

TECNOLOGICO NACIONAL DE MÉXICO INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:

Agosto - Diciembre 2025

MATERIA:

Patrones de diseño

TÍTULO ACTIVIDAD:

Examen Unidad 2

UNIDAD A EVALUAR:

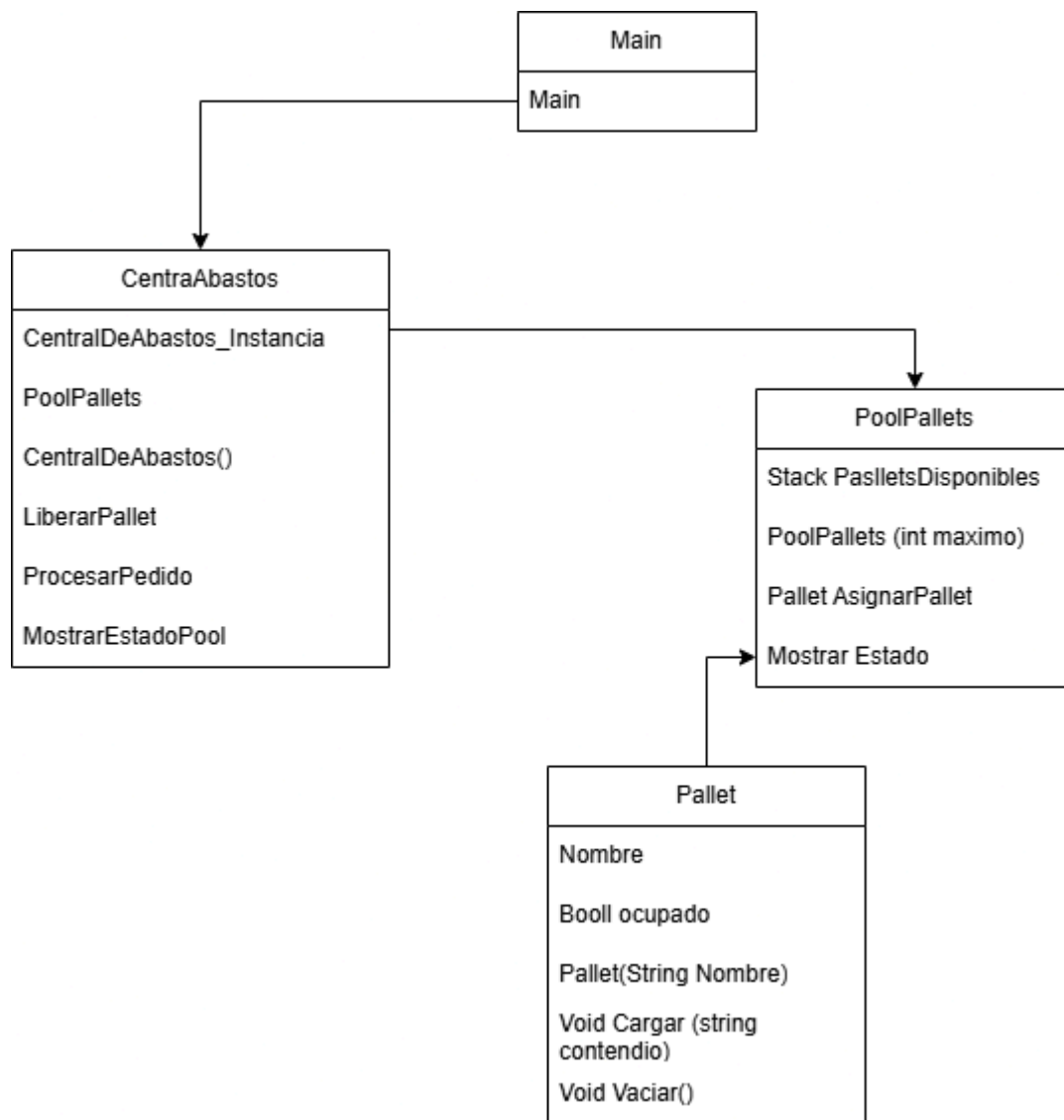
Unidad 2

NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

Soberanes Oregel Cristopher Daniel C20211465

NOMBRE DEL MAESTRO (A):

