# TECNOLÓGICO NACIONAL DE MÉXICO
## INSTITUTO TECNOLÓGICO DE TIJUANA

### SUBDIRECCIÓN ACADÉMICA
### DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

**SEMESTRE:**
Agosto - Diciembre 2025

**MATERIA:**
Patrones de diseño

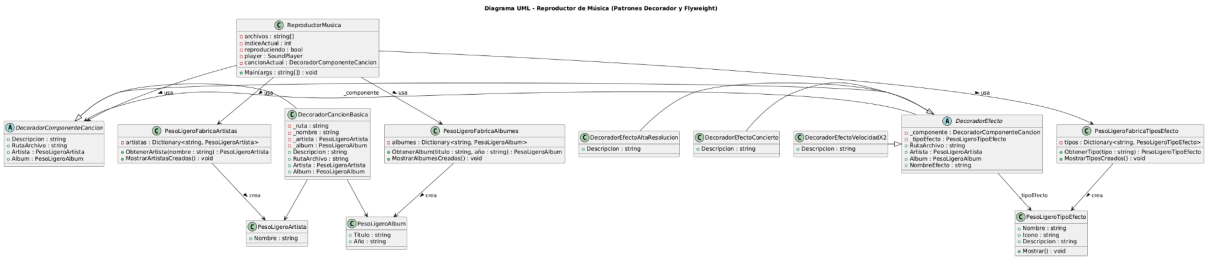
**TÍTULO ACTIVIDAD:**
Examen Unidad 3

**UNIDAD A EVALUAR:**
Unidad 3

**NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:**

Soberanes Oregel Cristopher Daniel C20211465

**NOMBRE DEL MAESTRO (A):**
Maribel Guerrero Luis




```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public class DecoradorCancionBasica :
DecoradorComponenteCancion
    {
        private string _ruta;
        private string _nombre;
        private PesoLigeroArtista _artista;
        private PesoLigeroAlbum _album;

        public DecoradorCancionBasica(string ruta,
PesoLigeroArtista artista = null, PesoLigeroAlbum album = null)
        {
            _ruta = ruta;
            _nombre = Path.GetFileName(ruta);
            _artista = artista;
            _album = album;
        }

        public override string Descripcion
        {
            get
            {
                string desc = $" {_nombre}";
                if (_artista != null)
                    desc += $"\n\t Artista: {_artista.Nombre}";
                if (_album != null)
```

```
                    desc += $"\n\t Álbum: {_album.Titulo}
({_album.Año})";
                desc += "\n\t- Reproducción Normal";
                return desc;
            }
        }

        public override string RutaArchivo => _ruta;
        public override PesoLigeroArtista Artista => _artista;
        public override PesoLigeroAlbum Album => _album;
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public abstract class DecoradorComponenteCancion
    {
        public abstract string Descripcion { get; }
        public abstract string RutaArchivo { get; }
        public abstract PesoLigeroArtista Artista { get; }
        public abstract PesoLigeroAlbum Album { get; }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public abstract class DecoradorEfecto :
DecoradorComponenteCancion
```

```
    {
        protected DecoradorComponenteCancion _componente;
        protected PesoLigeroTipoEfecto _tipoEfecto;

        public DecoradorEfecto(DecoradorComponenteCancion
componente, PesoLigeroTipoEfecto tipoEfecto)
        {
            _componente = componente;
            _tipoEfecto = tipoEfecto;
        }

        public override string RutaArchivo =>
_componente.RutaArchivo;
        public override PesoLigeroArtista Artista =>
_componente.Artista;
        public override PesoLigeroAlbum Album =>
_componente.Album;
        public string NombreEfecto => _tipoEfecto.Nombre;
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public class DecoradorEfectoAltaResolucion : DecoradorEfecto
    {
        public
DecoradorEfectoAltaResolucion(DecoradorComponenteCancion
componente)
            : base(componente,
PesoLigeroFabricaTiposEfecto.ObtenerTipo("AltaResolucion")) { }

        public override string Descripcion =>
_componente.Descripcion + $"\n\t-  {_tipoEfecto.Nombre}";
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public class DecoradorEfectoConcierto : DecoradorEfecto
    {
        public DecoradorEfectoConcierto(DecoradorComponenteCancion
componente)
            : base(componente,
PesoLigeroFabricaTiposEfecto.ObtenerTipo("Concierto")) { }

        public override string Descripcion =>
_componente.Descripcion + $"\n\t- {_tipoEfecto.Nombre}";
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public class DecoradorEfectoVelocidadX2 : DecoradorEfecto
    {
        public
DecoradorEfectoVelocidadX2(DecoradorComponenteCancion componente)
            : base(componente,
PesoLigeroFabricaTiposEfecto.ObtenerTipo("VelocidadX2")) { }

        public override string Descripcion =>
_componente.Descripcion + $"\n\t- {_tipoEfecto.Nombre}";
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public class PesoLigeroAlbum
    {
        public string Titulo { get; private set; }
        public string Año { get; private set; }

        public PesoLigeroAlbum(string titulo, string año)
        {
            Titulo = titulo;
            Año = año;
        }

        public override string ToString() => $"{Titulo} ({Año})";
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public class PesoLigeroArtista
    {
        public string Nombre { get; private set; }

        public PesoLigeroArtista(string nombre)
        {
            Nombre = nombre;
        }
```

```
        public override string ToString() => Nombre;
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public static class PesoLigeroFabricaAlbumes
