TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto-Diciembre 2025

MATERIA:
Patrones de Diseño de Software

TÍTULO ACTIVIDAD:

Examen unidad 4 y 5

UNIDAD A EVALUAR:
U4 Y U5

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

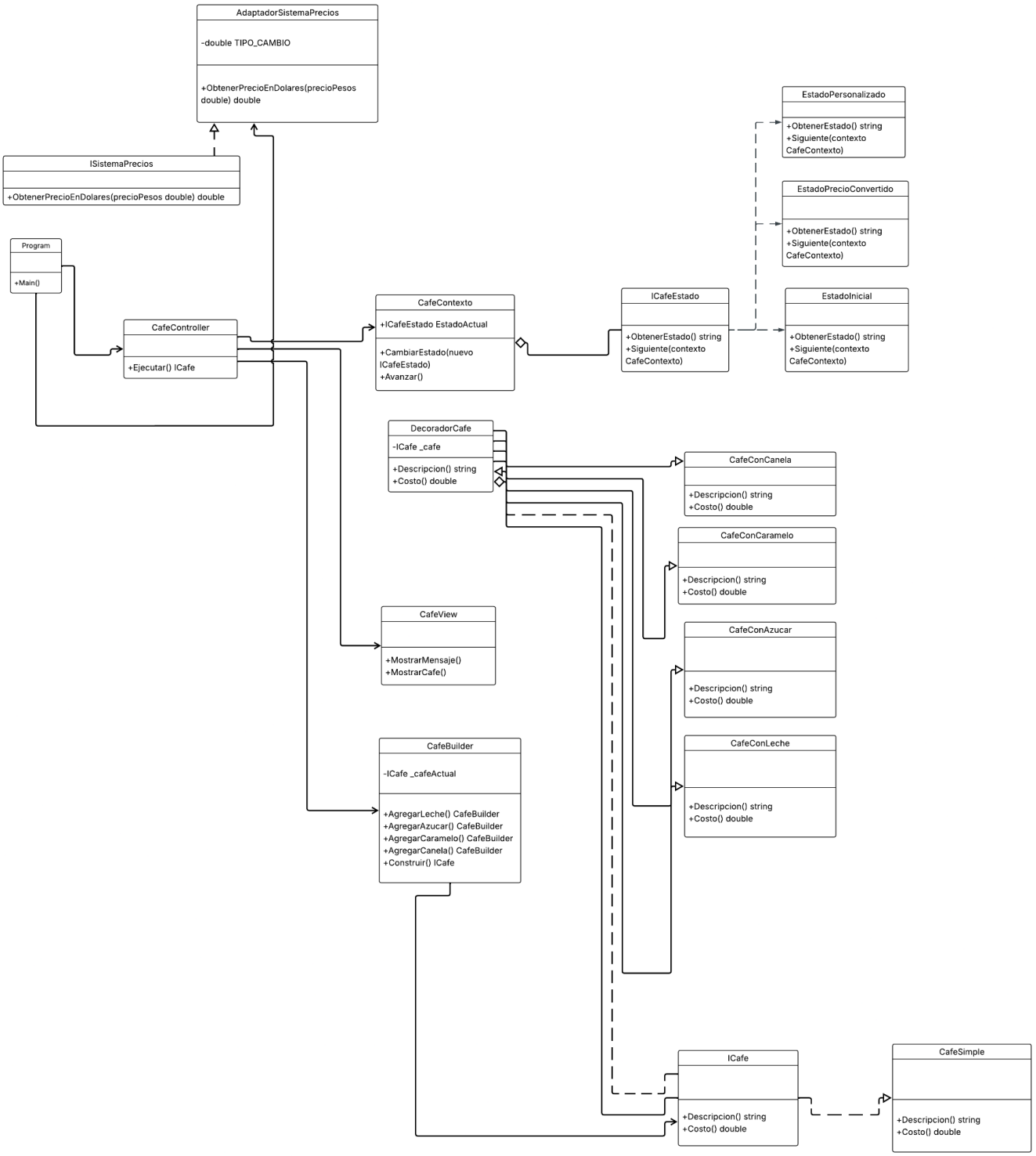
Liczi Citlali Flores Ramirez - 21210537

NOMBRE DEL MAESTRO (A):
MARIBEL GUERRERO LUIS

Introducción

En este proyecto desarrollé un simulador de cafetería donde el usuario puede armar su propio café agregando los ingredientes que quiera. La idea principal fue aplicar diferentes patrones de diseño y la arquitectura MVC de una forma práctica para entender cómo se organizan mejor los programas y cómo se hace el código más claro y fácil de mantener. Separé la lógica del café, la vista y el controlador en carpetas distintas para que se vea la estructura y para trabajar cada parte por separado. Además, cada patrón me ayudó a resolver un problema específico dentro del sistema del café, como agregar ingredientes, cambiar de estados o convertir precios. Con este proyecto aprendí a organizar mejor mis clases y a ver cómo estos patrones realmente sí facilitan el desarrollo.

UML


AdaptadorSistemaPrecios
-double TIPO_CAMBIO
+ObtenerPrecioEnDolares(precioPesos double) double
ISistemaPrecios
+ObtenerPrecioEnDolares(precioPesos double) double
Program
+Main()
CafeController
+Ejecutar() ICafe
CafeContexto
+ICafeEstado EstadoActual
+CambiarEstado(nuevo ICafeEstado)
+Avanzar()
ICafeEstado
+ObtenerEstado() string
+Siguiente(contexto CafeContexto)
EstadoPersonalizado
+ObtenerEstado() string
+Siguiente(contexto CafeContexto)
EstadoPrecioConvertido
+ObtenerEstado() string
+Siguiente(contexto CafeContexto)
EstadoInicial
+ObtenerEstado() string
+Siguiente(contexto CafeContexto)
DecoradorCafe
-ICafe _cafe
+Descripcion() string
+Costo() double
CafeConCanela
+Descripcion() string
+Costo() double
CafeConCaramelo
+Descripcion() string
+Costo() double
CafeConAzucar
+Descripcion() string
+Costo() double
CafeConLeche
+Descripcion() string
+Costo() double
CafeView
+MostrarMensaje()
+MostrarCafe()