Maribel Guerrero Luis



```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace CentralDeAbastosConPallets
8  {
9
10     // Clase pallet
11
12     13 references
13     public class Pallet
14     {
15         6 references
16         public string Nombre { get; private set; }
17         3 references
18         public bool Ocupado { get; private set; }
19
20         1 reference
21         public Pallet(string nombre)
22         {
23             Nombre = nombre;
24             Ocupado = false;
25         }
26
27         1 reference
28         public void Cargar(string contenido)
29         {
30             Ocupado = true;
31             Console.WriteLine($" {Nombre} está transportando {contenido}...");
32         }
33
34         1 reference
35         public void Vaciar()
36         {
37             Ocupado = false;
38             Console.WriteLine($" {Nombre} ha sido vaciado y está libre.");
39         }
40     }
41 }

```

```

namespace CentralDeAbastosConPallets
{
    3 references
    public class PoolPallets
    {
        //PRIVADO PROFE, pegueme pero no me deje
        //controla directamente los objetos(Pallets) que se pueden prestar y devolver
        private readonly Stack<Pallet> palletsDisponibles = new Stack<Pallet>();
        private readonly int maximoPallets = 3;

        1 reference
        public PoolPallets(int maximo)
        {
            maximoPallets = maximo;
            for (int i = 1; i <= maximo; i++)
                palletsDisponibles.Push(new Pallet($"Pallet-{i}"));

            Console.WriteLine($" Pool inicializado con {maximo} pallets disponibles.\n");
        }

        1 reference
        public Pallet AsignarPallet()
        {
            if (palletsDisponibles.Count > 0)
            {
                Pallet p = palletsDisponibles.Pop();
                Console.WriteLine($" {p.Nombre} ha sido asignado.");
                Console.WriteLine($" Pallets disponibles: {palletsDisponibles.Count}\n");
                return p;
            }
            else
            {
                Console.WriteLine(" No hay pallets disponibles actualmente.\n");
                return null;
            }
        }
    }
}

```

```

1 reference
public void LiberarPallet(Pallet p)
{
    if (palletsDisponibles.Count < maximoPallets)
    {
        p.Vaciar();
        palletsDisponibles.Push(p);
        Console.WriteLine($" {p.Nombre} regresó al pool.");
        Console.WriteLine($" Pallets disponibles: {palletsDisponibles.Count}\n");
    }
    else
    {
        Console.WriteLine(" El pool ya está completo, no se puede devolver más pallets.\n");
    }
}

1 reference
public void MostrarEstado()
{
    Console.WriteLine($" Pallets disponibles actualmente: {palletsDisponibles.Count}\n");
}
}

```

```

namespace CentralDeAbastosConPallets
{
    10 references
    public class CentralDeAbastos
    {
        private static CentralDeAbastos _instance;
        private static readonly object _lock = new object();
        //Aqui se hace privada PROFE
        private PoolPallets poolPallets; // Object Pool interno

        // Constructor privado
        1 reference
        private CentralDeAbastos()
        {
            Console.WriteLine(" Central de abastos, que va a querer WERITO...\n");
            //Y aqui se usa la privada( si te ganan los nervios revisa la linea 16 pa saber porque se hace privada)
            poolPallets = new PoolPallets(3); // Pool con 3 pallets
        }

        // Propiedad Singleton
        3 references
        public static CentralDeAbastos Instance
        {
            get
            {
                if (_instance == null)
                {
                    lock (_lock)
                    {
                        if (_instance == null)
                            _instance = new CentralDeAbastos();
                    }
                }
                return _instance;
            }
        }
    }
}

