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

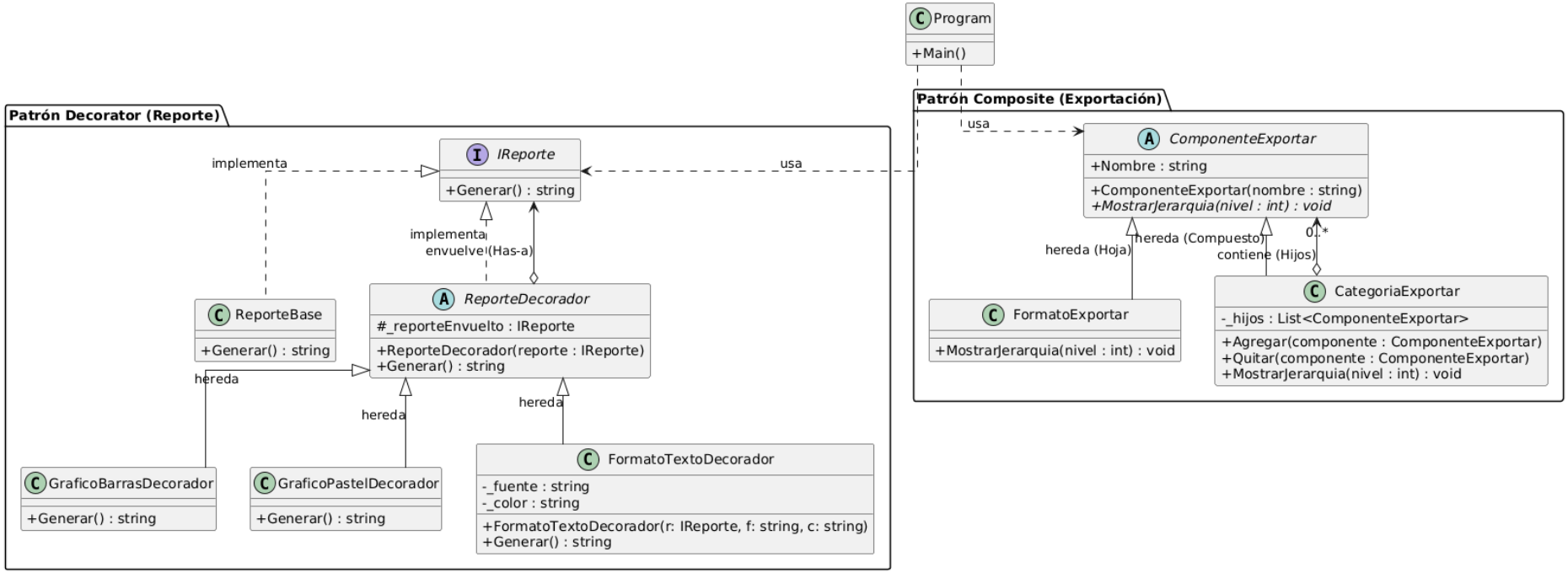
TÍTULO ACTIVIDAD:
Examen U3

UNIDAD A EVALUAR:
Unidad 3

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:
Angulo Berrelleza Fernando 21212322

NOMBRE DEL MAESTRO (A):
Maribel Guerrero Luis

Diagrama


Program
+Main()
Patrón Decorator (Reporte)
IReporte
+Generar() : string
implementa
usa
implementa
envuelve
(Has-a)
ReporteBase
+Generar() : string
ReporteDecorador
#_reporteEnvuelto : IReporte
+ReporteDecorador(reporte : IReporte)
+Generar() : string
hereda
hereda
hereda
GraficoBarrasDecorador
+Generar() : string
GraficoPastelDecorador
+Generar() : string
FormatoTextoDecorador
-_fuente : string
-_color : string
+FormatoTextoDecorador(r: IReporte, f: string, c: string)
+Generar() : string
Patrón Composite (Exportación)
usa
ComponenteExportar
+Nombre : string
+ComponenteExportar(nombre : string)
+MostrarJerarquia(nivel : int) : void
hereda (Hoja)
hereda (Compuesto)
contiene (Hijos)
0..*
FormatoExportar
+MostrarJerarquia(nivel : int) : void
CategoriaExportar
-_hijos : List<ComponenteExportar>
+Agregar(componente : ComponenteExportar)
+Quitar(componente : ComponenteExportar)
+MostrarJerarquia(nivel : int) : void

