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:

Datos de clase privada

UNIDAD A EVALUAR:
Examen 3

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

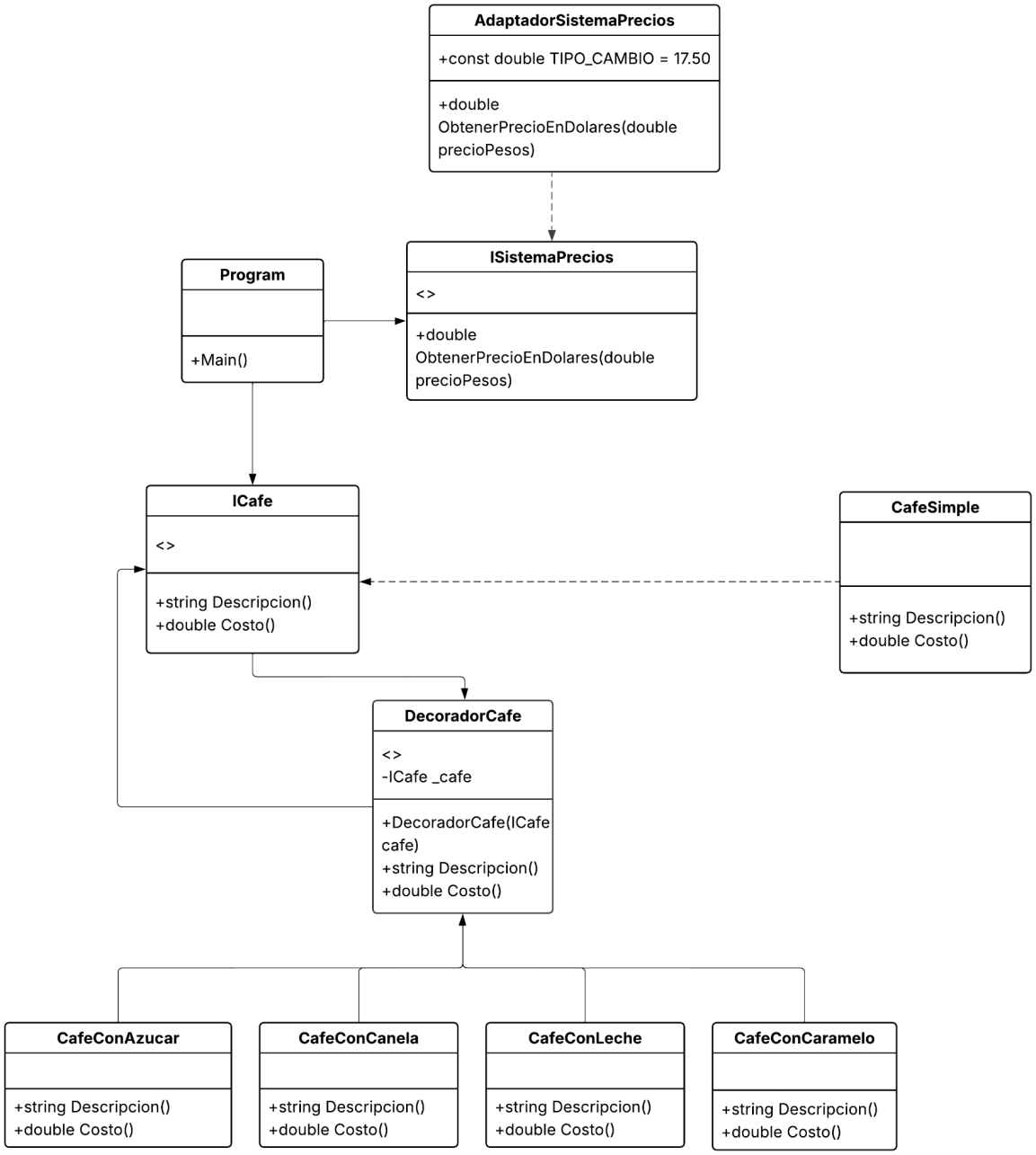
Liczi Citlali Flores Ramirez - 21210537

NOMBRE DEL MAESTRO (A):
MARIBEL GUERRERO LUIS

Introducción

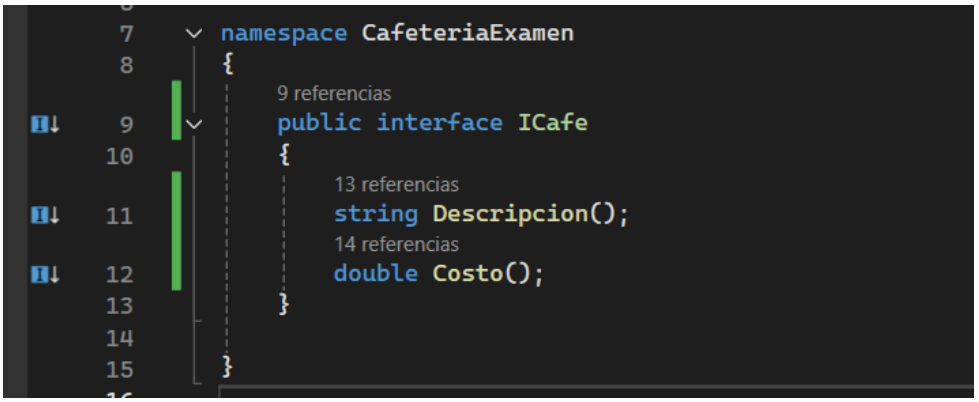
Desarrollé un simulador de café en C# utilizando el modo consola. El objetivo principal fue aplicar los patrones de diseño Decorador y Adaptador dentro de un mismo sistema funcional. El programa simula una cafetería donde el usuario inicia con un café simple y tiene la posibilidad de personalizarlo agregando ingredientes opcionales como leche, azúcar, canela o caramelo, cada uno con su costo adicional.El patrón Decorador permitió agregar estos ingredientes de forma dinámica, sin necesidad de modificar la estructura base del café. Cada vez que el usuario selecciona un nuevo ingrediente, se crea una nueva capa de decoración sobre el objeto principal.Además, se aplicó el patrón Adaptador para convertir el precio total del café de pesos a dólares, implementando una clase adaptadora que realiza la conversión a otro sistema de valores sin alterar el funcionamiento del programa base.En conjunto, el proyecto demuestra cómo estos patrones pueden trabajar de manera complementaria dentro de una misma aplicación orientada a objetos.

