TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

SEMESTRE:
Agosto - Diciembre 2025

MATERIA:
Patrones de diseño

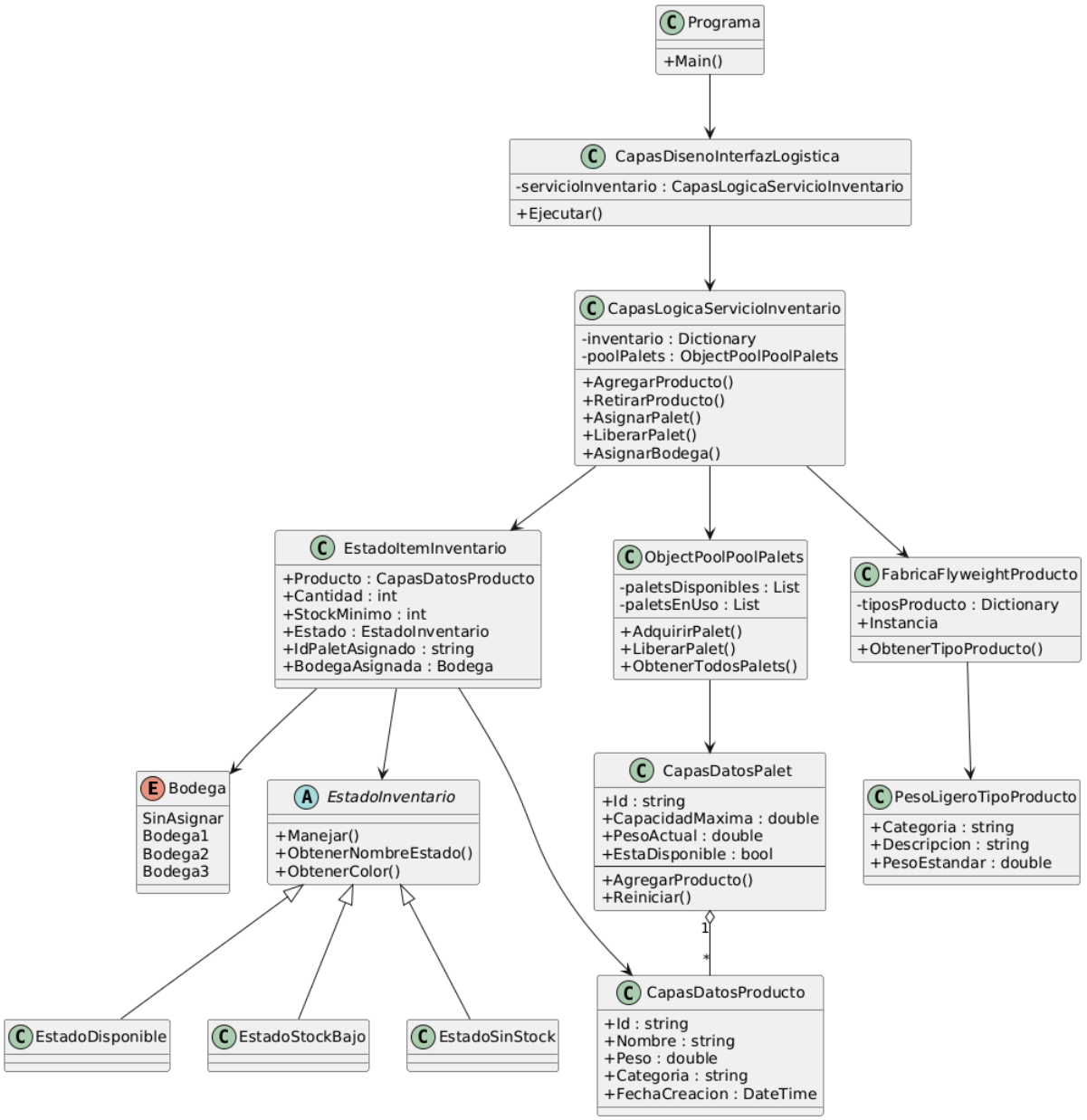
TÍTULO ACTIVIDAD:
Examen Unidad 4 y 5

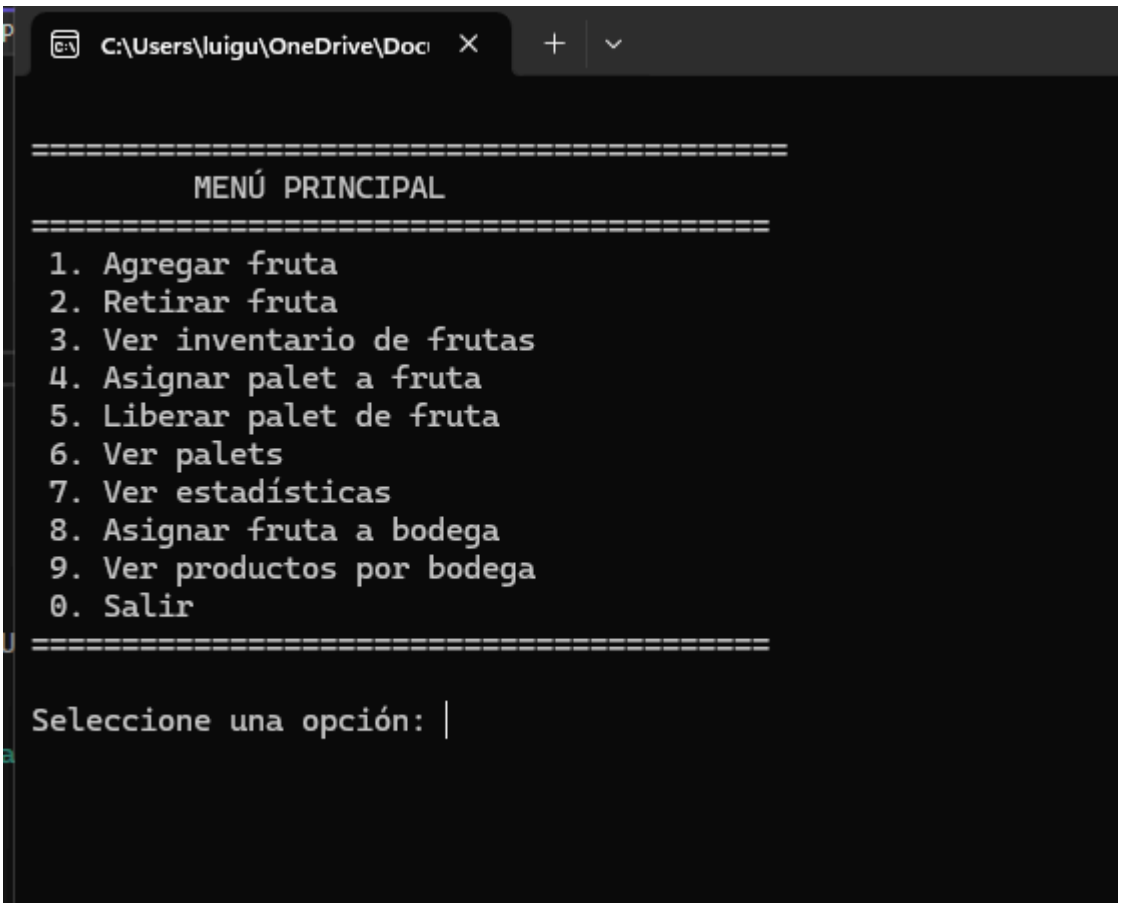
UNIDAD A EVALUAR:
Unidad 4 y 5

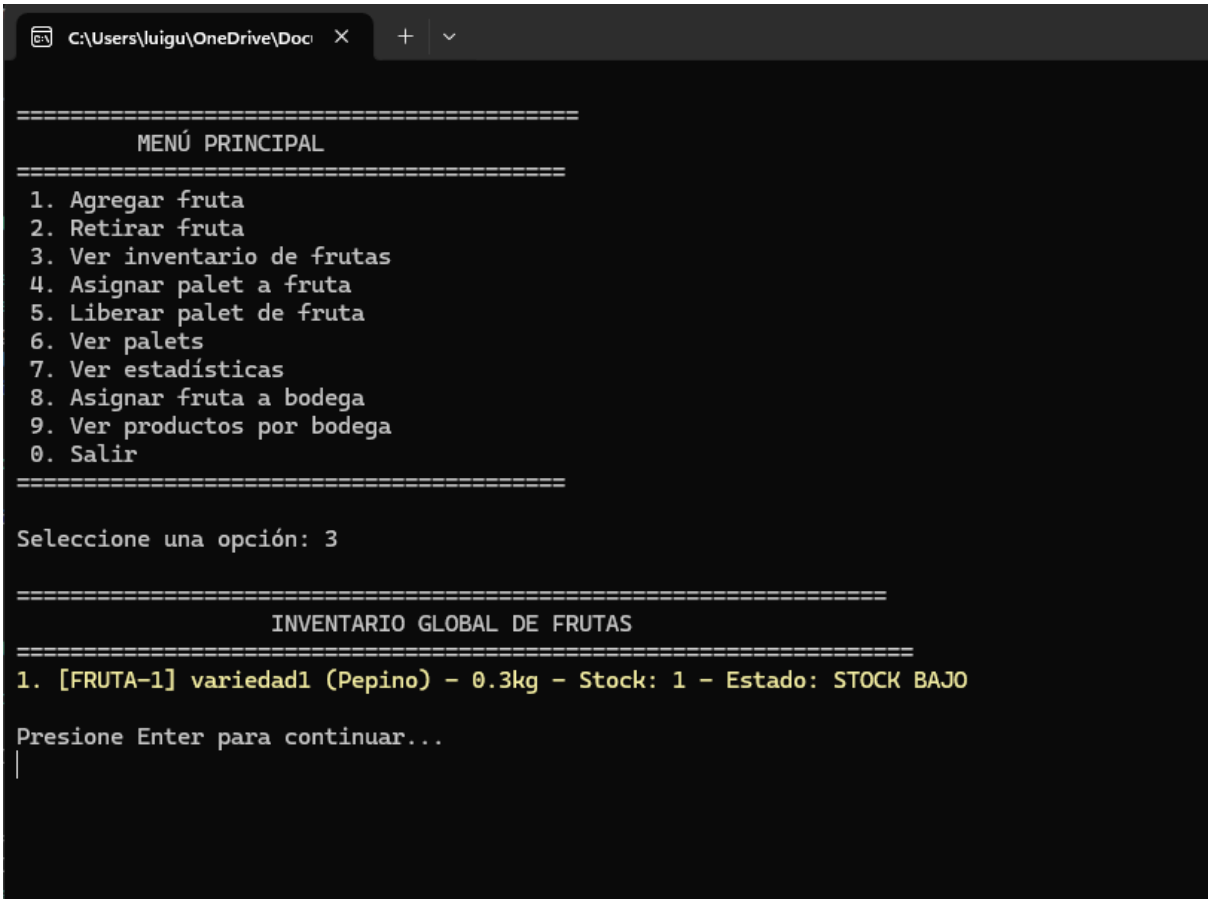
NOMBRE Y NÚMERO DE CONTROL DEL ALUMNO:

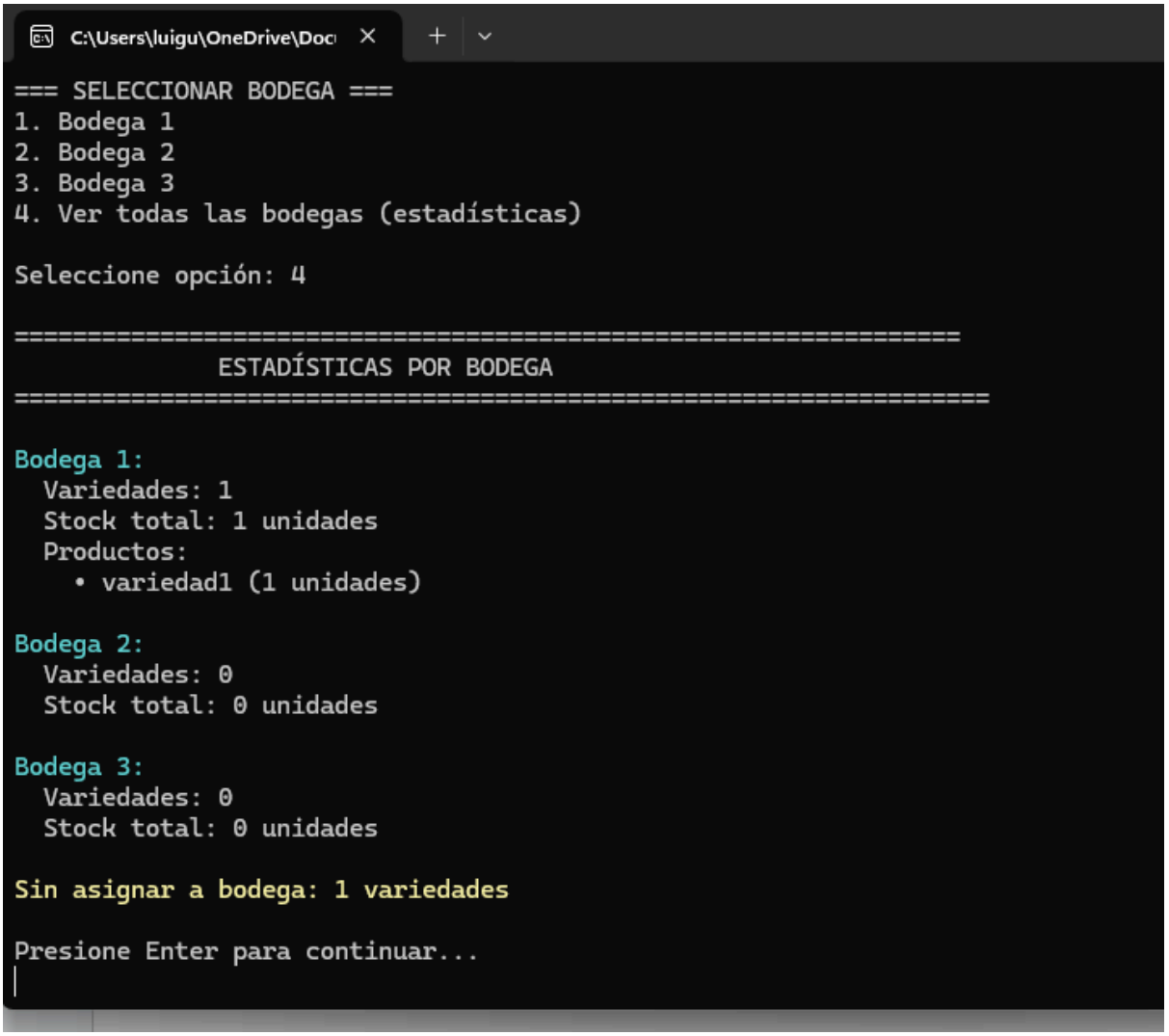
Soberanes Oregel Cristopher Daniel C20211465

NOMBRE DEL MAESTRO (A):
Maribel Guerrero Luis


Programa
+Main()
CapasDisenoInterfazLogistica
-servicioInventario : CapasLogicaServicioInventario
+Ejecutar()
CapasLogicaServicioInventario
-inventario : Dictionary
-poolPalets : ObjectPoolPoolPalets
+AgregarProducto()
+RetirarProducto()
+AsignarPalet()
+LiberarPalet()
+AsignarBodega()
