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

Звіт

з лабораторної роботи № 2 з дисципліни
«Основи розробки програмного забезпечення на платформі Microsoft.NET»

«LINQ to Objects»

Варіант 5

Виконав студент ІП-13 Григоренко Родіон Ярославович
(шифр, прізвище, ім'я, по батькові)

Перевірила Ліщук Катерина Ігорівна
(прізвище, ім'я, по батькові)

Київ 2023

Лабораторна робота 2

LINQ to XML

Мета: - ознайомитися з обробкою XML документів з використанням технології LINQ to XML

Постановка задачі:

При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1) Розробити структуру XML для зберігання даних згідно варіантів, наведених нижче.

2) Створити XML-файл з використанням XmlWriter. Дані необхідно вводити з консолі, зберегти.

3) Завантажити створені файли (або заздалегідь створені) з використанням XmlDocument та XmlSerializer.

4) LINQ to XML:

a. Вивести зміст файлу, створеного в п.2

b. Для файлу, створеного в п.2 розробити як мінімум 15 різних запитів, використовуючи різні дії над отриманими даними. Запити не повинні повторюватись.

5) Створити програмне забезпечення, котре реалізує обробку даних з використання бібліотеки LINQ to XML.

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

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

Варіант 5

Розробити структуру даних для зберігання інформації для відстеження фінансових показників роботи пункту прокату автомобілів. В автопарк компанії входить кілька автомобілів різних марок, вартостей і типів. Кожен автомобіль має свою ціну прокату. В пункт прокату звертаються клієнти. Всі клієнти проходять обов'язкову реєстрацію - про них збирається стандартна інформація (ПІБ, адреса, телефон). Кожен клієнт може звертатися в пункт прокату кілька разів. Всі звернення клієнтів фіксуються, при цьому по кожній угоді запам'ятовуються дата видачі та

очікувана дата повернення. Перед отриманням автомобіля клієнт залишає деяку заставну суму, яка йому повністю повертається після успішного повернення автомобіля. Вартість прокату автомобіля залежить не тільки від самого автомобіля, але і від терміну його прокату, а також від року випуску.

1. Архітектура Проекту.

Виконуючи другу лабораторну роботу, було створено консольне застосування на мові C#, котре реалізує обробку даних з використанням бібліотеки LINQ to XML. Було створено 5 класів, а саме:

Program.cs - власне клас програми з виконуваною функцією Main

Agency.cs - клас агенції який містить списки автівок, клієнтів та угод

Auto.cs - клас, який відображає сутність автомобіль

Client.cs - клас, який відображає сутність Клієнт

Deal.cs - клас, який відображає сутність Угода

Код класів:

Agency.cs

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

```
namespace Lab2DotNet
{
    [XmlRoot("Agency")]
    public class Agency
    {
        [XmlElement("AutoPark")]
        public List<Auto> AutoPark = new List<Auto>();

        [XmlElement("Clients")]
        public List<Client> Clients = new List<Client>();

        [XmlElement("Deals")]
        public List<Deal> Deals = new List<Deal>();
    }
}
```

Auto.cs

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

namespace Lab2DotNet
{
    public class Auto
```

```
    {
        public string Brand;
        public string Model;
        public int Cost;
        public string Type;
        public int RentalPrice;
        public int YearOfProduction;
        public int RentalPeriod;

        public Auto() { }
        public Auto(string brand, string model, int cost, string type, int
yearOfProduction, int rentalPeriod)
        {
            Brand = brand;
            Model = model;
            Cost = cost;
            Type = type;
            YearOfProduction = yearOfProduction;
            RentalPeriod = rentalPeriod;
            RentalPrice = (int)(((float)Cost / 100f) * (1f - (float)rentalPeriod /
5000f) * (((float)YearOfProduction - 1970f) / 53f));
        }
    }
}
```

Client.cs

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

namespace Lab2DotNet
{
    public class Client
    {
        public string Name;
        public string Address;
        public string PhoneNumber;

        public Client() { }
        public Client(string name, string address, string phoneNumber)
        {
            Name = name;
            Address = address;
            PhoneNumber = phoneNumber;
        }
    }
}
```

Deal.cs

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

namespace Lab2DotNet
{
    public class Deal
    {
        public Client Client;
        public Auto Auto;
        public DateTime DateOfIssue;
        public DateTime DateOfReturn;

        public Deal() { }
        public Deal(Client client, Auto auto, DateTime dateOfIssue, DateTime
dateOfReturn)
        {
            Client = client;
            Auto = auto;
            DateOfIssue = dateOfIssue;
            DateOfReturn = dateOfReturn;
        }
    }
}
```

Program.cs

```
using System;
using System.Linq;
using System.Xml;
using System.Xml.Linq;
using System.Xml.Serialization;
using System.IO;
using System.Text;


namespace Lab2DotNet
{
    class Program
    {
        static void Main(string[] args)
        {
            Agency agency = new Agency();


            Auto auto1 = new Auto("Ford", "Raptor", 150000, "off-road", 2023, 0);
            Auto auto2 = new Auto("Mercedes", "G-Class", 200000, "off-road",
2023, 0);
            Auto auto3 = new Auto("Audi", "RS Q8", 210000, "off-road", 2023, 0);
            Auto auto4 = new Auto("BMW", "X6", 112000, "crossover", 2020,
500);
            Auto auto5 = new Auto("Volkswagen", "Touareg", 25000, "crossover",
2017, 1000);
```

```
        Auto auto6 = new Auto("Lada", "Priora", 10000, "sedan", 2008, 2000);
        Auto auto7 = new Auto("Nissan", "Qashqai", 14500, "crossover", 2017,
1500);
        Auto auto8 = new Auto("Toyota", "Corolla", 7000, "sedan", 2010,
3000);
        Auto auto9 = new Auto("Chrysler", "300 C", 11000, "sedan", 2008,
2300);
        Auto auto10 = new Auto("Porsche", "Cayenne", 160000, "crossover",
2023, 0);


        agency.AutoPark.Add(auto1);
        agency.AutoPark.Add(auto2);
        agency.AutoPark.Add(auto3);
        agency.AutoPark.Add(auto4);
        agency.AutoPark.Add(auto5);
        agency.AutoPark.Add(auto6);
        agency.AutoPark.Add(auto7);
        agency.AutoPark.Add(auto8);
        agency.AutoPark.Add(auto9);
        agency.AutoPark.Add(auto10);


        Client client1 = new Client("Radulf Schulte", "Mecklenburg-
Vorpommern Greifswald Kurfürstendamm 23", "+49 0383 37 68 78");
        Client client2 = new Client("Dietrich Frei", "Niedersachsen Werpeloh
Rudolstaedter Strasse 45", "+49 05952 53 26 46");
        Client client3 = new Client("Helmfried Schreier", "Baden-Württemberg
Heilbronn Sontheim Kurfuerstendamm 85", "+49 079 99 33 63");
        Client client4 = new Client("Leon Nagel", "Nordrhein-Westfalen
Oberhausen Stadtmitte Friedrichstrasse 36", "+49 0208 77 56 75");
        Client client5 = new Client("Claus Esser", "Hamburg  Hamburg
Bergedorf Waßmannsdorfer Chaussee 69", "+49 040 44 81 77");
```

```
Client client6 = new Client("Karl Stark", "Rheinland-Pfalz Meckenheim Invalidenstrasse 65", "+49 06326 49 53 30");

Client client7 = new Client("Emmerich Schumacher", "Niedersachsen Norden Kastanienallee 68", "+49 04931 60 37 51");

Client client8 = new Client("Wilfried Knopf", "Baden-Württemberg Pforzheim Buckenberg Pasewalker Straße 38", "+49 07231 59 54 52");

Client client9 = new Client("Astor Arnold", "Freistaat Thüringen Saalfeld Eschenweg 6", "+49 03671 26 79 41");

Client client10 = new Client("Mathis Groß", "Freistaat Bayern Biberbach Alt Reinickendorf 3", "+49 08271 22 11 44");


agency.Clients.Add(client1);

agency.Clients.Add(client2);

agency.Clients.Add(client3);

agency.Clients.Add(client4);

agency.Clients.Add(client5);

agency.Clients.Add(client6);

agency.Clients.Add(client7);

agency.Clients.Add(client8);

agency.Clients.Add(client9);

agency.Clients.Add(client10);


Deal deal1 = new Deal(client8, auto2, new DateTime(2021, 6, 18), new DateTime(2021, 6, 20));

Deal deal2 = new Deal(client3, auto7, new DateTime(2020, 5, 18), new DateTime(2020, 6, 10));

Deal deal3 = new Deal(client5, auto4, new DateTime(2020, 9, 16), new DateTime(2020, 9, 28));

Deal deal4 = new Deal(client7, auto6, new DateTime(2020, 6, 7), new DateTime(2020, 8, 8));

Deal deal5 = new Deal(client9, auto5, new DateTime(2020, 3, 26), new DateTime(2020, 7, 25));
```

```
        Deal deal6 = new Deal(client2, auto9, new DateTime(2022, 5, 15), new DateTime(2022, 6, 20));
        Deal deal7 = new Deal(client4, auto3, new DateTime(2020, 3, 17), new DateTime(2020, 3, 29));
        Deal deal8 = new Deal(client1, auto8, new DateTime(2022, 2, 5), new DateTime(2023, 1, 1));
        Deal deal9 = new Deal(client10, auto10, new DateTime(2021, 6, 18), new DateTime(2021, 7, 5));
        Deal deal10 = new Deal(client6, auto1, new DateTime(2021, 1, 31), new DateTime(2021, 3, 10));

        agency.Deals.Add(deal1);
        agency.Deals.Add(deal2);
        agency.Deals.Add(deal3);
        agency.Deals.Add(deal4);
        agency.Deals.Add(deal5);
        agency.Deals.Add(deal6);
        agency.Deals.Add(deal7);
        agency.Deals.Add(deal8);
        agency.Deals.Add(deal9);
        agency.Deals.Add(deal10);

        XmlSerializer serializer = new XmlSerializer(typeof(Agency));

        using(TextWriter writer = new StreamWriter("data.xml"))
        {
            serializer.Serialize(writer, agency);
        }
```

```
        using (TextReader reader = new StreamReader("data.xml"))
        {
            Agency ag = (Agency)serializer.Deserialize(reader);

            Console.WriteLine("\nAutos:\n");

            foreach(Auto auto in ag.AutoPark)
            {

Console.WriteLine($"\t{auto.Brand}\t{auto.Model}\t{auto.Cost}\t{auto.Type}\
t{auto.YearOfProduction}\t{auto.RentalPrice}");

            }

            Console.WriteLine("\nClients:\n");

            foreach (Client client in ag.Clients)
            {

Console.WriteLine($"\t{client.Name}\t{client.Address}\t{client.PhoneNumber
}");

            }
        }



        XDocument data = XDocument.Load("data.xml");

            //this query selects all clients' names
            Console.WriteLine("1.Clients' names:\n");
```

```
        var query1 = from c in data.Descendants("Clients") select
c.Element("Name").Value;
        foreach (var name in query1) Console.WriteLine($"\t{name}");

        //this query selects all autos' names and rental prices
        Console.WriteLine("\n2.Autos and their rental prices");
        var query2 = from a in data.Descendants("AutoPark")
                     select new
                     {
                        brand = a.Element("Brand").Value,
                        model = a.Element("Model").Value,
                        rentalPrice = a.Element("RentalPrice").Value
                     };
        foreach (var auto in query2)
Console.WriteLine($"\t{auto.brand}\t{auto.model}\t{auto.rentalPrice}");

        //this query selects all autos whose type is crossover
        Console.WriteLine("\n3.Autos whose type is crossover:");
        var query3 = data.Descendants("AutoPark").Where(a =>
a.Element("Type").Value == "crossover");
           agency.AutoPark.Select(a => a).Where(a => a.Type == "crossover");
        foreach (var auto in query3)
Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model
").Value}\t{auto.Element("Type").Value}");

        //this query calculates and selects average rental price for all autos
        Console.WriteLine("\n4.Average rental price");
        var query4 = data.Descendants("AutoPark").Average(a =>
Convert.ToInt32(a.Element("RentalPrice").Value));
        Console.WriteLine($"\t{Math.Round(query4, 2)}");
```

```
        //this query selects all autos and their costs
        Console.WriteLine("\n5.Autos and their costs:");
        var query5 = from a in data.Descendants("AutoPark")
                     orderby Convert.ToInt32(a.Element("Cost").Value) descending
                     select new
                     {
                        brand = a.Element("Brand").Value,
                        model = a.Element("Model").Value,
                        cost = a.Element("Cost").Value
                     };
        foreach (var auto in query5)
Console.WriteLine($"\t{auto.brand}\t{auto.model}\t{auto.cost}");


        //this query selects all clients' names and phone numbers that begins
with +49
        Console.WriteLine("\n6.Clients whose phone numbers starts with +49");
        var query6 = data.Descendants("Clients").Where(n =>
n.Element("PhoneNumber").Value.StartsWith("+49"));
        foreach (var client in query6)
Console.WriteLine($"\t{client.Element("Name").Value}\t{client.Element("Pho
neNumber").Value}");


        //this query group autos by types and calculates their amount in each
type
        Console.WriteLine("\n7.Amount of autos by types:");
        var query7 = from a in data.Descendants("AutoPark")
                     group a by a.Element("Type").Value into type
                     select new
                     {
```

```
                    name = type.Key,
                    amount = type.Count()
                };
        foreach (var item in query7)
Console.WriteLine($"\t{item.name}\t{item.amount}");

        //this query selects all autos and their year of production
        Console.WriteLine("\n8.Autos and their years of production:");
        var query8 = data.Descendants("AutoPark").OrderByDescending(c =>
Convert.ToInt32(c.Element("Cost").Value)).
            OrderByDescending(y =>
Convert.ToInt32(y.Element("YearOfProduction").Value));
        foreach (var auto in query8)

Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model
").Value}\t{auto.Element("YearOfProduction").Value}");

        //this query selects dates of issues and returns of all deals
        Console.WriteLine("\n9.List of deals(dates of issue and return)");
        var query9 = from d in data.Descendants("Deals")
                select new
                {
                    dateOfIssue = d.Element("DateOfIssue").Value,
                    dateOfReturn = d.Element("DateOfReturn").Value
                };
        foreach (var deal in query9)
Console.WriteLine($"\t{deal.dateOfIssue}\t{deal.dateOfReturn}");

        //this query selects clients and autos they rent
        Console.WriteLine("\n10.Clients and autos they rent");
```

```
        var query10 = from a in data.Descendants("AutoPark")
                      from c in data.Descendants("Clients")
                      from d in data.Descendants("Deals")
                      where (d.Element("Auto").Value == a.Value &&
d.Element("Client").Value == c.Value)
                      select new
                      {
                         name = c.Element("Name").Value,
                         brand = a.Element("Brand").Value,
                         model = a.Element("Model").Value
                      };
        foreach (var item in query10)
Console.WriteLine($"\t{item.name}\t{item.brand}\t{item.model}");


        //this query selects autos whose cost is over 100000
        Console.WriteLine("\n11.Autos whose cost is over 100000:");
        var query11 = data.Descendants("AutoPark").Where(c =>
Convert.ToInt32(c.Element("Cost").Value) > 100000);
        foreach (var auto in query11)

Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model
").Value}\t{auto.Element("Cost").Value}");


        //this query selects all clients and put them in alphabetical order by their
names
        Console.WriteLine("\n12.List of clients in alphabetical order:");
        var query12 = from c in data.Descendants("Clients")
                      orderby c.Element("Name").Value
                      select new
                      {
```

```
                    name = c.Element("Name").Value
                };
        foreach (var client in query12) Console.WriteLine($"\t{client.name}");

        //this query selects all autos and put them in descending order by their
costs
        Console.WriteLine("\n13.List of autos in their costs descending order:");
        var query13 = data.Descendants("AutoPark").OrderByDescending(c =>
c.Element("Cost").Value);
        foreach (var auto in query13)

Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model
").Value}\t{auto.Element("Cost").Value}");

        //this query selects all clients from Niedersachsen
        Console.WriteLine("\n14.Clients in Niedersachsen Land:");
        var query14 = from c in data.Descendants("Clients")
                where c.Element("Address").Value.Contains("Niedersachsen")
                select new
                {
                    name = c.Element("Name").Value,
                    address = c.Element("Address").Value
                };
        foreach (var client in query14)
Console.WriteLine($"\t{client.name}\t{client.address}");

        //this query group all autos by their year of production and calculates
their amount in each group
        Console.WriteLine("\n15.Autos grouped by their year of production:");
        var query15 = from a in data.Descendants("AutoPark")
```

```
                group a by a.Element("YearOfProduction").Value into year
                select new
                {
                    name = year.Key,
                    amount = year.Count()
                };
        foreach (var item in query15)
Console.WriteLine($"\t{item.name}\t{item.amount}");


    }
  }
}
```

2.Запис даних у файл

```
XmlSerializer serializer = new XmlSerializer(typeof(Agency));

using(TextWriter writer = new StreamWriter("data.xml"))
{
    serializer.Serialize(writer, agency);
}
```

3.Вивід даних з файлу

```
using (TextReader reader = new StreamReader("data.xml"))
{
    Agency ag = (Agency)serializer.Deserialize(reader);

    Console.WriteLine("\nAutos:\n");

    foreach(Auto auto in ag.AutoPark)
    {
        Console.WriteLine($"\t{auto.Brand}\t{auto.Model}\t{auto.Cost}\t{auto.Type}\t{auto.YearOfProduction}\t{auto.RentalPrice}");

    }

    Console.WriteLine("\nClients:\n");

    foreach (Client client in ag.Clients)
    {
        Console.WriteLine($"\t{client.Name}\t{client.Address}\t{client.PhoneNumber}");

    }
}
```

```
Autos:

        Ford    Raptor  150000  off-road        2023    1500
        Mercedes        G-Class 200000  off-road        2023    2000
        Audi    RS Q8   210000  off-road        2023    2100
        BMW     X6      112000  crossover       2020    950
        Volkswagen      Touareg 25000   crossover       2017    177
        Lada    Priora  10000   sedan   2008    43
        Nissan  Qashqai 14500   crossover       2017    90
        Toyota  Corolla 7000    sedan   2010    21
        Chrysler        300 C   11000   sedan   2008    42
        Porsche Cayenne 160000  crossover       2023    1600

Clients:

        Radulf Schulte  Mecklenburg-Vorpommern Greifswald Kurfurstendamm 23     +49 0383 37 68 78
        Dietrich Frei   Niedersachsen Werpeloh Rudolstaedter Strasse 45 +49 05952 53 26 46
        Helmfried Schreier      Baden-Wurttemberg Heilbronn Sontheim Kurfuerstendamm 85 +49 079 99 33 63
        Leon Nagel      Nordrhein-Westfalen Oberhausen Stadtmitte Friedrichstrasse 36   +49 0208 77 56 75
        Claus Esser     Hamburg  Hamburg Bergedorf Wa?mannsdorfer Chaussee 69   +49 040 44 81 77
        Karl Stark      Rheinland-Pfalz Meckenheim  Invalidenstrasse 65 +49 06326 49 53 30
        Emmerich Schumacher     Niedersachsen Norden Kastanienallee 68  +49 04931 60 37 51
        Wilfried Knopf  Baden-Wurttemberg Pforzheim Buckenberg Pasewalker Stra?e 38     +49 07231 59 54 52
        Astor Arnold    Freistaat Thuringen Saalfeld Eschenweg 6        +49 03671 26 79 41
        Mathis Gro?     Freistaat Bayern Biberbach Alt Reinickendorf 3  +49 08271 22 11 44
```

4.Запити

1. this query selects all clients' names

```
Console.WriteLine("1.Clients' names:\n");
var query1 = from c in data.Descendants("Clients") select c.Element("Name").Value;
foreach (var name in query1) Console.WriteLine($"\t{name}");
```

```
1.Clients' names:

        Radulf Schulte
        Dietrich Frei
        Helmfried Schreier
        Leon Nagel
        Claus Esser
        Karl Stark
        Emmerich Schumacher
        Wilfried Knopf
        Astor Arnold
        Mathis Gro?
```

2. this query selects all autos' names and rental prices

```
Console.WriteLine("\n2.Autos and their rental prices");
var query2 = from a in data.Descendants("AutoPark")
             select new
             {
                 brand = a.Element("Brand").Value,
                 model = a.Element("Model").Value,
                 rentalPrice = a.Element("RentalPrice").Value
             };
foreach (var auto in query2) Console.WriteLine($"\t{auto.brand}\t{auto.model}\t{auto.rentalPrice}");
```

```
2.Autos and their rental prices
        Ford    Raptor  1500
        Mercedes        G-Class 2000
        Audi    RS Q8   2100
        BMW     X6      950
        Volkswagen      Touareg 177
        Lada    Priora  43
        Nissan  Qashqai 90
        Toyota  Corolla 21
        Chrysler        300 C   42
        Porsche Cayenne 1600
```

3. this query selects all autos whose type is crossover

```
Console.WriteLine("\n3.Autos whose type is crossover:");
var query3 = data.Descendants("AutoPark").Where(a => a.Element("Type").Value == "crossover");
    agency.AutoPark.Select(a => a).Where(a => a.Type == "crossover");
foreach (var auto in query3)
    Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model").Value}\t{auto.Element("Type").Value}");
```

```
3.Autos whose type is crossover:
        BMW     X6      crossover
        Volkswagen      Touareg crossover
        Nissan  Qashqai crossover
        Porsche Cayenne crossover
```

4. this query calculates and selects average rental price for all autos

```
Console.WriteLine("\n4.Average rental price");
var query4 = data.Descendants("AutoPark").Average(a => Convert.ToInt32(a.Element("RentalPrice").Value));
Console.WriteLine($"\t{Math.Round(query4, 2)}");
```

```
4.Average rental price
        852,3
```

5. this query selects all autos and their costs

```
Console.WriteLine("\n5.Autos and their costs:");
var query5 = from a in data.Descendants("AutoPark")
             orderby Convert.ToInt32(a.Element("Cost").Value) descending
             select new
             {
                 brand = a.Element("Brand").Value,
                 model = a.Element("Model").Value,
                 cost = a.Element("Cost").Value
             };
foreach (var auto in query5) Console.WriteLine($"\t{auto.brand}\t{auto.model}\t{auto.cost}");
```

```
5.Autos and their costs:
        Audi    RS Q8   210000
        Mercedes        G-Class 200000
        Porsche Cayenne 160000
        Ford    Raptor  150000
        BMW     X6      112000
        Volkswagen      Touareg 25000
        Nissan  Qashqai 14500
        Chrysler        300 C   11000
        Lada    Priora  10000
        Toyota  Corolla 7000
```

6. this query selects all clients' names and phone numbers that begins with +49

```
Console.WriteLine("\n6.Clients whose phone numbers starts with +49");
var query6 = data.Descendants("Clients").Where(n => n.Element("PhoneNumber").Value.StartsWith("+49"));
foreach (var client in query6) Console.WriteLine($"\t{client.Element("Name").Value}\t{client.Element("PhoneNumber").Value}");
```

```
6.Clients whose phone numbers starts with +49
        Radulf Schulte  +49 0383 37 68 78
        Dietrich Frei   +49 05952 53 26 46
        Helmfried Schreier      +49 079 99 33 63
        Leon Nagel      +49 0208 77 56 75
        Claus Esser     +49 040 44 81 77
        Karl Stark      +49 06326 49 53 30
        Emmerich Schumacher     +49 04931 60 37 51
        Wilfried Knopf  +49 07231 59 54 52
        Astor Arnold    +49 03671 26 79 41
        Mathis Gro?     +49 08271 22 11 44
```

7. this query group autos by types and calculates their amount in each type

```
Console.WriteLine("\n7.Amount of autos by types:");
var query7 = from a in data.Descendants("AutoPark")
             group a by a.Element("Type").Value into type
             select new
             {
                 name = type.Key,
                 amount = type.Count()
             };
foreach (var item in query7) Console.WriteLine($"\t{item.name}\t{item.amount}");
```

```
7.Amount of autos by types:
        off-road        3
        crossover       4
        sedan   3
```

8. this query selects all autos and their year of production

```
var query8 = data.Descendants("AutoPark").OrderByDescending(c => Convert.ToInt32(c.Element("Cost").Value)).
    OrderByDescending(y => Convert.ToInt32(y.Element("YearOfProduction").Value));
foreach (var auto in query8)
    Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model").Value}\t{auto.Element("YearOfProduction").Value}");
```

```
8.Autos and their years of production:
        Audi    RS Q8   2023
        Mercedes        G-Class 2023
        Porsche Cayenne 2023
        Ford    Raptor  2023
        BMW     X6      2020
        Volkswagen      Touareg 2017
        Nissan  Qashqai 2017
        Toyota  Corolla 2010
        Chrysler        300 C   2008
        Lada    Priora  2008
```

9. this query selects dates of issues and returns of all deals

```
Console.WriteLine("\n9.List of deals(dates of issue and return)");
var query9 = from d in data.Descendants("Deals")
             select new
             {
                 dateOfIssue = d.Element("DateOfIssue").Value,
                 dateOfReturn = d.Element("DateOfReturn").Value
             };
foreach (var deal in query9) Console.WriteLine($"\t{deal.dateOfIssue}\t{deal.dateOfReturn}");
```

```
9.List of deals(dates of issue and return)
        2021-06-18T00:00:00     2021-06-20T00:00:00
        2020-05-18T00:00:00     2020-06-10T00:00:00
        2020-09-16T00:00:00     2020-09-28T00:00:00
        2020-06-07T00:00:00     2020-08-08T00:00:00
        2020-03-26T00:00:00     2020-07-25T00:00:00
        2022-05-15T00:00:00     2022-06-20T00:00:00
        2020-03-17T00:00:00     2020-03-29T00:00:00
        2022-02-05T00:00:00     2023-01-01T00:00:00
        2021-06-18T00:00:00     2021-07-05T00:00:00
        2021-01-31T00:00:00     2021-03-10T00:00:00
```

10. this query selects clients and autos they rent

```
Console.WriteLine("\n10.Clients and autos they rent");
var query10 = from a in data.Descendants("AutoPark")
              from c in data.Descendants("Clients")
              from d in data.Descendants("Deals")
              where (d.Element("Auto").Value == a.Value && d.Element("Client").Value == c.Value)
              select new
              {
                  name = c.Element("Name").Value,
                  brand = a.Element("Brand").Value,
                  model = a.Element("Model").Value
              };
foreach (var item in query10) Console.WriteLine($"\t{item.name}\t{item.brand}\t{item.model}");
```

```
10.Clients and autos they rent
        Karl Stark      Ford    Raptor
        Wilfried Knopf  Mercedes        G-Class
        Leon Nagel      Audi    RS Q8
        Claus Esser     BMW     X6
        Astor Arnold    Volkswagen      Touareg
        Emmerich Schumacher     Lada    Priora
        Helmfried Schreier      Nissan  Qashqai
        Radulf Schulte  Toyota  Corolla
        Dietrich Frei   Chrysler        300 C
        Mathis Gro?     Porsche Cayenne
```

11. this query selects autos whose cost is over 100000

```
Console.WriteLine("\n11.Autos whose cost is over 100000:");
var query11 = data.Descendants("AutoPark").Where(c => Convert.ToInt32(c.Element("Cost").Value) > 100000);
foreach (var auto in query11)
    Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model").Value}\t{auto.Element("Cost").Value}");
```

```
11.Autos whose cost is over 100000:
        Ford    Raptor  150000
        Mercedes        G-Class 200000
        Audi    RS Q8   210000
        BMW     X6      112000
        Porsche Cayenne 160000
```

12. this query selects all clients and put them in alphabetical order by their names

```
Console.WriteLine("\n12.List of clients in alphabetical order:");
var query12 = from c in data.Descendants("Clients")
              orderby c.Element("Name").Value
              select new
              {
                  name = c.Element("Name").Value
              };
foreach (var client in query12) Console.WriteLine($"\t{client.name}");
```

```
12.List of clients in alphabetical order:
        Astor Arnold
        Claus Esser
        Dietrich Frei
        Emmerich Schumacher
        Helmfried Schreier
        Karl Stark
        Leon Nagel
        Mathis Gro?
        Radulf Schulte
        Wilfried Knopf
```

13. this query selects all autos and put them in descending order by their costs

```
Console.WriteLine("\n13.List of autos in their costs descending order:");
var query13 = data.Descendants("AutoPark").OrderByDescending(c => c.Element("Cost").Value);
foreach (var auto in query13)
    Console.WriteLine($"\t{auto.Element("Brand").Value}\t{auto.Element("Model").Value}\t{auto.Element("Cost").Value}");
```

```
13.List of autos in their costs descending order:
        Toyota  Corolla 7000
        Volkswagen      Touareg 25000
        Audi    RS Q8   210000
        Mercedes        G-Class 200000
        Porsche Cayenne 160000
        Ford    Raptor  150000
        Nissan  Qashqai 14500
        BMW     X6      112000
        Chrysler        300 C   11000
        Lada    Priora  10000
```

14. this query selects all clients from Niedersachsen

```
Console.WriteLine("\n14.Clients in Niedersachsen Land:");
var query14 = from c in data.Descendants("Clients")
              where c.Element("Address").Value.Contains("Niedersachsen")
              select new
              {
                  name = c.Element("Name").Value,
                  address = c.Element("Address").Value
              };
foreach (var client in query14) Console.WriteLine($"\t{client.name}\t{client.address}");
```

```
14.Clients in Niedersachsen Land:
        Dietrich Frei   Niedersachsen Werpeloh Rudolstaedter Strasse 45
        Emmerich Schumacher     Niedersachsen Norden Kastanienallee 68
```

15. this query group all autos by their year of production and calculates their amount in each group

```
Console.WriteLine("\n15.Autos grouped by their year of production:");
var query15 = from a in data.Descendants("AutoPark")
              group a by a.Element("YearOfProduction").Value into year
              select new
              {
                  name = year.Key,
                  amount = year.Count()
              };
foreach (var item in query15) Console.WriteLine($"\t{item.name}\t{item.amount}");
```

```
15.Autos grouped by their year of production:
        2023    4
        2020    1
        2017    2
        2008    2
        2010    1
```

Висновок

Виконуючи дану лабораторну роботу, я покращив свої навички створення програмного забезпечення на мові C#, створив структуру даних, навчився обробляти дані за допомогою бібліотеки LINQ to XML, та закріпив теоретичні знання на практиці.