FleetManager S.A

Iris Pérez Aparicio

2º DAM

Campus FP Emprende Humanes

Sistema de Gestión de Vehículos

Resultados de aprendizaje y criterios de evaluación

RA1. Comprender y aplicar los fundamentos del lenguaje C# en el desarrollo de aplicaciones	~~1a~~	~~Se han identificado los fundamentos de la programación estructurada y el lenguaje C# (tipos de datos, variables, operadores).~~
	~~1b~~	~~Se han identificado los fundamentos de la programación orientada a objetos.~~
	1c	Se han implementado clases con atributos, métodos y constructores aplicando encapsulación.
	1d	Se han aplicado los principios de herencia y polimorfismo en ejercicios prácticos.

Enunciado

La empresa **FleetManager S.A.** gestiona una flota de vehículos de transporte de pasajeros y de carga. Necesita un sistema flexible para calcular el **Costo Operacional por Kilómetro** de cada unidad.

Todo vehículo tiene una **Matrícula** y un **Consumo de Combustible** (litros por cada 100 km). Su **Costo Operacional Base** es fijo (0.15€ por litro).

Código (Archivo **Vehiculo.cs**)

```
namespace IrisPerezAparicio_ASP_A2_11_FleetManager_SA
{
    // Clase base reutilizable con lógica común
    class Vehiculo
    {
        // Propiedad automática
        public string Matricula { get; set; }

        // Propiedad no automática con validación
        private double _consumo; // litros por 100 km
        public double Consumo
        {
            get => _consumo;
            set => _consumo = value < 0 ? 0.0 : value;
        }

        // Costo operacional base por litro (solo lectura)
        protected const double COSTO_OPERACIONAL_BASE = 0.15;

        // Constructor base
        public Vehiculo(string matricula, double consumo)
        {
            Matricula = matricula;
            Consumo = consumo;
        }

        // Método polimórfico: costo por km base
        // Puede ser reutilizado por las clases derivadas
        public virtual double CalcularCostoPorKm()
        {
            // Según enunciado: Coste base = Consumo x Costo Operacional
Base
            return Consumo * COSTO_OPERACIONAL_BASE;
        }
```

```
        // ToString polimórfico (solo datos comunes)
        public override string ToString()
        {
            return $"Matrícula: {Matricula}, Consumo: {Consumo:F2}
L/100km";
        }
    }
}
```

Los vehículos de **transporte de pasajeros** tienen además una **Capacidad Máxima**. En el caso de los **Autobuses** se aplica un **Factor de Desgaste** del 1.2€ al costo operacional base: *Coste Por KM = Consumo x Costo Operacional Base x Factor de Desgaste*.

Código (Archivo **Autobus.cs**)

```
namespace IrisPerezAparicio_ASP_A2_11_FleetManager_SA

{
    class Autobus : Vehiculo
    {
        private double _capacidadMaxima;
        public double CapacidadMaxima
        {
            get => _capacidadMaxima;
            set => _capacidadMaxima = value < 0 ? 0.0 : value;
        }

        // Factor de desgaste (solo lectura)
        private const double FACTOR_DESGASTE = 1.2;

        public Autobus(string matricula, double consumo, double
capacidadMaxima)
            : base(matricula, consumo)
        {
            CapacidadMaxima = capacidadMaxima;
        }

        // Coste Por KM = (coste base) x Factor de Desgaste
        // Reutiliza el método polimórfico de la clase base
```

```
        public override double CalcularCostoPorKm()
        {
            return base.CalcularCostoPorKm() * FACTOR_DESGASTE;
        }

        public override string ToString()
        {
            return $"[Autobús] {base.ToString()}, Capacidad Máxima:
{CapacidadMaxima:F0} pasajeros";
        }
    }
}
```

Los vehículos de **transporte de carga** tienen además un **Peaje Anual**.
En el caso de los **Camiones** su costo operacional por kilómetro es el costo base más un **Costo Fijo de Peaje por Kilómetro**, calculado como **Peaje Anual / 100,000 km**: *Coste Por KM = Consumo x Costo Operacional Base x Peaje Anual / 100,000.0.*

Código (Archivo **Camion.cs**)

```
namespace IrisPerezAparicio_ASP_A2_11_FleetManager_SA
{
    class Camion : Vehiculo
    {
        private double _peajeAnual;
        public double PeajeAnual
        {
            get => _peajeAnual;
            set => _peajeAnual = value < 0 ? 0.0 : value;
        }

        public Camion(string matricula, double consumo, double
peajeAnual)
            : base(matricula, consumo)
        {
            PeajeAnual = peajeAnual;
        }

        // Coste Por KM = coste base + (Peaje Anual / 100000)
        // Se apoya en el método base para el coste de combustible
```

```
        public override double CalcularCostoPorKm()
        {
            double costoBaseCombustible = base.CalcularCostoPorKm();
            double costoPeajePorKm = PeajeAnual / 100000.0;
            return costoBaseCombustible + costoPeajePorKm;
        }

        public override string ToString()
        {
            return $"[Camión] {base.ToString()}, Peaje Anual:
{PeajeAnual:F2}";
        }
    }
}
```

El sistema debe **calcular** el **Coste Por Km** de cada vehículo de su flota.

Todos los atributos numéricos críticos (**consumo, capacidad, carga, peaje, …**) deben validarse de forma que si se les quiere asignar un **valor negativo**, el sistema debe asignarle automáticamente **0.0** como medida preventiva.