EstadoItemInventario
+Producto : CapasDatosProducto
+Cantidad : int
+StockMinimo : int
+Estado : EstadoInventario
+IdPaletAsignado : string
+BodegaAsignada : Bodega
ObjectPoolPoolPalets
-paletsDisponibles : List
-paletsEnUso : List
+AdquirirPalet()
+LiberarPalet()
+ObtenerTodosPalets()
FabricaFlyweightProducto
-tiposProducto : Dictionary
+Instancia
+ObtenerTipoProducto()
Bodega
SinAsignar
Bodega1
Bodega2
Bodega3
EstadoInventario
+Manejar()
+ObtenerNombreEstado()
+ObtenerColor()
CapasDatosPalet
+Id : string
+CapacidadMaxima : double
+PesoActual : double
+EstaDisponible : bool
+AgregarProducto()
+Reiniciar()
PesoLigeroTipoProducto
+Categoria : string
+Descripcion : string
+PesoEstandar : double
1
*
EstadoDisponible
EstadoStockBajo
EstadoSinStock
CapasDatosProducto
+Id : string
+Nombre : string
+Peso : double
+Categoria : string
+FechaCreacion : DateTime


C:\Users\luigu\OneDrive\Doc
MENÚ PRINCIPAL
1. Agregar fruta
2. Retirar fruta
3. Ver inventario de frutas
4. Asignar palet a fruta
5. Liberar palet de fruta
6. Ver palets
7. Ver estadísticas
8. Asignar fruta a bodega
9. Ver productos por bodega
0. Salir
Seleccione una opción:


C:\Users\luigu\OneDrive\Doc
MENÚ PRINCIPAL
1. Agregar fruta
2. Retirar fruta
3. Ver inventario de frutas
4. Asignar palet a fruta
5. Liberar palet de fruta
6. Ver palets
7. Ver estadísticas
8. Asignar fruta a bodega
9. Ver productos por bodega
0. Salir
Seleccione una opción: 3
INVENTARIO GLOBAL DE FRUTAS
1. [FRUTA-1] variedad1 (Pepino) - 0.3kg - Stock: 1 - Estado: STOCK BAJO
Presione Enter para continuar...




```cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public enum Bodega
    {
        SinAsignar = 0,
        Bodega1 = 1,
        Bodega2 = 2,
        Bodega3 = 3
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class CapasDatosPalet
    {
        public string Id { get; set; }
        public double CapacidadMaxima { get; set; }
        public double PesoActual { get; set; }
        public List<CapasDatosProducto> Productos { get; set; }
        public bool EstaDisponible { get; set; }

        public CapasDatosPalet(string id, double capacidadMaxima)
        {
            Id = id;
            CapacidadMaxima = capacidadMaxima;
            PesoActual = 0;
            Productos = new List<CapasDatosProducto>();
            EstaDisponible = true;
        }

        public bool AgregarProducto(CapasDatosProducto producto)
        {
            if (PesoActual + producto.Peso <= CapacidadMaxima)
            {
                Productos.Add(producto);
                PesoActual += producto.Peso;
                return true;
            }
            return false;
        }

        public void Reiniciar()
        {
            Productos.Clear();
            PesoActual = 0;
```

```
            EstaDisponible = true;
        }

        public override string ToString()
        {
            return $"{Id} - {PesoActual}/{CapacidadMaxima}kg -
{Productos.Count} productos - {(EstaDisponible ? "Disponible" :
"En uso")}";
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class CapasDatosProducto
    {
        public string Id { get; set; }
        public string Nombre { get; set; }
        public double Peso { get; set; }
        public string Categoria { get; set; }
        public DateTime FechaCreacion { get; set; }

        public CapasDatosProducto(string id, string nombre, double
peso, string categoria)
        {
            Id = id;
            Nombre = nombre;
            Peso = peso;
            Categoria = categoria;
            FechaCreacion = DateTime.Now;
        }

        public override string ToString()
        {
            return $"[{Id}] {Nombre} ({Categoria}) - {Peso}kg";
```

```
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class CapasDisenoInterfazLogistica
    {
        private CapasLogicaServicioInventario servicioInventario =
new CapasLogicaServicioInventario();

        public void Ejecutar()
        {

Console.WriteLine("================================================
====================");
            Console.WriteLine("   SISTEMA DE GESTIÓN LOGÍSTICA DE
FRUTAS                          ");
            Console.WriteLine("   Frutas disponibles: Pepino,
Jicama, Mango, Fresa             ");
            Console.WriteLine("   Bodegas disponibles: Bodega 1,
2, 3                            ");
            Console.WriteLine("   Patrones Implementados:
");
            Console.WriteLine("   • Object Pool (Creacional) -
PoolPalets [MÁXIMO: 15]        ");
            Console.WriteLine("   • Flyweight (Estructural) -
FabricaFlyweightProducto        ");
            Console.WriteLine("   • State (Comportamiento) -
EstadoInventario                ");
            Console.WriteLine("   • Capas (Arquitectura) - 3 capas
separadas                     ");

Console.WriteLine("================================================
====================\n");
```

```
bool salir = false;
while (!salir)
{
    MostrarMenu();
    string opcion = Console.ReadLine();

    switch (opcion)
    {
        case "1":
            MenuAgregarProducto();
            break;
        case "2":
            MenuRetirarProducto();
            break;
        case "3":
            MostrarInventario();
            break;
        case "4":
            MenuAsignarPalet();
            break;
        case "5":
            MenuLiberarPalet();
            break;
        case "6":
            MostrarPalets();
            break;
        case "7":
            servicioInventario.MostrarEstadisticas();
            break;
        case "8":
            MenuAsignarBodega();
            break;
        case "9":
            MenuVerPorBodega();
            break;
        case "0":
            salir = true;
            Console.WriteLine("\n¡Hasta luego!");
            break;
        default:
            Console.WriteLine("Opción no válida");
```

```
                        break;
                }

                if (!salir)
                {
                    Console.WriteLine("\nPresione Enter para
continuar...");
                    Console.ReadLine();
                }
            }
        }

        private void MostrarMenu()
        {
            Console.Clear();
            int paletsRestantes =
servicioInventario.ObtenerPaletsRestantes();

Console.WriteLine("\n========================================");
            Console.WriteLine("         MENÚ PRINCIPAL
");

Console.WriteLine("========================================");
            Console.WriteLine(" 1. Agregar fruta
");
            Console.WriteLine(" 2. Retirar fruta
");
            Console.WriteLine(" 3. Ver inventario de frutas
");
            Console.WriteLine(" 4. Asignar palet a fruta
");
            Console.WriteLine(" 5. Liberar palet de fruta
");
            Console.WriteLine(" 6. Ver palets
");
            Console.WriteLine(" 7. Ver estadísticas
");
            Console.WriteLine(" 8. Asignar fruta a bodega
");
            Console.WriteLine(" 9. Ver productos por bodega
");
            Console.WriteLine(" 0. Salir
```

```
");

Console.WriteLine("========================================");
            Console.ForegroundColor = paletsRestantes > 0 ?
