

TECNOLÓGICO 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

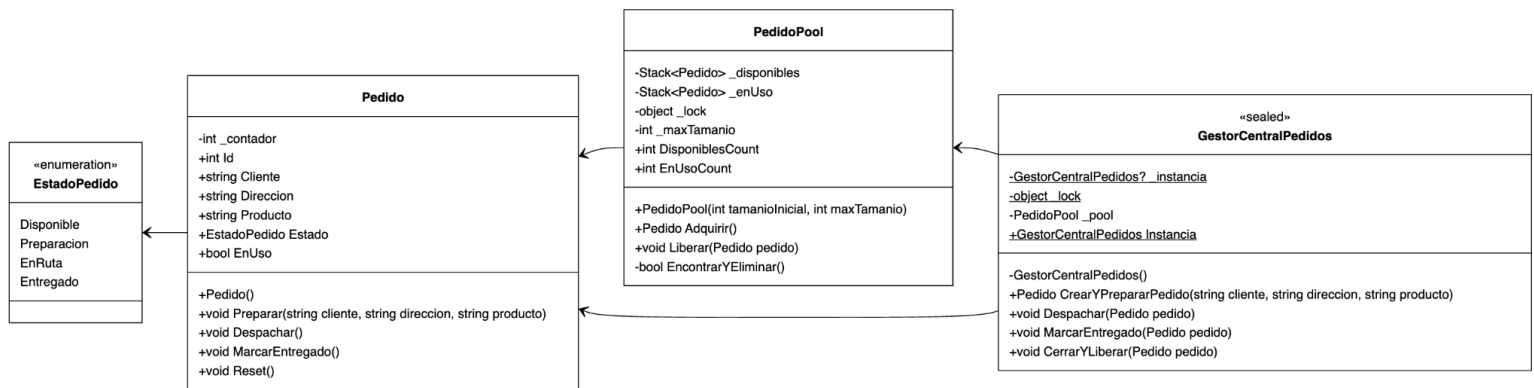
NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

Vizuet Acosta Andre 21212372

NOMBRE DEL MAESTRO (A):

Maribel Guerrero Luis

UML



EstadoPedido

```

8 references
1  enum EstadoPedido
2  {
3      1 reference
4      Disponible,
5      2 references
6      Preparacion,
7      2 references
8      EnRuta,
9      2 references
10     Entregado
11 }
  
```

Pedido

```
18 references
1  class Pedido
2  {
3      3 references
4      public int Id { get; private set; }
5      5 references
6      public string Cliente { get; private set; } = "";
7      5 references
8      public string Direccion { get; private set; } = "";
9      5 references
10     public string Producto { get; private set; } = "";
11     10 references
12     public EstadoPedido Estado { get; private set; }
13     2 references
14     public bool EnUso { get; set; }
15
16     1 reference
17     private static int _contador = 1;
18     2 references
19     public Pedido()
20     {
21         Id = _contador++;
22         Reset();
23     }
24
25     1 reference
26     public void Preparar(string cliente, string direccion, string producto)
27     {
28         Cliente = cliente;
29         Direccion = direccion;
30         Producto = producto;
31         Estado = EstadoPedido.Preparacion;
32         Console.WriteLine($"Pedido en {this.Estado}: {this.Cliente}, {this.Direccion}, {this.Producto}");
33     }
34 }
```

```

25
26 1 reference
27 public void Despachar()
28 {
29     if (Estado != EstadoPedido.Preparacion)
30         throw new InvalidOperationException("Solo se puede despachar un pedido en preparación.");
31     Estado = EstadoPedido.EnRuta;
32     Console.WriteLine($"Pedido {this.Estado}: {this.Cliente}, {this.Direccion}, {this.Producto}");
33 }
34
35 1 reference
36 public void MarcarEntregado()
37 {
38     if (Estado != EstadoPedido.EnRuta)
39         throw new InvalidOperationException("Solo se puede marcar entregado un pedido en ruta.");
40     Estado = EstadoPedido.Entregado;
41     Console.WriteLine($"Pedido {this.Estado}: {this.Cliente}, {this.Direccion}, {this.Producto}");
42 }
43
44 2 references
45 public void Reset()
46 {
47     Cliente = "";
48     Direccion = "";
49     Producto = "";
50     Estado = EstadoPedido.Disponible;
51     EnUso = false;
52 }
53 }

```

Singleton

```
6 references
1 sealed class GestorCentralPedidos
2
3     4 references
4     private static GestorCentralPedidos? _instancia = null;
5     1 reference
6     private static readonly object _lock = new object();
7
8     1 reference
9     public static GestorCentralPedidos Instancia
10    {
11        get
12        {
13            if (_instancia == null)
14            {
15                lock (_lock)
16                {
17                    if (_instancia == null)
18                        _instancia = new GestorCentralPedidos();
19                }
20            }
21            return _instancia;
22        }
23    }
24
25    3 references
26    private readonly PedidoPool _pool;
27
28    1 reference
29    private GestorCentralPedidos()
30    {
31        _pool = new PedidoPool(tamanoInicial: 2, maxTamano: 100);
32    }
33
34    2 references
35    public Pedido CrearYPrepararPedido(string cliente, string direccion, string producto)
36    {
37        Pedido pedido = _pool.Adquirir();
38        pedido.Preparar(cliente, direccion, producto);
39        return pedido;
40    }
41
42    2 references
43    public void Despachar(Pedido pedido) => pedido.Despachar();
44
45    2 references
46    public void MarcarEntregado(Pedido pedido) => pedido.MarcasEntregado();
47
48    2 references
49    public void CerrarYLiberar(Pedido pedido)
50    {
51        if (pedido.Estado != EstadoPedido.Entregado)
52            throw new InvalidOperationException("No se puede liberar un pedido que no ha sido entregado.");
53        _pool.Liberar(pedido);
54    }
55 }
```