    {
        private static Dictionary<string, PesoLigeroAlbum> albumes
= new Dictionary<string, PesoLigeroAlbum>();

        public static PesoLigeroAlbum ObtenerAlbum(string titulo,
string año)
        {
            string clave = $"{titulo.ToLower().Trim()}_{año}";
            if (!albumes.ContainsKey(clave))
            {
                albumes[clave] = new PesoLigeroAlbum(titulo, año);
                Console.ForegroundColor = ConsoleColor.DarkYellow;
                Console.WriteLine($"  [Peso Pluma] Nuevo álbum
creado: {titulo} ({año})");
                Console.ResetColor();
            }
            return albumes[clave];
        }

        public static void MostrarAlbumesCreados()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("  ÁLBUMES EN MEMORIA (Peso
liguerito)");

            Console.ResetColor();
```

```
            if (albumes.Count == 0)
            {
                Console.WriteLine("  Ningún álbum registrado
aún.");
            }
            else
            {
                int i = 1;
                foreach (var album in albumes.Values)
                {
                    Console.WriteLine($"  {i}.  {album.Titulo}
({album.Año})");
                    i++;
                }
            }
            Console.WriteLine($"\n  Total de objetos Album en
memoria: {albumes.Count}");
        }

        public static int CantidadAlbumes() => albumes.Count;
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public static class PesoLigeroFabricaArtistas
    {
        private static Dictionary<string, PesoLigeroArtista>
artistas = new Dictionary<string, PesoLigeroArtista>();

        public static PesoLigeroArtista ObtenerArtista(string
nombre)
        {
            string clave = nombre.ToLower().Trim();
            if (!artistas.ContainsKey(clave))
```

```
            {
                artistas[clave] = new PesoLigeroArtista(nombre);
                Console.ForegroundColor = ConsoleColor.DarkYellow;
                Console.WriteLine($"  [Flyweight] Nuevo artista
creado: {nombre}");
                Console.ResetColor();
            }
            return artistas[clave];
        }

        public static void MostrarArtistasCreados()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine("\n----------------------------------------");
            Console.WriteLine("  ARTISTAS EN MEMORIA
(FLYWEIGHT)");

Console.WriteLine("----------------------------------------");
            Console.ResetColor();

            if (artistas.Count == 0)
            {
                Console.WriteLine("  Ningún artista registrado
aún.");
            }
            else
            {
                int i = 1;
                foreach (var artista in artistas.Values)
                {
                    Console.WriteLine($"  {i}.
{artista.Nombre}");
                    i++;
                }
            }
            Console.WriteLine($"\n  Total de objetos Artista en
memoria: {artistas.Count}");
        }

        public static int CantidadArtistas() => artistas.Count;
    }
```

```
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
{
    public static class PesoLigeroFabricaTiposEfecto
    {
        private static Dictionary<string, PesoLigeroTipoEfecto>
tipos = new Dictionary<string, PesoLigeroTipoEfecto>();

        public static PesoLigeroTipoEfecto ObtenerTipo(string
tipo)
        {
            if (!tipos.ContainsKey(tipo))
            {
                switch (tipo)
                {
                    case "AltaResolucion":
                        tipos[tipo] = new
PesoLigeroTipoEfecto("Alta Resolución", "", "HD Audio de alta
calidad");
                        break;
                    case "VelocidadX2":
                        tipos[tipo] = new
PesoLigeroTipoEfecto("Velocidad x2", "", "Reproducción
acelerada");
                        break;
                    case "Concierto":
                        tipos[tipo] = new
PesoLigeroTipoEfecto("Modo Concierto", "", "Efecto Reverb
ambiental");
                        break;
                    default:
                        throw new ArgumentException($"Tipo de
efecto desconocido: {tipo}");
                }
```

```
            }
            return tipos[tipo];
        }

        public static void MostrarTiposCreados()
        {
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("\n");
            Console.WriteLine("  TIPOS DE EFECTOS EN MEMORIA
(FLYWEIGHT)");
            Console.ResetColor();

            if (tipos.Count == 0)
            {
                Console.WriteLine("  Ningún tipo de efecto creado
aún.");
            }
            else
            {
                foreach (var tipo in tipos.Values)
