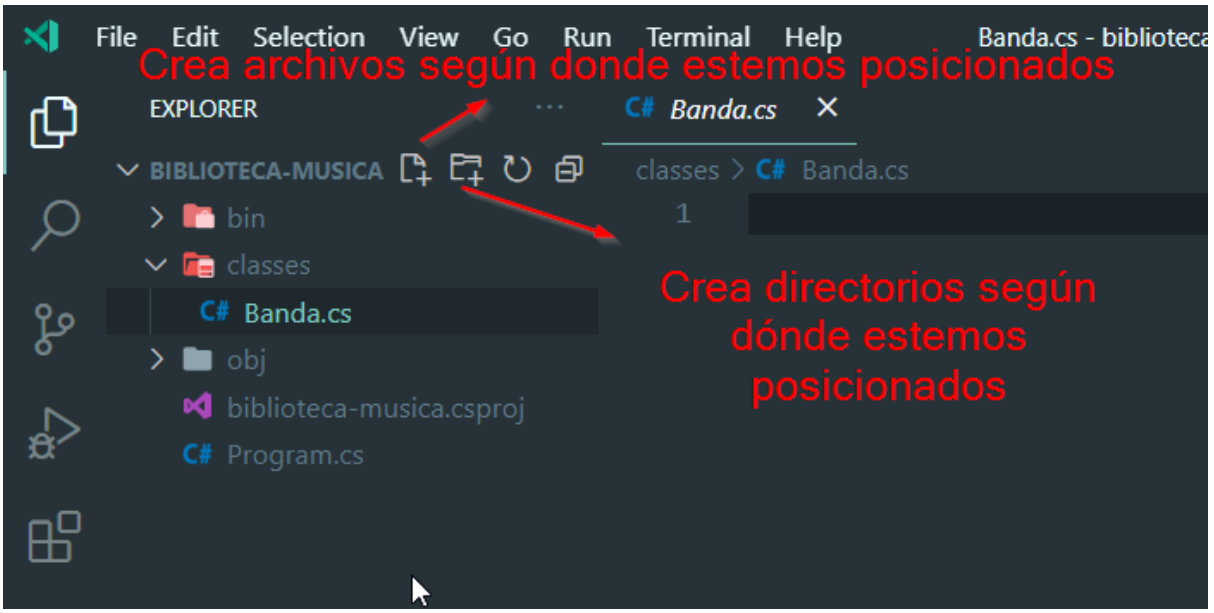
- Ingresamos al nuevo directorio (escribimos "**cd** biblioteca-musica" en la línea de comandos), y abrimos el proyecto en nuestro editor "**code .**".

  Recordemos que el comando **cd** (**C**hange **D**irectory) nos permite movernos entre directorios.

  Cuando queremos ingresar a un subdirectorio de nuestro cwd (Current Working Directory, representado por el punto "**.**"), debemos escribir el nombre del mismo tras el comando, y para ir un directorio atrás, utilizamos los dos puntos tras el comando "**..**".

- Crear una clase llamada **Banda**.

  Para esto, debemos crear un directorio en la raíz de nuestro proyecto con el nombre "**classes**". Dentro de ésta, un archivo llamado "**Banda.cs**"



Dentro de Banda.cs, tendremos los siguientes atributos:

- Nombre
- Género
- Integrantes
- Álbumes
- Canciones

No olvidemos crear el ***constructor*** para esta clase.

Por ***encapsulamiento***, todos estos atributos deben ser ***privados*** con sus correspondientes ***getters*** y ***setters***, así que debemos crear las funciones correspondientes.

Un esbozo posible es el siguiente:

```
0 references
public class Banda {
    3 references
    private string nombre;
    0 references
    public Banda (string xNombre) {
        this.nombre = xNombre;
    }
    0 references
    public string getNombre() {
        return this.nombre;
    }
    0 references
    public void setNombre(string xNombre) {
        this.nombre = xNombre;
    }
}
```

- Inicialmente, podríamos pensar que ***integrantes***, ***álbumes*** y ***canciones*** podrían ser una lista de strings, pero, idealmente, deberíamos crear más clases que representen dichos atributos.