```

```

// Método para procesar pedidos y asignar un pallet
4 references
public Pallet ProcesarPedido(string fruta)
{
    Console.WriteLine($" Pedido recibido: {fruta}");
    Pallet pallet = poolPallets.AsignarPallet();

    if (pallet != null)
    {
        pallet.Cargar(fruta);
        Console.WriteLine($" {fruta} está siendo transportada en {pallet.Nombre}.\n");
    }
    else
    {
        Console.WriteLine($" No hay pallets disponibles para transportar {fruta}. Espere un momento...\n");
    }

    return pallet;
}

// Método para liberar un pallet (regresarlo al pool)
1 reference
public void LiberarPallet(Pallet pallet)
{
    poolPallets.LiberarPallet(pallet);
}

// Método para verificar el Object Pool
1 reference
public void MostrarEstadoPool()
{
    poolPallets.MostrarEstado();
}
}

```

```

0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Console.Title = "Central de Abastos - Singleton + Object Pool";

        //Aquí es donde se genera el objeto y se hace el pedido
        CentralDeAbastos central1 = CentralDeAbastos.Instance;
        CentralDeAbastos central2 = CentralDeAbastos.Instance;
        CentralDeAbastos central3 = CentralDeAbastos.Instance;

        //Aquí se comprueba el pedido que cuaje como singleton
        Console.WriteLine(" COMPROBACIÓN DEL PATRÓN SINGLETON:");
        bool aqui_va_el_singleton_profe = ReferenceEquals(central1, central2) &&
            ReferenceEquals(central2, central3);

        Console.WriteLine($"¿Lleva singleton profe? {aqui_va_el_singleton_profe}\n");

        // Un menu así bien chikistrikis, pa que se vea en acción todo esta vaina
        Console.WriteLine("PEDIDOS ENTRANTES A LA CENTRAL DE ABASTOS\n" +
            "1. Manzanas\n" +
            "2. Bananas\n" +
            "3. Naranjas\n" +
            "4. Uvas\n" +
            "5. Liberar un pallet\n" +
            "6. Mostrar estado del pool\n" +
            "X. Salir\n");
    }
}

```

```

List<Pallet> palletsEnUso = new List<Pallet>();
string opcion;

do
{
    Console.Write("\nSeleccione una opción: ");
    opcion = Console.ReadLine();

    switch (opcion.ToLower())
    {
        case "1":
            palletsEnUso.Add(central1.ProcesarPedido("Manzanas"));
            break;
        case "2":
            palletsEnUso.Add(central1.ProcesarPedido("Bananas"));
            break;
        case "3":
            palletsEnUso.Add(central1.ProcesarPedido("Naranjas"));
            break;
        case "4":
            palletsEnUso.Add(central1.ProcesarPedido("Uvas"));
            break;
        case "5":
            if (palletsEnUso.Count > 0)
            {
                Pallet p = palletsEnUso[0];
                palletsEnUso.RemoveAt(0);
                central1.LiberarPallet(p);
            }
            else
            {
                Console.WriteLine(" ! No hay pallets en uso para liberar.");
            }
    }
}

