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 №3

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

на тему: «Робота з json»

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 в форматі JSON в C#

Постановка задачі комп‘ютерного практикуму № 3
При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

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

2) Продемонструвати роботу з JSON з використанням технології серіалізації/десеріалізації.

3) Продемонструвати роботу з JSON з використанням об’єктної моделі документів з використанням обох варіантів:

a. JsonDocument;
b. JsonNode.

4) Створити відповідне програмне забезпечення, котре реалізує завдання згідно п.1-3. Дані повинні зберігатись в файли та зчитуватись з файлів.

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

6) 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

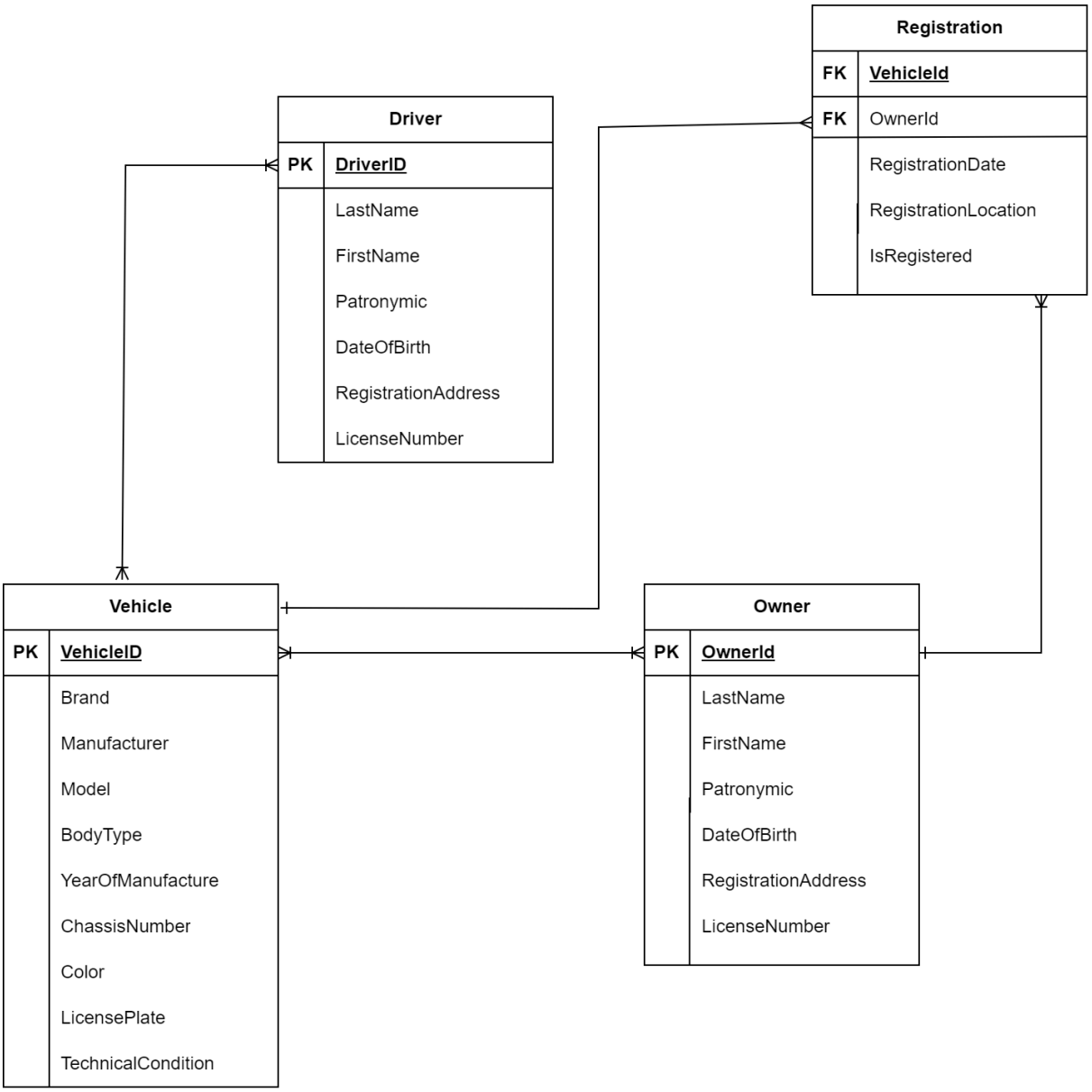


Рисунок 1. ER diagram

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:

