Desafío Practico 3:

Asignatura: Programación de Algoritmos 02L.

Docente: ing. Aida Quintanilla.

Temas: Structs, Archivos de Texto.

Integrante:

Harold Albeiro Quintanilla Rodriguez. (QR241622)

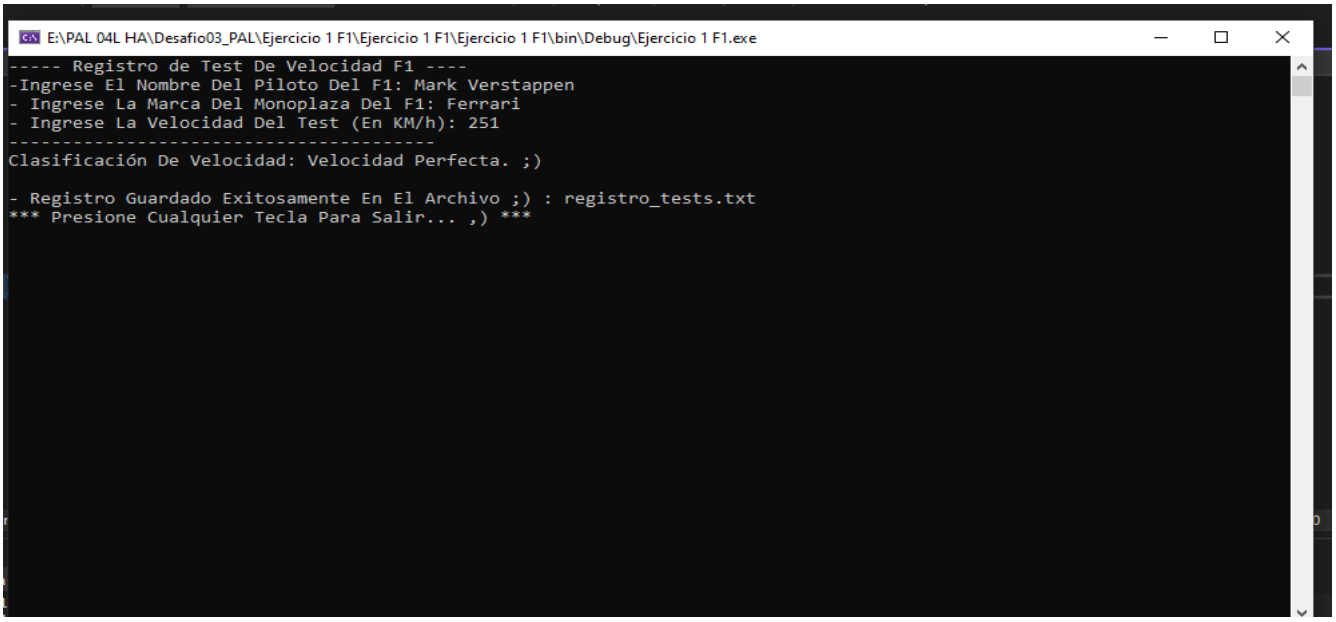
Ejercicio 1:

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

  public struct Piloto
  {// Realizado Por:
   // Harold Quintanilla
   //Dia:9/101/2025

    public string NombrePiloto;
    public string MarcaMonoplaza;
    public double VelocidadTest;

    // Método para clasificar la velocidad y obtener el mensaje
    public string ClasificarVelocidad()
    {
        if (VelocidadTest < 0)
        {
            return "Error: La Velocidad No Puede Ser Negativa.";
        }
        else if (VelocidadTest >= 0 && VelocidadTest <= 90)
        {
         // Si la Velocidad es de 0 a 90km/h Velocidad baja.
            return "Velocidad Baja. Corre Hombre ;(";
        }
        else if (VelocidadTest >= 91 && VelocidadTest <= 150)
        {
         // Si la Velocidad es de 91km/h a 150km/h Velocidad moderada.
            return "Velocidad Moderada. ;)";
        }
        else if (VelocidadTest >= 151 && VelocidadTest <= 250)
        {
         // Si la Velocidad es de 151km/h a 250km/h Velocidad media.
            return "Velocidad Media. ;)";
        }
        else if (VelocidadTest >= 251 && VelocidadTest <= 350)
        {
         // Si la Velocidad es de 251km/h a 350km/h Velocidad perfecta.
            return "Velocidad Perfecta. ;)";
        }
        else
        {
         // Si la velocidad es Mayor de 351km/h Motor Reventado. X.X
            return "Motor reventado. X.X Que Dundo SOS ;(";
        }
    }