Diagrama UML



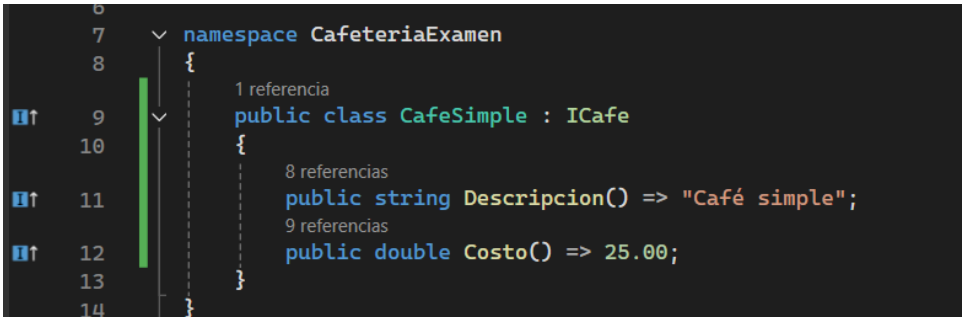
CÓDIGO

Interface ICafe



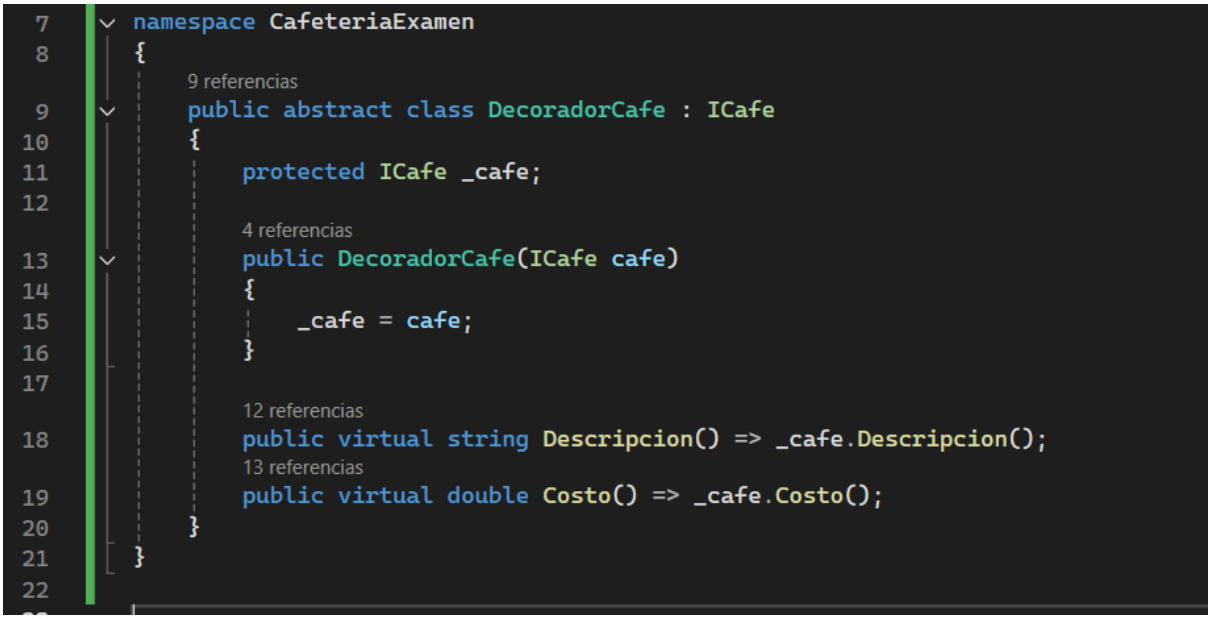
```
namespace CafeteriaExamen
{
    9 referencias
    public interface ICafe
    {
        13 referencias
        string Descripcion();
        14 referencias
        double Costo();
    }
}
```

class CafeSimple



```
namespace CafeteriaExamen
{
    1 referencia
    public class CafeSimple : ICafe
    {
        8 referencias
        public string Descripcion() => "Café simple";
        9 referencias
        public double Costo() => 25.00;
    }
}
```

abstract class DecoradorCafe



```
namespace CafeteriaExamen
{
    9 referencias
    public abstract class DecoradorCafe : ICafe
    {
        protected ICafe _cafe;

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