CafeBuilder
-ICafe _cafeActual
+AgregarLeche() CafeBuilder
+AgregarAzucar() CafeBuilder
+AgregarCaramelo() CafeBuilder
+AgregarCanela() CafeBuilder
+Construir() ICafe
ICafe
+Descripcion() string
+Costo() double
CafeSimple
+Descripcion() string
+Costo() double

CÓDIGO

interface ICafe

```
namespace CafeteriaExamen
{
    //todo tipo de café (simple o decorado) debe poder decir su descripción y su costo."
    12 referencias
    public interface ICafe
    {
        13 referencias
        string Descripcion();
        14 referencias
        double Costo();
    }

}

```

class CafeSimple : ICafe

```
namespace CafeteriaExamen
{
    //Clase concreta que representa el café base sin ingredientes.
    1 referencia
    public class CafeSimple : ICafe
    {
        8 referencias
        public string Descripcion() => "Café simple"; // descripción base
        9 referencias
        public double Costo() => 20.00;  // precio base en pesos
    }
}

```

public abstract class DecoradorCafe

```
namespace CafeteriaExamen

//Clase abstracta DecoradorCafe - sirve como plantilla para todos los cafés decorados.
{
    9 referencias
    public abstract class DecoradorCafe : ICafe
    {
        protected ICafe _cafe;  // // referencia al objeto que se decora


        //actúa como envoltura para añadir comportamiento sin tocar la clase base.

        4 referencias
        public DecoradorCafe(ICafe cafe)
        {
            _cafe = cafe;
        }

        12 referencias
        public virtual string Descripcion() => _cafe.Descripcion();
        13 referencias
        public virtual double Costo() => _cafe.Costo();
    }
}
```

public class CafeConAzucar : DecoradorCafe

```
namespace CafeteriaExamen
{
    2 referencias
    public class CafeConAzucar : DecoradorCafe
    {
        1 referencia
        public CafeConAzucar(ICafe cafe) : base(cafe) { }

        9 referencias
        public override string Descripcion() => _cafe.Descripcion() + ", Con azúcar";
        10 referencias
        public override double Costo() => _cafe.Costo() + 5.00;
    }
}
```