ConsoleColor.Green : ConsoleColor.Red;
            Console.ResetColor();
            Console.Write("\nSeleccione una opción: ");
        }

        private void MenuAgregarProducto()
        {
            Console.WriteLine("\n=== AGREGAR FRUTA ===");
            Console.WriteLine("Tipos de fruta: Pepino, Jicama,
Mango, Fresa");
            Console.Write("Tipo de fruta: ");
            string categoria = Console.ReadLine();

            Console.Write("Nombre/Variedad: ");
            string nombre = Console.ReadLine();

            Console.Write("Peso por unidad (kg, 0 para peso
estándar): ");
            double peso = double.Parse(Console.ReadLine() ?? "0");

            Console.Write("Cantidad de unidades: ");
            int cantidad = int.Parse(Console.ReadLine() ?? "1");

            servicioInventario.AgregarProducto(categoria, nombre,
peso, cantidad);
        }

        private void MenuRetirarProducto()
        {
            MostrarInventario();
            Console.Write("\nIngrese el número de fruta: ");
            int indice = int.Parse(Console.ReadLine() ?? "0") - 1;

            var inventario =
servicioInventario.ObtenerInventario();
            if (indice >= 0 && indice < inventario.Count)
            {
                Console.Write("Cantidad de unidades a retirar: ");
```

```
                int cantidad = int.Parse(Console.ReadLine() ??
"1");

                var claveProducto =
$"{inventario[indice].Producto.Categoria}-{inventario[indice].Prod
ucto.Nombre}";
                servicioInventario.RetirarProducto(claveProducto,
cantidad);
            }
        }

        private void MostrarInventario()
        {

Console.WriteLine("\n=============================================
====================");
            Console.WriteLine("                    INVENTARIO
GLOBAL DE FRUTAS                    ");

Console.WriteLine("=============================================
====================");

            var inventario =
servicioInventario.ObtenerInventario();
            if (inventario.Count == 0)
            {
                Console.WriteLine("El inventario de frutas está
vacío.");
                return;
            }

            for (int i = 0; i < inventario.Count; i++)
            {
                var item = inventario[i];
                Console.ForegroundColor =
item.Estado.ObtenerColor();
                Console.WriteLine($"{i + 1}. {item}");
                Console.ResetColor();
            }
        }

        private void MenuAsignarPalet()
```

```
        {
            MostrarInventario();
            Console.Write("\nIngrese el número de fruta: ");
            int indice = int.Parse(Console.ReadLine() ?? "0") - 1;

            var inventario =
servicioInventario.ObtenerInventario();
            if (indice >= 0 && indice < inventario.Count)
            {
                var claveProducto =
$"{inventario[indice].Producto.Categoria}-{inventario[indice].Prod
ucto.Nombre}";
                servicioInventario.AsignarPalet(claveProducto);
            }
        }

        private void MenuLiberarPalet()
        {
            MostrarInventario();
            Console.Write("\nIngrese el número de fruta: ");
            int indice = int.Parse(Console.ReadLine() ?? "0") - 1;

            var inventario =
servicioInventario.ObtenerInventario();
            if (indice >= 0 && indice < inventario.Count)
            {
                var claveProducto =
$"{inventario[indice].Producto.Categoria}-{inventario[indice].Prod
ucto.Nombre}";
                servicioInventario.LiberarPalet(claveProducto);
            }
        }

        private void MostrarPalets()
        {

Console.WriteLine("\n=============================================
==================");
            Console.WriteLine("         PALETS REUTILIZABLES
(Object Pool - Máx: 15)          ");

Console.WriteLine("=============================================
```

```
====================");

            var palets = servicioInventario.ObtenerPalets();
            if (palets.Count == 0)
            {
                Console.WriteLine("No hay palets creados aún.");
                return;
            }

            foreach (var palet in palets)
            {
                Console.ForegroundColor = palet.EstaDisponible ?
ConsoleColor.Green : ConsoleColor.Cyan;
                Console.WriteLine(palet);
                Console.ResetColor();

                if (palet.Productos.Count > 0)
                {
                    Console.WriteLine("  Productos en el palet:");
                    foreach (var producto in palet.Productos)
                    {
                        Console.WriteLine($"    • {producto}");
                    }
                }
            }
        }


        private void MenuAsignarBodega()
        {
            MostrarInventario();
            Console.Write("\nIngrese el número de fruta: ");
            int indice = int.Parse(Console.ReadLine() ?? "0") - 1;

            var inventario =
servicioInventario.ObtenerInventario();
            if (indice >= 0 && indice < inventario.Count)
            {
                Console.WriteLine("\n=== BODEGAS DISPONIBLES
===");
                Console.WriteLine("1. Bodega 1");
                Console.WriteLine("2. Bodega 2");
```

```
                Console.WriteLine("3. Bodega 3");
                Console.Write("\nSeleccione bodega: ");
                int numBodega = int.Parse(Console.ReadLine() ??
