TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto-Diciembre 2025

CARRERA:
INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA:
Patrones de diseño

TÍTULO ACTIVIDAD :
Examen unidad 2 Patrones de diseño

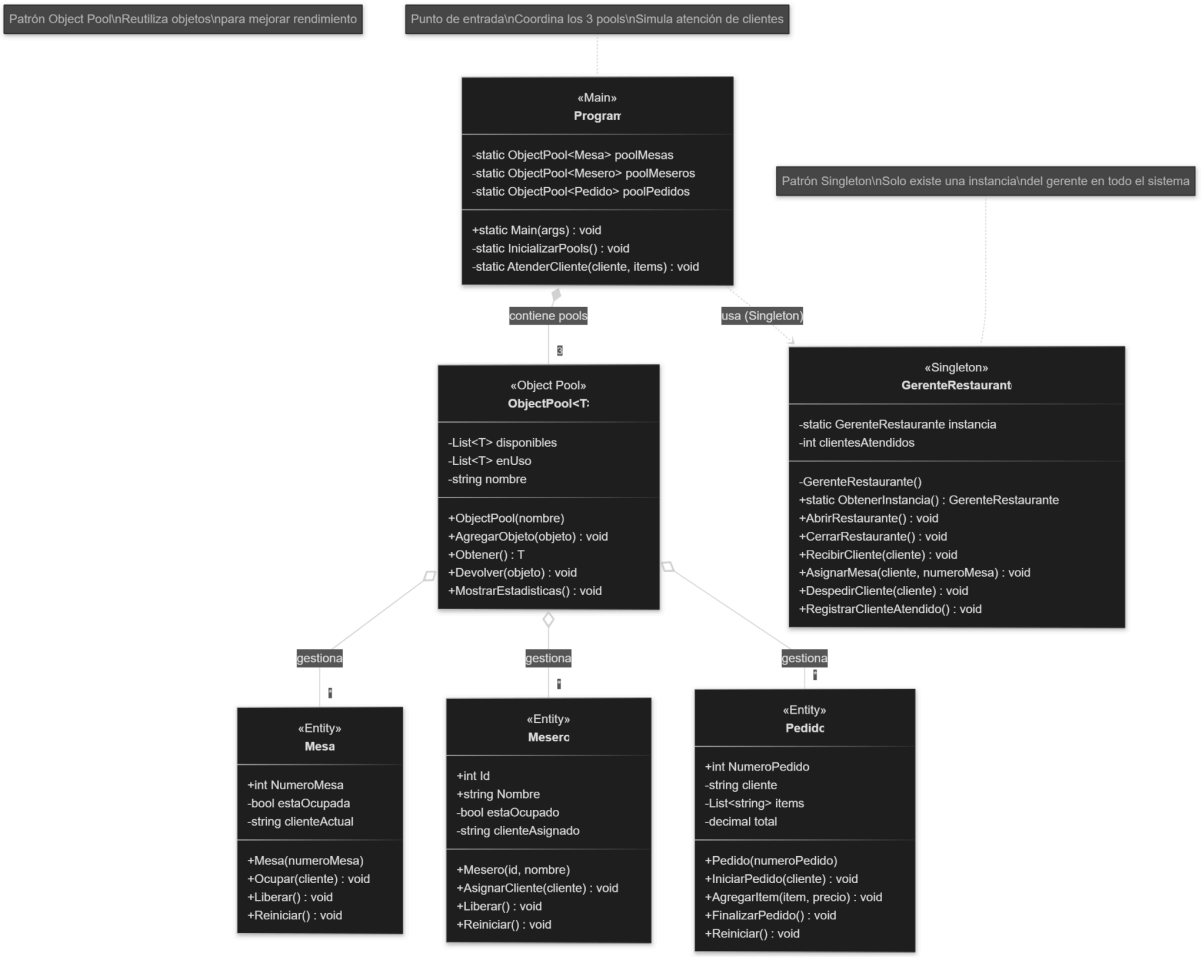
UNIDAD 2

NOMBRE Y NÚMERO DE CONTROL:
NARVAEZ MATA ALEJANDRO- 22210325

NOMBRE DEL MAESTRO (A):