```

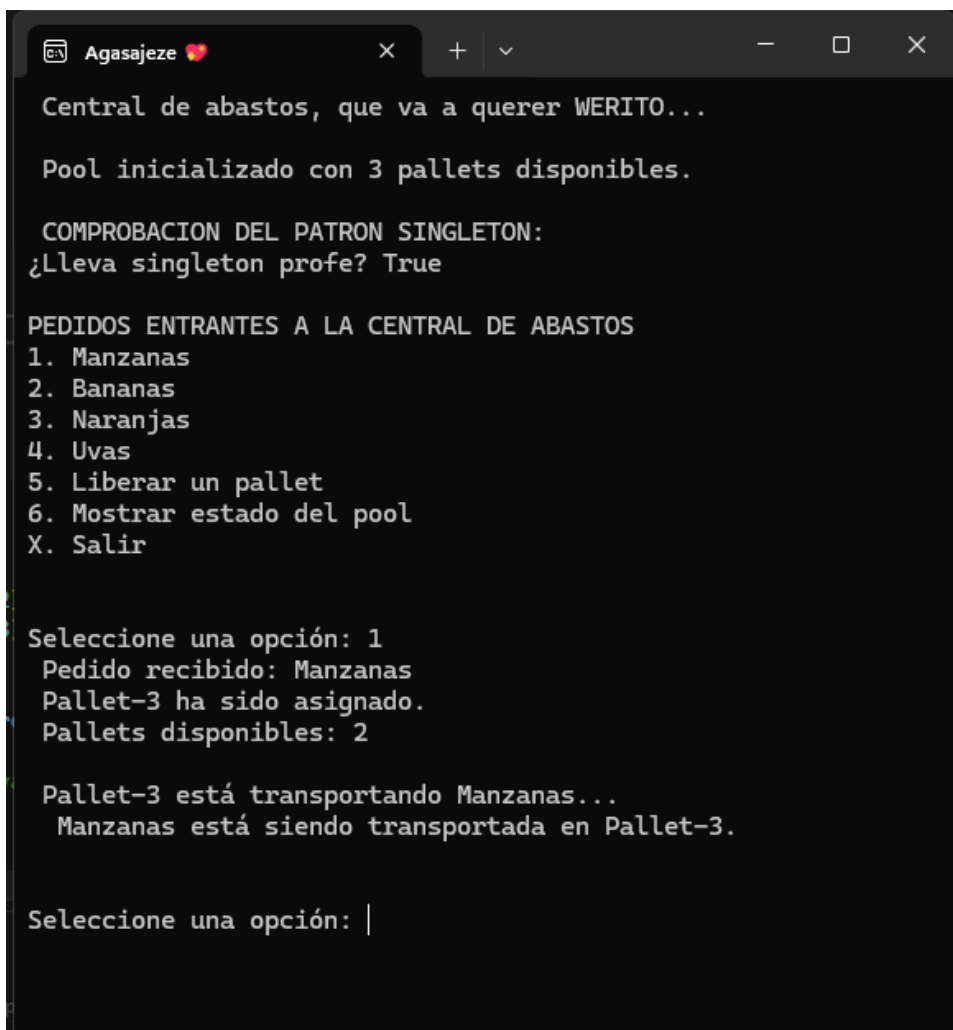
```

    }
    else
    {
        Console.WriteLine(" ! No hay pallets en uso para liberar.");
    }
    break;
case "6":
    central1.MostrarEstadoPool();
    break;
case "x":
    Console.WriteLine("\n Cerrando pedidos...");
    break;
default:
    Console.WriteLine(" Opción no válida.");
    break;
}

} while (opcion.ToLower() != "x");

Console.WriteLine("\nPresione cualquier tecla para salir...");
Console.ReadKey();
}
}

```



```

Central de abastos, que va a querer WERITO...

Pool inicializado con 3 pallets disponibles.

COMPROBACION DEL PATRON SINGLETON:
¿Lleva singleton profe? True

PEDIDOS ENTRANTES A LA CENTRAL DE ABASTOS
1. Manzanas
2. Bananas
3. Naranjas
4. Uvas
5. Liberar un pallet
6. Mostrar estado del pool
X. Salir

Seleccione una opción: 1
Pedido recibido: Manzanas
Pallet-3 ha sido asignado.
Pallets disponibles: 2

Pallet-3 está transportando Manzanas...
Manzanas está siendo transportada en Pallet-3.

Seleccione una opción: |

```

Conclusion:

Combinar Object Pool con Singleton es una forma muy práctica de manejar recursos de manera eficiente. El pool está chido ya que combina con el Singleton y asegura que no se creen múltiples instancias innecesarias y que todos los objetos reutilizables están organizados desde un solo punto. Como me dio dolores de cabeza combinarlos.