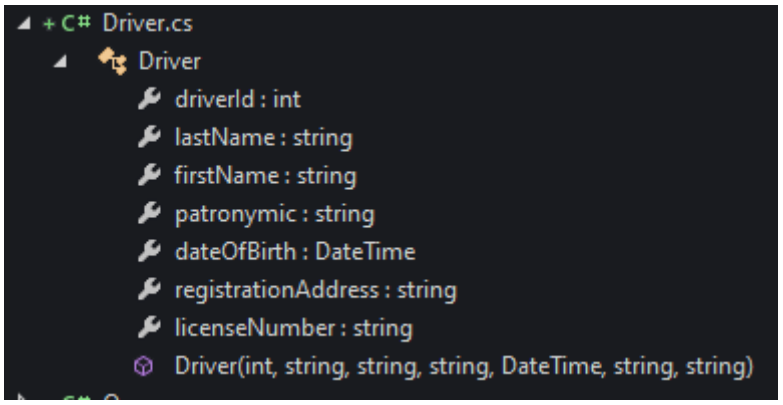


Рисунок 2. Клас водія

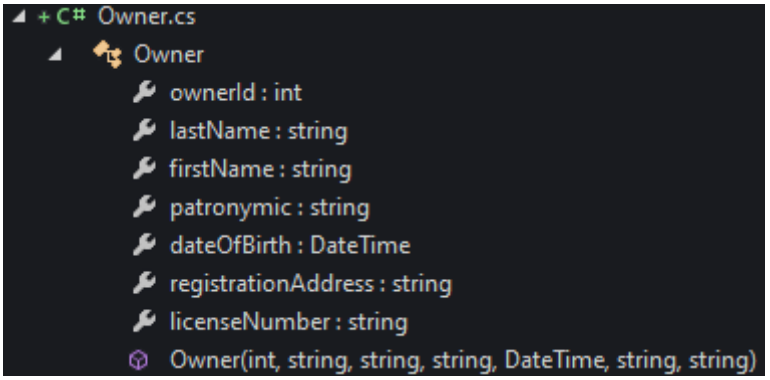


Рисунок 3. Клас власника

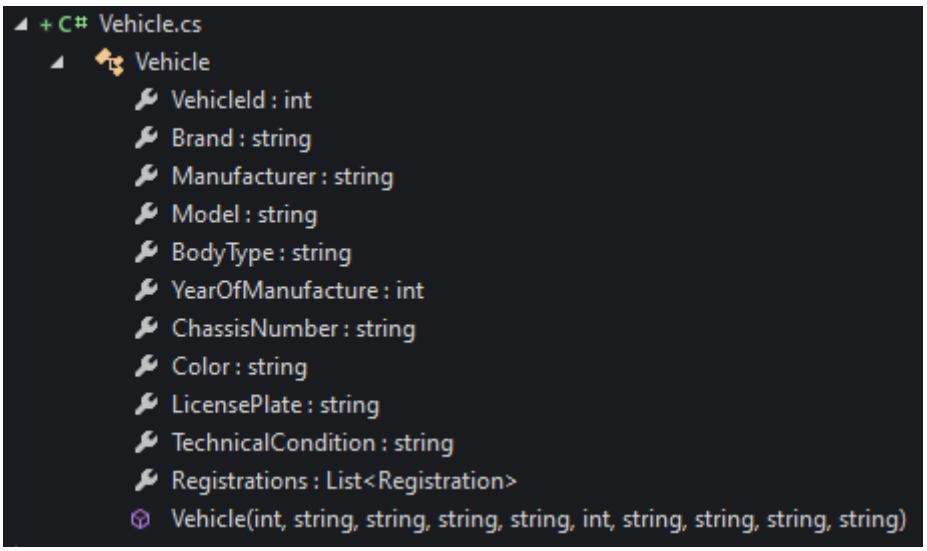


Рисунок 4. Клас транспортного засобу

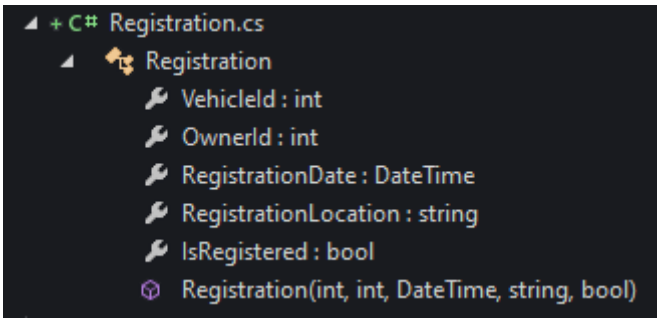


Рисунок 5. Клас реєстрації

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

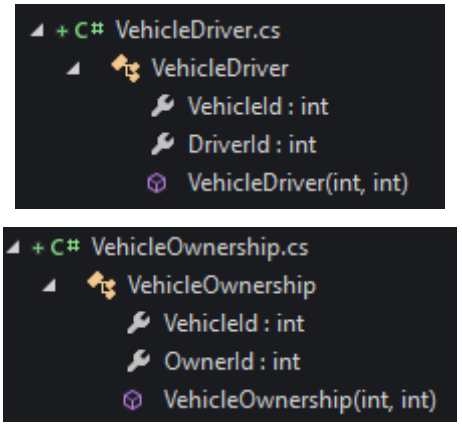


Рисунок 6. Допоміжні класи для зв’язків

Json-файли

Рисунок 7. Файли Json

Допоміжні класи, що реалізують логіку обробки Json документів

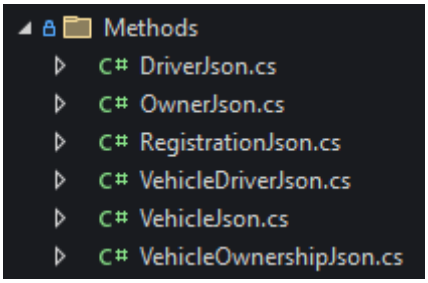


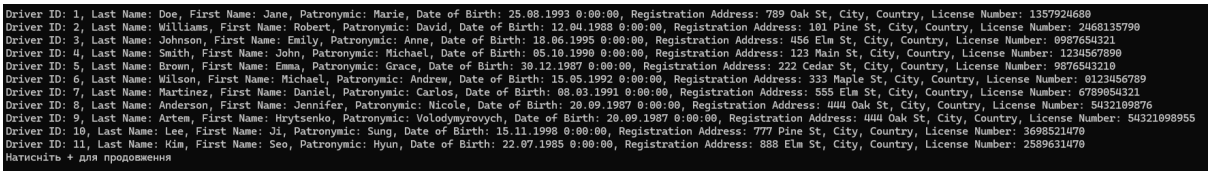
Рисунок 8. Допоміжні класи для обробки Json документів

Приклад роботи програми

```
Головне меню

1. Серіалізація списку об'єктів Driver
2. Десеріалізація списку об'єктів Driver
3. Отримати список об'єктів Driver за домогою JsonDocument
4. Отримати список об'єктів Driver за допомогою JsonNode
5. Серіалізація списку об'єктів Owner
6. Десеріалізація списку об'єктів Owner
7. Отримати список об'єктів Owner за допомогою JsonDocument
8. Отримати список об'єктів Owner за допомогою JsonNode
9. Серіалізація списку об'єктів Registration
10. Десеріалізація списку об'єктів Registration
11. Отримати список об'єктів Registration за допомогою JsonDocument
12. Отримати список об'єктів Registration за допомогою JsonNode
13. Серіалізація списку об'єктів Vehicle
14. Десеріалізація списку об'єктів Vehicle
15. Отримати список об'єктів Vehicle за допомогою JsonDocument
16. Отримати список об'єктів за допомогою Vehicle JsonNode
17. Серіалізація списоку об'єктів VehicleDriver
18. Десеріалізація списку об'єктів VehicleDriver
19. Отримати список об'єктів VehicleDriver за допомогою JsonDocument
20. Отримати список об'єктів VehicleDriver за допомогою JsonNode
21. Серіалізація списку об'єктів VehicleOwnership
22. Десеріалізація списку об'єктів VehicleOwnership
23. Отримати список об'єктів VehicleOwnership за допомогою JsonDocument
24. Отримати список об'єктів VehicleOwnership за допомогою JsonNode

Ваш вибір:
```