"0");

                if (numBodega >= 1 && numBodega <= 3)
                {
                    var claveProducto =
$"{inventario[indice].Producto.Categoria}-{inventario[indice].Prod
ucto.Nombre}";

servicioInventario.AsignarBodega(claveProducto,
(Bodega)numBodega);
                }
                else
                {
                    Console.WriteLine("Bodega no válida");
                }
            }
        }

        private void MenuVerPorBodega()
        {
            Console.WriteLine("\n=== SELECCIONAR BODEGA ===");
            Console.WriteLine("1. Bodega 1");
            Console.WriteLine("2. Bodega 2");
            Console.WriteLine("3. Bodega 3");
            Console.WriteLine("4. Ver todas las bodegas
(estadísticas)");
            Console.Write("\nSeleccione opción: ");
            int opcion = int.Parse(Console.ReadLine() ?? "0");

            if (opcion >= 1 && opcion <= 3)
            {
                var bodega = (Bodega)opcion;
                var productos =
servicioInventario.ObtenerProductosPorBodega(bodega);


Console.WriteLine($"\n=========================================
====================");
                Console.WriteLine($"              PRODUCTOS EN
```

```
BODEGA {opcion}                                ");

Console.WriteLine("==========================================================
====================");

                if (productos.Count == 0)
                {
                    Console.WriteLine("No hay productos en esta
bodega.");
                    return;
                }

                for (int i = 0; i < productos.Count; i++)
                {
                    var item = productos[i];
                    Console.ForegroundColor =
item.Estado.ObtenerColor();
                    Console.WriteLine($"{i + 1}. {item}");
                    Console.ResetColor();
                }
            }
            else if (opcion == 4)
            {
                servicioInventario.MostrarEstadisticasBodegas();
            }
            else
            {
                Console.WriteLine("Opción no válida");
            }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class CapasLogicaServicioInventario
```

```
    {
        private Dictionary<string, EstadoItemInventario>
inventario = new Dictionary<string, EstadoItemInventario>();
        private ObjectPoolPoolPalets poolPalets = new
ObjectPoolPoolPalets();
        private int siguienteIdProducto = 1;


        public void AgregarProducto(string categoria, string
nombre, double peso, int cantidad)
        {
            var tipoProducto =
FabricaFlyweightProducto.Instancia.ObtenerTipoProducto(categoria);
            var claveProducto = $"{categoria}-{nombre}";

            if (inventario.ContainsKey(claveProducto))
            {
                inventario[claveProducto].AgregarStock(cantidad);
                Console.WriteLine($"[INVENTARIO] Agregado stock a
{nombre}: +{cantidad} unidades");
            }
            else
            {
                var producto = new CapasDatosProducto(
                    $"FRUTA-{siguienteIdProducto++}",
                    nombre,
                    peso > 0 ? peso : tipoProducto.PesoEstandar,
                    categoria
                );
                var item = new EstadoItemInventario(producto,
cantidad);
                inventario[claveProducto] = item;
                Console.WriteLine($"[INVENTARIO] Nueva fruta
agregada: {producto.Nombre} ({categoria})");
            }
        }

        public bool RetirarProducto(string claveProducto, int
cantidad)
        {
            if (inventario.ContainsKey(claveProducto))
            {
```

```
                if
(inventario[claveProducto].RetirarStock(cantidad))
                {
                    Console.WriteLine($"[INVENTARIO] Retirado:
{cantidad} unidades");
                    return true;
                }
                else
                {
                    Console.WriteLine($"[INVENTARIO] Stock
insuficiente");
                    return false;
                }
            }
            Console.WriteLine($"[INVENTARIO] Producto no
encontrado");
            return false;
        }

        public bool AsignarPalet(string claveProducto)
        {
            if (inventario.ContainsKey(claveProducto))
            {
                var item = inventario[claveProducto];
                if (item.IdPaletAsignado == null)
                {
                    var palet = poolPalets.AdquirirPalet();
                    if (palet == null)
                    {
                        Console.WriteLine($"[LOGÍSTICA] No se pudo
asignar palet: límite alcanzado");
                        return false;
                    }

                    if (palet.AgregarProducto(item.Producto))
                    {
                        item.IdPaletAsignado = palet.Id;
                        Console.WriteLine($"[LOGÍSTICA] Palet
{palet.Id} asignado a {item.Producto.Nombre}");
                        return true;
                    }
                }
```

```
                else
                {
                    Console.WriteLine($"[LOGÍSTICA] El producto ya
tiene un palet asignado");
                }
            }
            return false;
        }

        public bool LiberarPalet(string claveProducto)
        {
            if (inventario.ContainsKey(claveProducto))
            {
                var item = inventario[claveProducto];
                if (item.IdPaletAsignado != null)
                {
                    var palet =
poolPalets.ObtenerTodosPalets().Find(p => p.Id ==
item.IdPaletAsignado);
                    if (palet != null)
                    {
                        poolPalets.LiberarPalet(palet);
                        item.IdPaletAsignado = null;
                        Console.WriteLine($"[LOGÍSTICA] Palet
liberado de {item.Producto.Nombre}");
                        return true;
                    }
                }
            }
            return false;
        }


        public bool AsignarBodega(string claveProducto, Bodega
bodega)
        {
            if (inventario.ContainsKey(claveProducto))
            {
                var item = inventario[claveProducto];
                item.BodegaAsignada = bodega;
                Console.ForegroundColor = ConsoleColor.Magenta;
                Console.WriteLine($"[BODEGA]
```

```
{item.Producto.Nombre} asignado a Bodega {(int)bodega}");
                Console.ResetColor();
                return true;
            }
            Console.WriteLine($"[BODEGA] Producto no encontrado");
            return false;
        }

        public List<EstadoItemInventario>
ObtenerProductosPorBodega(Bodega bodega)
        {
            return inventario.Values.Where(item =>
item.BodegaAsignada == bodega).ToList();
        }

        public void MostrarEstadisticasBodegas()
        {