Código

CategoriaExportar

```
namespace Examen
{
    4 referencias
    public class CategoriaExportar : ComponenteExportar
    {
        private List<ComponenteExportar> _hijos = new List<ComponenteExportar>();
        3 referencias
        public CategoriaExportar(string nombre) : base(nombre) { }
        8 referencias
        public void Agregar(ComponenteExportar componente)
        {
            _hijos.Add(componente);
        }
        0 referencias
        public void Quitar(ComponenteExportar componente)
        {
            _hijos.Remove(componente);
        }
        3 referencias
        public override void MostrarJerarquia(int nivel)
        {
            Console.WriteLine(new string ('+', nivel) + " " + Nombre);

            foreach (var hijo in _hijos)
            {
                hijo.MostrarJerarquia(nivel + 1);
            }
        }
    }
}
```

ComponenteExportar

```
namespace Examen
{
    9 referencias
    public abstract class ComponenteExportar
    {
        3 referencias
        public string Nombre { get; set; }
        2 referencias
        public ComponenteExportar(string nombre)
        {
            Nombre = nombre;
        }
        4 referencias
        public abstract void MostrarJerarquia(int nivel);
    }
}
```

FormatoExportar

```
namespace Examen
{
    7 referencias
    public class FormatoExportar : ComponenteExportar
    {
        6 referencias
        public FormatoExportar(string nombre) : base(nombre) { }
        2 referencias
        public override void MostrarJerarquia(int nivel)
        {
            Console.WriteLine(new string('-', nivel) + " " + Nombre);
        }
    }
}
```

FormatoTextoDecorador

```
namespace Examen
{
    2 referencias
    public class FormatoTextoDecorador : ReporteDecorador
    {
        private string _fuente;
        private string _color;
        1 referencia
        public FormatoTextoDecorador(IReporte reporte, string fuente, string color) : base(reporte)
        {
            _fuente = fuente;
            _color = color;
        }
        8 referencias
        public override string Generar()
        {
            return base.Generar() + $"\n + [Formato: Fuente={_fuente}, Color={_color}] aplicado";
        }
    }
}
```

GraficoBarrasDecorador

```
namespace Examen
{

    public class GraficoBarrasDecorador : ReporteDecorador
    {

        public GraficoBarrasDecorador(IReporte reporte) : base(reporte)
        {
        }
        8 referencias
        public override string Generar()
        {
            return base.Generar() + "\n + [Gráfico de Barras] Añadido";
        }
    }
}
```

GraficoPastelDecorador

```
namespace Examen
{

    public class GraficoBarrasDecorador : ReporteDecorador
    {

        public GraficoBarrasDecorador(IReporte reporte) : base(reporte)
        {
        }
        8 referencias
        public override string Generar()
        {
            return base.Generar() + "\n + [Gráfico de Barras] Añadido";
        }
    }
}
```

IReporte

```
namespace Examen
{
    8 referencias
    public interface IReporte
    {
        11 referencias
        string Generar();
    }
}
```

ReporteBase

```
namespace Examen
{
    1 referencia
    public class ReporteBase : IReporte
    {

        4 referencias
        public string Generar()
        {
            return "---REPORTE BASE ---";
        }
    }
}
```

