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

Практикум №1

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

на тему: «LINQ to Objects»

Викладач: Виконав:

Крамар Ю.М. студент 2 курсу

групи ІП-15 ФІОТ

Костін В.А.

Київ-2023

Комп‘ютерний практикум № 1.

**Тема:** LINQ to Objects

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

**Завдання:**

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-добагатьох і багато-до-багатьох.

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

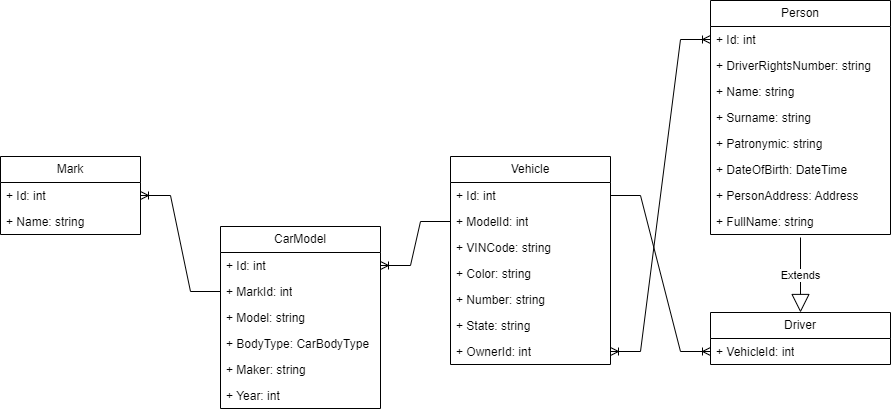
3) Створити програмне забезпечення, котре реалізує обробку даних з використання бібліотеки LINQ to Objects.

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.

2) 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).

Діаграма класів



**Клас:** Mark

**Призначення:** марка транспортного засобу

**Атрибути:** Id – ідентифікатор марки; name – назва марки.

**Клас:** CarModel

**Призначення:** модель транспортного засобу

**Атрибути:** Id – ідентифікатор моделі; MarkId – зовнішній ключ на марку транспортного засобу; Model – назва моделі; BodyType – тип кузову автомобіля; Maker - виробник; Year – рік випуску.

**Клас:** Vehicle

**Призначення:** транспортний засіб

**Атрибути:** Id – ідентифікатор транспортного засобу; ModelId – зовнішній ключ на модель; VINCode – номер шасі транспортного засобу; Color – колір транспортного засобу; Number – номерний знак транспортного засобу; State – стан транспортного засобу; OwnerId – зовнішній ключ на власника транспортного засобу.

**Клас:** Person

**Призначення:** людина(власник транспортного засобу)

**Атрибути:** Id – ідентифікатор людини; DriverRightsNumber – номер прав людини; Name – ім’я людини; Surname – прізвище людини; Patronymic – по-батькові людини; DateOfBirth – дата народження людини; PersonAddress – адреса людини; FullName – повне ім’я людини(в конструкторі або через Builder це поле не задається, а отримується в результаті конкатенації рядків ім’я, прізвища та по-батькові).

**Клас:** Driver(Унаслідується від класу Person).

**Призначення:** водій

**Атрибути:** VehicleId – зовнішній ключ на транспортний засіб, яким керує водій.

**Текст запитів**

Код програми можна переглянути в репозиторії: <https://github.com/VadimkaKostin/.Net>

1) Запит для фільрування водіїв по даті народження:

Drivers.Where(driver => driver.DateOfBirth.Year < 1990)

.Foreach(driver => Console.WriteLine("{0} {1}", driver.FullName, driver.DateOfBirth.ToShortDateString()));

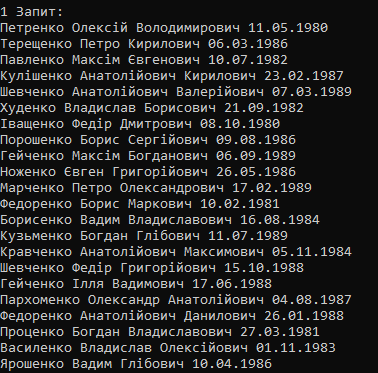


Рисунок 1.2 – результат 1-го запиту.

2) Запит для фільтрування транспортних засобів по даті випуску моделей:

var NewestVehicles = from vehicle in Vehicles

join model in CarModels

on vehicle.ModelId equals model.Id

where model.Year >= 2020

select vehicle;

NewestVehicles.Join(CarModels,

vehicle => vehicle.ModelId,

model => model.Id,

(vehicle, model) =>

new

{

vehicle.Id,

model.Year

})

.Foreach(vehicle => Console.WriteLine("{0} {1}", vehicle.Id, vehicle.Year));

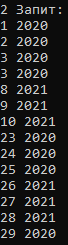


Рисунок 1.3 – результат 2-го запиту.

3) Запит для фільтрування транспортних засобів по марці автомобіля:

var ToyotaVehicles = from vehicle in Vehicles

join model in CarModels

on vehicle.ModelId equals model.Id

join mark in Marks

on model.MarkId equals mark.Id

where mark.Name == "Toyota"

select vehicle;

ToyotaVehicles.Join(CarModels,

vehicle => vehicle.ModelId,

model => model.Id,

(vehicle, model) =>

new

{

vehicle.Id,

model.Model,

model.MarkId

})

.Join(Marks,

vehicle => vehicle.MarkId,

mark => mark.Id,

(vehicle, mark) =>

new

{

Id = vehicle.Id,

Mark = mark.Name,

Model = vehicle.Model

})

.Foreach(vehicle => Console.WriteLine("{0} {1} {2}", vehicle.Id, vehicle.Mark, vehicle.Model));

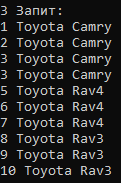


Рисунок 1.4 – результат 3-го запиту.

4) Запит сортування власників транспортних засобів за датою народження:

var ownersSorted = from owner in Owners

orderby owner.DateOfBirth

select new { Name = owner.Name, DateOfBirth = owner.DateOfBirth };

ownersSorted.Foreach(owner =>

Console.WriteLine($"{owner.Name} {owner.DateOfBirth.ToString().Split(' ')[0]}"));

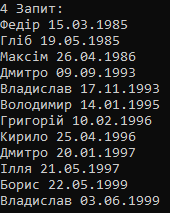


Рисунок 1.5 – результат 4-го запиту.

5) Запит сортування моделей авто по року випуску:

CarModels.OrderByDescending(model => model.Year)

.ThenBy(model => model.Model)

.Select(model => new { Name = model.Model, Year = model.Year })

.Foreach(model => Console.WriteLine($"{model.Name} {model.Year}"));

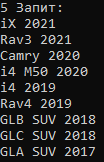


Рисунок 1.6 – результат 5-го запиту.

6) Запиит об'єднання моделей та марок в одну таблицю:

var ModelsWithMarks = from model in CarModels

join mark in Marks

on model.MarkId equals mark.Id

select new

{

Id = model.Id,

Mark = mark.Name,

Name = model.Model,

BodyType = model.BodyType,

Maker = model.Maker,

Year = model.Year

};

ModelsWithMarks.Foreach(model =>

Console.WriteLine($"{model.Id} {model.Mark} {model.Name} {model.BodyType} {model.Maker}" +

$"{model.Year}"));

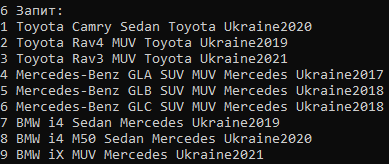


Рисунок 1.7 – результат 6-го запиту.

7) Запит для об'єднання транспортних засобів із моделями:

var VehiclesWithModels = Vehicles.Join(ModelsWithMarks,

vehicle => vehicle.ModelId,

model => model.Id,

(vehicle, model) =>

new

{

Id = vehicle.Id,

Mark = model.Mark,

Model = model.Name,

BodyType = model.BodyType,

Maker = model.Maker,

Year = model.Year,

VINCode = vehicle.VINCode,

Color = vehicle.Color,

Number = vehicle.Number,

State = vehicle.State,

OwnerId = vehicle.OwnerId

});

VehiclesWithModels.Foreach(vehicle =>

Console.WriteLine($"{vehicle.Id} {vehicle.Mark} {vehicle.Model} {vehicle.BodyType} " +

$"{vehicle.Maker} {vehicle.Year} {vehicle.VINCode} {vehicle.Color} {vehicle.Number} " +

$"{vehicle.State} {vehicle.OwnerId}"));

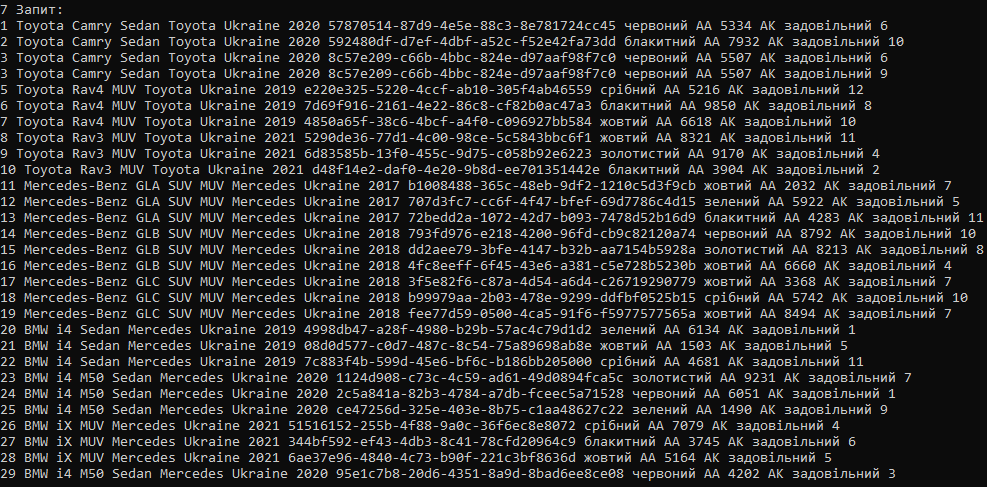


Рисунок 1.8 – результат 7-го запиту.

8) Запит для об'єднання трансопртних засобів і їх власників:

VehiclesWithModels.Join(Owners,

vehicle => vehicle.OwnerId,

owner => owner.Id,

(vehicle, owner) =>

new

{

Id = vehicle.Id,

Mark = vehicle.Mark,

Model = vehicle.Model,

BodyType = vehicle.BodyType,

Maker = vehicle.Maker,

Year = vehicle.Year,

VINCode = vehicle.VINCode,

Color = vehicle.Color,

Number = vehicle.Number,

State = vehicle.State,

Owner = owner.Name + " " + owner.Surname + " " + owner.Patronymic

})

.Foreach(vehicle =>

Console.WriteLine($"{vehicle.Id} {vehicle.Mark} {vehicle.Model} {vehicle.BodyType} " +

$"{vehicle.Maker} {vehicle.Year} {vehicle.VINCode} {vehicle.Color} {vehicle.Number} " +

$"{vehicle.State} {vehicle.Owner}"));

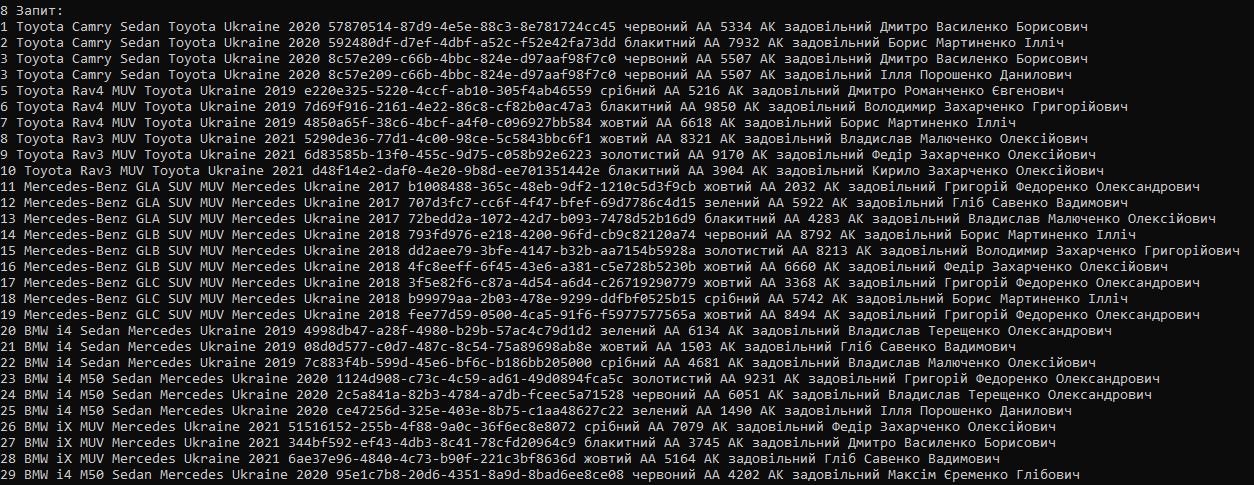


Рисунок 1.9 – результат 8-го запиту.

9) Запит для підрахунку кількості водіїв на кожний траспорт:

Drivers.GroupBy(driver => driver.VehicleId)

.Foreach(group =>

{

Console.WriteLine("Vehicle: {0} Drivers: {1}", group.Key, group.Count());

});

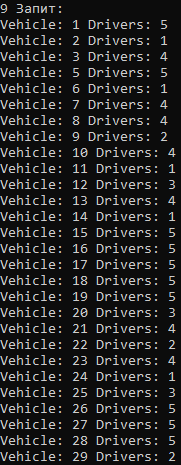


Рисунок 1.10 – результат 9-го запиту.

10) Запит для виведення транспортів та їх водіїв:

var vehiclesDriversAmountQuerry = from driver in Drivers

group driver by driver.VehicleId;

vehiclesDriversAmountQuerry.Foreach(group =>

{

Console.Write("Vehicle: " + group.Key + " Drivers: ");

group.Foreach(driver => Console.Write(driver.FullName + " | "));

Console.WriteLine();

});

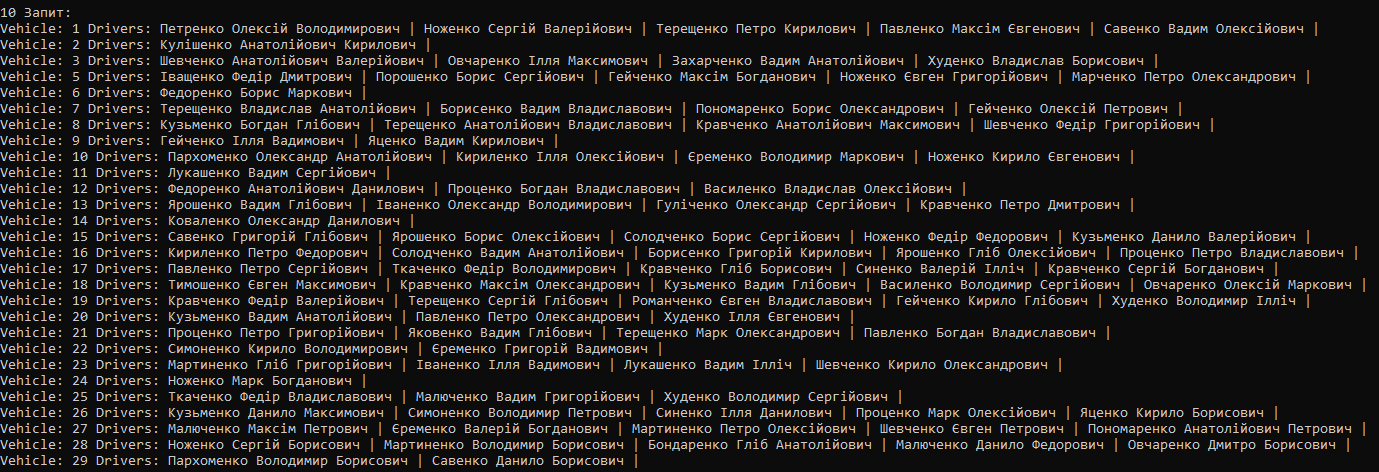


Рисунок 1.11 – результат 10-го запиту.

11) Запит для підрахунку кількості транспортних засобів у кожного власника:

VehiclesWithModels.GroupBy(vehicle => vehicle.OwnerId)

.Join(Owners, group => group.Key, owner => owner.Id, (group, owner) =>

new

{

Id = group.Key,

Name = owner.FullName,

Amount = group.Count()

})

.OrderByDescending(owner => owner.Amount)

.ThenBy(owner => owner.Name)

.Foreach(owner => Console.WriteLine("Owner: {0} {1} Vehicles: {2}", owner.Id, owner.Name, owner.Amount));

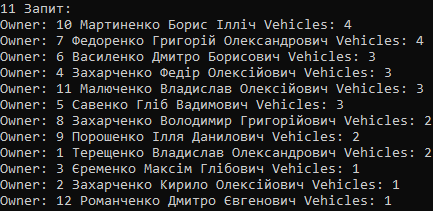


Рисунок 1.12 – результат 11-го запиту.

12) Запит для підрахунку кількості власників у кожного тарнспортного засобу:

VehiclesWithModels.GroupBy(vehicle => vehicle.Id)

.Select(group => new { Id = group.Key, Amount = group.Count() })

.Foreach(vehicle => Console.WriteLine("Vehicle: {0} Owners: {1}", vehicle.Id, vehicle.Amount));

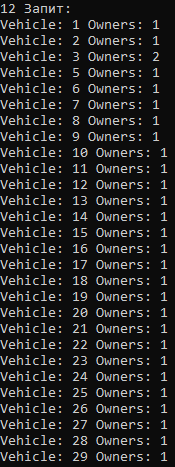


Рисунок 1.13 – результат 12-го запиту.

13) Запит для виведення перших 5 власників із найбільшою кількістю транспортних засобів:

int counter = 1;

VehiclesWithModels.GroupBy(vehicle => vehicle.OwnerId)

.Select(group => new { Id = group.Key, Amount = group.Count() })

.Join(Owners, group => group.Id, owner => owner.Id,

(group, owner) => new

{

Id = owner.Id,

Name = owner.FullName,

Amount = group.Amount

})

.OrderByDescending(owner => owner.Amount)

.Take(5)

.Foreach(owner =>

{

Console.WriteLine(counter++ + ")" + owner.Name + " " + owner.Amount);

});

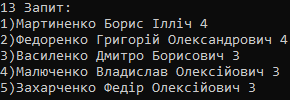


Рисунок 1.14 – результат 13-го запиту.

14) Запит для виведення другого за алфавітом водія:

var SecondDriver = Drivers.OrderBy(driver => driver.FullName)

.Take(2)

.Skip(1)

.First();

Console.WriteLine(SecondDriver.FullName);



Рисунок 1.15 – результат 14-го запиту.

15) Запит для виведення транспортних засобів, у яких більше 3 водіїв:

var VehiclesWithDrivers =

from groupDriver in

(from driver in Drivers group driver by driver.VehicleId)

where groupDriver.Count() > 3

join groupVehicle in (from vehicle in VehiclesWithModels group vehicle by vehicle.Id)

on groupDriver.Key equals groupVehicle.Key

let Amount = groupDriver.Count()

orderby Amount descending

select new

{

Id = groupVehicle.Key,

Amount

};

VehiclesWithDrivers.Foreach(vehicle =>

{

Console.WriteLine($"Vehicle:{vehicle.Id} Drivers: {vehicle.Amount}");

});

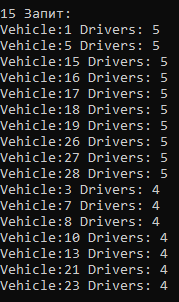


Рисунок 1.16 – результат 15-го запиту.

16) Запит для об'єднання колекцій автомобілів з фільтуванням по марці та року випуску моделі:

ToyotaVehicles.Union(NewestVehicles)

.OrderBy(vehicle => vehicle.Id)

.Foreach(vehicle => Console.WriteLine(vehicle.Id));

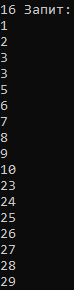


Рисунок 1.17 – результат 16-го запиту.

17) Запит для об'єднання колекцій автомобілів з фільтуванням по марці та року випуску моделі через конкатенацію та видалення дублікатів. Результат має вийти той самий, що і в минулому запиті:

ToyotaVehicles.Concat(NewestVehicles)

.Distinct()

.OrderBy(vehicle => vehicle.Id)

.Foreach(vehicle => Console.WriteLine(vehicle.Id));

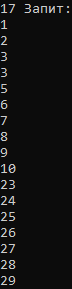


Рисунок 1.18 – результат 17-го запиту.

18) Запит для перевірки чи є хочаб один водій який живе на певній вулиці:

bool result = Drivers.Any(driver =>

driver.PersonAddress.Street == "вул.Велика Васильківська");

Console.WriteLine(result);



Рисунок 1.19 – результат 18-го запиту.

19) Запит для перевірки чи всі власники народились до 2000 року:

result = Owners.All(owner => owner.DateOfBirth.Year < 2000);

Console.WriteLine(result);



Рисунок 1.20 – результат 19-го запиту.

20) Запит для підрахунку середнього віку усіх водіїв:

double average = Drivers.Average(driver => GetAge(driver.DateOfBirth));

Console.WriteLine(Math.Round(average, 2));



Рисунок 1.21 – результат 20-го запиту.

**Висновок**

Протягом першого комп’ютерного практикуму ми ознайомитися з обробкою даних з використанням бібліотеки LINQ to Objects. Для створення запитів використовувались класичні варіанти запитів та запити у формі методів розширення. Під час виконання були використанні запити LINQ наступних груп операцій: фільтри, проекція, сортування, групування, операції над множинами, елементи, квантори, агрегатори.