MARIBEL GUERRERO LUIS

UML


Patrón Object Pool\nReutiliza objetos\npara mejorar rendimiento
Punto de entrada\nCoordina los 3 pools\nSimula atención de clientes
«Main»
Program
-static ObjectPool<Mesa> poolMesas
-static ObjectPool<Mesero> poolMeseros
-static ObjectPool<Pedido> poolPedidos
+static Main(args) : void
-static InicializarPools() : void
-static AtenderCliente(cliente, items) : void
Patrón Singleton\nSolo existe una instancia\ndel gerente en todo el sistema
contiene pools
3
usa (Singleton)
«Object Pool»
ObjectPool<T:
-List<T> disponibles
-List<T> enUso
-string nombre
+ObjectPool(nombre)
+AgregarObjeto(objeto) : void
+Obtener() : T
+Devolver(objeto) : void
+MostrarEstadisticas() : void
«Singleton»
GerenteRestaurant
-static GerenteRestaurante instancia
-int clientesAtendidos
-GerenteRestaurante()
+static ObtenerInstancia() : GerenteRestaurante
+AbrirRestaurante() : void
+CerrarRestaurante() : void
+RecibirCliente(cliente) : void
+AsignarMesa(cliente, numeroMesa) : void
+DespedirCliente(cliente) : void
+RegistrarClienteAtendido() : void
gestiona
gestiona
gestiona
«Entity»
Mesa
+int NumeroMesa
-bool estaOcupada
-string clienteActual
+Mesa(numeroMesa)
+Ocupar(cliente) : void
+Liberar() : void
+Reiniciar() : void
«Entity»
Meserc
+int Id
+string Nombre
-bool estaOcupado
-string clienteAsignado
+Mesero(id, nombre)
+AsignarCliente(cliente) : void
+Liberar() : void
+Reiniciar() : void
«Entity»
Pedidc
+int NumeroPedido
-string cliente
-List<string> items
-decimal total
+Pedido(numeroPedido)
+IniciarPedido(cliente) : void
+AgregarItem(item, precio) : void
+FinalizarPedido() : void
+Reiniciar() : void

Código

program.cs

```
using System;


namespace RestauranteSimulacion
{
    class Program
    {
        // Aquí guardo los pools que voy a usar en todo el programa
        static ObjectPool<Mesa> poolMesas = null!; // Pool de mesas
        static ObjectPool<Mesero> poolMeseros = null!; // Pool de
meseros
        static ObjectPool<Pedido> poolPedidos = null!; // Pool de
pedidos


        // Aquí es donde empieza el programa
        static void Main(string[] args)
        {
            // Obtengo el gerente (Singleton - solo existe uno)
            var gerente = GerenteRestaurante.ObtenerInstancia();
            gerente.AbrirRestaurante(); // Abro el restaurante


            // Creo e inicializo los pools
            InicializarPools();
```

```
            // Atiendo a los clientes
            AtenderCliente("Frenkie de Jong", new[] { ("Hamburguesa",
150m), (" Bacardi con coca ", 30m) });
            AtenderCliente("Aitana Bombati", new[] { ("Pizza", 200m)
});


            // Muestro las estadísticas de los pools
            poolMesas.MostrarEstadisticas();
            poolMeseros.MostrarEstadisticas();
            poolPedidos.MostrarEstadisticas();


            // Cierro el restaurante
            gerente.CerrarRestaurante();
            Console.ReadKey(); // Espero a que presionen una tecla
        }


        // Aquí creo los pools y les agrego objetos
        static void InicializarPools()
        {
            // Creo el pool de mesas y agrego 3 mesas
            poolMesas = new ObjectPool<Mesa>("Mesas");
            poolMesas.AgregarObjeto(new Mesa(1)); // Agrego mesa 1
            poolMesas.AgregarObjeto(new Mesa(2)); // Agrego mesa 2
            poolMesas.AgregarObjeto(new Mesa(3)); // Agrego mesa 3
```

```
            // Creo el pool de meseros y agrego 2 meseros
            poolMeseros = new ObjectPool<Mesero>("Meseros");
            poolMeseros.AgregarObjeto(new Mesero(1, "Gilberto")); // Agrego a Gilberto
            poolMeseros.AgregarObjeto(new Mesero(2, "Marcela")); // Agrego a Marcela

            // Creo el pool de pedidos y agrego 2 pedidos
            poolPedidos = new ObjectPool<Pedido>("Pedidos");
            poolPedidos.AgregarObjeto(new Pedido(1)); // Agrego pedido 1
            poolPedidos.AgregarObjeto(new Pedido(2)); // Agrego pedido 2
        }

        // Aquí simulo la atención de un cliente completo
        static void AtenderCliente(string cliente, (string item, decimal precio)[] items)
        {
            Console.WriteLine($"\n{new string('=', 60)}"); // Aquí creo una línea de 60 signos de igual
            Console.WriteLine($"[CLIENTE] {cliente} llego al restaurante");
            var gerente = GerenteRestaurante.ObtenerInstancia(); // Obtengo el gerente
            gerente.RecibirCliente(cliente); // El gerente recibe al cliente
```

```
            Console.WriteLine("\n--- Solicitando recursos del pool
---");

            // Paso 1: Pido una mesa del pool

            Mesa mesa = poolMesas.Obtener();

            if (mesa == null) return; // Si no hay mesas, me detengo



            gerente.AsignarMesa(cliente, mesa.NumeroMesa); // El
gerente asigna la mesa



            // Paso 2: Pido un mesero del pool

            Mesero mesero = poolMeseros.Obtener();

            if (mesero == null) // Si no hay meseros

            {

                poolMesas.Devolver(mesa); // Devuelvo la mesa

                return; // Me detengo

            }



            // Paso 3: Pido un pedido del pool

            Pedido pedido = poolPedidos.Obtener();

            if (pedido == null) // Si no hay pedidos

            {

                poolMesas.Devolver(mesa); // Devuelvo la mesa

                poolMeseros.Devolver(mesero); // Devuelvo el mesero

                return; // Me detengo

            }
```

```
            Console.WriteLine("\n--- Usando los recursos ---");
            // Paso 4: Uso los recursos que obtuve
            mesa.Ocupar(cliente); // Ocupo la mesa
            mesero.AsignarCliente(cliente); // Asigno el mesero
            pedido.IniciarPedido(cliente); // Inicio el pedido

            // Paso 5: Agrego items al pedido
            foreach (var (item, precio) in items) // Recorro cada item
            {
                pedido.AgregarItem(item, precio); // Agrego el item
            }

            pedido.FinalizarPedido(); // Finalizo el pedido

            Console.WriteLine("\n--- Devolviendo recursos al pool
---");
            // Paso 6: Devuelvo los recursos al pool
            poolPedidos.Devolver(pedido); // Devuelvo el pedido
            poolMeseros.Devolver(mesero); // Devuelvo el mesero
            poolMesas.Devolver(mesa); // Devuelvo la mesa

            // Paso 7: Registro que atendí al cliente
            gerente.RegistrarClienteAtendido(); // Registro en el
gerente
```

```
            gerente.DespedirCliente(cliente); // El gerente despide al
cliente

            Console.WriteLine($"[CLIENTE] {cliente} se fue del
restaurante");

        }

    }

}
```