ObjectPool

```
3 references
1 class PedidoPool
2 {
3     4 references
4     private readonly Stack<Pedido> _disponibles = new Stack<Pedido>();
5     4 references
6     private readonly Stack<Pedido> _enUso = new Stack<Pedido>();
7     4 references
8     private readonly object _lock = new object();
9     2 references
10    private readonly int _maxTamano;
11
12    2 references
13    public int DisponiblesCount { get { lock (_lock) return _disponibles.Count; } }
14    2 references
15    public int EnUsoCount { get { lock (_lock) return _enUso.Count; } }
16
17    1 reference
18    public PedidoPool(int tamanoInicial, int maxTamano)
19    {
20        _maxTamano = maxTamano;
21        for (int i = 0; i < tamanoInicial; i++)
22            _disponibles.Push(new Pedido());
23    }
24 }
```

```

17 1 reference
18 public Pedido Adquirir()
19 {
20     lock (_lock)
21     {
22         Pedido p;
23         if (DisponiblesCount > 0)
24             p = _disponibles.Pop();
25         else
26         {
27             int totalActual = DisponiblesCount + EnUsoCount;
28             if (totalActual ≥ _maxTamano)
29                 throw new InvalidOperationException("El pool alcanzó su tamaño máximo.");
30             p = new Pedido();
31         }
32         p.EnUso = true;
33         _enUso.Push(p);
34         return p;
35     }
36 }
37
38 1 reference
39 public void Liberar(Pedido pedido)
40 {
41     if (pedido == null) return;
42     lock (_lock)
43     {
44         bool EncontrarYEliminar()
45         {
46             if (EnUsoCount == 0) return false;
47             Pedido top = _enUso.Pop();
48             if (top == pedido)
49             {
50                 return true;
51             }
52             bool encontradoEnResto = EncontrarYEliminar();
53             _enUso.Push(top);
54             return encontradoEnResto;
55         }
56         pedido.Reset();
57         _disponibles.Push(pedido);
58     }
59 }
60 }

```

Main

```
1 using System;
2 using System.Collections.Generic;
3 class Program
4 {
5     static void Main(string[] args)
6     {
7         GestorCentralPedidos gestor = GestorCentralPedidos.Instance;
8
9         Console.WriteLine("== Pedido 1 ==");
10        Pedido pedido1 = gestor.CrearYPrepararPedido(
11            cliente: "Edgar Manuel",
12            direccion: "Sta. Ximena 49",
13            producto: "Azzaro The Most Wanted EDP Intense 100ml"
14        );
15        gestor.Despachar(pedido1);
16        gestor.MarcarEntregado(pedido1);
17        gestor.CerrarYLiberar(pedido1);
18
19        Console.WriteLine("\n== Pedido 2 ==");
20        Pedido pedido2 = gestor.CrearYPrepararPedido(
21            cliente: "Andre Vizuet",
22            direccion: "Villas del Real",
23            producto: "Jean Paul Gaultier Le Male Elixir 125ml"
24        );
25        gestor.Despachar(pedido2);
26        gestor.MarcarEntregado(pedido2);
27        gestor.CerrarYLiberar(pedido2);
28
29        Console.WriteLine($"Comprobación de reutilización: \n Pedido 1: {pedido1.Id}\n Pedido 2: {pedido2.Id}");
30        Console.WriteLine($"Comprobacion de referencia: {ReferenceEquals(pedido1, pedido2)}");
31    }
32 }
```

Ejecución

```
ExamenUnidad2_Patrones_VizuetAcostaAndre/src on  main [!] via .NET v9.0.102 net9.0
➤ dotnet run
== Pedido 1 ==
Pedido en Preparacion: Edgar Manuel, Sta. Ximena 49, Azzaro The Most Wanted EDP Intense 100ml
Pedido EnRuta: Edgar Manuel, Sta. Ximena 49, Azzaro The Most Wanted EDP Intense 100ml
Pedido Entregado: Edgar Manuel, Sta. Ximena 49, Azzaro The Most Wanted EDP Intense 100ml

== Pedido 2 ==
Pedido en Preparacion: Andre Vizuet, Villas del Real, Jean Paul Gaultier Le Male Elixir 125ml
Pedido EnRuta: Andre Vizuet, Villas del Real, Jean Paul Gaultier Le Male Elixir 125ml
Pedido Entregado: Andre Vizuet, Villas del Real, Jean Paul Gaultier Le Male Elixir 125ml

Comprobación de reutilización:
Pedido 1: 2
Pedido 2: 2
Comprobacion de referencia: True
```

Conclusión

Con singleton centralizamos todo en una sola instancia que va a gestionar los recursos y optimiza el uso de objetos mediante la reutilización con el patrón objectpool. Esta combinación va a mejorar el rendimiento, evita los duplicados de instancias, el código es más fácil de mantener y eficiente.