  Por ejemplo, para **Integrantes**, podríamos crear la clase **Persona** con ***nombre*** y ***apellido***. Entonces, el atributo **Integrantes** pasará a ser una lista de personas.

```
public class Banda
{
    3 references
    private string nombre;
    3 references
    private string genero;
    1 reference
    private List<Persona> integrantes;
    0 references
    public Banda(
        string xNombre,
        string xGenero,
        List<Persona> xIntegrantes
    )
    {
        this.nombre = xNombre;
        this.genero = xGenero;
        this.integrantes = xIntegrantes;
    }

```

A continuación, un ejemplo de cómo podemos instanciar personas, añadirlas a una lista y más:

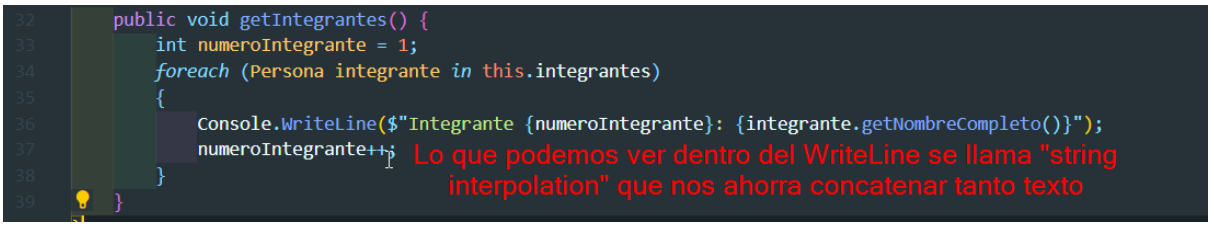
```
C# Banda.cs    C# Program.cs    C# Persona.cs
C# Program.cs > {} biblioteca_musica > biblioteca_musica.Program > Main(string[] args)
        static void Main(string[] args)
        {
            string[] listaPersonas = {
                "Myles Kennedy",
                "Mark Tremonti",
                "Brian Marshall",
                "Scott Phillips"
            };
            // Instancio una nueva lista de miembros
            List<Persona> integrantes = new List<Persona>();
            // Recorro listaPersonas
            foreach (var persona in listaPersonas)
            {
                // Separo nombre y apellido
                string[] datosPersona = persona.Split(" ");
                // Instancio una nueva persona
                Persona instanciaPersona = new Persona(datosPersona[0], datosPersona[1]);
                // Agrego persona a la lista
                integrantes.Add(instanciaPersona);
            }
            // Instancio una nueva banda e imprimo integrantes
            Banda instanciaBanda = new Banda("Alter Bridge", "Post-Grunge", integrantes);
            instanciaBanda.getIntegrantes();
        }
    }

PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

PS C:\progprojects\biblioteca-musica> dotnet run
Integrante 1: Myles Kennedy
Integrante 2: Mark Tremonti
Integrante 3: Brian Marshall
Integrante 4: Scott Phillips
PS C:\progprojects\biblioteca-musica>
```

Nota: idealmente, estos datos deben ingresarse de forma dinámica con Console.ReadLine().

Como se puede observar, inclusive tenemos una función propia de la instancia que devuelve a los integrantes llamada "**getIntegrantes**" que no devuelve directamente la lista, sino que la recorre e imprime directamente en pantalla.



Así mismo, la clase **Persona** también tiene un método que devuelve el nombre completo de la persona, que nos podría ser útil.

- Deberíamos repetir un proceso similar para **Álbumes** y **Canciones**.

  Podríamos crear una clase **Álbum**, y así nuestro atributo **Álbum** en **Banda** sería una lista de **Álbumes**.

  Cada Álbum podría tener una lista de **Canciones**, y así nos ahorraríamos un atributo de la clase **Banda**.

  Podríamos así tener una función en **Banda** que devuelva todas las **Canciones**: que simplemente recorra todas las **Canciones** para todos los **Álbumes** (doble foreach).

  Al hacer todo esto, trabajamos con el concepto de **Composición de Clases** y **Encapsulamiento**.

Una vez trabajadas las clases, dentro del Main del archivo Program.cs, deberíamos tener un programa capaz de:

- Agregar bandas.
- Listar bandas.
- Listar canciones de una banda en específico.
- Eliminar una banda.
- Eliminar un álbum o canción de una banda.
- Colocar un integrante como activo/inactivo.

OPCIONAL:

Los datos deberían ingresarse dinámicamente (Console.ReadLine()).

PREGUNTAS:

- ¿Qué sucede cuando a un método le colocamos "void"?
- ¿Qué modificadores de acceso existen en C# y cuáles son sus diferencias?
- ¿Qué es un constructor? ¿Por qué puedo tener varios?

- ¿Qué son los getters y los setters? ¿A qué principio de la programación orientada a objetos está asociada?¿Hay otras formas de definirlos en C#?