                {
                    Console.Write("  ");
                    tipo.Mostrar();
                }
            }
            Console.WriteLine($"\n  Total de objetos TipoEfecto en
memoria: {tipos.Count}");

Console.WriteLine("═══════════════════════════════════════════════
═══════\n");
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ReproductorMusicaDecorador
```

```
{
    public class PesoLigeroTipoEfecto
    {
        public string Nombre { get; private set; }

        public string Descripcion { get; private set; }

        public PesoLigeroTipoEfecto(string nombre, string icono,
string descripcion)
        {
            Nombre = nombre;

            Descripcion = descripcion;
        }

        public void Mostrar()
        {
            Console.WriteLine($" {Nombre}: {Descripcion}");
        }
    }
}


using System;
using System.IO;
using System.Media;
using System.Threading;
using System.Collections.Generic;

namespace ReproductorMusicaDecorador
{
    class ReproductorMusica
    {
        static Dictionary<string, PesoLigeroArtista>
artistasPorCancion = new Dictionary<string, PesoLigeroArtista>();
        static Dictionary<string, PesoLigeroAlbum>
albumesPorCancion = new Dictionary<string, PesoLigeroAlbum>();
        static SoundPlayer player;
        static string[] archivos;
        static int indiceActual = 0;
        static bool reproduciendo = false;
        static DecoradorComponenteCancion cancionActual;
```

```
        static void Main(string[] args)
        {
            Console.Title = "Reproductor de Música ";
            Console.ForegroundColor = ConsoleColor.Cyan;

            Console.WriteLine("REPRODUCTOR DE MUSICA - CONSOLA");
            Console.ResetColor();

            Console.Write("Ingresa la ruta de la carpeta con
archivos .wav: ");
            string carpeta = Console.ReadLine();

            if (!Directory.Exists(carpeta))
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Error: La carpeta no existe.");
                Console.ResetColor();
                Console.ReadKey();
                return;
            }

            archivos = Directory.GetFiles(carpeta, "*.wav");

            if (archivos.Length == 0)
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("No se encontraron archivos .wav
en la carpeta.");
                Console.ResetColor();
                Console.ReadKey();
                return;
            }

            player = new SoundPlayer();
            cancionActual = new
DecoradorCancionBasica(archivos[indiceActual]);
            MostrarMenu();
        }

        static void MostrarMenu()
```

```
        {
            while (true)
            {
                Console.Clear();
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine($"  CANCION ACTUAL:");
                Console.ResetColor();
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine(cancionActual.Descripcion);
                Console.ResetColor();

                // Mostrar efectos activos
                string efectosActivos = ObtenerEfectosActivos();
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine($"\n  Estado: {(reproduciendo ?
" REPRODUCIENDO" : " PAUSADO")}");
                Console.ForegroundColor = ConsoleColor.Magenta;
                Console.WriteLine($"  Efectos: {efectosActivos}");
                Console.ResetColor();
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine($"  Pista {indiceActual + 1} de
{archivos.Length}");

Console.WriteLine("-----------------------------------------\n");
                Console.ResetColor();

                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine("LISTA DE CANCIONES:");
                Console.ResetColor();

                for (int i = 0; i < archivos.Length; i++)
                {
                    if (i == indiceActual)
                        Console.ForegroundColor =
ConsoleColor.Cyan;
                    else
                        Console.ForegroundColor =
ConsoleColor.White;

                    Console.WriteLine($"  {i + 1}.
{Path.GetFileName(archivos[i])}");
                }
```

```
                Console.ResetColor();


Console.WriteLine("\n-----------------------------------------")
;
                Console.WriteLine("CONTROLES:");
                Console.WriteLine("  [P] Reproducir/Pausar");
                Console.WriteLine("  [N] Siguiente cancion");
                Console.WriteLine("  [A] Canción anterior");
                Console.WriteLine("  [S] Detener");
                Console.WriteLine("  [E] Agregar efectos");
                Console.WriteLine("  [R] Resetear efectos");
                Console.WriteLine("  [M] Editar Informacion