ReporteDecorador

```
namespace Examen
{
    7 referencias
    public abstract class ReporteDecorador : IReporte
    {
        protected IReporte _reporteEnvuelto;
        3 referencias
        public ReporteDecorador(IReporte reporte)
        {
            _reporteEnvuelto = reporte;
        }
        10 referencias
        public virtual string Generar()
        {
            return _reporteEnvuelto.Generar();
        }
    }
}
```

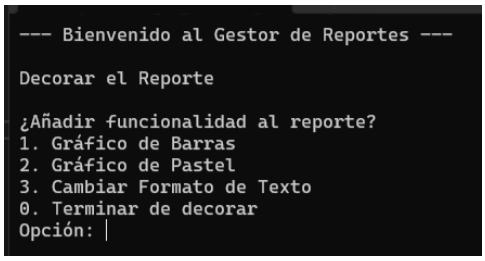
Program

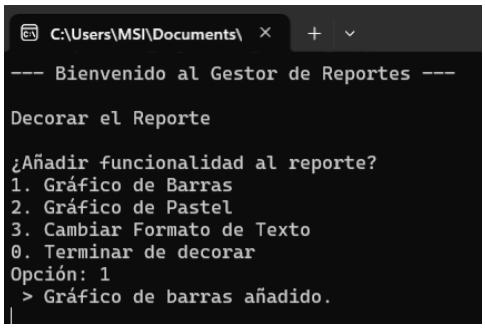
```
namespace Examen
{
    0 referencias
    internal class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            Console.WriteLine("--- Bienvenido al Gestor de Reportes ---");

            Console.WriteLine("\nDecorar el Reporte");

            IReporte miReporte = new ReporteBase();

            bool seguirDecorando = true;
            while (seguirDecorando)
            {
                Console.WriteLine("\n¿Añadir funcionalidad al reporte?");
                Console.WriteLine("1. Gráfico de Barras");
                Console.WriteLine("2. Gráfico de Pastel");
                Console.WriteLine("3. Cambiar Formato de Texto");
                Console.WriteLine("0. Terminar de decorar");
                Console.Write("Opción: ");
                string opcion = Console.ReadLine();

                switch (opcion)
                {
                    case "1":
                        miReporte = new GraficoBarrasDecorador(miReporte);
                        Console.WriteLine(" > Gráfico de barras añadido.");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                    case "2":
                        miReporte = new GraficoPastelDecorador(miReporte);
                        Console.WriteLine(" > Gráfico de pastel añadido.");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                    case "3":
                        Console.Write("  Ingrese nombre de la fuente (ej. Arial): ");
                        string fuente = Console.ReadLine();
                        Console.Write("  Ingrese color del texto (ej. Azul): ");
                        string color = Console.ReadLine();
```

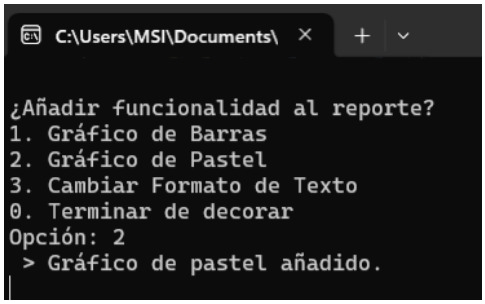
```
                        miReporte = new FormatoTextoDecorador(miReporte, fuente, color);
                        Console.WriteLine(" > Formato de texto aplicado.");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                    case "0":
                        seguirDecorando = false;
                        Console.ReadKey();
                        Console.Clear();
                        break;
                    default:
                        Console.WriteLine("Opción no válida.");
                        Console.ReadKey();
                        Console.Clear();
                        break;
                }
            }

            Console.WriteLine("\n--- Vista Previa del Reporte Final ---");
            Console.WriteLine(miReporte.Generar());
            Console.ReadKey();
            Console.Clear();


            Console.WriteLine("\nExportar el Reporte (Jerarquía)");
            Console.WriteLine("Construyendo árbol de formatos de exportación.");

            var raizExportar = new CategoriaExportar("Todos los Formatos");

            var categoriaDocumento = new CategoriaExportar("Documento");
            var categoriaImagen = new CategoriaExportar("Imagen");

            raizExportar.Agregar(categoriaDocumento);
            raizExportar.Agregar(categoriaImagen);


            categoriaDocumento.Agregar(new FormatoExportar("PDF"));
            categoriaDocumento.Agregar(new FormatoExportar("Excel"));
            categoriaDocumento.Agregar(new FormatoExportar("Word"));
            categoriaDocumento.Agregar(new FormatoExportar("PowerPoint"));
```