Object pool.cs

```
using System;

using System.Collections.Generic;



namespace RestauranteSimulacion

{

    // Implementación del patrón Object Pool para reutilización de
objetos

    public class ObjectPool<T> where T : class

    {

        private readonly List<T> _disponibles; // Aquí guardo los
objetos listos para usar

        private readonly List<T> _enUso; // Aquí guardo los objetos que
están en uso

        public string NombrePool { get; private set; } // Aquí está el
nombre del pool



        // Constructor: aquí inicializo el pool

        public ObjectPool(string nombrePool)

        {

            NombrePool = nombrePool; // Guardo el nombre que me dan

            _disponibles = new List<T>(); // Creo la lista de
disponibles vacía

            _enUso = new List<T>(); // Creo la lista de en uso vacía
```

```
            Console.WriteLine($"[POOL] Pool '{nombrePool}' creado"); // Aviso que ya estoy listo
        }

        // Aquí recibo objetos nuevos para el pool
        public void AgregarObjeto(T objeto)
        {
            _disponibles.Add(objeto); // Pongo el objeto en disponibles
            Console.WriteLine($"[POOL-PUSH] ✓ Objeto agregado al pool '{NombrePool}' (Total disponibles: {_disponibles.Count})");
        }

        // Aquí entrego un objeto cuando me lo piden
        public T Obtener()
        {
            if (_disponibles.Count > 0) // Verifico si tengo objetos disponibles
            {
                T objeto = _disponibles[0]; // Tomo el primer objeto que tengo
                _disponibles.RemoveAt(0); // Lo saco de disponibles
                _enUso.Add(objeto); // Lo pongo en la lista de en uso
                Console.WriteLine($"[POOL-POP] ← Objeto SACADO del pool '{NombrePool}' (Disponibles: {_disponibles.Count} | En uso: {_enUso.Count})");
                return objeto; // Se lo devuelvo al que lo pidió
            }
```

```
            Console.WriteLine($"[POOL-ERROR] ✗ Pool '{NombrePool}'
vacío - No hay objetos disponibles");

            return null!; // No tengo nada disponible, devuelvo null

        }



        // Aquí recibo de vuelta los objetos que ya terminaron de usar

        public void Devolver(T objeto)

        {

            if (objeto == null) return; // Si me dan null, no hago nada



            if (_enUso.Remove(objeto)) // Lo busco y saco de la lista
en uso

            {

                ReiniciarObjeto(objeto); // Lo reinicio para que esté
listo de nuevo

                _disponibles.Add(objeto); // Lo pongo de vuelta en
disponibles

                Console.WriteLine($"[POOL-RETURN] → Objeto DEVUELTO al
pool '{NombrePool}' (Disponibles: {_disponibles.Count} | En uso:
{_enUso.Count})");

            }

        }



        // Aquí reinicio el objeto según su tipo

        private void ReiniciarObjeto(T objeto)

        {
```

```
            if (objeto is Mesa mesa) mesa.Reiniciar(); // Si es mesa, la reinicio
            else if (objeto is Mesero mesero) mesero.Reiniciar(); // Si es mesero, lo reinicio
            else if (objeto is Pedido pedido) pedido.Reiniciar(); // Si es pedido, lo reinicio
        }

        // Aquí muestro cuántos objetos tengo
        public void MostrarEstadisticas()
        {
            Console.WriteLine($"\n[{NombrePool}] Disponibles: {_disponibles.Count} | En uso: {_enUso.Count}"); // Muestro el resumen
        }
    }
}
```

