13.0-Introduction to LINQ

Program

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

class Car
{
    public string Model { get; set; }
    public string Make { get; set; }
    public int Year { get; set; }
    public string Color { get; set; }

    public override string ToString() => $"{Year} {Make} {Model} - Color: {Color}";
}

class Program
{
    static void Main(string[] args)
    {
        List<Car> cars = new List<Car>
        {
            new Car { Model = "Accord", Make = "Honda", Year = 2020, Color = "Red" },
            new Car { Model = "Civic", Make = "Honda", Year = 2019, Color = "Blue" },
            new Car { Model = "Corolla", Make = "Toyota", Year = 2021, Color = "Black" },
            new Car { Model = "Camry", Make = "Toyota", Year = 2022, Color = "White" },
```

```
};

Console.WriteLine("Car Information System");

while (true)
{
    Console.WriteLine("\n1. List all cars");
    Console.WriteLine("2. Search for cars");
    Console.WriteLine("3. Exit");
    Console.Write("Enter your choice: ");

    if (int.TryParse(Console.ReadLine(), out int choice))
    {
        switch (choice)
        {
            case 1:
                ListAllCars(cars);
                break;
            case 2:
                SearchCars(cars);
                break;
            case 3:
                Console.WriteLine("Exiting the program. Goodbye!");
                return;
            default:
                Console.WriteLine("Invalid choice. Please enter a valid option.");
                break;
```

```
            }
        }
        else
        {
            Console.WriteLine("Invalid input. Please enter a valid number.");
        }
    }
}

static void ListAllCars(List<Car> cars)
{
    Console.WriteLine("\nAll Cars:");
    cars.ForEach(Console.WriteLine);
}

static void SearchCars(List<Car> cars)
{
    Console.Write("Enter car make: ");
    string make = Console.ReadLine();

    Console.Write("Enter car model: ");
    string model = Console.ReadLine();

    var matchingCars = cars.Where(c =>
        c.Make.Equals(make, StringComparison.OrdinalIgnoreCase)
        && c.Model.Equals(model, StringComparison.OrdinalIgnoreCase)).ToList();
```

```
        Console.WriteLine(matchingCars.Any()
            ? "\nMatching Cars:\n" + string.Join("\n", matchingCars)
            : "No matching cars found.");
    }
}
```

Exercise

Write a C# program that manages information about cars.

Describe the Car class.

Explain its properties:

Model: Represents the model of the car.

Make: Represents the manufacturer of the car.

Year: Represents the year of manufacture.

Color: Represents the color of the car.

Highlight the ToString() method which overrides the default ToString() method to provide a custom string representation of a car object.

Mention that it's the entry point of the program.

Describe the initialization of a list of cars (List<Car>).

Provide details about each car object including model, make, year, and color.

Explain the menu loop which allows the user to interact with the program.

Display options for the user:

List all cars.

Search for cars.

Exit the program.

User Input Handling:

Explain how the program handles user input.

It validates user input to ensure it's a valid integer.

Switches to different cases based on the user's choice:

Handles invalid input gracefully by displaying an error message.

Describe the functionality of listing all cars.

Iterate through the list of cars and display details of each car.

Explain the functionality of searching for cars.

Prompt the user to enter the make and model of the car they want to search for.

Utilize LINQ to filter the list of cars based on the user's input.

Display the matching cars if any, or notify the user if no matching cars are found.

Hint

Implement a while loop to ensure the program runs continuously until the user chooses to exit.

Use while (true) to create an infinite loop.

Print the menu options to the console inside the loop.

Use Console.WriteLine() to display each option on a new line.

Prompt the user to enter their choice.

Use Console.ReadLine() to capture the user's input as a string.

Utilize int.TryParse() to convert the input into an integer:

If successful, proceed to the corresponding menu option.

If unsuccessful, handle invalid input gracefully.

Utilize a switch statement to handle different user choices.

Switch on the integer variable representing the user's choice.

Implement cases for each menu option:

Case 1: List all cars.

Case 2: Search for cars.

Case 3: Exit the program.

Default case: Handle invalid choices by displaying an error message.

Call the appropriate functions based on the user's choice:

Invoke the ListAllCars() function for option 1.

Invoke the SearchCars() function for option 2.

Implement program termination for option 3 using return.

Ensure the program continues to prompt the user for input until a valid choice is entered.

Display an error message for invalid input.

Explanation

This program is a simple car information system implemented in C#. It allows users to interactively view a list of cars and search for specific cars based on their make and model. Here's how the program works:

The Car class defines the properties of a car, including its model, make, year, and color. The ToString() method is overridden to provide a formatted string representation of a car's information.

In the Main method, a list of Car objects is initialized with some sample car data. Then, an infinite while loop is used to continuously display a menu of options to the user and handle their input.

Within the loop, the user is prompted to enter their choice, and the program reads their input using Console.ReadLine(). The input is parsed into an integer using int.TryParse() to ensure it's a valid numeric choice.

A switch statement is used to handle different menu choices:

If the user selects option 1, all cars in the list are displayed by calling the ListAllCars() function.

If the user selects option 2, they are prompted to enter the make and model of the car they want to search for. The program then filters the list of cars using LINQ's Where method to find matching cars and displays them.

If the user selects option 3, the program exits gracefully with a goodbye message.

The ListAllCars() function iterates through the list of cars and prints each car's information to the console using Console.WriteLine().

The SearchCars() function prompts the user to enter the make and model of the car they want to search for. It then uses LINQ's Where method to filter the list of cars based on the user's input and displays the matching cars or a message if no matches are found.