(Artista/Álbum)");
                Console.WriteLine("  [F] Ver Flyweight
(memoria)");
                Console.WriteLine("  [1-9] Seleccionar canción");
                Console.WriteLine("  [Q] Salir");

Console.WriteLine("-----------------------------------------
\n");

                Console.Write("Opción: ");
                var tecla = Console.ReadKey(true);

                switch (char.ToUpper(tecla.KeyChar))
                {
                    case 'P':
                        ReproducirPausar();
                        break;
                    case 'N':
                        Siguiente();
                        break;
                    case 'A':
                        Anterior();
                        break;
                    case 'S':
                        Detener();
                        break;
                    case 'E':
                        AgregarEfectos();
                        break;
```

```
                    case 'R':
                        ResetearEfectos();
                        break;
                    case 'F':
                        MostrarFlyweight();
                        break;
                    case 'M':
                        EditarMetadatos();
                        break;
                    case 'Q':
                        Detener();
                        return;
                    default:
                        if (char.IsDigit(tecla.KeyChar))
                        {
                            int num =
int.Parse(tecla.KeyChar.ToString());
                            if (num > 0 && num <= archivos.Length)
                            {
                                indiceActual = num - 1;
                                CambiarCancion();
                            }
                        }
                        break;
                }

                Thread.Sleep(100);
            }
        }

        static void EditarMetadatos()
        {
            Console.Clear();
            Console.ForegroundColor = ConsoleColor.Magenta;
            Console.WriteLine(" EDITAR INFO DE CANCION  \n");
            Console.ResetColor();

            string ruta = archivos[indiceActual];
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine($"Canción:
{Path.GetFileName(ruta)}\n");
            Console.ResetColor();
```

```
            Console.Write("Nombre del Artista: ");
            string nombreArtista = Console.ReadLine();

            Console.Write("Título del Álbum: ");
            string tituloAlbum = Console.ReadLine();

            Console.Write("Año del Álbum: ");
            string añoAlbum = Console.ReadLine();
            if (string.IsNullOrWhiteSpace(añoAlbum))
                añoAlbum = "Desconocido";

            PesoLigeroArtista artista = null;
            PesoLigeroAlbum album = null;

            if (!string.IsNullOrWhiteSpace(nombreArtista))
            {
                artista =
PesoLigeroFabricaArtistas.ObtenerArtista(nombreArtista);
                artistasPorCancion[ruta] = artista;
            }

            if (!string.IsNullOrWhiteSpace(tituloAlbum))
            {
                album =
PesoLigeroFabricaAlbumes.ObtenerAlbum(tituloAlbum, añoAlbum);
                albumesPorCancion[ruta] = album;
            }


            cancionActual = new DecoradorCancionBasica(ruta,
artista, album);

            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("\n✓ Metadatos actualizados
correctamente");
            Console.ResetColor();
            Thread.Sleep(2000);
        }
```

```
        static void MostrarFlyweight()
        {
            Console.Clear();


            PesoLigeroFabricaTiposEfecto.MostrarTiposCreados();


            PesoLigeroFabricaArtistas.MostrarArtistasCreados();


            PesoLigeroFabricaAlbumes.MostrarAlbumesCreados();


            Console.ForegroundColor = ConsoleColor.Yellow;

            Console.WriteLine("\nPresiona cualquier tecla para
continuar...");
            Console.ReadKey();
        }

        static void AgregarEfectos()
        {
            Console.Clear();
            Console.ForegroundColor = ConsoleColor.Magenta;

            Console.WriteLine("MENÚ DE EFECTOS DE AUDIO
\n");
            Console.ResetColor();

            Console.WriteLine("Selecciona un efecto para
aplicar:");
            Console.WriteLine("  1) Alta Resolucion (HD Audio)");
            Console.WriteLine("  2) Velocidad x2");
            Console.WriteLine("  3) Modo Concierto");
            Console.WriteLine("  8) Volver");

            Console.Write("\nOpción: ");
            string opcion = Console.ReadLine();

            switch (opcion)
            {
```