Gerente restaurante.cs (singleton)

```
using System;


namespace RestauranteSimulacion
{
    // Aquí implemento el patrón Singleton - Solo puedo existir una vez
    public sealed class GerenteRestaurante
    {
        private static GerenteRestaurante _instancia = null!; // Aquí guardo mi única instancia
        private static readonly object _lock = new object(); // Aquí controlo el acceso de múltiples hilos


        public int ClientesAtendidos { get; private set; } // Aquí cuento los clientes que atiendo


        // Constructor privado - Solo yo puedo crearme
        private GerenteRestaurante()
        {
            ClientesAtendidos = 0; // Empiezo sin clientes atendidos
            Console.WriteLine("\n[SINGLETON] ✓ INSTANCIA ÚNICA CREADA - Gerente del restaurante");
        }


        // Aquí entrego mi única instancia (o me creo si no existo)
        public static GerenteRestaurante ObtenerInstancia()
```

```
        {
            if (_instancia == null) // Verifico si ya existo
            {
                lock (_lock) // Bloqueo para que solo un hilo entre
                {
                    if (_instancia == null) // Verifico de nuevo dentro del bloqueo
                    {
                        _instancia = new GerenteRestaurante(); // Me creo por primera vez
                    }
                }
            }
            else
            {
                Console.WriteLine("  [SINGLETON] → Reutilizando instancia única existente");
            }
            return _instancia; // Me devuelvo a quien me llamó
        }


        // Aquí abro el restaurante
        public void AbrirRestaurante()
        {
            Console.WriteLine("\n[RESTAURANTE] El gerente ha abierto el restaurante\n");
```

```
        }

        // Aquí cierro el restaurante
        public void CerrarRestaurante()
        {
            Console.WriteLine($"\n[RESTAURANTE] El gerente ha cerrado el restaurante - Clientes atendidos: {ClientesAtendidos}\n");
        }

        // Aquí recibo a los clientes en la entrada
        public void RecibirCliente(string nombreCliente)
        {
            Console.WriteLine($"  [SINGLETON-HOST] Gerente recibe a {nombreCliente} en la entrada");
        }

        // Aquí asigno una mesa al cliente
        public void AsignarMesa(string nombreCliente, int numeroMesa)
        {
            Console.WriteLine($"  [SINGLETON-HOST] Gerente asigna Mesa {numeroMesa} a {nombreCliente}");
        }

        // Aquí registro cada cliente que atiendo
        public void RegistrarClienteAtendido()
        {
```

```
            ClientesAtendidos++; // Sumo uno al contador

            Console.WriteLine($"  [SINGLETON-HOST] Gerente registra
cliente #{ClientesAtendidos} atendido");

        }



        // Aquí me despido de los clientes

        public void DespedirCliente(string nombreCliente)

        {

            Console.WriteLine($"  [SINGLETON-HOST] Gerente despide a
{nombreCliente}");

        }

    }

}
```

