



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



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 4 y 5

UNIDAD A EVALUAR:

4 y 5

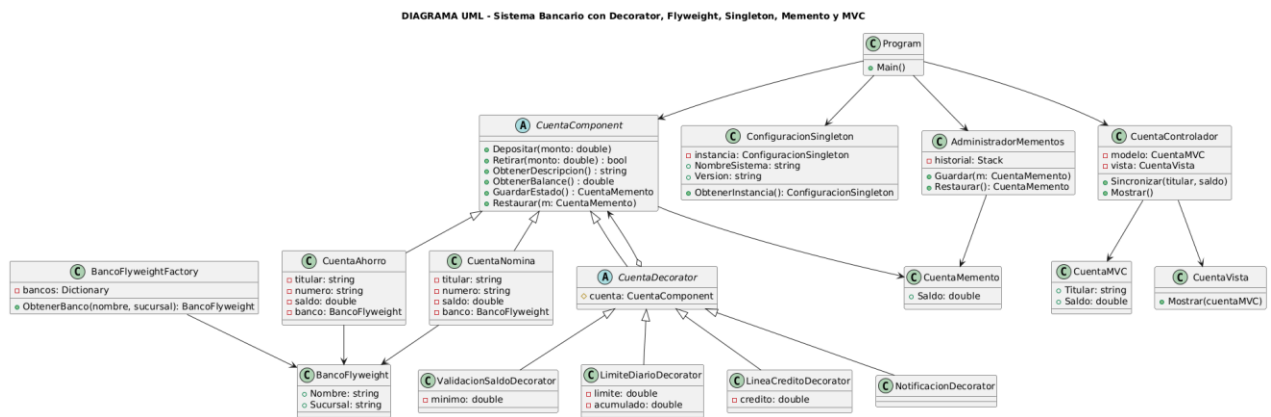
NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

López Hernandez Víctor Manuel 21212350

NOMBRE DEL MAESTRO (A):

Maribel Guerrero Luis

Diagrama UML



Código

```

AdministradorMementos.cs
namespace ExaDecorador
{
    2 referencias
    public class AdministradorMementos
    {
        private Stack<CuentaMemento> historial = new Stack<CuentaMemento>();

        2 referencias
        public void Guardar(CuentaMemento m)
        {
            historial.Push(m);
        }

        1 referencia
        public CuentaMemento Restaurar()
        {
            if (historial.Count > 0)
                return historial.Pop();

            return null;
        }
    }
}

```

```

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

namespace ExaDecorador
{
    7 referencias
    public class BancoFlyweight
    {
        3 referencias
        public string Nombre { get; private set; }
        3 referencias
        public string Sucursal { get; private set; }

        1 referencia
        public BancoFlyweight(string nombre, string sucursal)
        {
            Nombre = nombre;
            Sucursal = sucursal;
        }
    }
}

```

```
BancoFlyweightFactory.cs  [X]
ExaDecorador
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ExaDecorador
8  {
9      2 referencias
10     public class BancoFlyweightFactory
11     {
12         private static Dictionary<string, BancoFlyweight> bancos = new Dictionary<string, BancoFlyweight>();
13
14         2 referencias
15         public static BancoFlyweight ObtenerBanco(string nombre, string sucursal)
16         {
17             string clave = nombre + sucursal;
18
19             if (!bancos.ContainsKey(clave))
20             {
21                 bancos[clave] = new BancoFlyweight(nombre, sucursal);
22             }
23
24             return bancos[clave];
25         }
26     }
27 }
```

```
ConfiguracionSingleton.cs  [X]
ExaDecorador
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ExaDecorador
8  {
9      6 referencias
10     public class ConfiguracionSingleton
11     {
12         private static ConfiguracionSingleton instancia = null;
13
14         2 referencias
15         public string NombreSistema { get; private set; }
16
17         2 referencias
18         public string Version { get; private set; }
19
20         1 referencia
21         private ConfiguracionSingleton()
22         {
23             NombreSistema = "Sistema Bancario Tec";
24             Version = "2.0";
25         }
26
27         1 referencia
28         public static ConfiguracionSingleton ObtenerInstancia()
29         {
30             if (instancia == null)
31                 instancia = new ConfiguracionSingleton();
32
33             return instancia;
34         }
35     }
36 }
```