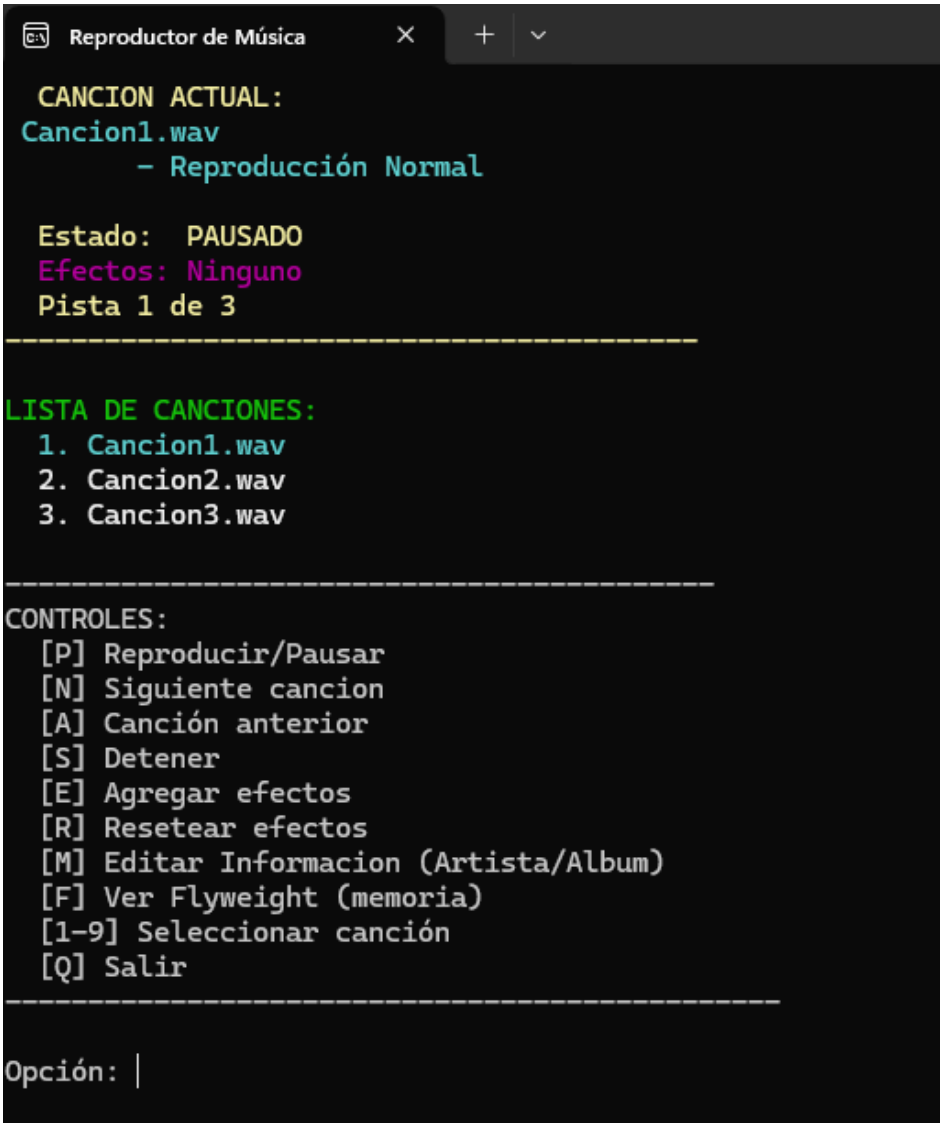
```
                case "1":
                    cancionActual = new
DecoradorEfectoAltaResolucion(cancionActual);
                    MostrarMensajeEfecto(" Alta Resolucion
aplicada");
                    break;
                case "2":
                    cancionActual = new
DecoradorEfectoVelocidadX2(cancionActual);
                    MostrarMensajeEfecto(" Velocidad x2
aplicada");
                    break;
                case "3":
                    cancionActual = new
DecoradorEfectoConcierto(cancionActual);
                    MostrarMensajeEfecto(" Modo Concierto
activado");
                    break;
            }
        }

        static void MostrarMensajeEfecto(string mensaje)
        {
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine($"\n{mensaje}");
            Console.ResetColor();
            Console.WriteLine("\n¿Deseas agregar otro efecto?");
            Console.WriteLine("1) Sí");
            Console.WriteLine("2) No");
            Console.Write("\nOpción: ");
            string respuesta = Console.ReadLine();

            if (respuesta == "1")
            {
                AgregarEfectos();
            }
        }

        static void ResetearEfectos()
        {

            PesoLigeroArtista artista = cancionActual.Artista;
```

