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

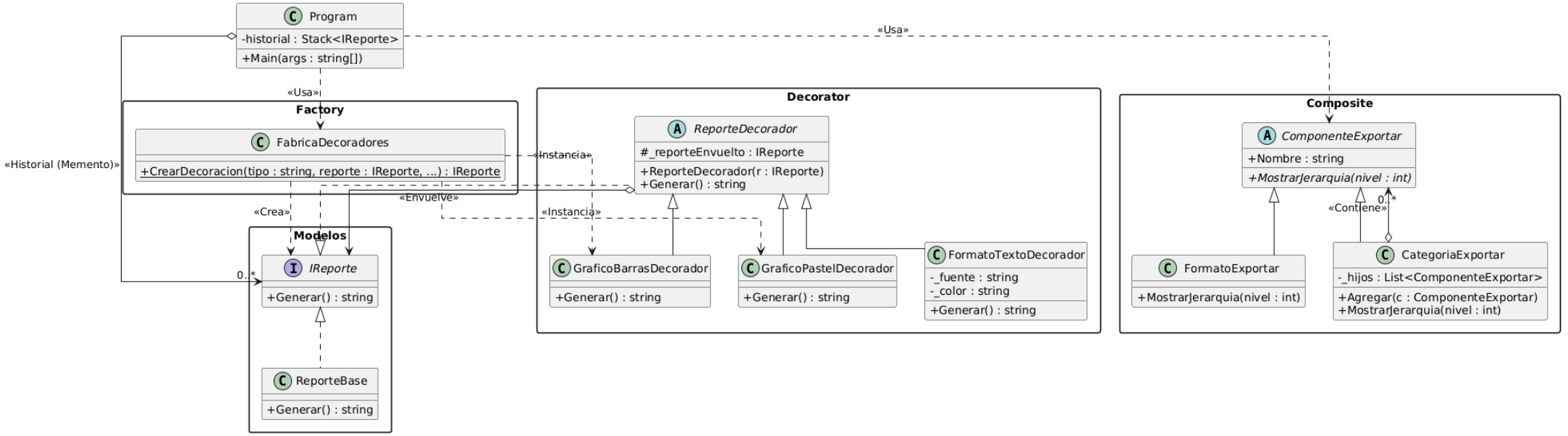
TÍTULO ACTIVIDAD:
Examen U4 – U5

UNIDAD A EVALUAR:
Unidad 4 - 5

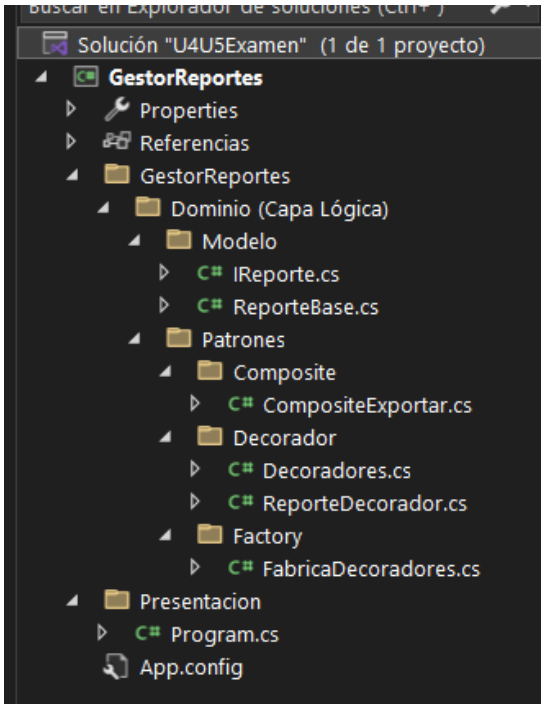
NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:
Angulo Berrelleza Fernando 21212322

NOMBRE DEL MAESTRO (A):
Maribel Guerrero Luis

Diagrama


Program
-historial : Stack<IReporte>
+Main(args : string[])
«Usa»
«Usa»
Factory
FabricaDecoradores
+CrearDecoracion(tipo : string, reporte : IReporte, ...) : IReporte
«Historial (Memento)»
«Instancia»
«Envuelve»
«Crea»
Modelos
IReporte
+Generar() : string
0..*
ReporteBase
+Generar() : string
Decorator
ReporteDecorador
#_reporteEnvuelto : IReporte
+ReporteDecorador(r : IReporte)
+Generar() : string
«Instancia»
GraficoBarrasDecorador
+Generar() : string
GraficoPastelDecorador
+Generar() : string
FormatoTextoDecorador
-_fuente : string
-_color : string
+Generar() : string
Composite
ComponenteExportar
+Nombre : string
+MostrarJerarquia(nivel : int)
«Contiene»
0..*
FormatoExportar
+MostrarJerarquia(nivel : int)
CategoriaExportar
-_hijos : List<ComponenteExportar>
+Agregar(c : ComponenteExportar)
+MostrarJerarquia(nivel : int)

Explorador de soluciones



Código: IReporte

```
namespace GestorReportes.GestorReportes.Dominio__Capa_Lógica_.Modelo
{
    12 referencias
    public interface IReporte
    {
        11 referencias
        string Generar();
    }
}
```

ReporteBase

```
namespace GestorReportes.GestorReportes.Dominio__Capa_Lógica_.Modelo
{
    1 referencia
    public class ReporteBase : IReporte
    {
        4 referencias
        public string Generar()
        {
            return "--- REPORTE BASE ---";
        }
    }
}
```

Patrón compuesto: CompositeExportar

```
namespace GestorReportes.GestorReportes.Dominio__Capa_Lógica_.Patrones.Composite
{
    8 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);
    }

    6 referencias
    public class FormatoExportar : ComponenteExportar
    {
        5 referencias
        public FormatoExportar(string nombre) : base(nombre) { }

        2 referencias
        public override void MostrarJerarquia(int nivel)
        {
            Console.WriteLine(new string('-', nivel) + " " + Nombre);
        }
    }

    4 referencias
    public class CategoriaExportar : ComponenteExportar
    {
        private List<ComponenteExportar> _hijos = new List<ComponenteExportar>();

        3 referencias
        public CategoriaExportar(string nombre) : base(nombre) { }

        7 referencias
        public void Agregar(ComponenteExportar componente)
        {
            _hijos.Add(componente);
        }

        3 referencias
        public override void MostrarJerarquia(int nivel)
        {
            Console.WriteLine(new string('+', nivel) + " " + Nombre);
            foreach (var hijo in _hijos)
            {
                hijo.MostrarJerarquia(nivel + 2);
            }
        }
    }
}
```

Patrón Decorador: Decoradores

```
namespace GestorReportes.GestorReportes.Dominio__Capa_Lógica_.Patrones.Decorador
{
    2 referencias
    public class GraficoBarrasDecorador : ReporteDecorador
    {
        1 referencia
        public GraficoBarrasDecorador(IReporte reporte) : base(reporte) { }

        8 referencias
        public override string Generar()
        {
            return base.Generar() + "\n + [Gráfico de Barras] añadido";
        }
    }

    2 referencias
    public class GraficoPastelDecorador : ReporteDecorador
    {
        1 referencia
        public GraficoPastelDecorador(IReporte reporte) : base(reporte) { }

        8 referencias
        public override string Generar()
        {
            return base.Generar() + "\n + [Gráfico de Pastel] añadido";
        }
    }

    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";
        }
    }
}
```

ReporteDecorador

```
namespace GestorReportes.GestorReportes.Dominio__Capa_Lógica_.Patrones.Decorador
{
    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();
        }
    }
}
```

Patrón Factory Method: FabricaDecoradores

```
namespace GestorReportes.GestorReportes.Dominio__Capa_Lógica_.Patrones.Factory
{
    3 referencias
    public static class FabricaDecoradores
    {
        3 referencias
        public static IReporte CrearDecoracion(string tipo, IReporte reporteActual, string arg1 = "", string arg2 = "")
        {
            switch (tipo.ToLower())
            {
                case "barras":
                    return new GraficoBarrasDecorador(reporteActual);
                case "pastel":
                    return new GraficoPastelDecorador(reporteActual);
                case "texto":
                    return new FormatoTextoDecorador(reporteActual, arg1, arg2);
                default:
                    Console.WriteLine("Tipo de decoración no encontrado.");
                    return reporteActual;
            }
        }
    }
}
```

Program

```
14      static void Main(string[] args)
15      {
16          Stack<IReporte> historial = new Stack<IReporte>();
17
18          IReporte miReporte = new ReporteBase();
19
20          bool editando = true;
21
22          while (editando)
23          {
24              Console.ForegroundColor = ConsoleColor.Cyan;
25              Console.WriteLine("\n VISTRA PREVIA ACTUAL");
26              Console.WriteLine(miReporte.Generar());
27              Console.ResetColor();
28
29              Console.WriteLine("\nOpciones de Edición:");
30              Console.WriteLine("1. Agregar Gráfico de Barras");
31              Console.WriteLine("2. Agregar Gráfico de Pastel");
32              Console.WriteLine("3. Aplicar Formato de Texto");
33              Console.WriteLine("4. DESHACER último cambio (Memento)");
34              Console.WriteLine("0. Terminar edición y Exportar");
35              Console.Write("Seleccione: ");
36              string op = Console.ReadLine();
37
38              if (op == "1" || op=="2" || op== "3")
39              {
40                  historial.Push(miReporte);
41              }
42
```

```
43              switch (op)
44              {
45                  case "1":
46                      Console.Clear();
47                      miReporte = FabricaDecoradores.CrearDecoracion("barras", miReporte);
48                      break;
49
50                  case "2":
51                      Console.Clear();
52                      miReporte = FabricaDecoradores.CrearDecoracion("pastel", miReporte);
53                      break;
54
55                  case "3":
56                      Console.Clear();
57                      Console.Write("Fuente: "); string f = Console.ReadLine();
58                      Console.Write("Color: "); string c = Console.ReadLine();
59                      miReporte = FabricaDecoradores.CrearDecoracion("texto", miReporte, f, c);
60                      break;
61
62                  case "4":
63                      Console.Clear();
64                      if (historial.Count > 0)
65                      {
66                          miReporte = historial.Pop();
67                          Console.WriteLine("-> Cambio deshecho exitosamente.");
68                      }
69                      else
70                      {
71                          Console.WriteLine("-> No hay acciones para deshacer.");
72                      }
73                      break;
74
75                  case "0":
76                      Console.Clear();
77                      editando = false;
78                      break;
79
80                  default:
81                      Console.Clear();
82                      Console.WriteLine("Opción no válida");
83                      break;
84              }
85          }
86
```

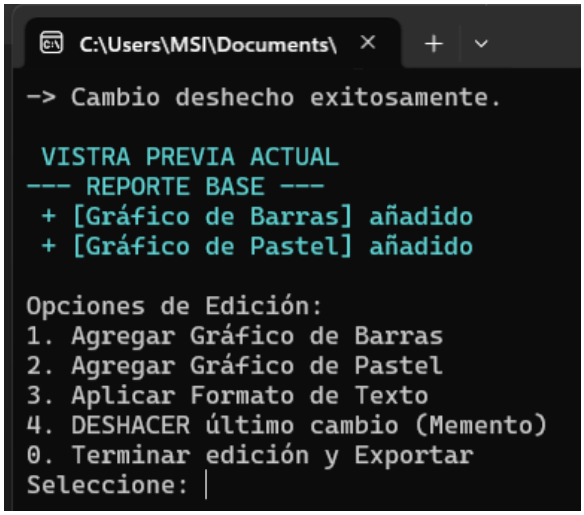
```
87                  Console.WriteLine("\n\n\tEXPORTACIÓN DEL REPORTE");
88                  var raiz = new CategoriaExportar("Formatos");
89                  var docs = new CategoriaExportar("Documentos");
90                  var imgs = new CategoriaExportar("Imágenes");
91
92                  raiz.Agregar(docs);
93                  raiz.Agregar(imgs);
94
95                  docs.Agregar(new FormatoExportar("PDF"));
96                  docs.Agregar(new FormatoExportar("Word"));
97                  docs.Agregar(new FormatoExportar("Excel"));
98
99                  imgs.Agregar(new FormatoExportar("JPG"));
100                 imgs.Agregar(new FormatoExportar("PNG"));
101
102                 raiz.MostrarJerarquia(0);
103
104                 Console.Write("Escriba el formato deseado (ej. PDF, JPG): ");
105                 string formatoElegido = Console.ReadLine();
106
107                 if (!string.IsNullOrWhiteSpace(formatoElegido))
108                 {
109                     Console.ForegroundColor = ConsoleColor.Green;
110                     Console.WriteLine($"\n--- EXPORTANDO REPORTE A {formatoElegido.ToUpper()} ---");
111                     Console.WriteLine(miReporte.Generar());
112                     Console.WriteLine($"--- ARCHIVO GUARDADO COMO: reporte_final.{formatoElegido.ToLower()} ---");
113                     Console.ResetColor();
114                 }
115                 else
116                 {
117                     Console.WriteLine(">> Exportación cancelada por el usuario.");
118                 }
119
120                 Console.WriteLine("\nPresione Enter para salir");
121                 Console.ReadKey();
122             }
123         }
124     }
125
```

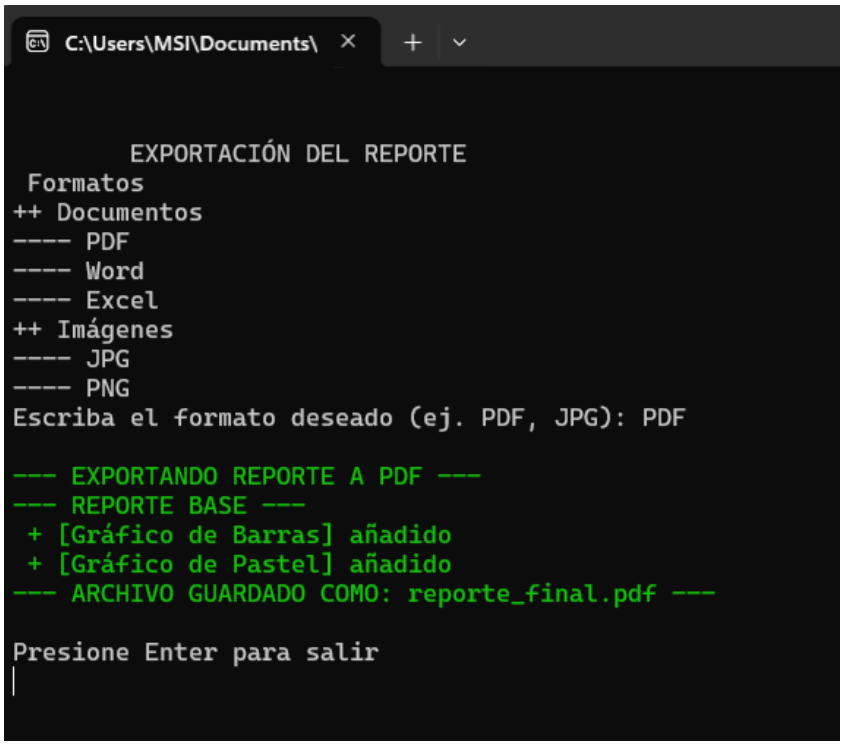
Ejecución

```
 VISTRA PREVIA ACTUAL
--- REPORTE BASE ---

Opciones de Edición:
1. Agregar Gráfico de Barras
2. Agregar Gráfico de Pastel
3. Aplicar Formato de Texto
4. DESHACER último cambio (Memento)
0. Terminar edición y Exportar
Seleccione:
```

```
C:\Users\MSI\Documents\
Fuente: Arial
Color: Negro

 VISTRA PREVIA ACTUAL
--- REPORTE BASE ---
 + [Gráfico de Barras] añadido
 + [Gráfico de Pastel] añadido
 + [Formato: Fuente=Arial, Color=Negro] aplicado

Opciones de Edición:
1. Agregar Gráfico de Barras
2. Agregar Gráfico de Pastel
3. Aplicar Formato de Texto
4. DESHACER último cambio (Memento)
0. Terminar edición y Exportar
Seleccione:
```

```
C:\Users\MSI\Documents\
Fuente: Arial
Color: Negro

 VISTRA PREVIA ACTUAL
--- REPORTE BASE ---
 + [Gráfico de Barras] añadido
 + [Gráfico de Pastel] añadido
 + [Formato: Fuente=Arial, Color=Negro] aplicado

Opciones de Edición:
1. Agregar Gráfico de Barras
2. Agregar Gráfico de Pastel
3. Aplicar Formato de Texto
4. DESHACER último cambio (Memento)
0. Terminar edición y Exportar
Seleccione: 4
```


C:\Users\MSI\Documents\
-> Cambio deshecho exitosamente.
VISTRA PREVIA ACTUAL
--- REPORTE BASE ---
+ [Gráfico de Barras] añadido
+ [Gráfico de Pastel] añadido
Opciones de Edición:
1. Agregar Gráfico de Barras
2. Agregar Gráfico de Pastel
3. Aplicar Formato de Texto
4. DESHACER último cambio (Memento)
0. Terminar edición y Exportar
Seleccione:


C:\Users\MSI\Documents\
EXPORTACIÓN DEL REPORTE
Formatos
++ Documentos
---- PDF
---- Word
---- Excel
++ Imágenes
---- JPG
---- PNG
Escriba el formato deseado (ej. PDF, JPG): PDF
--- EXPORTANDO REPORTE A PDF ---
--- REPORTE BASE ---
+ [Gráfico de Barras] añadido
+ [Gráfico de Pastel] añadido
--- ARCHIVO GUARDADO COMO: reporte_final.pdf ---
Presione Enter para salir

Conclusión

La integración de una arquitectura en capas y los patrones Factory Method, Decorator, Composite y Memento han proporcionado al Gestor de Reportes una extraordinaria capacidad de mantenimiento, robustez y flexibilidad. Por otro lado, el Factory Method separa la creación de objetos, posibilitando que se incorporen formatos o tipos gráficos nuevos sin necesidad de cambiar el programa central. La cooperación entre Composite y Decorator hace más simple la personalización dinámica del contenido y el tratamiento uniforme de la jerarquía de exportación. Por último, aplicar Memento aumenta considerablemente la experiencia del usuario al permitir que se anulen los cambios.