Pedido.cs

```
using System;
using System.Collections.Generic;


namespace RestauranteSimulacion
{
    // Aquí represento un pedido de comida (objeto reciclable del pool)
    public class Pedido
    {
        public int NumeroPedido { get; private set; } // Aquí guardo mi número de pedido
        public decimal Total { get; private set; } // Aquí guardo el total del pedido


        // Aquí me crean con mi número de pedido
        public Pedido(int numeroPedido)
        {
            NumeroPedido = numeroPedido; // Guardo el número que me asignan
            Total = 0; // Empiezo con total en cero
        }


        // Aquí inicio un pedido para un cliente
        public void IniciarPedido(string cliente)
        {
```

```
            Console.WriteLine($"  [PEDIDO] Pedido {NumeroPedido} → INICIADO para {cliente}");
        }

        // Aquí agrego items al pedido
        public void AgregarItem(string item, decimal precio)
        {
            Total += precio; // Sumo el precio al total
            Console.WriteLine($"    + {item} (${precio})");
        }

        // Aquí finalizo el pedido y muestro el total
        public void FinalizarPedido()
        {
            Console.WriteLine($"  [PEDIDO] → FINALIZADO - Total: ${Total}");
        }

        // Aquí me reinician para que pueda ser usado de nuevo
        public void Reiniciar()
        {
            Total = 0; // Limpio el total
            Console.WriteLine($"  [PEDIDO] Pedido {NumeroPedido} → LIMPIADO y listo para reusar");
        }
    }
```

```
}
```

Mesero.cs

```
using System;


namespace RestauranteSimulacion
{
    // Aquí represento a un mesero del restaurante (objeto reciclable del pool)
    public class Mesero
    {
        public string Nombre { get; private set; } // Aquí guardo mi nombre
        public bool EstaDisponible { get; private set; } // Aquí guardo si estoy libre o ocupado


        // Aquí me crean con mi nombre
        public Mesero(int id, string nombre)
        {
            Nombre = nombre; // Guardo el nombre que me dan
            EstaDisponible = true; // Empiezo disponible para atender
        }


        // Aquí me asignan a un cliente
        public void AsignarCliente(string cliente)
        {
            EstaDisponible = false; // Me marco como ocupado
```

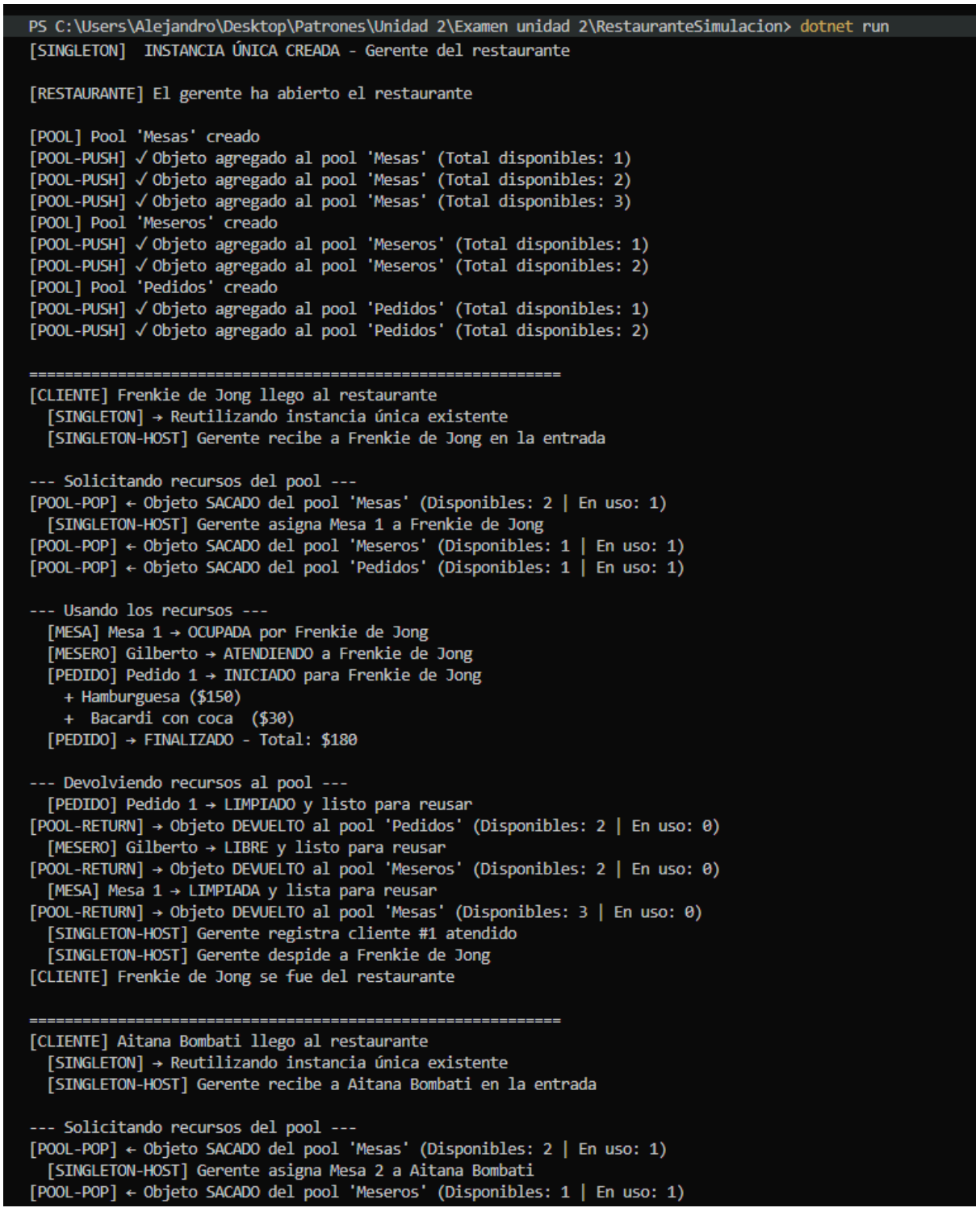
```
            Console.WriteLine($"  [MESERO] {Nombre} → ATENDIENDO a
{cliente}");

        }



        // Aquí me reinician para que pueda atender a otro cliente

        public void Reiniciar()

        {

            EstaDisponible = true; // Me libero para el siguiente
cliente

            Console.WriteLine($"  [MESERO] {Nombre} → LIBRE y listo
para reusar");

        }

    }

}
```