CuentaAhorro.cs

ExaDecorador

ExaDecorador.CuentaAhorro

```
2 referencias
public class CuentaAhorro : CuentaComponent
{
    private string titular;
    private string numero;
    public double saldo;
    private BancoFlyweight banco;

    1 referencia
    public CuentaAhorro(string titular, string numero, double saldo)
    {
        this.titular = titular;
        this.numero = numero;
        this.saldo = saldo;
        this.banco = BancoFlyweightFactory.ObtenerBanco("TecBank", "Sucursal Centro");
    }

    3 referencias
    public override void Depositar(double monto)
    {
        saldo += monto;
    }

    3 referencias
    public override bool Retirar(double monto)
    {
        if (saldo >= monto)
        {
            saldo -= monto;
            return true;
        }
        Console.WriteLine("Saldo insuficiente.");
        return false;
    }

    3 referencias
    public override string ObtenerDescripcion()
    {
        return "Cuenta de Ahorro\nTitular: " + titular +
            "\nNúmero: " + numero +
            "\nBanco: " + banco.Nombre +
            "\nSucursal: " + banco.Sucursal;
    }

    5 referencias
    public override double ObtenerBalance()
    {
        return saldo;
    }

    // MEMENTO
    4 referencias
    public override CuentaMemento GuardarEstado()
    {
        return new CuentaMemento(this.saldo);
    }

    3 referencias
    public override void Restaurar(CuentaMemento m)
    {
        if (m != null)
            this.saldo = m.Saldo;
    }
}
```

77 %

No se encontraron problemas.

CuentaComponent.cs

C# ExaDecorador

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ExaDecorador
8  {
9      public abstract class CuentaComponent
10     {
11         public abstract void Depositar(double monto);
12         public abstract bool Retirar(double monto);
13         public abstract string ObtenerDescripcion();
14         public abstract double ObtenerBalance();
15
16         // MEMENTO
17         public abstract CuentaMemento GuardarEstado();
18         public abstract void Restaurar(CuentaMemento m);
19     }
20 }
21
```

CuentaControlador.cs

C# ExaDecorador

```
7  namespace ExaDecorador
8  {
9      public class CuentaControlador
10     {
11         private CuentaMVC modelo;
12         private CuentaVista vista;
13
14         public CuentaControlador(CuentaMVC m, CuentaVista v)
15         {
16             modelo = m;
17             vista = v;
18         }
19
20         public void Sincronizar(string titular, double saldo)
21         {
22             modelo.Titular = titular;
23             modelo.Saldo = saldo;
24         }
25
26         public void Mostrar()
27         {
28             vista.Mostrar(modelo);
29         }
30     }
31 }
32
```

CuentaDecorator.cs

ExaDecorador

ExaDecorador.CuentaDecorator

```

7 namespace ExaDecorador
8 {
9     9 referencias
10     public abstract class CuentaDecorator : CuentaComponent
11     {
12         protected CuentaComponent cuenta;
13
14         4 referencias
15         public CuentaDecorator(CuentaComponent cuenta)
16         {
17             this.cuenta = cuenta;
18
19         5 referencias
20         public override void Depositar(double monto) { cuenta.Depositar(monto); }
21         11 referencias
22         public override bool Retirar(double monto) { return cuenta.Retirar(monto); }
23         5 referencias
24         public override string ObtenerDescripcion() { return cuenta.ObtenerDescripcion(); }
25         5 referencias
26         public override double ObtenerBalance() { return cuenta.ObtenerBalance(); }
27
28         4 referencias
29         public override CuentaMemento GuardarEstado() { return cuenta.GuardarEstado(); }
30         3 referencias
31         public override void Restaurar(CuentaMemento m) { cuenta.Restaurar(m); }
32     }
33 }

```

CuentaMemento.cs

ExaDecorador

Exa

```

7 namespace ExaDecorador
8 {
9     15 referencias
10     public class CuentaMemento
11     {
12         3 referencias
13         public double Saldo { get; private set; }
14
15         2 referencias
16         public CuentaMemento(double saldo)
17         {
18             Saldo = saldo;
19         }
20     }
21 }

```

CuentaMVC.cs

ExaDecorador

```

7 namespace ExaDecorador
8 {
9     5 referencias
10     public class CuentaMVC
11     {
12         2 referencias
13         public string Titular { get; set; }
14         2 referencias
15         public double Saldo { get; set; }
16     }
17 }

```

CuentaNomina.cs

ExaDecorador

ExaDecorador.CuentaNomina

```
2 referencias
public class CuentaNomina : CuentaComponent
{
    private string titular;
    private string numero;
    public double saldo;
    private BancoFlyweight banco;