Console.WriteLine("\n==============================================
====================");
            Console.WriteLine("              ESTADÍSTICAS POR
BODEGA                          ");

Console.WriteLine("==============================================
====================");

            for (int i = 1; i <= 3; i++)
            {
                var bodega = (Bodega)i;
                var productos = ObtenerProductosPorBodega(bodega);
                var totalStock = productos.Sum(p => p.Cantidad);

                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine($"\nBodega {i}:");
                Console.ResetColor();
                Console.WriteLine($"  Variedades:
{productos.Count}");
                Console.WriteLine($"  Stock total: {totalStock}
unidades");

                if (productos.Count > 0)
                {
```

```
                    Console.WriteLine("  Productos:");
                    foreach (var item in productos)
                    {
                        Console.WriteLine($"    • 
{item.Producto.Nombre} ({item.Cantidad} unidades)");
                    }
                }
            }

            var sinAsignar = 
ObtenerProductosPorBodega(Bodega.SinAsignar);
            if (sinAsignar.Count > 0)
            {
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine($"\nSin asignar a bodega: 
{sinAsignar.Count} variedades");
                Console.ResetColor();
            }
        }

        // *** FIN NUEVOS MÉTODOS ***

        public List<EstadoItemInventario> ObtenerInventario()
        {
            return inventario.Values.ToList();
        }

        public List<CapasDatosPalet> ObtenerPalets()
        {
            return poolPalets.ObtenerTodosPalets();
        }

        public void MostrarEstadisticas()
        {
            poolPalets.MostrarEstadisticasPool();
            Console.WriteLine($"\n=== Estadísticas de Flyweight 
===");
            Console.WriteLine($"Tipos de frutas en caché: 
{FabricaFlyweightProducto.Instancia.ObtenerTotalTipos()}");
            Console.WriteLine($"Variedades únicas en inventario: 
{inventario.Count}");
        }
```

```
        public int ObtenerPaletsRestantes()
        {
            return poolPalets.ObtenerPaletsRestantes();
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class EstadoDisponible : EstadoInventario
    {
        public override EstadoInventario
Manejar(EstadoItemInventario contexto)
        {
            if (contexto.Cantidad == 0)
                return new EstadoSinStock();
            else if (contexto.Cantidad <= contexto.StockMinimo)
                return new EstadoStockBajo();
            return this;
        }

        public override string ObtenerNombreEstado() =>
"DISPONIBLE";
        public override ConsoleColor ObtenerColor() =>
ConsoleColor.Green;
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace TESTFINAL
{
    public abstract class EstadoInventario
    {
        public abstract EstadoInventario
Manejar(EstadoItemInventario contexto);
        public abstract string ObtenerNombreEstado();
        public abstract ConsoleColor ObtenerColor();
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class EstadoItemInventario
    {
        public CapasDatosProducto Producto { get; set; }
        public int Cantidad { get; set; }
        public int StockMinimo { get; set; }
        public EstadoInventario Estado { get; set; }
        public string IdPaletAsignado { get; set; }
        public Bodega BodegaAsignada { get; set; }

        public EstadoItemInventario(CapasDatosProducto producto,
int cantidad, int stockMinimo = 10)
        {
            Producto = producto;
            Cantidad = cantidad;
            StockMinimo = stockMinimo;
            Estado = new EstadoDisponible();
            IdPaletAsignado = null;
            BodegaAsignada = Bodega.SinAsignar;
            ActualizarEstado();
        }

        public void AgregarStock(int cantidad)
```

```
        {
            Cantidad += cantidad;
            ActualizarEstado();
        }

        public bool RetirarStock(int cantidad)
        {
            if (Cantidad >= cantidad)
            {
                Cantidad -= cantidad;
                ActualizarEstado();
                return true;
            }
            return false;
        }

        private void ActualizarEstado()
        {
            Estado = Estado.Manejar(this);
        }

        public override string ToString()
        {
            string infoPalet = IdPaletAsignado != null ? $"
[Palet: {IdPaletAsignado}]" : "";
            string infoBodega = BodegaAsignada !=
Bodega.SinAsignar ? $" [Bodega: {(int)BodegaAsignada}]" : "";
            return $"{Producto} - Stock: {Cantidad} - Estado:
{Estado.ObtenerNombreEstado()}{infoPalet}{infoBodega}";
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class EstadoSinStock : EstadoInventario
```

```
    {
        public override EstadoInventario
Manejar(EstadoItemInventario contexto)
        {
            if (contexto.Cantidad > 0)
                return new EstadoStockBajo();
            return this;
        }

        public override string ObtenerNombreEstado() => "SIN
STOCK";
        public override ConsoleColor ObtenerColor() =>
ConsoleColor.Red;
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class EstadoStockBajo : EstadoInventario
    {
        public override EstadoInventario
Manejar(EstadoItemInventario contexto)
        {
            if (contexto.Cantidad == 0)
                return new EstadoSinStock();
            else if (contexto.Cantidad > contexto.StockMinimo)
                return new EstadoDisponible();
            return this;
        }

        public override string ObtenerNombreEstado() => "STOCK
BAJO";
        public override ConsoleColor ObtenerColor() =>
ConsoleColor.Yellow;
    }
```

```
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class ObjectPoolPoolPalets
    {
        private List<CapasDatosPalet> paletsDisponibles = new
List<CapasDatosPalet>();
        private List<CapasDatosPalet> paletsEnUso = new
List<CapasDatosPalet>();
        private int siguienteId = 1;
        private readonly double capacidadPorDefecto;
        private const int MAXIMO_PALETS = 15;

        public ObjectPoolPoolPalets(double capacidadPorDefecto =
1000.0)
        {
            this.capacidadPorDefecto = capacidadPorDefecto;
        }


        public CapasDatosPalet AdquirirPalet()
        {
            CapasDatosPalet palet;

            if (paletsDisponibles.Count > 0)
            {

                palet = paletsDisponibles[0];
                paletsDisponibles.RemoveAt(0);
                Console.WriteLine($"[POOL] Reutilizando palet
{palet.Id}");
            }
            else
            {
```

```
                if (siguienteId > MAXIMO_PALETS)
                {
                    Console.ForegroundColor = ConsoleColor.Red;
                    Console.WriteLine($"[POOL] ¡LÍMITE ALCANZADO!
