Arthur Lamidel

TD7

Program.cs

```csharp
using Models;

Console.OutputEncoding = System.Text.Encoding.UTF8;
// Load the trucks from the file
Truck.AddFromFile("camions.txt");
// Display them
Truck.Display();
// Add a new truck
new Truck("Iveco", "Stralis", new DateTime(2006, 1, 1), 310, 36540);
// Display trucks satisfying a predicate
Console.WriteLine("Trucks whose selling price is under 30,000€");
Truck.Display(truck => truck.SellPrice < 30000);
// Remove a truck
Truck.RemoveAt(4);
// Save them to a new text file
Truck.SaveToFile("camions_AS_2.txt");
```

Models/Truck.cs

```csharp
using System.Text.RegularExpressions;

namespace Models;

public class Truck
{
    public static List<Truck> Trucks { get; } = new List<Truck>();

    public string Make { get; init; }
    public string Type { get; init; }
    public DateTime MakeYear { get; init; }
    public int HorsePower { get; init; }
    public int SellPrice { get; init; }

    // Constructor used internally to instantiate without the console output
    private Truck()
    {
        // Parameter initialization is done using init constructor
        // Should only ever be used inside the class

        Trucks.Add(this); // Add the truck to the general list
    }

    // Normal constructor that outputs the creation of the object to the console
    public Truck
    (
        string make,
        string type,
        DateTime makeYear,
        int horsePower,
        int sellPrice
    )
    {
        this.Make = make;
        this.Type = type;
        this.MakeYear = makeYear;

        // HorsePower cannot be negative
        if (horsePower <= 0) throw new ArgumentException("HorsePower cannot be negative");
        this.HorsePower = horsePower;

        // However, SellPrice could be negative (although rare)
        this.SellPrice = sellPrice;

        Trucks.Add(this); // Add the truck to the general list

        // Output the sucessfull addition of the truck
        Console.WriteLine($"Adding {this}\n---");
    }
    public static void AddFromFile(string fileName)
    {
        // Open the text file
        using var file = new StreamReader(fileName);
        // Create a regex to check that a given line is valid
        var truckRegex = new Regex(@"^\w+_[\w\d]+_\d+_\d+_\d+$");

        string line;
        // Until the end of the file is reached, read a line
        while ((line = file.ReadLine()!) != null)
        {
            // If the line is not valid, ignore it
            if (!truckRegex.Match(line).Success) continue;
            // Extract each parameter and parse them accordingly
            // Use them to create the corresponding truck (silently)
            var param = line.Split('_');
            var truck = new Truck
            {
                Make = param[0],
                Type = param[1],
                // Parse a DateTime from a string with only the year
                MakeYear = DateTime.ParseExact(param[2], "yyyy", null),
                HorsePower = int.Parse(param[3]),
                SellPrice = int.Parse(param[4])
            };
        }
    }

    public static void SaveToFile(string filename)
    {
        // Open the text file
        using var file = new StreamWriter(filename);

        // Write each line
        foreach (var truck in Trucks)
            file.WriteLine($"
{truck.Make}_{truck.Type}_{truck.MakeYear.Year}_{truck.HorsePower}_{truck.SellPrice}");

        // Write the output
        Console.WriteLine($"Saved to file {filename}\n---");
    }

    public static void RemoveAt(int index)
    {
        // Write the output
        Console.WriteLine($"Removing {Trucks[index-1]}\n---");
        // Delete the truck for the general list
        Trucks.RemoveAt(index-1);
    }

    public static void Display()
    {
        // Write each truck and their index
        for (int i = 0; i < Trucks.Count; i++)
            Console.WriteLine($"{i+1} : {Trucks[i]}");
        // Write the total number of trucks
        Console.WriteLine($"Total : {Trucks.Count} truck(s)\n---");
    }

    public static void Display(Func<Truck, bool> predicate)
    {
        // Display each truck satisfying the predicate
        Trucks
            .Where(predicate)
            .ToList()
            .ForEach(truck => Console.WriteLine(truck));
        Console.WriteLine("---");
    }

    public override string ToString() => $"{this.Make} {this.Type}";
}
```