Mesa.cs

```
using System;


namespace RestauranteSimulacion
{
    // Aquí represento una mesa del restaurante (objeto reciclable del pool)
    public class Mesa
    {
        public int NumeroMesa { get; private set; } // Aquí guardo mi número de mesa
        public bool EstaOcupada { get; private set; } // Aquí guardo si estoy ocupada o no


        // Aquí me crean con mi número de mesa
        public Mesa(int numeroMesa)
        {
            NumeroMesa = numeroMesa; // Guardo el número que me asignan
            EstaOcupada = false; // Empiezo disponible
        }


        // Aquí me ocupan con un cliente
        public void Ocupar(string cliente)
        {
            EstaOcupada = true; // Me marco como ocupada
```

```
            Console.WriteLine($"  [MESA] Mesa {NumeroMesa} → OCUPADA
por {cliente}");

        }



        // Aquí me reinician para que pueda ser usada de nuevo

        public void Reiniciar()

        {

            EstaOcupada = false; // Me libero para el siguiente cliente

            Console.WriteLine($"  [MESA] Mesa {NumeroMesa} → LIMPIADA y
lista para reusar");

        }

    }

}
```

Captura



```
PS C:\Users\Alejandro\Desktop\Patrones\Unidad 2\Examen unidad 2\RestauranteSimulacion> dotnet run
[SINGLETON]  INSTANCIA ÚNICA CREADA - Gerente del restaurante

[RESTAURANTE] El gerente ha abierto el restaurante

[POOL] Pool 'Mesas' creado
[POOL-PUSH] ✓ Objeto agregado al pool 'Mesas' (Total disponibles: 1)
[POOL-PUSH] ✓ Objeto agregado al pool 'Mesas' (Total disponibles: 2)
[POOL-PUSH] ✓ Objeto agregado al pool 'Mesas' (Total disponibles: 3)
[POOL] Pool 'Meseros' creado
[POOL-PUSH] ✓ Objeto agregado al pool 'Meseros' (Total disponibles: 1)
[POOL-PUSH] ✓ Objeto agregado al pool 'Meseros' (Total disponibles: 2)
[POOL] Pool 'Pedidos' creado
[POOL-PUSH] ✓ Objeto agregado al pool 'Pedidos' (Total disponibles: 1)
[POOL-PUSH] ✓ Objeto agregado al pool 'Pedidos' (Total disponibles: 2)

============================================================
[CLIENTE] Frenkie de Jong llego al restaurante
  [SINGLETON] → Reutilizando instancia única existente
  [SINGLETON-HOST] Gerente recibe a Frenkie de Jong en la entrada

--- Solicitando recursos del pool ---
[POOL-POP] ← Objeto SACADO del pool 'Mesas' (Disponibles: 2 | En uso: 1)
  [SINGLETON-HOST] Gerente asigna Mesa 1 a Frenkie de Jong
[POOL-POP] ← Objeto SACADO del pool 'Meseros' (Disponibles: 1 | En uso: 1)
[POOL-POP] ← Objeto SACADO del pool 'Pedidos' (Disponibles: 1 | En uso: 1)

--- Usando los recursos ---
  [MESA] Mesa 1 → OCUPADA por Frenkie de Jong
  [MESERO] Gilberto → ATENDIENDO a Frenkie de Jong
  [PEDIDO] Pedido 1 → INICIADO para Frenkie de Jong
    + Hamburguesa ($150)
    +  Bacardi con coca  ($30)
  [PEDIDO] → FINALIZADO - Total: $180

--- Devolviendo recursos al pool ---
  [PEDIDO] Pedido 1 → LIMPIADO y listo para reusar
[POOL-RETURN] → Objeto DEVUELTO al pool 'Pedidos' (Disponibles: 2 | En uso: 0)
  [MESERO] Gilberto → LIBRE y listo para reusar
