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 de software

TÍTULO ACTIVIDAD:

Examen unidad 4 y 5

UNIDAD A EVALUAR:
4 y 5

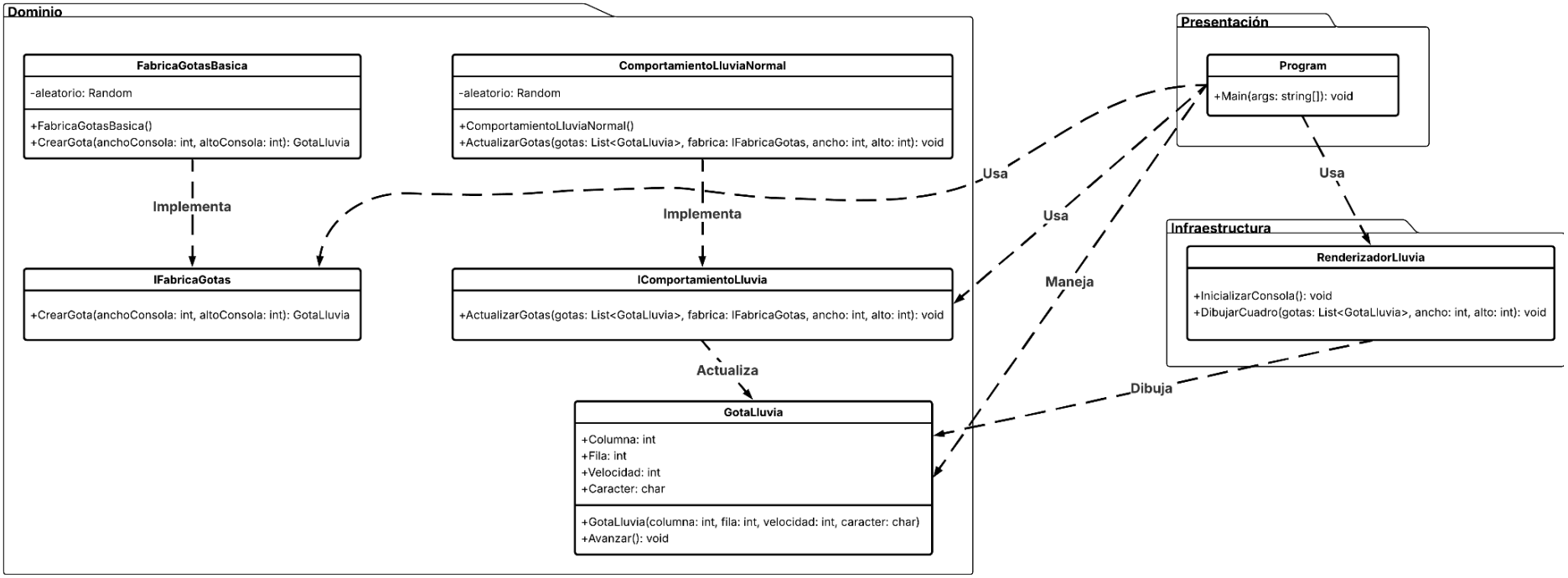
NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

Herrera Sandoval Jose Miguel 21212344

NOMBRE DEL MAESTRO (A):

Maribel Guerrero Luis

Diagrama UML



Código

[Program.cs](#)

```
namespace PatronesU5v2
{
    0 referencias
    class Program
    {
        0 referencias
        static void Main(string[] args)
        {
            Console.Title = "Lluvia ASCII con patrones";
            Console.CursorVisible = false;

            int ancho = Console.WindowWidth;
            int alto = Console.WindowHeight - 1;

            var gotas = new List<GotaLluvia>();

            IFabricaGotas fabrica = new FabricaGotasBasica();
            IComportamientoLluvia comportamiento = new ComportamientoLluviaNormal();
            var renderizador = new RenderizadorLluvia();

            renderizador.InicializarConsola();

            Console.WriteLine("Lluvia ASCII");
            Console.WriteLine("Presiona ESC para salir.");
            Thread.Sleep(1000);

            while (true)
            {
                if (Console.KeyAvailable)
                {
                    var tecla = Console.ReadKey(true);
                    if (tecla.Key == ConsoleKey.Escape)
                    {
                        break;
                    }
                }

                comportamiento.ActualizarGotas(gotas, fabrica, ancho, alto);

                renderizador.DibujarCuadro(gotas, ancho, alto);

                Thread.Sleep(50);
            }

            Console.CursorVisible = true;
        }
    }
}
```

GotaLluvia.cs

```
namespace PatronesU5v2
{
    public class GotaLluvia
    {
        public int Columna { get; set; }
        6 referencias
        public int Fila { get; set; }
        2 referencias
        public int Velocidad { get; set; }
        2 referencias
        public char Caracter { get; set; }

        1 referencia
        public GotaLluvia(int columna, int fila, int velocidad, char caracter)
        {
            Columna = columna;
            Fila = fila;
            Velocidad = velocidad;
            Caracter = caracter;
        }

        1 referencia
        public void Avanzar()
        {
            Fila += Velocidad;
        }
    }
}
```

IFabricaGotas.cs **(Factory Method)**

```
namespace PatronesU5v2
{
    4 referencias
    public interface IFabricaGotas
    {
        2 referencias
        GotaLluvia CrearGota(int anchoConsola, int altoConsola);
    }
}
```

FabricaGotasBasica.cs (Implementación Factory Method)

```
namespace PatronesU5v2
{
    2 referencias
    public class FabricaGotasBasica : IFabricaGotas
    {
        private readonly Random _aleatorio;

        1 referencia
        public FabricaGotasBasica()
        {
            _aleatorio = new Random();
        }

        2 referencias
        public GotaLluvia CrearGota(int anchoConsola, int altoConsola)
        {
            int columna = _aleatorio.Next(0, Math.Max(1, anchoConsola - 1));
            int filaInicial = 0;
            int velocidad = _aleatorio.Next(1, 3);
            char caracter = '|';

            return new GotaLluvia(columna, filaInicial, velocidad, caracter);
        }
    }
}
```

IComportamientoLluvia.cs (Strategy)

```
namespace PatronesU5v2
{
    2 referencias
    public interface IComportamientoLluvia
    {
        2 referencias
        void ActualizarGotas(List<GotaLluvia> gotas, IFabricaGotas fabrica, int ancho, int alto);
    }
}
```

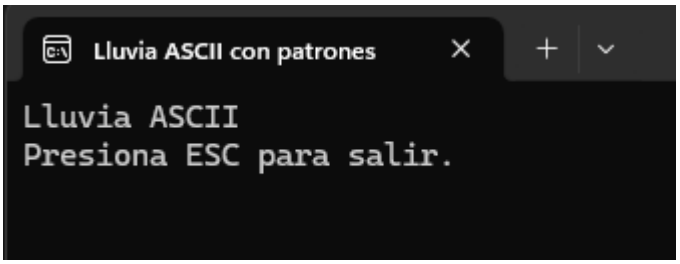
ComportamientoLluviaNormal.cs (Implementación Strategy)

```
namespace PatronesU5v2
{
    2 referencias
    public class ComportamientoLluviaNormal : IComportamientoLluvia
    {
        private readonly Random _aleatorio;

        1 referencia
        public ComportamientoLluviaNormal()
        {
            _aleatorio = new Random();
        }

        2 referencias
        public void ActualizarGotas(List<GotaLluvia> gotas, IFabricaGotas fabrica, int ancho, int alto)
        {
            for (int i = 0; i < gotas.Count; i++)
            {
                gotas[i].Avanzar();
            }

            for (int i = gotas.Count - 1; i >= 0; i--)
            {
                if (gotas[i].Fila >= alto)
                {
                    gotas.RemoveAt(i);
                }
            }

            int nuevas = _aleatorio.Next(2, 4);
            for (int i = 0; i < nuevas; i++)
            {
                var nuevaGota = fabrica.CrearGota(ancho, alto);
                gotas.Add(nuevaGota);
            }
        }
    }
}
```

RenderizadorLluvia.cs (Facade)

```
namespace PatronesU5v2
{
    1 referencia
    public class RenderizadorLluvia
    {
        1 referencia
        public void InicializarConsola()
        {
            Console.Clear();
        }

        1 referencia
        public void DibujarCuadro(List<GotaLluvia> gotas, int ancho, int alto)
        {
            char[,] buffer = new char[alto, ancho];

            for (int fila = 0; fila < alto; fila++)
            {
                for (int col = 0; col < ancho; col++)
                {
                    buffer[fila, col] = ' ';
                }
            }

            foreach (var gota in gotas)
            {
                if (gota.Fila >= 0 && gota.Fila < alto &&
                    gota.Columna >= 0 && gota.Columna < ancho)
                {
                    buffer[gota.Fila, gota.Columna] = gota.Caracter;
                }
            }

            Console.SetCursorPosition(0, 0);
            for (int fila = 0; fila < alto; fila++)
            {
                var linea = new char[ancho];
                for (int col = 0; col < ancho; col++)
                {
                    linea[col] = buffer[fila, col];
                }
                Console.WriteLine(linea);
            }
        }
    }
}
```

Capturas de pantalla (Ejecución)



Conclusión

Anteriormente se había implementado con Object Pool y Singleton el mismo programa para simular una lluvia estilo ASCII “|”, pero en este caso, se utilizó Fachada, Estrategia, Fábrica abstracta y Arquitectura de capas. La mayor diferencia que encontré es que es mucho más fácil de codificar utilizando estos 4 patrones, con funciones limitadas y solo pudiendo cambiar de manera manual la velocidad y la cantidad de la “lluvia”, pero se intentó implementar los patrones lo mejor posible. Aprendí que, en la mayoría de casos, utilizar varios patrones de diseño mejora significativamente el rendimiento y simplifica las líneas de código, evitando sobrecargar el proyecto.