public class CafeConCanela : DecoradorCafe

```
namespace CafeteriaExamen
{
    2 referencias
    public class CafeConCanela : DecoradorCafe
    {
        1 referencia
        public CafeConCanela(ICafe cafe) : base(cafe) { }

        9 referencias
        public override string Descripcion() => _cafe.Descripcion() + ", Con canela";
        10 referencias
        public override double Costo() => _cafe.Costo() + 8.00;
    }
}
```

public class CafeConLeche : DecoradorCafe

```
namespace CafeteriaExamen
{
    2 referencias
    public class CafeConLeche : DecoradorCafe
    {
        1 referencia
        public CafeConLeche(ICafe cafe) : base(cafe) { }

        9 referencias
        public override string Descripcion() => _cafe.Descripcion() + ", Con leche";
        10 referencias
        public override double Costo() => _cafe.Costo() + 10.00;
    }
}
```

public class CafeConCaramelo : DecoradorCafe

```
namespace CafeteriaExamen
{
    2 referencias
    public class CafeConCaramelo : DecoradorCafe
    {
        1 referencia
        public CafeConCaramelo(ICafe cafe) : base(cafe) { }

        9 referencias
        public override string Descripcion() => _cafe.Descripcion() + ", Con caramelo";
        10 referencias
        public override double Costo() => _cafe.Costo() + 12.00;
    }
}
```

internal interface ISistemaPrecios

```
namespace CafeteriaExamen
{

    //define el método de conversión.
    2 referencias
    internal interface ISistemaPrecios
    {
        2 referencias
        double ObtenerPrecioEnDolares(double precioPesos);  //Permite desacoplar el sistema de conversión del código del café.
    }
}
```

class AdaptadorSistemaPrecios : ISistemaPrecios

```
namespace CafeteriaExamen
{
    // Adaptador: convierte el precio de pesos a dólares

    //el sistema base no necesita saber cómo se hace la conversión, solo usa el adaptador
    1 referencia
    public class AdaptadorSistemaPrecios : ISistemaPrecios
    {
        private const double TIPO_CAMBIO = 18.30;

        2 referencias
        public double ObtenerPrecioEnDolares(double precioPesos)
        {
            return precioPesos / TIPO_CAMBIO;
        }
    }
}
```

public class CafeBuilder

```
namespace CafeteriaExamen.Model
{
    7 referencias
    public class CafeBuilder  // se arma paso a paso
    {
        private ICafe _cafeActual;

        1 referencia
        public CafeBuilder()
        {
            _cafeActual = new CafeSimple();    //  Punto de inicio (producto base)
        }

        1 referencia
        public CafeBuilder AgregarLeche()
        {
            _cafeActual = new CafeConLeche(_cafeActual);  // usa decorador dentro del builder
            return this;
        }

        1 referencia
        public CafeBuilder AgregarCaramelo()
        {
            _cafeActual = new CafeConCaramelo(_cafeActual);
            return this;
        }

        1 referencia
        public CafeBuilder AgregarAzucar()
        {
            _cafeActual = new CafeConAzucar(_cafeActual);
            return this;
        }

        1 referencia
        public CafeBuilder AgregarCanela()
        {
            _cafeActual = new CafeConCanela(_cafeActual);
            return this;
        }

        2 referencias
        public ICafe Construir()
        {
            return _cafeActual;    //  Entrega el café final
        }
    }
}
```

interface ICafeEstado

```
6
7   namespace CafeteriaExamen
8   {
        5 referencias
9       public interface ICafeEstado
10      {
            4 referencias
11          void Siguiente(CafeContexto contexto);
            4 referencias
12          string ObtenerEstado();
13      }
14  }
15
```

public class CafeContexto

```
6
7   namespace CafeteriaExamen
8   {
        7 referencias
9       public class CafeContexto
10      {
            4 referencias
11          public ICafeEstado EstadoActual { get; private set; }
12
            1 referencia
13          public CafeContexto()
14          {
15              EstadoActual = new EstadoInicial();   // estado inicial
16          }
17
            2 referencias
18          public void CambiarEstado(ICafeEstado nuevoEstado)
19          {
20              EstadoActual = nuevoEstado;   // transición entre estados
21          }
22
            1 referencia
23          public void Avanzar()
24          {
25              EstadoActual.Siguiente(this); // // ejecuta el cambio
26          }
27      }
28  }
29
```