[POOL-RETURN] → Objeto DEVUELTO al pool 'Meseros' (Disponibles: 2 | En uso: 0)
  [MESA] Mesa 1 → LIMPIADA y lista para reusar
[POOL-RETURN] → Objeto DEVUELTO al pool 'Mesas' (Disponibles: 3 | En uso: 0)
  [SINGLETON-HOST] Gerente registra cliente #1 atendido
  [SINGLETON-HOST] Gerente despide a Frenkie de Jong
[CLIENTE] Frenkie de Jong se fue del restaurante

============================================================
[CLIENTE] Aitana Bombati llego al restaurante
  [SINGLETON] → Reutilizando instancia única existente
  [SINGLETON-HOST] Gerente recibe a Aitana Bombati en la entrada

--- Solicitando recursos del pool ---
[POOL-POP] ← Objeto SACADO del pool 'Mesas' (Disponibles: 2 | En uso: 1)
  [SINGLETON-HOST] Gerente asigna Mesa 2 a Aitana Bombati
[POOL-POP] ← Objeto SACADO del pool 'Meseros' (Disponibles: 1 | En uso: 1)
```

Despliegue completo porque no se ve en la captura

PS C:\Users\Alejandro\Desktop\Patrones\Unidad 2\Examen unidad 2\RestauranteSimulacion> dotnet run

C:\Program Files\dotnet\sdk\9.0.305\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.EolTargetFrameworks.targets(32,5): warning NETSDK1138: The target framework 'net6.0' is out of support and

will not receive security updates in the future. Please refer to https://aka.ms/dotnet-core-support for more information about the support policy.

C:\Program Files\dotnet\sdk\9.0.305\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.EolTargetFrameworks.targets(32,5): warning NETSDK1138: The target framework 'net6.0' is out of support and

will not receive security updates in the future. Please refer to https://aka.ms/dotnet-core-support for more information about the support policy.

C:\Program Files\dotnet\sdk\9.0.305\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.EolTargetFrameworks.targets(32,5): warning NETSDK1138: The target framework 'net6.0' is out of support and

will not receive security updates in the future. Please refer to https://aka.ms/dotnet-core-support for more information about the support policy.

[SINGLETON] INSTANCIA ÚNICA CREADA - Gerente del restaurante

[RESTAURANTE] El gerente ha abierto el restaurante

[POOL] Pool 'Mesas' creado

[POOL-PUSH] ✓ Objeto agregado al pool 'Mesas' (Total disponibles: 1)

[POOL-PUSH] ✓ Objeto agregado al pool 'Mesas' (Total disponibles: 2)

[POOL-PUSH] ✓ Objeto agregado al pool 'Mesas' (Total disponibles: 3)

[POOL] Pool 'Meseros' creado

[POOL-PUSH] ✓ Objeto agregado al pool 'Meseros' (Total disponibles: 1)

[POOL-PUSH] ✓ Objeto agregado al pool 'Meseros' (Total disponibles: 2)

[POOL] Pool 'Pedidos' creado

[POOL-PUSH] ✓ Objeto agregado al pool 'Pedidos' (Total disponibles: 1)

[POOL-PUSH] ✓ Objeto agregado al pool 'Pedidos' (Total disponibles: 2)

==========

[CLIENTE] Frenkie de Jong llego al restaurante

[SINGLETON] → Reutilizando instancia única existente

[SINGLETON-HOST] Gerente recibe a Frenkie de Jong en la entrada

--- Solicitando recursos del pool ---

[POOL-POP] ← Objeto SACADO del pool 'Mesas' (Disponibles: 2 | En uso: 1)