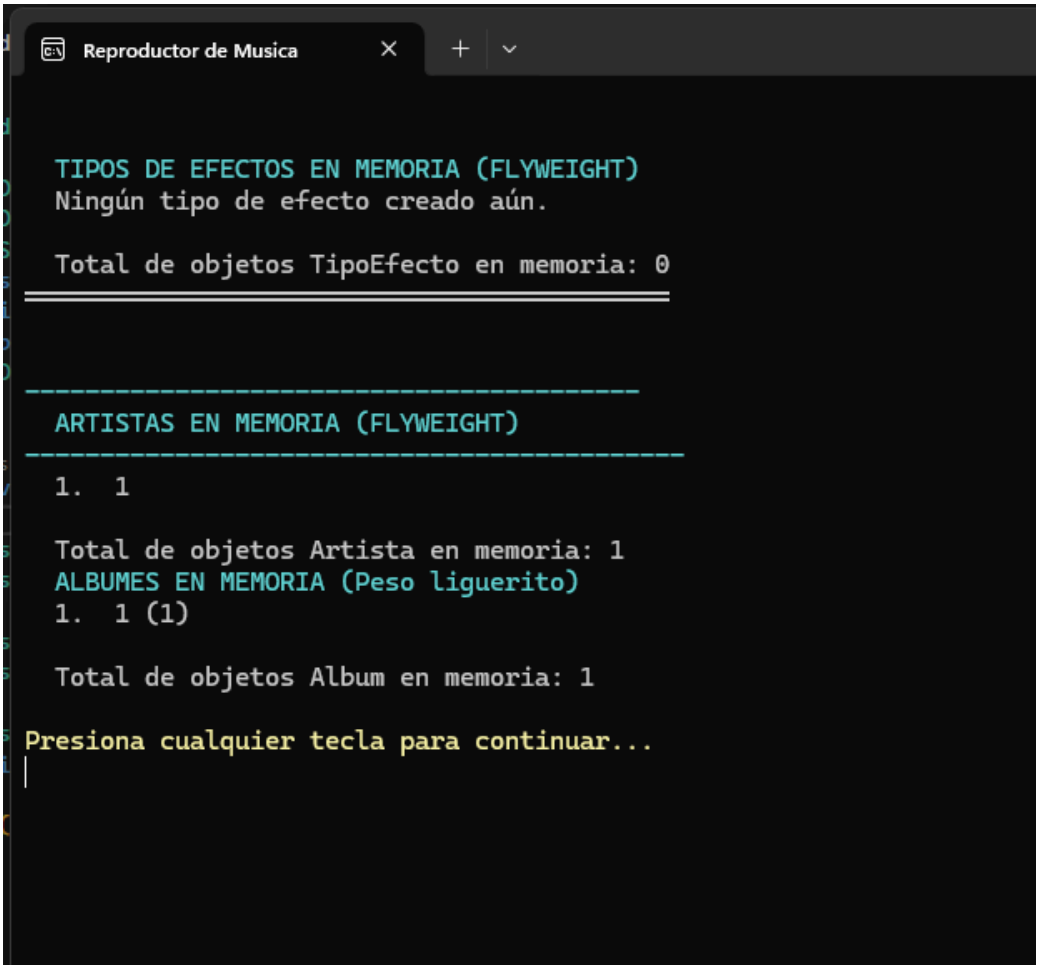
```
            PesoLigeroAlbum album = cancionActual.Album;
            cancionActual = new
DecoradorCancionBasica(archivos[indiceActual], artista, album);

            Console.Clear();
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine(" Todos los efectos han sido
removidos");
            Console.WriteLine("  (Metadatos de artista y álbum
conservados)");
            Console.ResetColor();
            Thread.Sleep(2000);
        }

        static void ReproducirPausar()
        {
            if (reproduciendo)
            {
                player.Stop();
                reproduciendo = false;
            }
            else
            {
                ReproducirCancion();
            }
        }

        static void ReproducirCancion()
        {
            try
            {
                player.SoundLocation = cancionActual.RutaArchivo;
                player.Play();
                reproduciendo = true;
            }
            catch (Exception ex)
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine($"\nError al reproducir:
{ex.Message}");
                Console.ResetColor();
                Thread.Sleep(2000);
```

```
                reproduciendo = false;
            }
        }

        static void CambiarCancion()
        {
            string ruta = archivos[indiceActual];
            PesoLigeroArtista artista =
artistasPorCancion.ContainsKey(ruta) ? artistasPorCancion[ruta] :
null;
            PesoLigeroAlbum album =
albumesPorCancion.ContainsKey(ruta) ? albumesPorCancion[ruta] :
null;

            cancionActual = new DecoradorCancionBasica(ruta,
artista, album);

            Console.Clear();
            Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine("-----------------------------------------")
;
            Console.ResetColor();

            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine($" Reproduciendo:
{Path.GetFileName(ruta)}");
            Console.ResetColor();

            if (artista != null)
            {
                Console.ForegroundColor = ConsoleColor.Green;
                Console.WriteLine($" Artista: {artista.Nombre}");
            }
            else
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine($" Artista: Desconocido");
            }

            if (album != null)
            {
```

```
                Console.ForegroundColor = ConsoleColor.Magenta;
                Console.WriteLine($" Álbum: {album.Titulo}
({album.Año})");
            }
            else
            {
                Console.ForegroundColor = ConsoleColor.DarkGray;
                Console.WriteLine($" Álbum: Desconocido");
            }

            Console.ResetColor();

Console.WriteLine("---------------------------------------\n");

            string efectos = ObtenerEfectosActivos();
            Console.ForegroundColor = ConsoleColor.Blue;
            Console.WriteLine($"Efectos: {efectos}");
            Console.ResetColor();

            if (reproduciendo)
                ReproducirCancion();

            Thread.Sleep(2000);
        }



        static void Siguiente()
        {
            indiceActual = (indiceActual + 1) % archivos.Length;
            CambiarCancion();
        }

        static void Anterior()
        {
            indiceActual = (indiceActual - 1 + archivos.Length) %
archivos.Length;
            CambiarCancion();
        }

        static void Detener()
        {
```

```
            player.Stop();
            reproduciendo = false;
        }

        static string ObtenerEfectosActivos()
        {
            var efectos = new List<string>();
            DecoradorComponenteCancion temp = cancionActual;

            while (temp is DecoradorEfecto decorador)
            {
                efectos.Add(decorador.NombreEfecto);
                var campo =
typeof(DecoradorEfecto).GetField("_componente",
                    System.Reflection.BindingFlags.NonPublic |
System.Reflection.BindingFlags.Instance);
                temp =
(DecoradorComponenteCancion)campo.GetValue(decorador);
            }

            if (efectos.Count == 0)
                return "Ninguno";

            efectos.Reverse();
            return string.Join(", ", efectos);
        }
    }
}
```