public class EstadoInicial : ICafeEstado

```
namespace CafeteriaExamen
{
    1 referencia
    public  class EstadoInicial : ICafeEstado
    {
        2 referencias
        public string ObtenerEstado() => "Café base creado.";

        2 referencias
        public void Siguiente(CafeContexto contexto)
        {
            contexto.CambiarEstado(new EstadoPersonalizado());
```

public class EstadoPersonalizado :ICafeEstado

```
namespace CafeteriaExamen
{
    1 referencia
    public class EstadoPersonalizado :ICafeEstado
    {
        2 referencias
        public string ObtenerEstado() => "Café personalizado con ingredientes.";

        2 referencias
        public void Siguiente(CafeContexto contexto)
        {
            contexto.CambiarEstado(new EstadoPrecioConvertido());
        }
    }
}
```

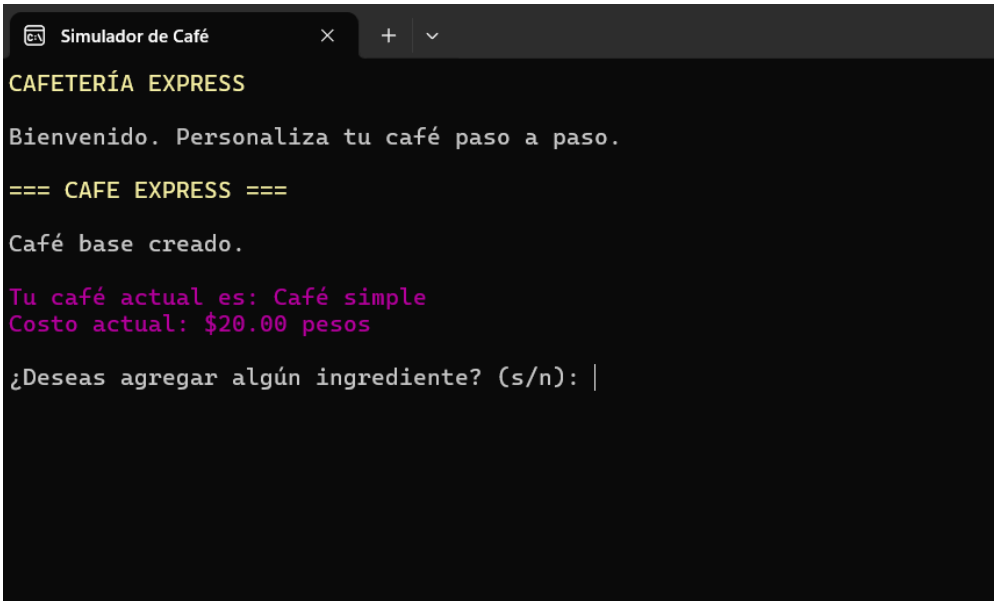
public class EstadoPrecioConvertido: ICafeEstado

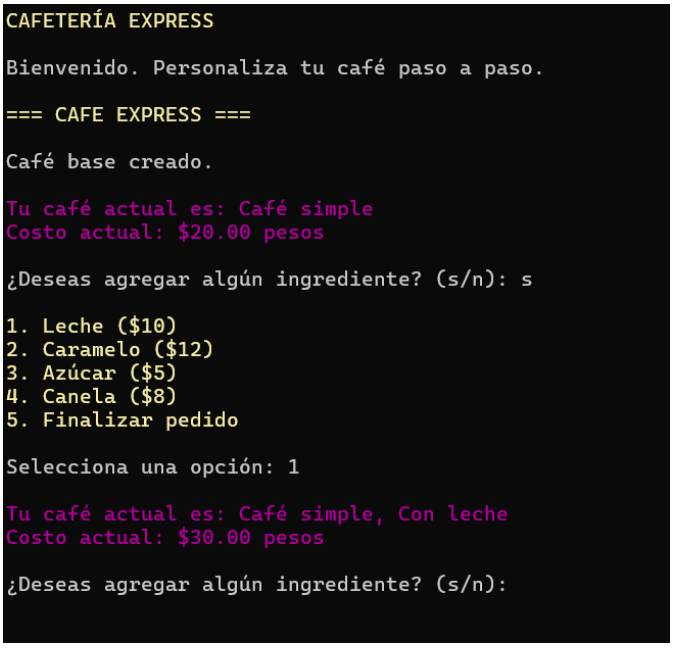
```
namespace CafeteriaExamen
{
    1 referencia
    public class EstadoPrecioConvertido: ICafeEstado
    {
        2 referencias
        public string ObtenerEstado() => "Precio convertido a dólares.";

        2 referencias
        public void Siguiente(CafeContexto contexto)
        {
            // Estado final.
        }
    }
}
```