Diseña una aplicación de consola que incluya un menú con la siguiente funcionalidad:

1. **Registrar Vehículo:** Permite al usuario seleccionar el tipo de vehículo (Autobús o Camión), introducir sus datos y añadir la instancia a una **colección interna de vehículos**.
2. **Ver Costos Operacionales:** Recorre la colección, mostrando los atributos del vehículo (usando `ToString()`) y su Costo Operacional Total (llamando a `CalcularCostoPorKm()`).
3. **Calcular Costo Total de Flota (Consumo Fijo):** Suma todos los Costos Operacionales Totales de la colección, asumiendo una distancia fija de **100,000.0 km** por vehículo.
4. **Salir:** Termina la ejecución de la aplicación.

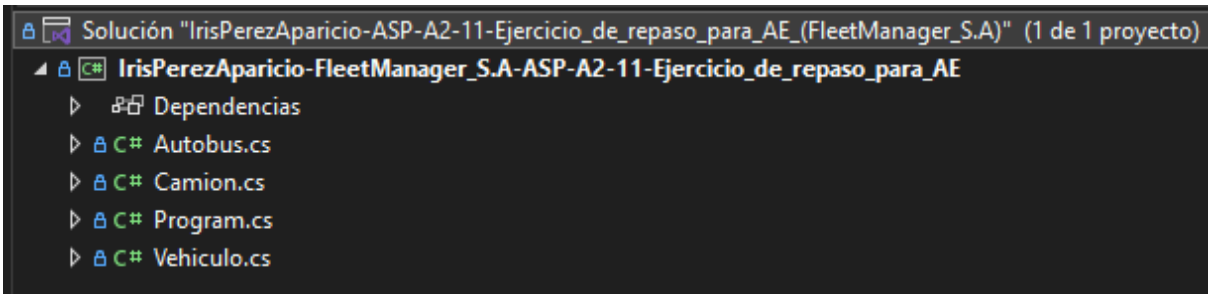
Código (Archivo **Program.cs**)

```
namespace IrisPerezAparicio_ASP_A2_11_FleetManager_SA
{
    class Program
    {
        // Colección polimórfica de vehículos
        static List<Vehiculo> flota = new List<Vehiculo>();

        static int LeerEntero()
        {
            int num;
            while (!int.TryParse(Console.ReadLine(), out num))
                Console.Write("Introduce un número entero válido: ");
            return num;
        }

        static double LeerDouble()
        {
            double num;
            while (!double.TryParse(Console.ReadLine(), out num))
                Console.Write("Introduce un número válido: ");
            return num;
        }

        static void RegistrarVehiculo()
        {
            Console.WriteLine("\nTIPO DE VEHÍCULO:");
            Console.WriteLine("1. Autobús");
            Console.WriteLine("2. Camión");
            Console.WriteLine("3. Volver al menú");
            Console.Write("Seleccione una opción: ");
            int opcion = LeerEntero();

            if (opcion < 1 || opcion > 3)
            {
                Console.WriteLine("Opción no válida. Volviendo al
menú...");
                return;
            }

            if (opcion == 3)
            {
                Console.WriteLine("Volviendo al menú...");
                return;
            }

            Console.Write("\nMatrícula: ");
```

```
            string matricula = Console.ReadLine();

            Console.Write("Consumo (L/100km): ");
            double consumo = LeerDouble();

            switch (opcion)
            {
                case 1:
                    Console.Write("Capacidad Máxima (número de
pasajeros): ");
                    double capacidad = LeerDouble();
                    flota.Add(new Autobus(matricula, consumo,
capacidad));
                    Console.WriteLine("Autobús registrado
correctamente.\n");
                    break;

                case 2:
                    Console.Write("Peaje Anual (€): ");
                    double peajeAnual = LeerDouble();
                    flota.Add(new Camion(matricula, consumo,
peajeAnual));
                    Console.WriteLine("Camión registrado
correctamente.\n");
                    break;
            }
        }

        static void VerCostosOperacionales()
        {
            Console.WriteLine("\nCOSTOS OPERACIONALES POR VEHÍCULO:");
            if (flota.Count == 0)
            {
                Console.WriteLine("No hay vehículos registrados.");
                return;
            }

            foreach (var vehiculo in flota)
            {
                double costoPorKm = vehiculo.CalcularCostoPorKm();
                Console.WriteLine($"{vehiculo} | Costo por Km:
{costoPorKm:F4} €");
            }
        }

        static void CalcularCostoTotalFlota()
        {
            if (flota.Count == 0)
```

```
            {
                Console.WriteLine("\nNo hay vehículos registrados.");
                return;
            }

            const double KM_FIJOS = 100000.0;
            double total = flota.Sum(v => v.CalcularCostoPorKm() *
KM_FIJOS);

            Console.WriteLine($"\nCosto Total de la Flota para
{KM_FIJOS:F0} km por vehículo: {total:F2} €");
        }

        static void Main()
        {
            while (true)
            {
                Console.WriteLine("\n··· MENÚ ···");
                Console.WriteLine("1. Registrar Vehículo");
                Console.WriteLine("2. Ver Costos Operacionales");
                Console.WriteLine("3. Calcular Costo Total de Flota
(100,000 km por vehículo)");
                Console.WriteLine("4. Salir");
                Console.Write("Seleccione una opción: ");

                int opcion = LeerEntero();

                switch (opcion)
                {
                    case 1:
                        RegistrarVehiculo();
                        break;
                    case 2:
                        VerCostosOperacionales();
                        break;
                    case 3:
                        CalcularCostoTotalFlota();
                        break;
                    case 4:
                        return;
                    default:
                        Console.WriteLine("Opción no válida (1-4).");
                        break;
                }
            }
        }
    }
}
```

ESTRUCTURA FINAL DE ARCHIVOS:



Enlace al código en **GitHub**:

https://github.com/IrisCampusFP/Actividades_ASP_NET/tree/main/IrisPerezAparicio-ASP-A2-11-Ejercicio_de_repaso_para_AE_(FleetManager_S.A)