    // Método para crear una Línea de Registro con los Datos
    public string CrearRegistro()
    {
```

```
        string clasificacion = ClasificarVelocidad();
        return $"Piloto: {NombrePiloto}, Monoplaza: {MarcaMonoplaza},
Velocidad: {VelocidadTest} km/h, Clasificación: {clasificacion}";
    }
  }

   public class ProgramaFIA
   {
    private const string NombreArchivo = "registro_tests.txt";

    public static void Main(string[] args)
    {
        Piloto monoplaza = new Piloto();

        Console.WriteLine("----- Registro de Test De Velocidad F1 ----");

        // --- 1. Solicitar el nombre del piloto ---
        Console.Write("-Ingrese El Nombre Del Piloto Del F1: ");
        monoplaza.NombrePiloto = Console.ReadLine();

        // --- 2. Solicitar la marca del monoplaza ---
        Console.Write("- Ingrese La Marca Del Monoplaza Del F1: ");
        monoplaza.MarcaMonoplaza = Console.ReadLine();

        // --- 3. Solicitar la velocidad del test con validación ---
        double velocidadIngresada = -1;
        bool entradaValida = false;

        while (!entradaValida)
        {
            Console.Write("- Ingrese La Velocidad Del Test (En KM/h): ");
            string entrada = Console.ReadLine();

            if (double.TryParse(entrada, out velocidadIngresada))
            {
                if (velocidadIngresada < 0)
                {
                    Console.WriteLine("- Error: La Velocidad No Puede Ser
Negativa. Intente De Nuevo. ;/");
                }
                else
                {
                    monoplaza.VelocidadTest = velocidadIngresada;
                    entradaValida = true;
                }
            }
            else
            {
                Console.WriteLine("- Error: Formato De Velocidad No Válido.
Intente De Nuevo. ;( ");
            }
        }

        Console.WriteLine(new string('-', 40));
        string mensajeClasificacion = monoplaza.ClasificarVelocidad();
```

```
        Console.WriteLine($"Clasificación De Velocidad:
{mensajeClasificacion}");

        // --- Implementamos archivos de texto (Guardar el registro De
Datos De F1) ---
        GuardarRegistroEnArchivo(monoplaza);

        Console.WriteLine($"\n- Registro Guardado Exitosamente En El
Archivo ;) : {NombreArchivo}");
        Console.WriteLine("*** Presione Cualquier Tecla Para Salir... ,)
***");
        Console.ReadKey();
    }


    private static void GuardarRegistroEnArchivo(Piloto p)
    {
        try
        {
            string registro = p.CrearRegistro();
            using (StreamWriter sw = new StreamWriter(NombreArchivo, true))
            {
                sw.WriteLine(registro);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"\n- Error Al Escribir En El Archivo :
{ex.Message}");

