МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Комп‘ютерний практикум №1

з курсу «Основи розробки програмного забезпечення на платформі Microsoft.NET»

на тему: «LINQ to Objects»

Перевірила:
доцент,
Ліщук К. І.

Виконав:
студент 2 курсу
групи ІП-21 ФІОТ
Гриценко А. В.

Київ 2024

Варіант 19

**Мета:** ознайомитися з обробкою даних з використанням бібліотеки LINQ to Objects

**Постановка задачі комп‘ютерного практикуму № 1**
При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1) Розробити структуру даних для зберігання згідно варіантів, наведених нижче. У кожному з варіантів має бути як мінімум 4 класи. В рамках реалізації повинні бути продемонстровані зв'язки між класами: один-до-багатьох і багато-до-багатьох.

2) Розробити як мінімум 20 різних запитів, використовуючи різні дії над даними колекцій: групування, сортування, фільтрацію, об'єднання результатів декількох запитів в один (join, concat) та інше. Крім того, необхідно використовувати обидва можливі варіанти реалізації LINQзапитів (класичний варіант та з використанням методів розширення), причому запити не повинні повторюватись.

3) Створити програмне забезпечення, котре реалізує обробку даних з використання бібліотеки LINQ to Objects.

4) Програмне забезпечення необхідно розробити у вигляді консольного застосування на мові C#.

5) Створити звіт. Звіт повинен містити: опис архітектури проекту, словесний опис запитів, текст програмного коду, скріншоти результатів виконання.

**Варіант індивідуального завдання:**

19) Розробити структуру даних для зберігання інформації про реєстрацію транспортних засобів. Для кожного транспортного засобу зберігається як мінімум марка авто, виробник, модель, тип кузову, рік випуску, номер шасі (VIN-код), колір, номерний знак, технічний стан, власник автомобіля,

перелік водіїв, котрі мають право керувати транспортним засобом, тощо. Для власників та тих персон, котрі мають право керувати транспортним засобом, - номер прав водія, прізвище, ім'я, по батькові, дата народження, адреса реєстрації. Необхідно врахувати, що транспортний засіб може мати декілька власників (тобто бути зареєстрованим декілька разів).

ER-diagram

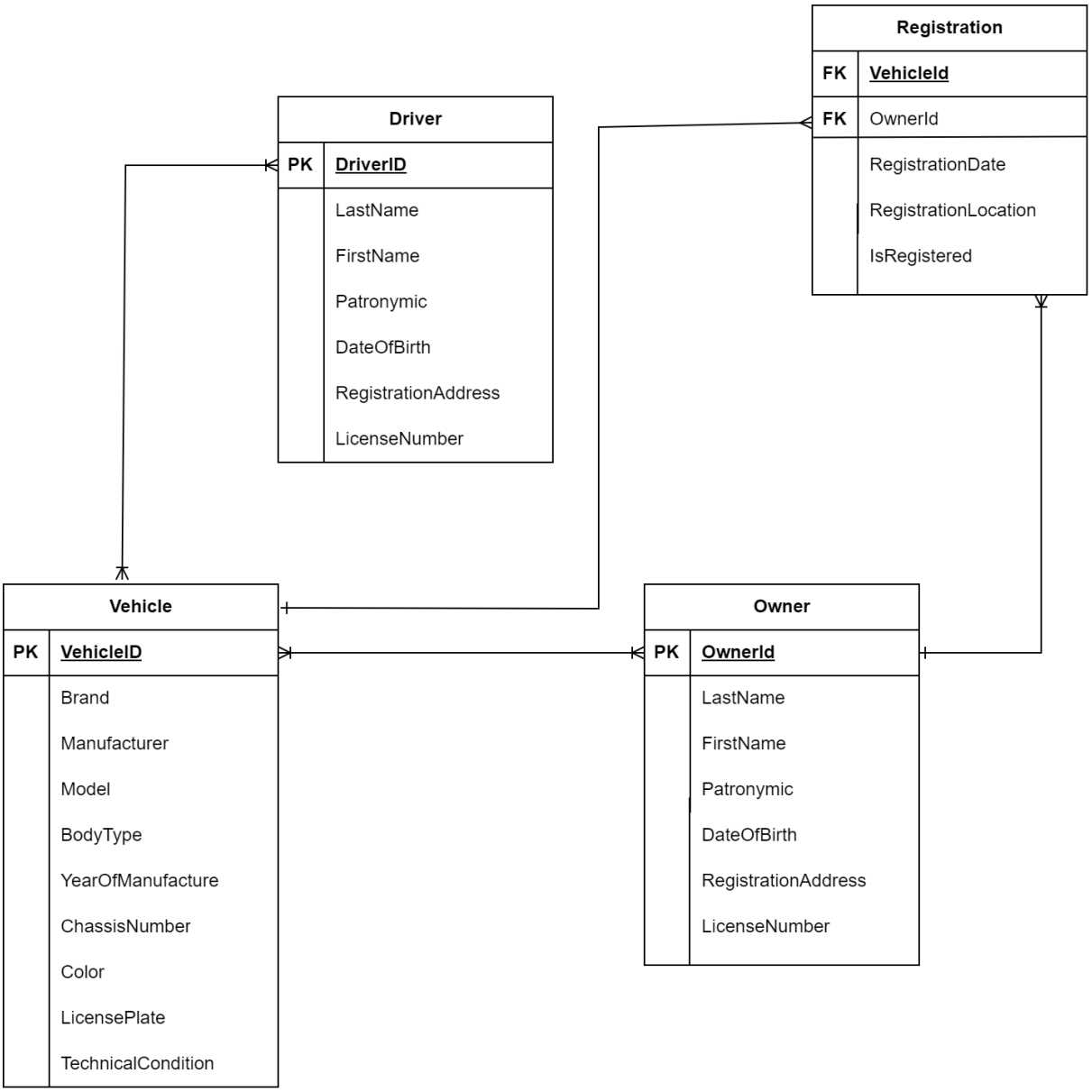


Рисунок 1. ER diagram

Vehicle - Driver (багато до багатьох): кілька водіїв можуть керувати кількома транспортними засобами, і навпаки. Наприклад, у службах

оренди або сімейних автомобілях, де кілька членів сім’ї можуть керувати одним транспортним засобом.