internal class CafeView

```
namespace CafeteriaExamen.Vista
{
    2 referencias
    internal class CafeView
    {

        1 referencia
        public static void MostrarBienvenida()
        {
            Console.Title = "Simulador de Café";
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("=== CAFE EXPRESS ===\n");
            Console.ResetColor();
        }

        1 referencia
        public static void MostrarCafe(string descripcion, double costo)
        {
            Console.ForegroundColor = ConsoleColor.Magenta;
            Console.WriteLine($"\nTu café actual es: {descripcion}");
            Console.WriteLine($"Costo actual: ${costo:0.00} pesos");
            Console.ResetColor();
        }
    }
}
```

```
            _contexto.Avanzar();
            return _builder.Construir();
        }

        1 referencia
        private void MostrarMenuIngredientes()
        {
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("\n1. Leche ($10)");
            Console.WriteLine("2. Caramelo ($12)");
            Console.WriteLine("3. Azúcar ($5)");
            Console.WriteLine("4. Canela ($8)");
            Console.WriteLine("5. Finalizar pedido");
            Console.ResetColor();

            Console.Write("\nSelecciona una opción: ");
            string opcion = Console.ReadLine();

            switch (opcion)
            {
                case "1": _builder.AgregarLeche(); break;
                case "2": _builder.AgregarCaramelo(); break;
                case "3": _builder.AgregarAzucar(); break;
                case "4": _builder.AgregarCanela(); break;
                case "5": _continuar = false; break;
                default: Console.WriteLine("Opción inválida."); break;
            }
        }
    }
}
```


Simulador de Café
CAFETERÍA EXPRESS
Bienvenido. Personaliza tu café paso a paso.
=== CAFE EXPRESS ===
Café base creado.
Tu café actual es: Café simple
Costo actual: $20.00 pesos
¿Deseas agregar algún ingrediente? (s/n):


CAFETERÍA EXPRESS
Bienvenido. Personaliza tu café paso a paso.
=== CAFE EXPRESS ===
Café base creado.
Tu café actual es: Café simple
Costo actual: $20.00 pesos
¿Deseas agregar algún ingrediente? (s/n): s
1. Leche ($10)
2. Caramelo ($12)
3. Azúcar ($5)
4. Canela ($8)
5. Finalizar pedido
Selecciona una opción: 1
Tu café actual es: Café simple, Con leche
Costo actual: $30.00 pesos
¿Deseas agregar algún ingrediente? (s/n):


TICKET
Café: Café simple, Con leche
Total en pesos: $30.00
Total en dólares: $1.64 USD
Gracias por tu compra :)

Conclusión

Al terminar este proyecto entendí mucho mejor cómo funcionan los patrones de diseño y por qué son tan útiles cuando se programa. El uso del Decorador me permitió agregar ingredientes sin complicarme y sin hacer muchas clases diferentes. El Builder hizo más fácil construir el café paso por paso, mientras que el patrón State me ayudó a organizar mejor las diferentes etapas del proceso. El Adaptador también fue importante porque me permitió convertir el precio sin mezclar esa lógica con la del café. Y gracias a la arquitectura MVC pude separar mi código en carpetas claras: Model, View y Controller, lo que hace que todo se vea más ordenado y profesional. En general, este proyecto me ayudó a mejorar mi forma de programar y me dejó más claro cómo aplicar buenas prácticas para que un programa sea más limpio y fácil de mantener