        }
    }
}
```

Captura 1 Del Funcionamiento:



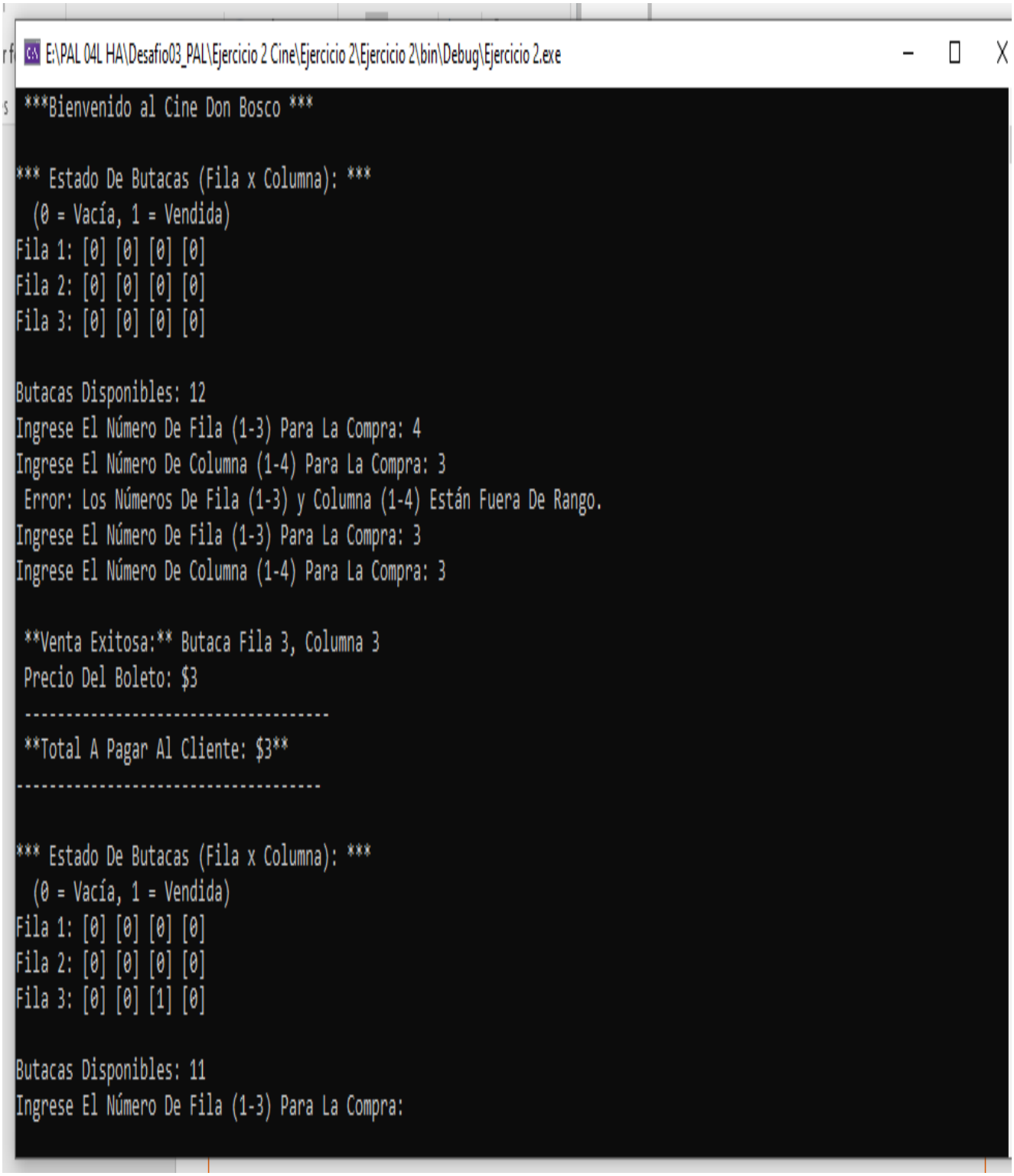
Ejercicio 2:

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

namespace Ejercicio_2
{
    internal class Program
    {
        public class CineDonBosco
        {// Realizado Por:
         // Harold Quintanilla
         //Dia:9/101/2025
            private const int PRECIO_BOLETO = 3;
            private const int MAX_BUTACAS = 12;

          // Matriz para Representar Las Butacas (4 filas x 4 columnas)
            private static int[,] butacas = new int[4, 4];
            private static int butacasVendidas = 0;

            public static void Main(string[] args)
            {
                // 1. Inicializar Butacas a 0 (vacías)
                InicializarButacas();

                Console.WriteLine(" ***Bienvenido al Cine Don Bosco *** ");

                // Bucle principal para la venta de boletos
                while (butacasVendidas < MAX_BUTACAS)
                {
                    MostrarEstadoButacas();
                    VenderBoleto();
                    Console.WriteLine("-------------------------------------");
                }

                // 4. Mensaje final al venderse todos los boletos
                Console.WriteLine("\n **¡ATENCIÓN! Se Han Vendido Todos Los
Boletos.** ");
                Console.WriteLine("   **La Función Va A Comenzar.**");
                Console.WriteLine("-------------------------------------");
            }

            // Inicializa todas las butacas a 0 (vacías).
            private static void InicializarButacas()
            {
                for (int i = 0; i < butacas.GetLength(0); i++)
                {
                    for (int j = 0; j < butacas.GetLength(1); j++)
                    {
                        butacas[i, j] = 0; // 0 = Vacía
                    }
                }
            }

            // Muestra el estado actual de las butacas (0: Vacía, 1: Vendida).
```

```
            private static void MostrarEstadoButacas()
            {
                Console.WriteLine("\n*** Estado De Butacas (Fila x Columna):
***");
                Console.WriteLine("  (0 = Vacía, 1 = Vendida)");

                for (int i = 0; i < butacas.GetLength(0); i++)
                {
                    // Solo mostramos las filas con butacas utilizables
                    if (i < 3)
                    {
                        Console.Write($"Fila {i + 1}: ");
                        for (int j = 0; j < butacas.GetLength(1); j++)
                        {
                            Console.Write($"[{butacas[i, j]}] ");
                        }
                        Console.WriteLine();
                    }
                }
                Console.WriteLine($"\nButacas Disponibles: {MAX_BUTACAS -
butacasVendidas}");
            }

            private static void VenderBoleto()
            {
                int fila = -1;