    1 referencia
    public CuentaNomina(string titular, string numero, double saldo)
    {
        this.titular = titular;
        this.numero = numero;
        this.saldo = saldo;
        this.banco = BancoFlyweightFactory.ObtenerBanco("TecBank", "Sucursal Centro");
    }

    3 referencias
    public override void Depositar(double monto)
    {
        saldo += monto;
    }

    3 referencias
    public override bool Retirar(double monto)
    {
        if (saldo >= monto)
        {
            saldo -= monto;
            return true;
        }
        Console.WriteLine("Saldo insuficiente.");
        return false;
    }

    3 referencias
    public override string ObtenerDescripcion()
    {
        return "Cuenta de Nómina\nTitular: " + titular +
            "\nNúmero: " + numero +
            "\nBanco: " + banco.Nombre +
            "\nSucursal: " + banco.Sucursal;
    }

    5 referencias
    public override double ObtenerBalance()
    {
        return saldo;
    }

    51
    52
    53
    // MEMENTO
    4 referencias
    public override CuentaMemento GuardarEstado()
    {
        return new CuentaMemento(this.saldo);
    }

    3 referencias
    public override void Restaurar(CuentaMemento m)
    {
        if (m != null)
            this.saldo = m.Saldo;
    }
}
```

77 %

No se encontraron problemas.

CuentaVista.cs

ExaDecorador

ExaDecorador.C

```
7 namespace ExaDecorador
8 {
9     4 referencias
10     public class CuentaVista
11     {
12         1 referencia
13         public void Mostrar(CuentaMVC c)
14         {
15             Console.WriteLine("\n===== INFORMACIÓN (MVC) =====");
16             Console.WriteLine("Titular: " + c.Titular);
17             Console.WriteLine("Saldo: $" + c.Saldo);
18         }
19     }
```

LimiteDiarioDecorator.cs

ExaDecorador

ExaDecorador.LimiteDiarioDecorator

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ExaDecorador
8 {
9     2 referencias
10     public class LimiteDiarioDecorator : CuentaDecorator
11     {
12         private double limite;
13         private double acumulado;
14
15         1 referencia
16         public LimiteDiarioDecorator(CuentaComponent cuenta, double limite) : base(cuenta)
17         {
18             this.limite = limite;
19             acumulado = 0;
20
21         8 referencias
22         public override bool Retirar(double monto)
23         {
24             if (acumulado + monto > limite)
25             {
26                 Console.WriteLine("Límite diario excedido.");
27                 return false;
28             }
29
30             bool ok = base.Retirar(monto);
31             if (ok) acumulado += monto;
32             return ok;
33         }
34     }
```

```
LineaCreditoDecorator.cs
C# ExaDecorador ExaDecorador.LineaCreditoDecorator

2 referencias
public class LineaCreditoDecorator : CuentaDecorator
{
    private double credito;

    1 referencia
    public LineaCreditoDecorator(CuentaComponent cuenta, double credito)
        : base(cuenta)
    {
        this.credito = credito;
    }

    8 referencias
    public override bool Retirar(double monto)
    {
        Console.WriteLine("\n¿Retirar de cuenta (1) o de línea de crédito (2)?");
        string op = Console.ReadLine();

        if (op == "1")
            return base.Retirar(monto);

        if (op == "2")
        {
            if (monto <= credito)
            {
                credito -= monto;
                Console.WriteLine("Retiro de crédito exitoso.");
                return true;
            }
            Console.WriteLine("Crédito insuficiente.");
            return false;
        }

        Console.WriteLine("Opción no válida.");
        return false;
    }

    5 referencias
    public override string ObtenerDescripcion()
    {
        return base.ObtenerDescripcion() + "\nLínea de crédito: $" + credito;
    }
}
```

```
NotificacionDecorator.cs
C# ExaDecorador ExaDecorador.NotificacionDecorator

namespace ExaDecorador
{
    2 referencias
    public class NotificacionDecorator : CuentaDecorator
    {
        1 referencia
        public NotificacionDecorator(CuentaComponent cuenta): base(cuenta) { }

        5 referencias
        public override void Depositar(double monto)
        {
            base.Depositar(monto);
            Console.WriteLine("Notificación: Depósito realizado.");
        }

        8 referencias
        public override bool Retirar(double monto)
        {
            bool ok = base.Retirar(monto);
            Console.WriteLine(ok ?
                "Notificación: Retiro exitoso." :
                "Notificación: Retiro denegado.");
            return ok;
        }
    }
}
```

```
7 namespace ExaDecorador
8 {
9     2 referencias
10    public class ValidacionSaldoDecorator : CuentaDecorator
11    {
12        private double minimo;
13
14        1 referencia
15        public ValidacionSaldoDecorator(CuentaComponent cuenta, double minimo)
16            : base(cuenta)
17        {
18            this.minimo = minimo;
19
20        8 referencias
21        public override bool Retirar(double monto)
22        {
23            if (cuenta.ObtenerBalance() - monto < minimo)
24            {
25                Console.WriteLine("No puede dejar menos de $" + minimo + ".");
26                return false;
27            }
28            return base.Retirar(monto);
29        }
30    }
```

Program.cs

C# ExaDecorador

ExaDecorador.Program

0 referencias

class Program

{

0 referencias

static void Main(string[] args)

{

// SINGLETON

ConfiguracionSingleton cfg = ConfiguracionSingleton.ObtenerInstancia();

Console.WriteLine("=== " + cfg.NombreSistema + " v" + cfg.Version + " ===");

Console.Write("Nombre del titular: ");

string nombre = Console.ReadLine();

Console.Write("Tipo de cuenta (1=Ahorro, 2=Nómina): ");

string tipo = Console.ReadLine();

Console.Write("Saldo inicial: \$");

double saldo = Convert.ToDouble(Console.ReadLine());

CuentaComponent cuenta;

if (tipo == "1")

cuenta = new CuentaAhorro(nombre, "A-001", saldo);

else

cuenta = new CuentaNomina(nombre, "N-001", saldo);

Console.WriteLine("\n¿Activar validación? (S/N): ");

if (Console.ReadLine().ToUpper() == "S")

cuenta = new ValidacionSaldoDecorator(cuenta, 100);

Console.WriteLine("¿Activar límite diario? (S/N): ");

if (Console.ReadLine().ToUpper() == "S")

{

Console.Write("Límite: \$");

double lim = Convert.ToDouble(Console.ReadLine());

cuenta = new LimiteDiarioDecorator(cuenta, lim);

}

Console.WriteLine("¿Activar línea de crédito? (S/N): ");

if (Console.ReadLine().ToUpper() == "S")

{

Console.Write("Crédito: \$");

double cred = Convert.ToDouble(Console.ReadLine());

cuenta = new LineaCreditoDecorator(cuenta, cred);

}

Console.WriteLine("¿Activar notificaciones? (S/N): ");

if (Console.ReadLine().ToUpper() == "S")

cuenta = new NotificacionDecorator(cuenta);

}

77 %

No se encontraron problemas.

```
Program.cs  [C#] ExaDecorador  ExaDecorador.Program
54      cuenta = new NotificacionDecorator(cuenta);
55
56      AdministradorMementos admin = new AdministradorMementos();
57
58      bool salir = false;
59      while (!salir)
60      {
61          Console.WriteLine("\n===== MENÚ =====");
62          Console.WriteLine("1. Depositar");
63          Console.WriteLine("2. Retirar");
64          Console.WriteLine("3. Ver información");
65          Console.WriteLine("4. Deshacer operación (Memento)");
66          Console.WriteLine("5. Ver información con MVC");
67          Console.WriteLine("6. Salir");
68
69          Console.Write("Opción: ");
70          string op = Console.ReadLine();
71
72          switch (op)
73          {
74              case "1":
75                  Console.Write("Monto: $");
76                  admin.Guardar(cuenta.GuardarEstado());
77                  cuenta.Depositar(Convert.ToDouble(Console.ReadLine()));
78                  break;
79
80              case "2":
81                  Console.Write("Monto: $");
82                  admin.Guardar(cuenta.GuardarEstado());
83                  cuenta.Retirar(Convert.ToDouble(Console.ReadLine()));
84                  break;
85
86              case "3":
87                  Console.WriteLine("\n" + cuenta.ObtenerDescripcion());
88                  Console.WriteLine("Saldo actual: $" + cuenta.ObtenerBalance());
89                  break;
90
91              case "4":
92                  cuenta.Restaurar(admin.Restaurar());
93                  Console.WriteLine("Operación deshecha.");
94                  break;
95
96              case "5":
97                  CuentaMVC modelo = new CuentaMVC();
98                  CuentaVista vista = new CuentaVista();
99                  CuentaControlador controlador = new CuentaControlador(modelo, vista);
100
101                  controlador.Sincronizar(nombre, cuenta.ObtenerBalance());
102
103                  controlador.Sincronizar(nombre, cuenta.ObtenerBalance());
104                  controlador.Mostrar();
105                  break;
106
107              case "6":
108                  salir = true;
109                  break;
110          }
111      }
112  }
113  }
```

Consola

```
C:\Users\Manny\source\repos x + v
=== Sistema Bancario Tec v2.0 ===
Nombre del titular:
```

```
C:\Users\Manny\source\repos x + v
=== Sistema Bancario Tec v2.0 ===
Nombre del titular: Victor Lopez
Tipo de cuenta (1=Ahorro, 2=Nómina):
```

```
C:\Users\Manny\source\repos x + v
=== Sistema Bancario Tec v2.0 ===
Nombre del titular: Victor Lopez
Tipo de cuenta (1=Ahorro, 2=Nómina): 1
Saldo inicial: $1500
¿Activar validación? (S/N):
```

```
C:\Users\Manny\source\repos x + v
=== Sistema Bancario Tec v2.0 ===
Nombre del titular: Victor Lopez
Tipo de cuenta (1=Ahorro, 2=Nómina): 1
Saldo inicial: $1500
¿Activar validación? (S/N): s
¿Activar límite diario? (S/N): s
Límite: $2000
¿Activar línea de crédito? (S/N): s
Crédito: $10000
¿Activar notificaciones? (S/N): s

===== MENÚ =====
1. Depositar
2. Retirar
3. Ver información
4. Deshacer operación (Memento)
5. Ver información con MVC
6. Salir
Opción: |
```



C:\Users\Manny\source\repos



=== Sistema Bancario Tec v2.0 ===

Nombre del titular: Victor Lopez

Tipo de cuenta (1=Ahorro, 2=Nómina): 1

Saldo inicial: \$1500

¿Activar validación? (S/N): s

¿Activar límite diario? (S/N): s

Límite: \$2000

¿Activar línea de crédito? (S/N): s

Crédito: \$10000

¿Activar notificaciones? (S/N): s

===== MENÚ =====

1. Depositar

2. Retirar

3. Ver información

4. Deshacer operación (Memento)

5. Ver información con MVC

6. Salir

Opción: 1

Monto: \$500

Notificación: Depósito realizado.

===== MENÚ =====

1. Depositar

2. Retirar

3. Ver información

4. Deshacer operación (Memento)

5. Ver información con MVC

6. Salir

Opción: 2

Monto: \$300

¿Retirar de cuenta (1) o de línea de crédito (2)?

1

Notificación: Retiro exitoso.

===== MENÚ =====

1. Depositar
2. Retirar
3. Ver información
4. Deshacer operación (Memento)
5. Ver información con MVC
6. Salir

Opción: 3

Cuenta de Ahorro

Titular: Victor Lopez

Número: A-001

Banco: TecBank

Sucursal: Sucursal Centro

Línea de crédito: \$10000

Saldo actual: \$1700

===== MENÚ =====

1. Depositar
2. Retirar
3. Ver información
4. Deshacer operación (Memento)
5. Ver información con MVC
6. Salir

Opción: 4

Operación deshecha.

===== MENÚ =====

1. Depositar
2. Retirar
3. Ver información
4. Deshacer operación (Memento)
5. Ver información con MVC
6. Salir

Opción: 3

Cuenta de Ahorro

Titular: Victor Lopez

Número: A-001

Banco: TecBank

Sucursal: Sucursal Centro

Línea de crédito: \$10000

Saldo actual: \$2000

```
===== INFORMACIÓN (MVC) =====  
Titular: Victor Lopez  
Saldo: $2000
```

Conclusión

Implementar todos estos patrones dentro de un mismo proyecto me ayudó a organizar mejor el sistema y hacer que cada parte cumpliera una función específica sin estorbar a las demás. El patrón Decorator permitió que la cuenta bancaria fuera personalizable, activando solo las funciones que el usuario necesita, el patrón Flyweight evitó repetir datos del banco y ayudó a que varias cuentas compartieran la misma información sin gastar memoria, el Singleton sirvió para manejar una configuración única del sistema, mientras que el Memento hizo posible deshacer operaciones y regresar el saldo a un estado anterior. Finalmente, el patrón MVC ayudó a mostrar la información de una forma más limpia y ordenada.