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

class CafeConCaramelo

```
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;
    }
}
```

cLass CafeConCanela

```
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;
    }
}
```

class CafeConLeche

```
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;
    }
}
```

class CafeConAzucar

```
namespace CafeteriaExamen
{
    2 referencias
    internal 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;
    }
}
```

interface ISistemaPrecios

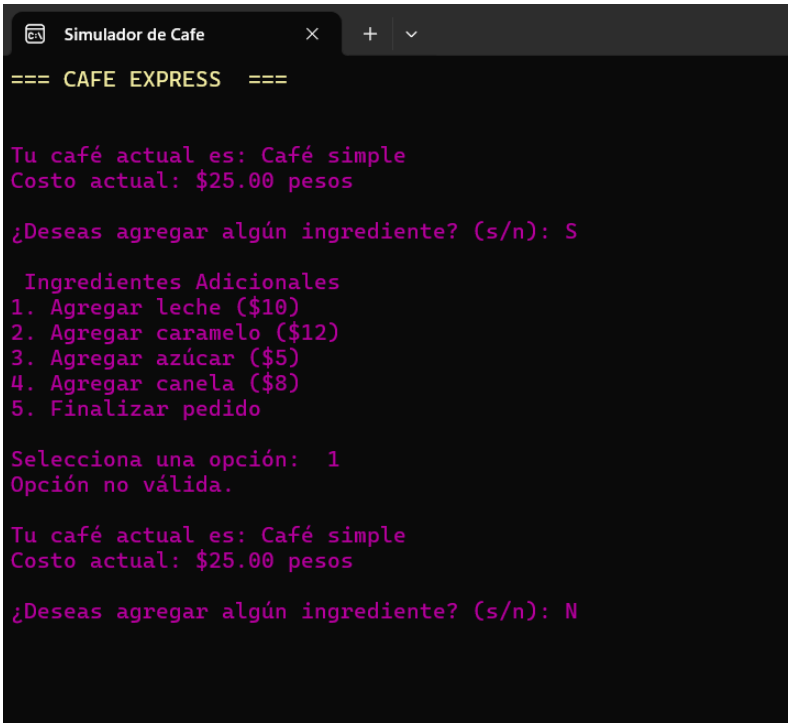
```
namespace CafeteriaExamen
{
    2 referencias
    internal interface ISistemaPrecios
    {
        2 referencias
        double ObtenerPrecioEnDolares(double precioPesos);
    }
}
```

class AdaptadorSistemaPrecios