                int columna = -1;
                bool entradaValida = false;

                while (!entradaValida)
                {
                    try
                    {
                        Console.Write("Ingrese El Número De Fila (1-3) Para La
Compra: ");
                        // Uso de int.Parse que puede lanzar FormatException
                        fila = int.Parse(Console.ReadLine()) - 1; // Ajuste a
índice 0

                        Console.Write("Ingrese El Número De Columna (1-4) Para
La Compra: ");
                        columna = int.Parse(Console.ReadLine()) - 1; // Ajuste
a índice 0

                        // Validación de rango (excepción no controlada por
defecto)
                        if (fila < 0 || fila >= 3 || columna < 0 || columna >=
4)
                        {
                            // Lanza una excepción para un manejo más limpio
del error
                            throw new IndexOutOfRangeException(" Error: Los
Números De Fila (1-3) y Columna (1-4) Están Fuera De Rango.");
                        }

                        // Validación de butaca ya vendida
                        if (butacas[fila, columna] == 1)
                        {
```

```
                            Console.WriteLine(" **Error:** La Butaca
Seleccionada Ya Está Vendida. Por Favor, Elija Otra.");
                        }
                        else
                        {
                            // Venta exitosa
                            butacas[fila, columna] = 1; // 2. Vender: Butaca se
representa con 1
                            butacasVendidas++;

                            // 3. Mostrar total de la venta
                            Console.WriteLine($"\n **Venta Exitosa:** Butaca
Fila {fila + 1}, Columna {columna + 1}");
                            Console.WriteLine($" Precio Del Boleto:
${PRECIO_BOLETO}");
                            Console.WriteLine(" ------------------------------
------");
                            Console.WriteLine($" **Total A Pagar Al Cliente:
${PRECIO_BOLETO}**");

                            entradaValida = true; // Salir del bucle while
                        }
                    }
                    // Bloque para manejar excepciones
                    catch (FormatException)
                    {
                        Console.WriteLine(" **Error de Formato:** Ingrese
Unicamente Números Enteros Para Fila y Columna.");
                    }
                    catch (IndexOutOfRangeException ex)
                    {
                        Console.WriteLine(ex.Message);
                    }
                    catch (Exception ex)
                    {
                        // Manejo de cualquier otra excepción imprevista
                        Console.WriteLine($" **Ocurrió Un Error Inesperado ;(
:** {ex.Message}");
                    }
                }
            }



        }
    }
}
```