No se pueden crear más de {MAXIMO_PALETS} palets.");
                    Console.ResetColor();
                    return null;
                }


                palet = new
CapasDatosPalet($"PAL-{siguienteId++}", capacidadPorDefecto);
                Console.ForegroundColor = ConsoleColor.Cyan;
                Console.WriteLine($"[POOL] Creando nuevo palet
{palet.Id} ({siguienteId - 1}/{MAXIMO_PALETS})");
                Console.ResetColor();
            }

            palet.EstaDisponible = false;
            paletsEnUso.Add(palet);
            return palet;
        }


        public void LiberarPalet(CapasDatosPalet palet)
        {
            if (paletsEnUso.Contains(palet))
            {
                paletsEnUso.Remove(palet);
                palet.Reiniciar();
                paletsDisponibles.Add(palet);
                Console.WriteLine($"[POOL] Palet {palet.Id}
devuelto al pool");
            }
        }

        public void MostrarEstadisticasPool()
        {
            Console.WriteLine($"\n=== Estadísticas del Pool de
Palets ===");
            Console.WriteLine($"Palets disponibles:
```

```
{paletsDisponibles.Count}");
            Console.WriteLine($"Palets en uso:
{paletsEnUso.Count}");
            Console.WriteLine($"Total creados: {siguienteId -
1}/{MAXIMO_PALETS}");
            Console.ForegroundColor = (siguienteId - 1) >=
MAXIMO_PALETS ? ConsoleColor.Red : ConsoleColor.Green;
            Console.WriteLine($"Estado: {(siguienteId - 1 >=
MAXIMO_PALETS ? "LÍMITE ALCANZADO" : "Disponible para crear
más")}");
            Console.ResetColor();
        }

        public List<CapasDatosPalet> ObtenerTodosPalets()
        {
            var todosPalets = new List<CapasDatosPalet>();
            todosPalets.AddRange(paletsDisponibles);
            todosPalets.AddRange(paletsEnUso);
            return todosPalets;
        }

        public int ObtenerPaletsRestantes()
        {
            return MAXIMO_PALETS - (siguienteId - 1);
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class FabricaFlyweightProducto
    {
        private Dictionary<string, PesoLigeroTipoProducto>
tiposProducto = new Dictionary<string, PesoLigeroTipoProducto>();
        private static FabricaFlyweightProducto instancia;
```

```
        private FabricaFlyweightProducto()
        {

            tiposProducto["Pepino"] = new
PesoLigeroTipoProducto("Pepino", "Pepino fresco", 0.3);
            tiposProducto["Jicama"] = new
PesoLigeroTipoProducto("Jicama", "Jicama fresca", 0.8);
            tiposProducto["Mango"] = new
PesoLigeroTipoProducto("Mango", "Mango fresco", 0.4);
            tiposProducto["Fresa"] = new
PesoLigeroTipoProducto("Fresa", "Fresa fresca", 0.15);
        }

        public static FabricaFlyweightProducto Instancia
        {
            get
            {
                if (instancia == null)
                    instancia = new FabricaFlyweightProducto();
                return instancia;
            }
        }

        public PesoLigeroTipoProducto ObtenerTipoProducto(string
categoria)
        {
            if (tiposProducto.ContainsKey(categoria))
                return tiposProducto[categoria];


            var nuevoTipo = new PesoLigeroTipoProducto(categoria,
$"Categoría {categoria}", 5.0);
            tiposProducto[categoria] = nuevoTipo;
            return nuevoTipo;
        }

        public int ObtenerTotalTipos()
        {
            return tiposProducto.Count;
        }
    }
```

```
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TESTFINAL
{
    public class PesoLigeroTipoProducto
    {
        public string Categoria { get; private set; }
        public string Descripcion { get; private set; }
        public double PesoEstandar { get; private set; }

        public PesoLigeroTipoProducto(string categoria, string
descripcion, double pesoEstandar)
        {
            Categoria = categoria;
            Descripcion = descripcion;
            PesoEstandar = pesoEstandar;
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;



namespace TESTFINAL
{

    class Programa
    {
        static void Main(string[] args)
        {
```

```
            Console.OutputEncoding = System.Text.Encoding.UTF8;


            var interfaz = new CapasDisenoInterfazLogistica();
            interfaz.Ejecutar();
        }
    }
}
```

Conclusión:

Sin duda este fue el código más demandante en cuanto a lógica y complicaciones al momento de desarrollarlo, no voy a decir que quedó perfecto, porque está lejos de quedar perfecto todavía tiene mucho margen de mejora. Pero sin lugar a dudas puedo decir que es el más completo y funcional en cuanto uso cotidiano en el día a día. Profesora lo mas seguro es que nos toque otro semestre juntos, le pido por favor déjeme formar a nosotros el equipo.