```
namespace CafeteriaExamen
{
    // Adaptador: convierte el precio de pesos a dólares
    1 referencia
    public class AdaptadorSistemaPrecios : ISistemaPrecios
    {
        private const double TIPO_CAMBIO = 18.30;

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

Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Net.Mime.MediaTypeNames;

namespace CafeteriaExamen
{
    0 referencias
    internal class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            Console.Title = " Simulador de Cafe ";
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("=== CAFE EXPRESS  ===\n");

            ICafe miCafe = new CafeSimple();
            bool continuar = true;

            while (continuar)
            {
                Console.ForegroundColor = ConsoleColor.Magenta ;
                Console.WriteLine($"\nTu café actual es: {miCafe.Descripcion()}");
                Console.WriteLine($"Costo actual: ${miCafe.Costo():0.00} pesos");
                Console.Write("\n¿Deseas agregar algún ingrediente? (s/n): ");
                string respuesta = Console.ReadLine()?.ToLower();

                if (respuesta == "s")
                {
                    Console.WriteLine("\n Ingredientes Adicionales  ");
                    Console.WriteLine("1. Agregar leche ($10)");
                    Console.WriteLine("2. Agregar caramelo ($12)");
                    Console.WriteLine("3. Agregar azúcar ($5)");
                    Console.WriteLine("4. Agregar canela ($8)");
                    Console.WriteLine("5. Finalizar pedido");

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

                    switch (opcion)
                    {
                        case "1": miCafe = new CafeConLeche(miCafe); break;
                        case "2": miCafe = new CafeConCaramelo(miCafe); break;
                        case "3": miCafe = new CafeConAzucar(miCafe); break;
                        case "4": miCafe = new CafeConCanela(miCafe); break;
                        case "5": continuar = false; break;
                        default: Console.WriteLine("Opción no válida."); break;
                    }
                }
                else
                {
                    continuar = false;
                }
            }

            Console.Clear();
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("═══════════════════════════════════");
            Console.WriteLine("            ORDEN FINAL ");
            Console.WriteLine("═══════════════════════════════════\n");
            Console.ResetColor();

            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine($"Tu pedido: {miCafe.Descripcion()}");
            Console.WriteLine($"Total en pesos: ${miCafe.Costo():0.00}");

            ISistemaPrecios adaptador = new AdaptadorSistemaPrecios();
            double totalDolares = adaptador.ObtenerPrecioEnDolares(miCafe.Costo());
            Console.WriteLine($"Total en dólares: ${totalDolares:0.00} USD");

            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("     ¡Gracias por tu compra! ");
            Console.ResetColor();

            Console.ReadKey();
        }

    }
}
```


Simulador de Cafe
=== CAFE EXPRESS ===
Tu café actual es: Café simple
Costo actual: $25.00 pesos
¿Deseas agregar algún ingrediente? (s/n): S
Ingredientes Adicionales
1. Agregar leche ($10)
2. Agregar caramelo ($12)
3. Agregar azúcar ($5)
4. Agregar canela ($8)
5. Finalizar pedido
Selecciona una opción: 1
Opción no válida.
Tu café actual es: Café simple
Costo actual: $25.00 pesos
¿Deseas agregar algún ingrediente? (s/n): N


ORDEN FINAL
Tu pedido: Café simple
Total en pesos: $25.00
Total en dólares: $1.37 USD
¡Gracias por tu compra!

CONCLUSIÓN

A través del desarrollo de este programa comprendí la importancia de los patrones de diseño en la programación orientada a objetos.El patrón Decorador me permitió entender cómo agregar funcionalidades adicionales a un objeto sin alterar su código original, logrando un sistema flexible. En cambio, el patrón Adaptador me enseñó cómo conectar sistemas o estructuras diferentes, adaptando interfaces sin necesidad de modificar las clases existentes.Unir ambos patrones en un mismo ejercicio resultó muy útil, ya que mientras el Decorador se enfocó en extender el comportamiento del café mediante ingredientes adicionales, el Adaptador permitió integrar la funcionalidad de conversión de precios de pesos a dólares.