Captura 2 Del Funcionamiento:



Ejercicio 3:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.IO;
using System.Text;

public class GasolineraDonBosco
{
    // Estructura Anidada Para Datos Del Vehículo
    public struct DatosVehiculo
    {
        public string Marca;
        public string Tipo; // Sedán, Camión, Camioneta
    }

    // Estructura Para Los Datos del Cliente
    public struct DatosCliente
    {
        public string Nombre;
        public string Correo;
        public DatosVehiculo Vehiculo;
    }

    // Estructura Para Almacenar Los Datos De Cada Venta
    public struct Venta
    {
        public DatosCliente Cliente;
        public string TipoCombustible;
        public double Litros;
```

```
    public double TotalVenta;
    public DateTime FechaVenta;
}

private const double PRECIO_ESPECIAL = 3.77;
private const double PRECIO_REGULAR = 3.51;
private const double PRECIO_DIESEL = 3.25;
private const string NOMBRE_ARCHIVO = "VentasGasolineraDonBosco.txt";

// Lista Estática Para Guardar Todas Las Ventas en Memoria
private static List<Venta> registroVentas = new List<Venta>();

public static void Main(string[] args)
{
    bool salir = false;

    Console.WriteLine("****** Gasolinera Don Bosco ******");

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

        try
        {
            switch (opcion)
            {
                case "1":
                    RealizarVenta();
                    break;
                case "2":
                    MostrarDatosVentas();
                    break;
```

```
                    case "3":
                        MostrarEstadisticas();
                        break;
                    case "4":
                        salir = true;
                        Console.WriteLine("\nSaliendo Del Programa...");
                        break;
                    default:
                        Console.WriteLine("\nOpción No Válida. Por Favor,
Seleccione Una Opción del 1 Al 4. ;)");
                        break;
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"\n** ¡Error Inesperado! **: {ex.Message}");
            }

            if (!salir)
            {
                Console.WriteLine("\nPresione Cualquier Tecla Para Continuar Al
Menú...");
                Console.ReadKey();
            }
        }

        CerrarPrograma();
    }

    private static void MostrarMenu()
    {
        Console.WriteLine("\n================================");
        Console.WriteLine("****** Gasolinera Don Bosco ******");
```

```
        Console.WriteLine("-1. Venta.");
        Console.WriteLine("-2. Mostrar Datos Ventas.");
        Console.WriteLine("-3. Estadisticas De Ventas.");
        Console.WriteLine("-4. Salir");
        Console.WriteLine("==================================");
        Console.Write("Seleccione Una Opción: ");
    }

    private static void RealizarVenta()
    {
        Venta nuevaVenta = new Venta();

        try
        {
            // 1. Captura de Datos del Cliente y Vehículo
            Console.WriteLine("\n---- Datos Del Cliente y Vehículo ----");
            nuevaVenta.Cliente = CapturarDatosCliente();

            // 2. Selección de Combustible
            Console.WriteLine("\n---- Selección De Combustible ----");
            nuevaVenta = SeleccionarCombustible(nuevaVenta);

            // 3. Capturamos de Litros
            Console.Write("Ingrese La Cantidad De Litros Vendidos: ");
            if (!double.TryParse(Console.ReadLine(), out nuevaVenta.Litros) ||
nuevaVenta.Litros <= 0)
            {
                throw new ArgumentException("La Cantidad De Litros Debe Ser Un
Número Positivo.");
            }

            // 4. Cálculomos de Venta Total
            double precioLitro = ObtenerPrecio(nuevaVenta.TipoCombustible);
```