[SINGLETON-HOST] Gerente asigna Mesa 1 a Frenkie de Jong

[POOL-POP] ← Objeto SACADO del pool 'Meseros' (Disponibles: 1 | En uso: 1)

[POOL-POP] ← Objeto SACADO del pool 'Pedidos' (Disponibles: 1 | En uso: 1)

--- Usando los recursos ---

[MESA] Mesa 1 → OCUPADA por Frenkie de Jong

[MESERO] Gilberto → ATENDIENDO a Frenkie de Jong

[PEDIDO] Pedido 1 → INICIADO para Frenkie de Jong

+ Hamburguesa ($150)

+ Bacardi con coca ($30)

[PEDIDO] → FINALIZADO - Total: $180

--- Devolviendo recursos al pool ---

[PEDIDO] Pedido 1 → LIMPIADO y listo para reusar

[POOL-RETURN] → Objeto DEVUELTO al pool 'Pedidos' (Disponibles: 2 | En uso: 0)

[MESERO] Gilberto → LIBRE y listo para reusar

[POOL-RETURN] → Objeto DEVUELTO al pool 'Meseros' (Disponibles: 2 | En uso: 0)

[MESA] Mesa 1 → LIMPIADA y lista para reusar

[POOL-RETURN] → Objeto DEVUELTO al pool 'Mesas' (Disponibles: 3 | En uso: 0)

[SINGLETON-HOST] Gerente registra cliente #1 atendido

[SINGLETON-HOST] Gerente despide a Frenkie de Jong

[CLIENTE] Frenkie de Jong se fue del restaurante

==========

[CLIENTE] Aitana Bombati llego al restaurante

[SINGLETON] → Reutilizando instancia única existente

[SINGLETON-HOST] Gerente recibe a Aitana Bombati en la entrada

--- Solicitando recursos del pool ---

[POOL-POP] ← Objeto SACADO del pool 'Mesas' (Disponibles: 2 | En uso: 1)

[SINGLETON-HOST] Gerente asigna Mesa 2 a Aitana Bombati

[POOL-POP] ← Objeto SACADO del pool 'Meseros' (Disponibles: 1 | En uso: 1)

[POOL-POP] ← Objeto SACADO del pool 'Pedidos' (Disponibles: 1 | En uso: 1)

--- Usando los recursos ---

[MESA] Mesa 2 → OCUPADA por Aitana Bombati

[MESERO] Marcela → ATENDIENDO a Aitana Bombati

[PEDIDO] Pedido 2 → INICIADO para Aitana Bombati

+ Pizza ($200)

[PEDIDO] → FINALIZADO - Total: $200

--- Devolviendo recursos al pool ---

[PEDIDO] Pedido 2 → LIMPIADO y listo para reusar

[POOL-RETURN] → Objeto DEVUELTO al pool 'Pedidos' (Disponibles: 2 | En uso: 0)

[MESERO] Marcela → LIBRE y listo para reusar

[POOL-RETURN] → Objeto DEVUELTO al pool 'Meseros' (Disponibles: 2 | En uso: 0)

[MESA] Mesa 2 → LIMPIADA y lista para reusar

[POOL-RETURN] → Objeto DEVUELTO al pool 'Mesas' (Disponibles: 3 | En uso: 0)

[SINGLETON-HOST] Gerente registra cliente #2 atendido

[SINGLETON-HOST] Gerente despide a Aitana Bombati

[CLIENTE] Aitana Bombati se fue del restaurante

[Mesas] Disponibles: 3 | En uso: 0

[Meseros] Disponibles: 2 | En uso: 0

[Pedidos] Disponibles: 2 | En uso: 0

[RESTAURANTE] El gerente ha cerrado el restaurante - Clientes atendidos: 2

Conclusión

Patrón Singleton (GerenteRestaurante): sólo existe un gerente en todo el restaurante. Esto tiene sentido en la vida real ya que pues no puedes tener varios gerentes haciendo cosas diferentes al mismo tiempo. El gerente en este caso hace todo, recibe clientes, asigna mesas y lleva el conteo de cuántos clientes atendimos.

Patrón Object Pool (Mesa, Mesero, Pedido): Aquí reutilizo objetos en lugar de crearlos y destruirlos a cada rato. Es como en un restaurante real ya que las mesas, meseros y órdenes se reutilizan para cada cliente. Cuando termino de usar un objeto, lo "limpio" con el método Reiniciar() y lo devuelvo al pool para que esté listo para el siguiente cliente. Esto mejora el rendimiento porque evito estar creando y destruyendo objetos todo el tiempo.