Reproductor de Música
CANCION ACTUAL:
Cancion1.wav
- Reproducción Normal
Estado: PAUSADO
Efectos: Ninguno
Pista 1 de 3
LISTA DE CANCIONES:
1. Cancion1.wav
2. Cancion2.wav
3. Cancion3.wav
CONTROLES:
[P] Reproducir/Pausar
[N] Siguiente cancion
[A] Canción anterior
[S] Detener
[E] Agregar efectos
[R] Resetear efectos
[M] Editar Informacion (Artista/Album)
[F] Ver Flyweight (memoria)
[1-9] Seleccionar canción
[Q] Salir
Opción:


Reproductor de Musica
TIPOS DE EFECTOS EN MEMORIA (FLYWEIGHT)
Ningún tipo de efecto creado aún.
Total de objetos TipoEfecto en memoria: 0
ARTISTAS EN MEMORIA (FLYWEIGHT)
1. 1
Total de objetos Artista en memoria: 1
ALBUMES EN MEMORIA (Peso liguerito)
1. 1 (1)
Total de objetos Album en memoria: 1
Presiona cualquier tecla para continuar...

**Conclusión:**
No invente profesora, mejor diga que me quería reprobar. Taba bien dificil el examen, cómo implementamos el patrón Peso ligero en un  reproductor de música, las opciones estaban muy limitadas, si bien estuvo  interesante y se practicó más el patrón, este resultaba difícil.