```
            nuevaVenta.TotalVenta = nuevaVenta.Litros * precioLitro;
            nuevaVenta.FechaVenta = DateTime.Now;

            // 5. Guardamos y Mostrar Venta
            registroVentas.Add(nuevaVenta);
            GuardarVentaEnArchivo(nuevaVenta);

            Console.WriteLine("\n---- Facturación Finalizada ----");
            Console.WriteLine($"Cliente: {nuevaVenta.Cliente.Nombre}");
            Console.WriteLine($"Vehículo: {nuevaVenta.Cliente.Vehiculo.Marca}
({nuevaVenta.Cliente.Vehiculo.Tipo})");
            Console.WriteLine($"Combustible: {nuevaVenta.TipoCombustible}");
            Console.WriteLine($"Litros: {nuevaVenta.Litros:N2}");
            Console.WriteLine($"Precio Por Litro: ${precioLitro:N2}");
            Console.WriteLine($"** VENTA TOTAL: ${nuevaVenta.TotalVenta:N2}
**");
            Console.WriteLine("-------------------------------------");
        }
        catch (ArgumentException ex)
        {
            Console.WriteLine($"\n** Error De Entrada De Datos **:
{ex.Message}");
        }
        catch (FormatException)
        {
            Console.WriteLine("\n** Error De Formato **: Por Favor, Ingrese Un
Valor Numérico Valido Para Los Litros.");
        }
    }

    private static void MostrarDatosVentas()
    {
        Console.WriteLine("\n---- Listado De Ventas En La Gasolinera ----");
        if (registroVentas.Count == 0)
```

```
        {
            Console.WriteLine("Aún No Se Han Registrado Ventas En Esta
Sesión.");
            return;
        }

        foreach (var venta in registroVentas)
        {
            Console.WriteLine($"Fecha: {venta.FechaVenta:dd/MM/yyyy HH:mm}");
            Console.WriteLine($"  Cliente: {venta.Cliente.Nombre}");
            Console.WriteLine($"  Vehículo: {venta.Cliente.Vehiculo.Marca}
({venta.Cliente.Vehiculo.Tipo})");
            Console.WriteLine($"  Combustible: {venta.TipoCombustible}, Litros:
{venta.Litros:N2}");
            Console.WriteLine($"  Total: ${venta.TotalVenta:N2}");
            Console.WriteLine("-----------------------------------");
        }
    }

    private static void MostrarEstadisticas()
    {
        Console.WriteLine("\n---- Estadísticas De Ventas En La Gasolinera ----");
        Console.WriteLine($"Total De Clientes Atendidos: {registroVentas.Count}
(Ventas Registradas En Esta Sesión)");
        Console.WriteLine($"Fecha Del Sistema: {DateTime.Now:dd/MM/yyyy
HH:mm:ss}");

        // Ejemplo adicional: Total facturado
        double totalFacturado = registroVentas.Sum(v => v.TotalVenta);
        Console.WriteLine($"Total Facturado (Sesión Actual):
${totalFacturado:N2}");
    }

    private static DatosCliente CapturarDatosCliente()
    {
        DatosCliente cliente = new DatosCliente();
```

```
        Console.Write("Nombre Del Cliente: ");
        cliente.Nombre = Console.ReadLine().Trim();
        if (string.IsNullOrWhiteSpace(cliente.Nombre))
            throw new ArgumentException("El Nombre Del Cliente No Puede Estar
Vacío.");

        Console.Write("Correo Del Cliente: ");
        cliente.Correo = Console.ReadLine().Trim();

        // Validación de Correo Simplificada
        if (string.IsNullOrWhiteSpace(cliente.Correo) ||
!cliente.Correo.Contains("@"))
            throw new ArgumentException("El Correo Del Cliente No Es Válido.");