Vehicle - Owner (багато до багатьох): кілька власників можуть мати право власності на декілька транспортних засобів, і навпаки.

Vehicle - Registration (один до багатьох): кожен транспортний засіб може мати декілька реєстраційних записів з часом, але кожен реєстраційний запис пов’язаний лише з одним транспортним засобом. У реальних сценаріях транспортні засоби можуть проходити кілька реєстрацій через зміну власності, місцезнаходження чи інші фактори, але кожна реєстрація стосується окремого автомобіля.

Owner - Registration (один до багатьох): кожен власник може мати кілька реєстраційних записів протягом певного часу, але кожен реєстраційний запис пов’язано лише з одним власником.

**Основні класи, що реалізують дану діаграму:**

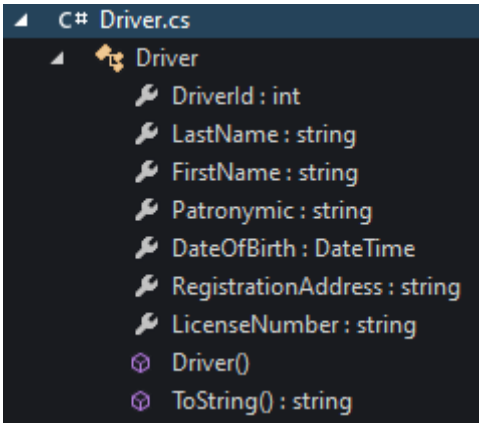


Рисунок 2. Клас водія

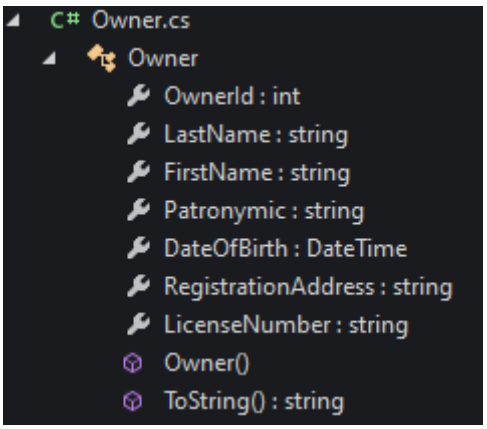


Рисунок 3. Клас власника

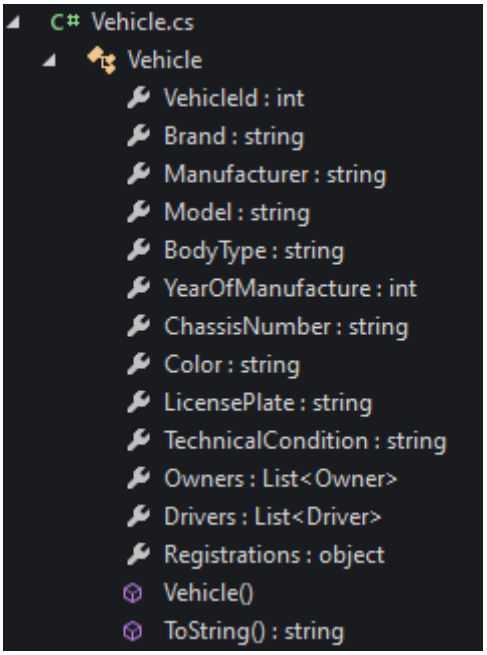


Рисунок 4. Клас транспортного засобу

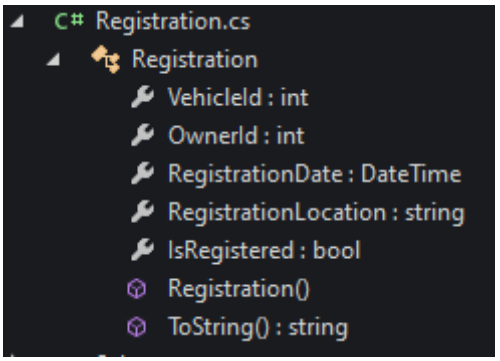


Рисунок 5. Клас реєстрації

Допоміжні класи, що реалізують зв’язок багато-до-багатьох

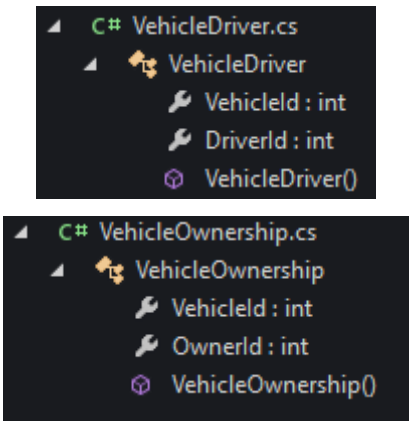


Рисунок 6. Допоміжні класи для зв’язків

Програмний код

**Реалізація основних класів:**

Driver.cs:

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

namespace laba1
{
   public class Driver
   {
      public int DriverId { get; init; }
      public string LastName { get; init; }
      public string FirstName { get; init; }
      public string Patronymic { get; init; }
      public DateTime DateOfBirth { get; init; }
      public string RegistrationAddress { get; init; }
      public string LicenseNumber { get; init; }

      public Driver()
      {
         DriverId = 0;
```

```
            LastName = string.Empty;
            FirstName = string.Empty;
            Patronymic = string.Empty;
            DateOfBirth = DateTime.MinValue;
            RegistrationAddress = string.Empty;
            LicenseNumber = string.Empty;
        }

        public override string ToString()
        {
            return $"\nDriver ID: {DriverId}\n" + // Include DriverId in the output
                   $"Прізвище: {LastName}\n" +
                   $"Ім'я: {FirstName}\n" +
                   $"По батькові: {Patronymic}\n" +
                   $"Дата народження: {DateOfBirth.ToShortDateString()}\n" +
                   $"Адреса реєстрації: {RegistrationAddress}\n" +
                   $"Номер водійського посвідчення: {LicenseNumber}";
        }
    }
}
```

Owner.cs:

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

namespace laba1
{
    public class Owner
    {
        public int OwnerId { get; init; }
        public string LastName { get; init; }
        public string FirstName { get; init; }
```

```
        public string Patronymic { get; init; }
        public DateTime DateOfBirth { get; init; }
        public string RegistrationAddress { get; init; }
        public string LicenseNumber { get; init; }

        public Owner()
        {
            OwnerId = 0;
            LastName = string.Empty;
            FirstName = string.Empty;
            Patronymic = string.Empty;
            DateOfBirth = DateTime.MinValue;
            RegistrationAddress = string.Empty;
            LicenseNumber = string.Empty;
        }

        public override string ToString()
        {
            return $"ID власника: {OwnerId}\n" +
                   $"Прізвище: {LastName}\n" +
                   $"Ім'я: {FirstName}\n" +
                   $"По батькові: {Patronymic}\n" +
                   $"Дата народження: {DateOfBirth.ToShortDateString()}\n" +
                   $"Адреса реєстрації: {RegistrationAddress}\n" +
                   $"Номер водійського посвідчення: {LicenseNumber}\n";
        }
    }
}
```

Vehicle.cs:

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

```
namespace laba1
{
    public class Vehicle
    {
        public int VehicleId { get; init; }
        public string Brand { get; init; }
        public string Manufacturer { get; init; }
        public string Model { get; init; }
        public string BodyType { get; init; }
        public int YearOfManufacture { get; init; }
        public string ChassisNumber { get; init; }
        public string Color { get; init; }
        public string LicensePlate { get; init; }
        public string TechnicalCondition { get; init; }
        public List<Owner> Owners { get; init; }
        public List<Driver> Drivers { get; init; }
        public object Registrations { get; internal set; }

        public Vehicle()
        {
            VehicleId = 0;
            Brand = string.Empty;
            Manufacturer = string.Empty;
            Model = string.Empty;
            BodyType = string.Empty;
            YearOfManufacture = 0;
            ChassisNumber = string.Empty;
            Color = string.Empty;
            LicensePlate = string.Empty;
            TechnicalCondition = string.Empty;
            Owners = new();
            Drivers = new();
        }

        public override string ToString()
        {
            StringBuilder infoAsStringBuilder = new StringBuilder();
```

```
            infoAsStringBuilder.AppendLine($"\nID транспортного засобу:
{VehicleId}");
            infoAsStringBuilder.AppendLine($"Марка: {Brand}");
            infoAsStringBuilder.AppendLine($"Виробник: {Manufacturer}");
            infoAsStringBuilder.AppendLine($"Модель: {Model}");
            infoAsStringBuilder.AppendLine($"Тип кузова: {BodyType}");
            infoAsStringBuilder.AppendLine($"Рік виробництва:
{YearOfManufacture}");
            infoAsStringBuilder.AppendLine($"Номер шасі: {ChassisNumber}");
            infoAsStringBuilder.AppendLine($"Колір: {Color}");
            infoAsStringBuilder.AppendLine($"Номерний знак: {LicensePlate}");
            infoAsStringBuilder.AppendLine($"Технічний стан:
{TechnicalCondition}");
            infoAsStringBuilder.Append("\n~Власники та водії~\n");
            foreach (Owner owner in Owners)
            {
                infoAsStringBuilder.AppendLine(owner.ToString());
            }
            foreach (Driver driver in Drivers)
            {
                infoAsStringBuilder.AppendLine(driver.ToString());
            }
            return infoAsStringBuilder.ToString();
        }
    }
}
```

Registration.cs:

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

namespace laba1
{
```

```
    public class Registration
    {
        public int VehicleId { get; init; }
        public int OwnerId { get; init; }
        public DateTime RegistrationDate { get; init; }
        public string RegistrationLocation { get; init; }
        public bool IsRegistered { get; set; }

        public Registration()
        {
            VehicleId = 0;
            OwnerId = 0;
            RegistrationDate = DateTime.MinValue;
            RegistrationLocation = string.Empty;
            IsRegistered = false;
        }

        public override string ToString()
        {
            return $"Vehicle ID: {VehicleId}\n" +
                   $"Owner ID: {OwnerId}\n" + // Include OwnerId in the output
                   $"Registration Date: {RegistrationDate}\n" +
                   $"Registration Location: {RegistrationLocation}\n" +
                   $"Is Registered: {(IsRegistered ? "Yes" : "No")}";
        }
    }
}
```

**Реалізація допоміжних класів:**

VehicleDriver.cs:

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

```
using System.Threading.Tasks;

namespace laba1
{
    public class VehicleDriver
    {
        public int VehicleId { get; init; }
        public int DriverId { get; init; }

        public VehicleDriver()
        {
            VehicleId = 0;
            DriverId = 0;
        }
    }
}
```

VehicleOwnership.cs:

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

namespace laba1
{
    public class VehicleOwnership
    {
        public int VehicleId { get; init; }
        public int OwnerId { get; init; }

        public VehicleOwnership()
        {
            VehicleId = 0;
            OwnerId = 0;
```

```
        }
    }
}
```

**Основна реалізація запитів:**

Queries.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.PortableExecutable;
using System.Runtime.ConstrainedExecution;
using System.Text;
using System.Threading.Tasks;

namespace laba1
{
    public class Queries
    {
        // #1. Отримати усі машини та відповідно їх власників
        public IEnumerable<Vehicle> GetSedanVehicles(IEnumerable<Vehicle>
vehicles)
        {
            var sedanVehicles = vehicles.Where(vehicle => vehicle.BodyType ==
"SUV");
            return sedanVehicles;
        }



        // #2. Отримати усі машини та відповідно їх власників
        public IEnumerable<(Vehicle vehicle, Owner owner)>
GetAllSUVsWithOwners(IEnumerable<Vehicle> vehicles,
IEnumerable<Owner> owners, IEnumerable<VehicleOwnership>
vehicleOwnerships)
        {
```

```
        var suvsWithOwners =
            from ownership in vehicleOwnerships
            join vehicle in vehicles.Where(v => v.BodyType == "SUV") on
ownership.VehicleId equals vehicle.VehicleId
            join owner in owners on ownership.OwnerId equals owner.OwnerId
            select (vehicle, owner);

        return suvsWithOwners;
    }

    // #3. Отримати перелік машин, що мають відмінні технічні показники
    public IEnumerable<Vehicle>
GetMachinesInExcellentCondition(IEnumerable<Vehicle> vehicles)
    {
        var machinesInExcellentCondition = vehicles
            .Where(vehicle => vehicle.TechnicalCondition.Equals("Awful",
StringComparison.OrdinalIgnoreCase));

        return machinesInExcellentCondition;
    }

    // #4. Отримати перелік людей (власників), вік яких входить у заданий
діапазон
    public IEnumerable<Owner>
GetOwnersWithinAgeRange(IEnumerable<Owner> owners, int minYear, int
maxYear)
    {
        var currentDate = DateTime.Now;
        var minDateOfBirth = currentDate.AddYears(-maxYear);
        var maxDateOfBirth = currentDate.AddYears(-minYear);

        return owners.Where(owner => owner.DateOfBirth >= minDateOfBirth
&& owner.DateOfBirth <= maxDateOfBirth);
    }

    // #5. Отримати водія, що має найдовше ім'я
    public Driver GetDriverWithLongestName(IEnumerable<Driver> drivers)
```

```
    {
        var query = from driver in drivers
                    let nameLength = driver.FirstName.Length +
driver.LastName.Length + driver.Patronymic.Length
                    orderby nameLength descending
                    select driver;

        return query.FirstOrDefault();
    }


    // #6. Отримати машину, що має найстаріший рік випуску
(manufacture)
    public Vehicle
GetVehicleWithEarliestManufactureYear(IEnumerable<Vehicle> vehicles)
    {
        var query = from vehicle in vehicles
                    orderby vehicle.YearOfManufacture
                    select vehicle;

        return query.FirstOrDefault();
    }



    // #7. Отримати відсортований перелік водіїв за датою народження в
порядку зростання
    public IEnumerable<Driver>
SortDriversByDateOfBirthAscendingWithBoundaries(IEnumerable<Driver>
drivers, int startYear, int endYear)
    {
        return drivers.Where(driver => driver.DateOfBirth.Year >= startYear
&& driver.DateOfBirth.Year <= endYear)
                      .OrderBy(driver => driver.DateOfBirth);
    }

    // #8. Отримати перелік машин, що не є зареєстрованими
```

```
    public IEnumerable<Vehicle>
GetUnregisteredVehicles(IEnumerable<Vehicle> vehicles,
IEnumerable<Registration> registrations)
    {
        var unregisteredVehiclesQuery =
            from vehicle in vehicles
            where !registrations.Any(registration => registration.VehicleId ==
vehicle.VehicleId)
            select vehicle;

        return unregisteredVehiclesQuery.ToList();
    }

    // #9. Отримати перелік машин, випущених до певного року (включно)
    public IEnumerable<Vehicle>
GetMachinesReleasedBeforeYear(IEnumerable<Vehicle> vehicles, int year)
    {
        return vehicles.Where(vehicle => vehicle.YearOfManufacture <= year);
    }

    // #10. Отримати машини, зареєстрованих в локації "Local Registration
Office"
    public IEnumerable<(string Brand, string Model)>
GetVehiclesRegisteredInLocalOffice(IEnumerable<Vehicle> vehicles,
IEnumerable<Registration> registrations)
    {
        var vehiclesRegisteredInLocalOffice =
            from registration in registrations
            where registration.RegistrationLocation == "Local Registration
Office"
            join vehicle in vehicles on registration.VehicleId equals
vehicle.VehicleId
            select (vehicle.Brand, vehicle.Model);

        return vehiclesRegisteredInLocalOffice;
    }
```

```
        // #11. Отримати машини, які не керуються
        public IEnumerable<(string Brand, string Model)>
GetVehiclesNotDrivenByDrivers(IEnumerable<Vehicle> vehicles,
IEnumerable<Driver> drivers, IEnumerable<VehicleDriver> vehicleDrivers)
        {
            var vehiclesNotDrivenByDrivers =
                from vehicle in vehicles
                where !drivers.Any(driver => vehicleDrivers.Any(vd => vd.VehicleId
== vehicle.VehicleId && vd.DriverId == driver.DriverId))
                select (vehicle.Brand, vehicle.Model);

            return vehiclesNotDrivenByDrivers;
        }

        // #12. Отримати перелік всіх унікальних власників машин та водіїв,
об'єднавши їх імена
        public IEnumerable<string>
GetUniqueOwnersAndDriversNames(IEnumerable<Owner> owners,
IEnumerable<Driver> drivers)
        {
            var ownerNames = owners.Select(owner => $"{owner.FirstName}
{owner.LastName}");
            var driverNames = drivers.Select(driver => $"{driver.FirstName}
{driver.LastName}");

            var uniqueNames = ownerNames.Concat(driverNames).Distinct();

            return uniqueNames;
        }

        // #13. Отримати перелік власників, що мають декілька машин
        public IEnumerable<Owner>
GetOwnersWithMultipleVehicles(IEnumerable<Owner> owners,
IEnumerable<VehicleOwnership> vehicleOwnerships)
        {
            var ownersWithMultipleVehicles = vehicleOwnerships.GroupBy(vo =>
vo.OwnerId)
```

```
                                        .Where(group => group.Count() > 1)
                                        .Select(group =>
owners.FirstOrDefault(owner => owner.OwnerId == group.Key));
            return ownersWithMultipleVehicles;
        }

        // #14. Отримати перелік машин, відсортовані за роком випуску
(спадання) з діапазоном років випуску автомобілів
        public IEnumerable<Vehicle>
SortVehiclesByModelYearDescendingWithBoundaries(IEnumerable<Vehicle>
vehicles, int startYear, int endYear)
        {
            var sortedVehicles = vehicles.Where(vehicle =>
vehicle.YearOfManufacture >= startYear && vehicle.YearOfManufacture <=
endYear)
                                .OrderByDescending(vehicle =>
vehicle.YearOfManufacture);
            return sortedVehicles;
        }



        // #15. Отримати перелік водіїв без машин
        public IEnumerable<Driver>
FindDriversWithoutVehicles(IEnumerable<Driver> drivers,
IEnumerable<VehicleDriver> vehicleDrivers)
        {
            var driversWithVehicles = vehicleDrivers.Select(vd =>
vd.DriverId).Distinct();
            var driversWithoutVehicles = drivers.Where(driver =>
!driversWithVehicles.Contains(driver.DriverId));

            return driversWithoutVehicles;
        }

        // #16. Отримати середній вік водіїв
        public double GetAverageAgeOfDrivers(IEnumerable<Driver> drivers)
        {
```

```
        double totalAge = drivers.Sum(driver => (DateTime.Now -
driver.DateOfBirth).TotalDays / 365.25);
        double averageAge = totalAge / drivers.Count();

        return averageAge;
    }

    // #17. Отримати перелік водіїв молодше певного віку
    public IEnumerable<Driver>
GetDriversYoungerThanAge(IEnumerable<Driver> drivers, int maxAge)
    {
        var currentDate = DateTime.Now;
        var maxDateOfBirth = currentDate.AddYears(-maxAge);

        var driversYoungerThanAge = drivers.OrderBy(driver =>
driver.DateOfBirth)
                                   .SkipWhile(driver => driver.DateOfBirth <=
maxDateOfBirth)
                                   .TakeWhile(driver => driver.DateOfBirth >
maxDateOfBirth);

        return driversYoungerThanAge;
    }

    // #18. Отримати список автомобілів, що були зареєстровані у якомусь
році
    public IEnumerable<(string Brand, string Model)>
GetVehiclesRegisteredInYear(IEnumerable<Vehicle> vehicles,
IEnumerable<Registration> registrations, int registrationYear)
    {
        var vehiclesRegisteredInYear =
            from registration in registrations
            where registration.RegistrationDate.Year == registrationYear
            join vehicle in vehicles on registration.VehicleId equals
vehicle.VehicleId
            select (vehicle.Brand, vehicle.Model);
```

```
            return vehiclesRegisteredInYear;
        }

        // #19. Отримати список водіїв, що народилися у конкретному році
        public IEnumerable<Driver>
GetDriversBornInYear(IEnumerable<Driver> drivers, int birthYear)
        {
            var driversBornInYear = drivers.Where(driver =>
driver.DateOfBirth.Year == birthYear);
            return driversBornInYear;
        }

        // #20. Отримати список автомобілів, які належать власнику з
конкретним ID
        public IEnumerable<(string Brand, string Model)>
GetCarsOwnedByOwner(IEnumerable<Vehicle> vehicles,
IEnumerable<Owner> owners, IEnumerable<VehicleOwnership>
vehicleOwnerships, int ownerId)
        {
            var ownerCar =
                from ownership in vehicleOwnerships
                join vehicle in vehicles on ownership.VehicleId equals
vehicle.VehicleId
                where ownership.OwnerId == ownerId
                select (vehicle.Brand, vehicle.Model);

            return ownerCar;
        }

    }
}
```

**Клас для виводу запитів:**

PrintQuery.cs

```
using System;
```

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

namespace laba1
{
   public class PrintAndQueriesConnector
   {
      private Queries _qryExecutor;
      private ConsolePrint _printer;
      private Data _dataLists;

      public PrintAndQueriesConnector(Queries qryExecutor, ConsolePrint
printer, Data dataLists)
      {
         this._qryExecutor = qryExecutor;
         this._printer = printer;
         this._dataLists = dataLists;
      }

      public void PrintAllVehicles()
      {
         _printer.Print("всі машини типу кузову SUV",
_qryExecutor.GetSedanVehicles(_dataLists.Vehicles));
      }

      public void PrintAllVehiclesWithOwners()
      {
         _printer.Print("машини типу кузову SUV та ім'я власників",
            _qryExecutor.GetAllSUVsWithOwners(_dataLists.Vehicles,
_dataLists.Owners, _dataLists.VehicleOwnerships));
      }

      public void PrintMachinesInExcellentCondition()
      {
         _printer.Print("машини з жахливим технічним станом",
```

```
_qryExecutor.GetMachinesInExcellentCondition(_dataLists.Vehicles));
        }

        public void PrintOwnersWithinAgeRange(int minYear, int maxYear)
        {
            _printer.Print($"власники з віком від {minYear} до {maxYear}",
                _qryExecutor.GetOwnersWithinAgeRange(_dataLists.Owners,
minYear, maxYear));
        }

        public void PrintDriverWithLongestName()
        {
            var driverWithLongestName =
_qryExecutor.GetDriverWithLongestName(_dataLists.Drivers);
            _printer.Print("водій з найдовшим ім'ям", new List<Driver> {
driverWithLongestName });
        }

        public void PrintVehicleWithEarliestManufactureYear()
        {
            var vehicleWithEarliestYear =
_qryExecutor.GetVehicleWithEarliestManufactureYear(_dataLists.Vehicles);
            _printer.Print("найстаріша машина (за роком випуску):", new
List<Vehicle> { vehicleWithEarliestYear });
        }


        public void
PrintDriversSortedByDateOfBirthAscendingWithBoundaries(int startYear, int
endYear)
        {
            _printer.Print($"відсортовані водії за датою народження в порядку
зростання з роками народження від {startYear} до {endYear}",

_qryExecutor.SortDriversByDateOfBirthAscendingWithBoundaries(_dataLists.
Drivers, startYear, endYear));
```

```
    }

    public void PrintUnregisteredVehicles()
    {
        var unregisteredVehiclesQuery =
            from vehicle in _dataLists.Vehicles
            where !_dataLists.Registrations.Any(registration =>
registration.VehicleId == vehicle.VehicleId)
            select vehicle;

        var unregisteredVehicles = unregisteredVehiclesQuery.ToList();
        _printer.Print("незареєстровані автомобілі:", unregisteredVehicles);
    }


    public void PrintMachinesReleasedBeforeYear(int year)
    {
        _printer.Print($"машини, випущені до {year} року включно",
            _qryExecutor.GetMachinesReleasedBeforeYear(_dataLists.Vehicles,
year));
    }

    public void PrintAllVehiclesRegisteredInLocalOffice()
    {
        var vehiclesRegisteredInLocalOffice =
_qryExecutor.GetVehiclesRegisteredInLocalOffice(_dataLists.Vehicles,
_dataLists.Registrations);

        _printer.Print("машини, зареєстровані в місцевому реєстраційному
офісі:", vehiclesRegisteredInLocalOffice);
    }


    public void PrintAllVehiclesNotDrivenByDrivers()
    {
```

```
            var vehiclesNotDrivenByDrivers =
_qryExecutor.GetVehiclesNotDrivenByDrivers(_dataLists.Vehicles,
_dataLists.Drivers, _dataLists.VehicleDrivers);

            _printer.Print("машини, які не керуються жодним водієм:",
vehiclesNotDrivenByDrivers);
        }

        public void PrintUniqueOwnersAndDriversNames()
        {
            _printer.Print("унікальні власники машин та водії, об'єднані за
іменами",

_qryExecutor.GetUniqueOwnersAndDriversNames(_dataLists.Owners,
_dataLists.Drivers));
        }

        public void PrintOwnersWithMultipleVehicles()
        {
            _printer.Print("власники, що мають декілька машин",
                _qryExecutor.GetOwnersWithMultipleVehicles(_dataLists.Owners,
_dataLists.VehicleOwnerships));
        }

        public void
PrintVehiclesSortedByModelYearDescendingWithBoundaries(int startYear, int
endYear)
        {
            _printer.Print($"машини, відсортовані за роком випуску у
спадаючому порядку з роками випуску від {startYear} до {endYear}",

_qryExecutor.SortVehiclesByModelYearDescendingWithBoundaries(_dataLists.
Vehicles, startYear, endYear));
        }

        public void PrintDriversWithoutVehicles()
        {
```

```
            _printer.Print("водії без машин",
                _qryExecutor.FindDriversWithoutVehicles(_dataLists.Drivers,
_dataLists.VehicleDrivers));
        }

        public void PrintAverageAgeOfDrivers()
        {
            double averageAge =
_qryExecutor.GetAverageAgeOfDrivers(_dataLists.Drivers);
            _printer.Print("середній вік водіїв", new List<double> { averageAge
});
        }

        public void PrintDriversYoungerThanAge(int maxAge)
        {
            _printer.Print($"водії молодше {maxAge} років",
                _qryExecutor.GetDriversYoungerThanAge(_dataLists.Drivers,
maxAge));
        }

        public void PrintVehiclesRegisteredInYear(int registrationYear)
        {
            var vehiclesRegisteredInYear =
_qryExecutor.GetVehiclesRegisteredInYear(_dataLists.Vehicles,
_dataLists.Registrations, registrationYear);

            _printer.Print($"автомобілі, зареєстровані у {registrationYear} році:",
vehiclesRegisteredInYear);
        }

        public void PrintDriversBornInYear(int birthYear)
        {
            _printer.Print($"водії, народжені у {birthYear} році",
                _qryExecutor.GetDriversBornInYear(_dataLists.Drivers, birthYear));
        }

        public void PrintCarsOwnedByOwner(int ownerId)
```

```
        {
            var carsOwnedByOwner =
_qryExecutor.GetCarsOwnedByOwner(_dataLists.Vehicles, _dataLists.Owners,
_dataLists.VehicleOwnerships, ownerId);

            _printer.Print($"автомобілі, що належать власнику з ID {ownerId}:",
carsOwnedByOwner);
        }
    }
}
```

**Класи для виводу меню та його контролю:**

Menu.cs:

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

namespace laba1
{
    public class Menu
    {
        public List<MenuItem> Items { get; set; }
        public int ItemsCount { get { return Items.Count; } }
        public bool IsExitWanted { get; set; }

        public Menu()
        {
            IsExitWanted = false;
            Items = new();
        }

        public void PrintMenu()
```

```
        {
            Console.WriteLine("Головне меню\n");
            foreach (MenuItem menuItem in Items)
            {
                Console.WriteLine(menuItem);
            }
        }

        public void ExecuteSelectedItem(int itemIndex)
        {
            Items[itemIndex].ExecuteSelectedAction();
        }

    }
}
```

MenuItem.cs:

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

namespace laba1
{
    public class MenuItem
    {
        private readonly string _title;
        private readonly Action _selectedAction;

        public MenuItem(string title, Action selectedAction)
        {
            _title = title;
            _selectedAction = selectedAction;
        }
```

```
        internal void ExecuteSelectedAction()
        {
            _selectedAction.Invoke();
        }

        public override string ToString()
        {
            return _title;
        }
    }
}
```

ConsolePrint.cs:

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

namespace laba1
{
    public class ConsolePrint
    {
        public void Print<TValue>(string message, IEnumerable<TValue>
singleCollectionQuery)
        {
            Console.Clear();
            Console.WriteLine($"Ви обрали {message}\n");
            foreach (TValue element in singleCollectionQuery)
            {
                Console.WriteLine(element);
            }
        }

        public void Print<TKey, TValue>(string message,
IEnumerable<IGrouping<TKey, TValue>> multyCollectionQuery)
```

```
        {
            Console.Clear();
            Console.WriteLine($"Ви обрали {message}\n");
            foreach (IGrouping<TKey, TValue> group in multyCollectionQuery)
            {
                foreach (TValue element in group)
                {
                    Console.WriteLine(element);
                }
                Console.WriteLine();
            }
        }

        public void Print<TKey, TValue>(string message,
            IEnumerable<IGrouping<TKey, IEnumerable<TValue>>>
multyCollectionquery)
        {
            foreach (var group in multyCollectionquery)
            {
                Console.WriteLine(group.Key);
                foreach (var element in group)
                    Console.WriteLine(element);
            }
        }
    }
}
```

**Основна програма:**

Program.cs:

```
using laba1;
using System;

namespace laba1
{
```

```
internal class Program
{
    static void Main()
    {
        Console.OutputEncoding = System.Text.Encoding.Unicode;

        Queries qryExecutor = new();
        ConsolePrint printer = new();
        Data dataLists = new();

        PrintAndQueriesConnector printQryCon = new(qryExecutor, printer,
dataLists);

        Menu menu = new();
        menu.Items = new()
        {
            new MenuItem("1. Вивести усі машини з типом кузову SUV",
printQryCon.PrintAllVehicles),

            new MenuItem("2. Вивести усі машини, що мають тип кузову
SUV та відповідно їх власників", printQryCon.PrintAllVehiclesWithOwners),

            new MenuItem("3. Вивести перелік машин, що мають жахливі
технічні показники", printQryCon.PrintMachinesInExcellentCondition),

            new MenuItem("4. Вивести перелік власників, вік яких входить у
діапазон від 10 до 40 років", () =>
printQryCon.PrintOwnersWithinAgeRange(10, 40)),

            new MenuItem("5. Вивести водія, що має найдовше ім'я",
printQryCon.PrintDriverWithLongestName),

            new MenuItem("6. Вивести машину, що має найстаріший рік
випуску (найбільш старенька)",
printQryCon.PrintVehicleWithEarliestManufactureYear),
```

```
        new MenuItem("7. Вивести відсортований перелік водіїв за датою народження в порядку зростання (діапазон з 1990 до 2000 року народження)", () => printQryCon.PrintDriversSortedByDateOfBirthAscendingWithBoundaries(1990, 2000)),

        new MenuItem("8. Вивести усі машини, що не є зареєстрованими", printQryCon.PrintUnregisteredVehicles),

        new MenuItem("9. Вивести перелік машин, випущених до 2016 року", () => printQryCon.PrintMachinesReleasedBeforeYear(2016)),

        new MenuItem("10. Вивести машини, зареєстрованих в локації \"Local Registration Office\"", printQryCon.PrintAllVehiclesRegisteredInLocalOffice),

        new MenuItem("11. Вивести машини, що не мають водія", printQryCon.PrintAllVehiclesNotDrivenByDrivers),

        new MenuItem("12. Вивести перелік всіх унікальних власників машин та водіїв, об'єднавши їх імена", printQryCon.PrintUniqueOwnersAndDriversNames),

        new MenuItem("13. Вивести перелік власників, що мають декілька машин", printQryCon.PrintOwnersWithMultipleVehicles),

        new MenuItem("14. Вивести перелік машин, відсортовані за роком випуску (спадання, діапазон 2000-2017)", () => printQryCon.PrintVehiclesSortedByModelYearDescendingWithBoundaries(2000, 2017)),

        new MenuItem("15. Вивести перелік водіїв без машин", printQryCon.PrintDriversWithoutVehicles),

        new MenuItem("16. Вивести середній вік водіїв", printQryCon.PrintAverageAgeOfDrivers),
```

```
            new MenuItem("17. Вивести перелік водіїв молодше 30", () =>
printQryCon.PrintDriversYoungerThanAge(30)),

            new MenuItem("18. Вивести список автомобілів, що були
зареєстровані у 2023 році", () =>
printQryCon.PrintVehiclesRegisteredInYear(2023)),

            new MenuItem("19. Вивести список водіїв, що народилися у 1990
році", () => printQryCon.PrintDriversBornInYear(1990)),

            new MenuItem("20. Вивести список автомобілів, які належать
власнику з ID: 1", () => printQryCon.PrintCarsOwnedByOwner(1)),

            new MenuItem("\n21. Вихід", () => menu.IsExitWanted = true)
        };

        MenuItemSelector selector = new(1, menu.ItemsCount);

        while (!menu.IsExitWanted)
        {
            menu.PrintMenu();
            menu.ExecuteSelectedItem(selector.SelectItem());

            Console.WriteLine("\nДля продовження натисніть будь-яку
клавішу");
            Console.ReadKey();
            Console.Clear();
            Console.SetCursorPosition(0, 0);
            Console.Clear();

        }
    }
  }
}
```

# Приклад роботи програми

```
Головне меню

1. Вивести усі машини з типом кузову SUV
2. Вивести усі машини, що мають тип кузову SUV та відповідно їх власників
3. Вивести перелік машин, що мають жахливі технічні показники
4. Вивести перелік власників, вік яких входить у діапазон від 10 до 40 років
5. Вивести водія, що має найдовше ім'я
6. Вивести машину, що має найстаріший рік випуску (найбільш старенька)
7. Вивести відсортований перелік водіїв за датою народження в порядку зростання (діапазон з 1990 до 2000 року народження)
8. Вивести усі машини, що не є зареєстрованими
9. Вивести перелік машин, випущених до 2016 року
10. Вивести машини, зареєстрованих в локації "Local Registration Office"
11. Вивести машини, що не мають водія
12. Вивести перелік всіх унікальних власників машин та водіїв, об'єднавши їх імена
13. Вивести перелік власників, що мають декілька машин
14. Вивести перелік машин, відсортовані за роком випуску (спадання, діапазон 2000-2017)
15. Вивести перелік водіїв без машин
16. Вивести середній вік водіїв
17. Вивести перелік водіїв молодше 30
18. Вивести список автомобілів, що були зареєстровані у 2023 році
19. Вивести список водіїв, що народилися у 1990 році
20. Вивести список автомобілів, які належать власнику з ID: 1

21. Вихід

Оберіть пункт:
```

```
Ви обрали відсортовані водії за датою народження в порядку зростання з роками народження від 1990 до 2000


Driver ID: 4
Прізвище: Smith
Ім'я: John
По батькові: Michael
Дата народження: 05.10.1990
Адреса реєстрації: 123 Main St, City, Country
Номер водійського посвідчення: 1234567890

Driver ID: 7
Прізвище: Martinez
Ім'я: Daniel
По батькові: Carlos
Дата народження: 08.03.1991
Адреса реєстрації: 555 Elm St, City, Country
Номер водійського посвідчення: 6789054321

Driver ID: 6
Прізвище: Wilson
Ім'я: Michael
По батькові: Andrew
Дата народження: 15.05.1992
Адреса реєстрації: 333 Maple St, City, Country
Номер водійського посвідчення: 0123456789

Driver ID: 1
Прізвище: Doe
Ім'я: Jane
По батькові: Marie
Дата народження: 25.08.1993
Адреса реєстрації: 789 Oak St, City, Country
Номер водійського посвідчення: 1357924680

Driver ID: 3
Прізвище: Johnson
Ім'я: Emily
По батькові: Anne
Дата народження: 18.06.1995
Адреса реєстрації: 456 Elm St, City, Country
Номер водійського посвідчення: 0987654321

Для продовження натисніть будь-яку клавішу
```

## Висновок

У даній роботі було створено 20 різноманітних запитів, що включають у собі різні операції над даними, такі як групування, сортування, фільтрацію та об'єднання результатів. Також було використано обидва можливі варіанти реалізації LINQ-запитів, уникаючи повторень. Консольне програмне забезпечення було створено на мові C# і відповідає всім вимогам завдання. Було засвоєно розуміння обробки даних з використанням LINQ to Objects та відповідне її втілення у вигляді програмного забезпечення.