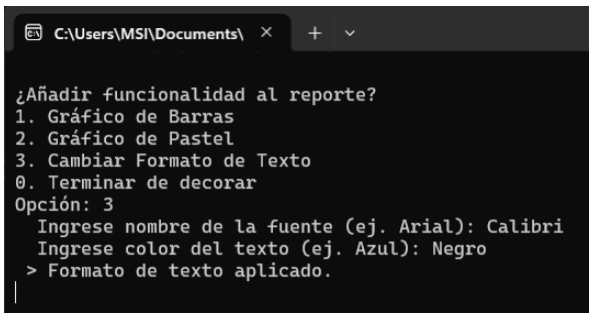
```
90
91              categoriaImagen.Agregar(new FormatoExportar("JPG"));
92              categoriaImagen.Agregar(new FormatoExportar("PNG"));
93
94
95              Console.WriteLine("\n--- Jerarquía de Exportación Disponible ---");
96              raizExportar.MostrarJerarquia(0);
97
98              Console.WriteLine("\nSeleccionar Formato de Exportación");
99              Console.Write("Escriba el formato en el que desea exportar (ej. PDF, JPG, Word): ");
100             string formatoElegido = Console.ReadLine();
101             Console.ReadKey();
102             Console.Clear();
103
104             if (!string.IsNullOrWhiteSpace(formatoElegido))
105             {
106                 Console.WriteLine($"\n--- EXPORTANDO REPORTE A {formatoElegido.ToUpper()} ---");
107                 Console.WriteLine(miReporte.Generar());
108                 Console.WriteLine($"--- REPORTE GUARDADO COMO report.{formatoElegido.ToLower()} ---");
109             }
110             else
111             {
112                 Console.WriteLine("No se seleccionó ningún formato. Exportación cancelada.");
113             }
114
115             Console.WriteLine("\nPresione Enter para salir.");
116             Console.ReadLine();
117         }
118     }
119 }
120
```

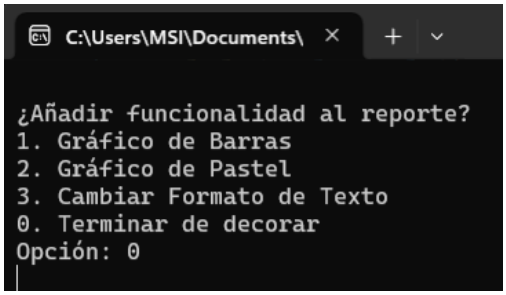
Código ejecutado

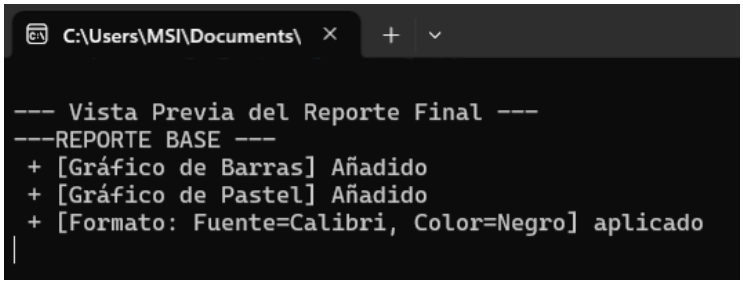


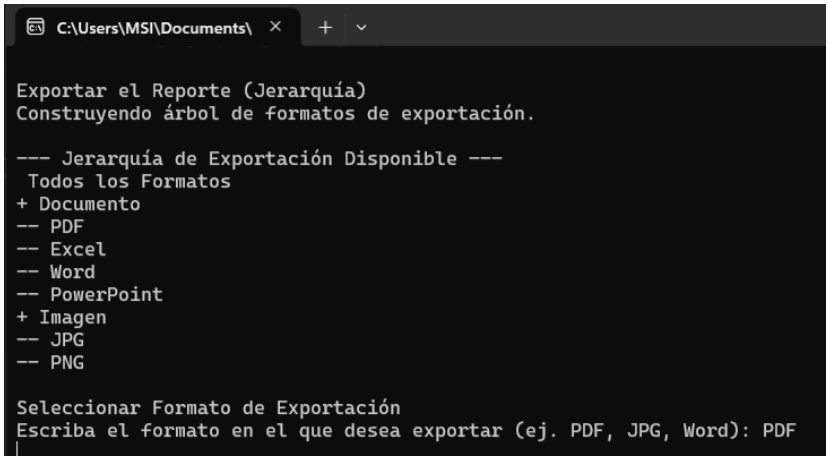


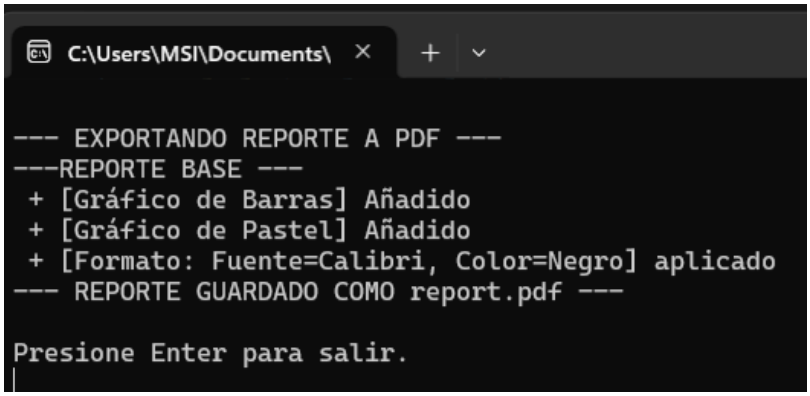







C:\Users\MSI\Documents\
--- Vista Previa del Reporte Final ---
---REPORTE BASE ---
+ [Gráfico de Barras] Añadido
+ [Gráfico de Pastel] Añadido
+ [Formato: Fuente=Calibri, Color=Negro] aplicado


C:\Users\MSI\Documents\
Exportar el Reporte (Jerarquía)
Construyendo árbol de formatos de exportación.
--- Jerarquía de Exportación Disponible ---
Todos los Formatos
+ Documento
-- PDF
-- Excel
-- Word
-- PowerPoint
+ Imagen
-- JPG
-- PNG
Seleccionar Formato de Exportación
Escriba el formato en el que desea exportar (ej. PDF, JPG, Word): PDF


C:\Users\MSI\Documents\
--- EXPORTANDO REPORTE A PDF ---
---REPORTE BASE ---
+ [Gráfico de Barras] Añadido
+ [Gráfico de Pastel] Añadido
+ [Formato: Fuente=Calibri, Color=Negro] aplicado
--- REPORTE GUARDADO COMO report.pdf ---
Presione Enter para salir.

Conclusión

El desarrollo del gestor de reportes nos enseño que el uso de estos dos patrones nos ayuda a crear un software flexible y escalable. Gracias al patrón Decorador, logramos personalizar los reportes añadiendo gráficos y formatos sin la necesidad de generar clases innecesarias, mientras que con el patrón Composite, simplificamos la exportación al tratar tanto a los archivos individuales como a las carpetas de manera uniforme.