        // Datos del Vehículo (Estructura Anidada)
        Console.Write("Marca Del Vehículo: ");
        cliente.Vehiculo.Marca = Console.ReadLine().Trim();
        if (string.IsNullOrWhiteSpace(cliente.Vehiculo.Marca))
            throw new ArgumentException("La Marca Del Vehículo No Puede Estar
Vacía.");

        string tipoVehiculo;
        do
        {
            Console.Write("Tipo de Vehículo (Sedan/Camion/Camioneta): ");
            tipoVehiculo = Console.ReadLine().Trim().ToLower();
            if (tipoVehiculo != "sedan" && tipoVehiculo != "camion" && tipoVehiculo
!= "camioneta")
            {
                Console.WriteLine("Tipo de vehículo No Reconocido. Intente De
Nuevo.");
            }
        } while (tipoVehiculo != "sedan" && tipoVehiculo != "camion" &&
tipoVehiculo != "camioneta");
```

```
            cliente.Vehiculo.Tipo = tipoVehiculo;
            return cliente;
        }

        private static Venta SeleccionarCombustible(Venta venta)
        {
            Console.WriteLine($"Tipos De Combustible Disponibles Para Un Vehículo Tipo
'{venta.Cliente.Vehiculo.Tipo}':");
            Console.WriteLine($"1. Regular (${PRECIO_REGULAR:N2})");
            Console.WriteLine($"2. Especial (${PRECIO_ESPECIAL:N2})");
            Console.WriteLine($"3. Diésel (${PRECIO_DIESEL:N2})");
            Console.Write("Seleccione El Combustible (1, 2 o 3): ");

            string opcionCombustible = Console.ReadLine();
            switch (opcionCombustible)
            {
                case "1":
                    venta.TipoCombustible = "Regular";
                    break;
                case "2":
                    venta.TipoCombustible = "Especial";
                    break;
                case "3":
                    venta.TipoCombustible = "Diésel";
                    break;
                default:
                    throw new ArgumentException("Opción De Combustible No Válida.");
            }

            // Validación de Tipo de Vehículo y Combustible
            if ((venta.Cliente.Vehiculo.Tipo == "sedan" || venta.Cliente.Vehiculo.Tipo
== "camioneta") && venta.TipoCombustible == "Diésel")
            {

            }
```

```
        return venta;
    }

    private static double ObtenerPrecio(string tipo)
    {
        switch (tipo)
        {
            case "Especial": return PRECIO_ESPECIAL;
            case "Regular": return PRECIO_REGULAR;
            case "Diésel": return PRECIO_DIESEL;
            default: return 0.0;
        }
    }

    // --- Aqui Tenemos El Manejo de Archivos ---
    private static void GuardarVentaEnArchivo(Venta venta)
    {
        string linea = $"{venta.FechaVenta:dd/MM/yyyy
HH:mm:ss}|{venta.Cliente.Nombre}|{venta.Cliente.Correo}|{venta.Cliente.Vehiculo.Mar
ca}|{venta.Cliente.Vehiculo.Tipo}|{venta.TipoCombustible}|{venta.Litros:N2}|{venta.
TotalVenta:N2}";

        try
        {
            File.AppendAllText(NOMBRE_ARCHIVO, linea + Environment.NewLine,
Encoding.UTF8);
        }
        catch (IOException ex)
        {
            Console.WriteLine($"\n** Error De Archivo **: No Se Pudo Guardar La
Venta. {ex.Message}");
        }
    }

    private static void CerrarPrograma()
```

```
    {
        // Se Abrirá Automáticamente el Archivo de Texto cuando el Programa se
cierre.
        try
        {
            if (File.Exists(NOMBRE_ARCHIVO))
            {
                // Abre el archivo usando la aplicación predeterminada (Bloc de
Notas)
                Process.Start(new ProcessStartInfo(NOMBRE_ARCHIVO) {
UseShellExecute = true });
            }
            else
            {
                Console.WriteLine("El Archivo De Ventas No Existe, No Se Puede
Abrir.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"\n** Error Al Abrir El Archivo **: {ex.Message}");
        }
    }
}
```

Captura 3 Del Funcionamiento: