TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto - Diciembre 2025

CARRERA:
Ingeniería Informática

MATERIA:
Patrones de Diseño

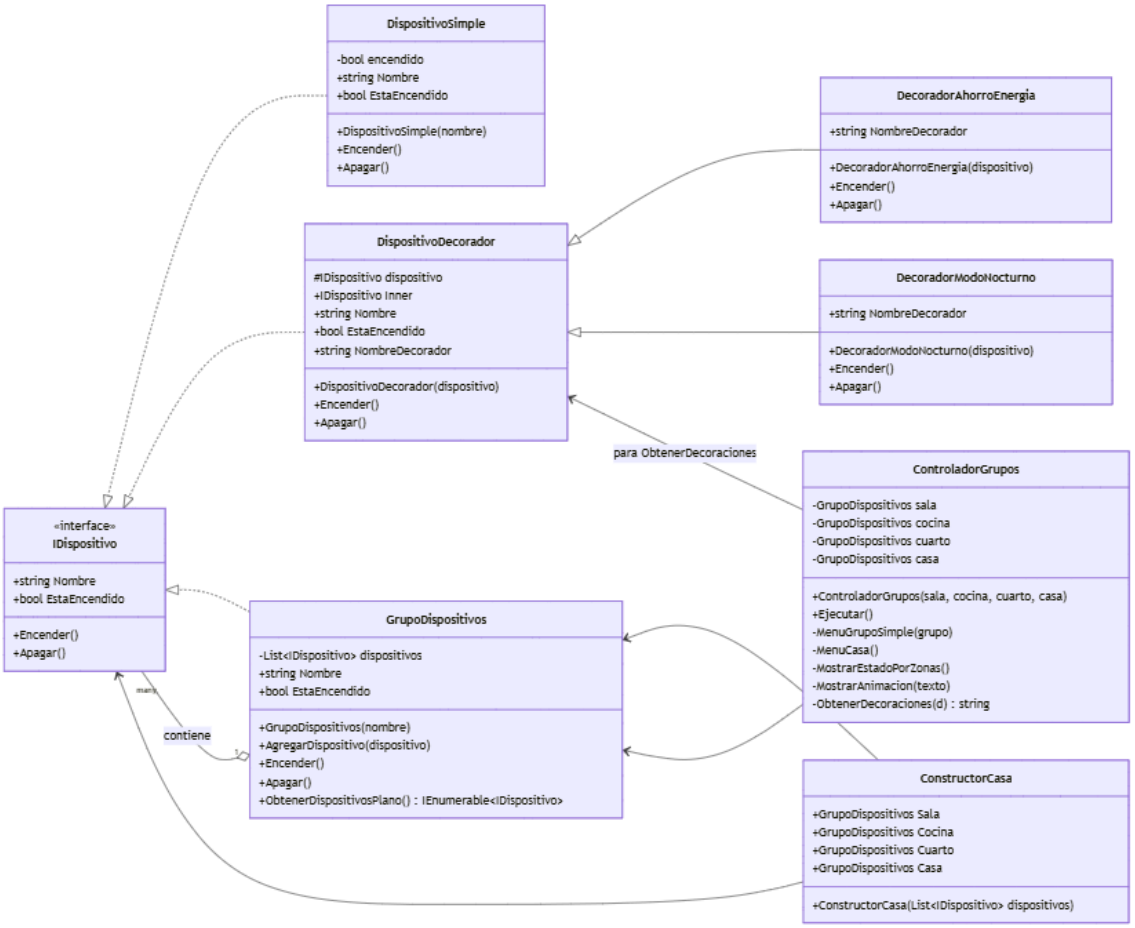
TÍTULO ACTIVIDAD:
Examen unidad 3

UNIDAD A EVALUAR:
3

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:
Gonzalez Gutierrez Daniel Alejandro 21212341

NOMBRE DEL MAESTRO (A):
Maribel Guerrero Luis

Diagrama UML



Código

Constructor Casa

```
namespace ExamenU3_Patrones_DanielAlejandroGonzalezGutierrez
{
    2 referencias
    public class ConstructorCasa
    {
        6 referencias
        public GrupoDispositivos Sala { get; }
        5 referencias
        public GrupoDispositivos Cocina { get; }
        5 referencias
        public GrupoDispositivos Cuarto { get; }
        5 referencias
        public GrupoDispositivos Casa { get; }

        1 referencia
        public ConstructorCasa(List<IDispositivo> dispositivos)
        {
            if (dispositivos == null || dispositivos.Count < 7)
                throw new ArgumentException("");

            Sala = new GrupoDispositivos("Sala");
            Sala.AgregarDispositivo(dispositivos[0]);
            Sala.AgregarDispositivo(dispositivos[1]);
            Sala.AgregarDispositivo(dispositivos[2]);

            Cocina = new GrupoDispositivos("Cocina");
            Cocina.AgregarDispositivo(dispositivos[3]);
            Cocina.AgregarDispositivo(dispositivos[4]);

            Cuarto = new GrupoDispositivos("Cuarto");
            Cuarto.AgregarDispositivo(dispositivos[5]);
            Cuarto.AgregarDispositivo(dispositivos[6]);

            Casa = new GrupoDispositivos("Casa");
            Casa.AgregarDispositivo(Sala);
            Casa.AgregarDispositivo(Cocina);
            Casa.AgregarDispositivo(Cuarto);
        }
    }
```

DispositivoDecorador

```
public abstract class DispositivoDecorador : IDispositivo
{
    protected IDispositivo dispositivo;

    2 referencias
    protected DispositivoDecorador(IDispositivo dispositivo)
    {
        this.dispositivo = dispositivo;
    }

    1 referencia
    public IDispositivo Inner => dispositivo;

    11 referencias
    public virtual string Nombre => dispositivo.Nombre;

    8 referencias
    public virtual bool EstaEncendido => dispositivo.EstaEncendido;

    3 referencias
    public virtual string NombreDecorador => "Decorador";

    7 referencias
    public virtual void Encender()
    {
        dispositivo.Encender();
    }

    7 referencias
    public virtual void Apagar()
    {
        dispositivo.Apagar();
    }
}
```

Controlador Grupos

```
2 referencias
public class ControladorGrupos
{
    private readonly GrupoDispositivos sala;
    private readonly GrupoDispositivos cocina;
    private readonly GrupoDispositivos cuarto;
    private readonly GrupoDispositivos casa;

    1 referencia
    public ControladorGrupos(GrupoDispositivos sala, GrupoDispositivos cocina, GrupoDispositivos cuarto, GrupoDispositivos casa
    {
        this.sala = sala;
        this.cocina = cocina;
        this.cuarto = cuarto;
        this.casa = casa;
    }

    1 referencia
    public void Ejecutar()
    {
        bool salir = false;

        while (!salir)
        {
            Console.Clear();
            Console.WriteLine("====   Control de dispositivos inteligentes   ====================\n");
            Console.WriteLine("Seleccione un grupo:");
            Console.WriteLine("1. Sala");
            Console.WriteLine("2. Cocina");
            Console.WriteLine("3. Cuarto");
            Console.WriteLine("4. Casa");
            Console.WriteLine("5. Salir");

            string opcionGrupo = Console.ReadLine();

            GrupoDispositivos grupoSeleccionado = null;

            if (opcionGrupo == "1") grupoSeleccionado = sala;
            else if (opcionGrupo == "2") grupoSeleccionado = cocina;
            else if (opcionGrupo == "3") grupoSeleccionado = cuarto;
            else if (opcionGrupo == "4") grupoSeleccionado = casa;
            else if (opcionGrupo == "5")
            {
                salir = true;
                Console.WriteLine("Saliendo de control de dispositivos.");
                Console.ReadKey();
            }

            if (grupoSeleccionado == null)
            {
                if (!salir)
                {
                    Console.WriteLine("Opción no válida");
                    Console.ReadKey();
                }
                continue;
            }
```

```
        if (grupoSeleccionado == casa)
        {
            MenuCasa();
        }
        else
        {
            MenuGrupoSimple(grupoSeleccionado);
        }
    }
}

1 referencia
private void MenuGrupoSimple(GrupoDispositivos grupo)
{
    Console.Clear();
    Console.WriteLine("Seleccione acción para el grupo " + grupo.Nombre + "\n");
    Console.WriteLine("1. Encender");
    Console.WriteLine("2. Apagar");
    Console.WriteLine("3. Regresar");
    string opcion = Console.ReadLine();

    var hojasGrupo = grupo.ObtenerDispositivosPlano().ToList();

    if (opcion == "1")
    {
        bool hayApagados = hojasGrupo.Any(d => !d.EstaEncendido);
        if (!hayApagados)
        {
            Console.Clear();
            Console.WriteLine("====   Grupo " + grupo.Nombre + "   ====================\n");
            Console.WriteLine("No hay dispositivos para encender");
            Console.WriteLine("\n=======================================\n");
            Console.WriteLine("\nPresione una tecla para regresar");
            Console.ReadKey();
        }
        else
        {
            Console.Clear();
            Console.WriteLine("====   Grupo " + grupo.Nombre + "   ====================");
            MostrarAnimacion("Encendiendo");
            grupo.Encender();
            Console.WriteLine("\n=======================================\n");
            Console.WriteLine("\nPresione una tecla para regresar");
            Console.ReadKey();
        }
    }
    else if (opcion == "2")
    {
        bool hayEncendidos = hojasGrupo.Any(d => d.EstaEncendido);
        if (!hayEncendidos)
        {
            Console.Clear();
            Console.WriteLine("====   Grupo " + grupo.Nombre + "   ====================\n");
            Console.WriteLine("No hay dispositivos para apagar");
            Console.WriteLine("\n=======================================\n");
            Console.WriteLine("\nPresione una tecla para regresar");
            Console.ReadKey();
        }
```

```
        else
        {
            Console.Clear();
            Console.WriteLine("====   Grupo " + grupo.Nombre + "   ====================");
            MostrarAnimacion("Apagando");
            grupo.Apagar();
            Console.WriteLine("\n=======================================\n");
            Console.WriteLine("\nPresione una tecla para regresar");
            Console.ReadKey();
        }
    }
}
```

```
private void MenuCasa()
{
    bool regresar = false;

    while (!regresar)
    {
        Console.Clear();
        Console.WriteLine("Seleccione acción para el grupo " + casa.Nombre + "\n");
        Console.WriteLine("1. Encender");
        Console.WriteLine("2. Apagar");
        Console.WriteLine("3. Mostrar estado de dispositivos");
        Console.WriteLine("4. Regresar");

        string opcion = Console.ReadLine();

        var hojasCasa = casa.ObtenerDispositivosPlano().ToList();

        if (opcion == "1")
        {
            bool hayApagados = hojasCasa.Any(d => !d.EstaEncendido);

            Console.Clear();
            Console.WriteLine("====   Grupo " + casa.Nombre + "   ====================");

            if (!hayApagados)
            {
                Console.WriteLine();
                Console.WriteLine("No hay dispositivos para encender");
            }
            else
            {
                MostrarAnimacion("Encendiendo");
                casa.Encender();

            }

            Console.WriteLine("\n======================================");
            Console.WriteLine("\nPresione una tecla para regresar");
            Console.ReadKey();
        }
        else if (opcion == "2")
        {
            bool hayEncendidos = hojasCasa.Any(d => d.EstaEncendido);

            Console.Clear();
            Console.WriteLine("====   Grupo " + casa.Nombre + "   ====================");

            if (!hayEncendidos)
            {
                Console.WriteLine();
                Console.WriteLine("No hay dispositivos para apagar");
            }
            else
            {
                MostrarAnimacion("Apagando");
                casa.Apagar();

            }

            Console.WriteLine("\n======================================");
            Console.WriteLine("\nPresione una tecla para regresar");
            Console.ReadKey();
        }
        else if (opcion == "3")
        {
            Console.Clear();
            Console.WriteLine("======================================");
            Console.WriteLine("Estado de dispositivos en " + casa.Nombre + ":");
            Console.WriteLine("======================================\n");
            MostrarEstadoPorZonas();
            Console.WriteLine("\n======================================");
            Console.WriteLine("\nPresione una tecla para regresar");
            Console.ReadKey();
        }
```

```
            }
            else if (opcion == "4")
            {
                regresar = true;
            }
            else
            {
                Console.WriteLine("Opción no válida");
                Console.ReadKey();
            }
        }
    }

    1 referencia
    private void MostrarEstadoPorZonas()
    {
        foreach (var sub in new[] { sala, cocina, cuarto })
        {
            Console.WriteLine();
            Console.WriteLine(sub.Nombre + ":");
            foreach (var d in sub.ObtenerDispositivosPlano())
            {
                string decor = ObtenerDecoraciones(d);
                Console.WriteLine("  - " + d.Nombre + " : " + (d.EstaEncendido ? "Encendido" : "Apagado") + decor);
            }
        }
    }

    4 referencias
    private static void MostrarAnimacion(string texto)
    {
        Console.Write("Cargando");
        int dotCount = 10;
        int dotsStartLeft = Console.CursorLeft;
        int dotsStartTop = Console.CursorTop;
        string prefix = "Cargando";


        for (int i = 0; i < dotCount; i++)
        {
            Thread.Sleep(200);
            Console.Write(".");
        }
        int startLeftFull = Math.Max(0, dotsStartLeft - prefix.Length);
        Console.SetCursorPosition(startLeftFull, dotsStartTop);
        Console.Write(new string(' ', prefix.Length + dotCount));
        Console.SetCursorPosition(0, dotsStartTop + 1);
    }

    1 referencia
    private static string ObtenerDecoraciones(IDispositivo d)
    {
        var decoradores = new List<string>();
        while (d is DispositivoDecorador dec)
        {
            decoradores.Add(dec.NombreDecorador);
            d = dec.Inner;
        }

        if (decoradores.Count == 0) return string.Empty;

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

DecoradorAhorroEnergia

```
public class DecoradorAhorroEnergia : DispositivoDecorador
{
    1 referencia
    public DecoradorAhorroEnergia(IDispositivo dispositivo) : base(dispositivo)
    {
    }

    2 referencias
    public override string NombreDecorador => "Ahorro de energía";

    6 referencias
    public override void Encender()
    {
        base.Encender();
        Console.WriteLine("(Activando modo ahorro de energía en " + Nombre + ")");
    }

    6 referencias
    public override void Apagar()
    {
        Console.WriteLine("(Guardando consumo de energía de " + Nombre + ")");
        base.Apagar();
    }
}
```

DecoradorModoNocturno

```
public class DecoradorModoNocturno : DispositivoDecorador
{
    1 referencia
    public DecoradorModoNocturno(IDispositivo dispositivo) : base(dispositivo)
    {
    }

    2 referencias
    public override string NombreDecorador => "Modo nocturno";

    6 referencias
    public override void Encender()
    {
        base.Encender();
        Console.WriteLine("(Ajustando brillo y volumen para modo nocturno en " + Nombre+")");
    }

    6 referencias
    public override void Apagar()
    {
        Console.WriteLine("(Restaurando configuraciones normales de " + Nombre+")");
        base.Apagar();
    }
}
```

DispositivoSimple

```
public class DispositivoSimple : IDispositivo
{
    10 referencias
    public string Nombre { get; }

    private bool encendido;

    8 referencias
    public bool EstaEncendido => encendido;

    7 referencias
    public DispositivoSimple(string nombre)
    {
        Nombre = nombre;
        encendido = false;
    }

    3 referencias
    public virtual void Encender()
    {
        if (!encendido)
        {
            encendido = true;
            Console.WriteLine("--- " + Nombre + " Encendido");
        }
    }

    3 referencias
    public virtual void Apagar()
    {
        if (encendido)
        {
            encendido = false;
            Console.WriteLine("--- " + Nombre + " Apagado");
        }
    }
}
```

GrupoDispositivos

```
20 referencias
public class GrupoDispositivos : IDispositivo
{
    private readonly List<IDispositivo> dispositivos = new List<IDispositivo>();

    20 referencias
    public string Nombre { get; }

    4 referencias
    public GrupoDispositivos(string nombre)
    {
        Nombre = nombre;
    }

    8 referencias
    public bool EstaEncendido
    {
        get
        {
            foreach (var d in dispositivos)
            {
                if (d.EstaEncendido) return true;
            }
            return false;
        }
    }

    10 referencias
    public void AgregarDispositivo(IDispositivo dispositivo)
    {
        dispositivos.Add(dispositivo);
    }

    5 referencias
    public void Encender()
    {
        Console.WriteLine();
        Console.WriteLine("Encendido grupo " + Nombre);
        foreach (var d in dispositivos)
        {
            d.Encender();
        }
    }

    5 referencias
    public void Apagar()
    {
        Console.WriteLine();
        Console.WriteLine("Apagado grupo " + Nombre);
        foreach (var d in dispositivos)
        {
            d.Apagar();
        }
    }

    4 referencias
    public IEnumerable<IDispositivo> ObtenerDispositivosPlano()
    {
        foreach (var d in dispositivos)
        {
            if (d is GrupoDispositivos g)
            {
                foreach (var sub in g.ObtenerDispositivosPlano())
                    yield return sub;
            }
            else
            {
                yield return d;
            }
        }
    }
}
```

IDispositivo

```
public interface IDispositivo
{
    29 referencias
    string Nombre { get; }
    10 referencias
    bool EstaEncendido { get; }
    11 referencias
    void Encender();
    11 referencias
    void Apagar();
}
```

Program

```
0 referencias
static void Main(string[] args)
{
    List<IDispositivo> dispositivos = CrearDispositivosIniciales();
    DecorarDispositivos(dispositivos);

    var constructor = new ConstructorCasa(dispositivos);
    var controlador = new ControladorGrupos(constructor.Sala, constructor.Cocina, constructor.Cuarto, constructor.Casa);

    controlador.Ejecutar();
}

1 referencia
static List<IDispositivo> CrearDispositivosIniciales()
{
    List<IDispositivo> lista = new List<IDispositivo>();
    lista.Add(new DispositivoSimple("Televisor"));
    lista.Add(new DispositivoSimple("Bocinas"));
    lista.Add(new DispositivoSimple("Foco sala"));
    lista.Add(new DispositivoSimple("Microondas"));
    lista.Add(new DispositivoSimple("Refrigerador"));
    lista.Add(new DispositivoSimple("Luces LED cuarto"));
    lista.Add(new DispositivoSimple("PC escritorio"));
    return lista;
}

1 referencia
static void DecorarDispositivos(List<IDispositivo> dispositivos)
{
    while (true)
    {
        Console.Clear();
        Console.WriteLine("====   Dispositivos disponibles para decorar:   ====================\n");

        for (int i = 0; i < dispositivos.Count; i++)
        {
            bool esDecorado = dispositivos[i] is DispositivoDecorador;
            string etiqueta = esDecorado ? " (decorado)" : "";
            Console.WriteLine((i + 1) + ". " + dispositivos[i].Nombre + etiqueta);
        }

        Console.WriteLine((dispositivos.Count + 1) + ". Terminar decoracion \n");

        Console.WriteLine("Seleccione una opción:");
        string entrada = Console.ReadLine();
        int opcion;

        if (!int.TryParse(entrada, out opcion))
        {
            Console.WriteLine("Opción inválida");
            Console.ReadKey();
            continue;
        }
```

```
if (opcion == dispositivos.Count + 1)
{
    Console.WriteLine("Decoracion finalizada presione una tecla para continuar");
    Console.ReadKey();
    break;
}

if (opcion < 1 || opcion > dispositivos.Count)
{
    Console.WriteLine("Opción inválida");
    Console.ReadKey();
    continue;
}

IDispositivo seleccionado = dispositivos[opcion - 1];

int decorador1 = ElegirTipoDecorador(seleccionado.Nombre);

if (decorador1 == 0)
    continue;

IDispositivo decorado1 = AplicarDecorador(seleccionado, decorador1);

int decorador2 = ElegirTipoDecorador(seleccionado.Nombre, decorador1);

if (decorador2 == 0)
{
    dispositivos[opcion - 1] = decorado1;
    Console.WriteLine("Decoración finalizada para " + seleccionado.Nombre + " (un decorador).");
    Console.WriteLine("Presione una tecla para continuar");
    Console.ReadKey();
    continue;
}

IDispositivo decoradoFinal = AplicarDecorador(decorado1, decorador2);

dispositivos[opcion - 1] = decoradoFinal;
```

```
static int ElegirTipoDecorador(string nombreDispositivo, int? decoradorYaElegido = null)
{
    while (true)
    {
        Console.Clear();

        Console.WriteLine();
        Console.WriteLine("=======================================");
        Console.WriteLine("Seleccione decorador para " + nombreDispositivo);
        Console.WriteLine("=======================================");
        Console.WriteLine("Elija decorador:\n");

        if (decoradorYaElegido != 1)
            Console.WriteLine("1. Decorador ahorro de energía");

        if (decoradorYaElegido != 2)
            Console.WriteLine("2. Decorador modo nocturno");

        if (decoradorYaElegido != null)
            Console.WriteLine("3. Terminar decoración");
        else
            Console.WriteLine("3. Regresar");

        string opcion = Console.ReadLine();

        if (opcion == "1" && decoradorYaElegido != 1) return 1;
        if (opcion == "2" && decoradorYaElegido != 2) return 2;
        if (opcion == "3") return 0;

        Console.WriteLine("Opción no válida, intente de nuevo.");
        Console.ReadKey();
    }
}

2 referencias
static IDispositivo AplicarDecorador(IDispositivo baseDispositivo, int tipo)
{
    if (tipo == 1) return new DecoradorAhorroEnergia(baseDispositivo);
    if (tipo == 2) return new DecoradorModoNocturno(baseDispositivo);
    return baseDispositivo;
}
```

Impresión

```
====   Dispositivos disponibles para decorar:   ====================

1. Televisor
2. Bocinas
3. Foco sala
4. Microondas
5. Refrigerador
6. Luces LED cuarto
7. PC escritorio
8. Terminar decoracion

Seleccione una opción:
```

```
========================================
Seleccione decorador para Televisor
========================================
Elija decorador:

1. Decorador ahorro de energía
2. Decorador modo nocturno
3. Regresar
```

```
====   Control de dispositivos inteligentes   ====================

Seleccione un grupo:
1. Sala
2. Cocina
3. Cuarto
4. Casa
5. Salir
```

```
Seleccione acción para el grupo Casa

1. Encender
2. Apagar
3. Mostrar estado de dispositivos
4. Regresar
```

```
====================================
Estado de dispositivos en Casa:
====================================


Sala:
  - Televisor : Apagado
  - Bocinas : Apagado
  - Foco sala : Apagado
```

```
Cocina:
  - Microondas : Apagado
  - Refrigerador : Apagado

Cuarto:
  - Luces LED cuarto : Apagado
  - PC escritorio : Apagado

====================================

Presione una tecla para regresar
```

Conclusiones

En conclusión, el uso de los dos patrones dentro de este proyecto ,El decorar diferentes dispositivos y a la vez que el controlador no le importa si es un grupo o un dispositivo, siempre llama Encender y Apagar.permitiendo tratar un dispositivo individual y un grupo de dispositivos con la misma interfaz.