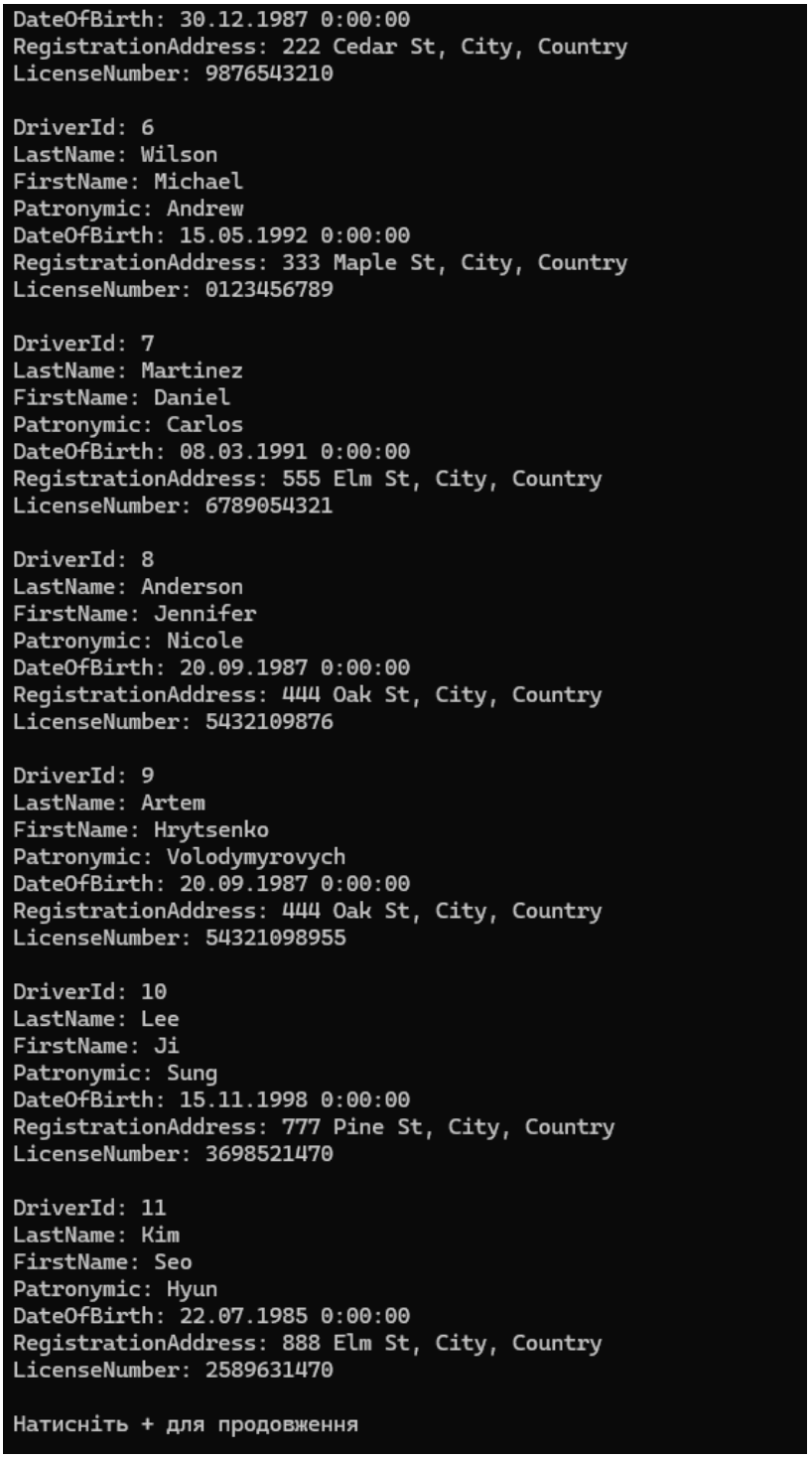


Рисунок 9. Приклади роботи програми

Словесний опис запитів

GetSUVVehicles : цей запит отримує всі транспортні засоби, які мають тип кузова "SUV". Потім відображається список цих транспортних засобів, включаючи їх марку, модель і рік випуску, якщо такі є.

GetAllSUVsWithOwners : цей запит отримує всі позашляховики разом із відповідними власниками. Він об’єднує інформацію з кількох таблиць, щоб зіставити кожен SUV з його власником, а потім відображає цю інформацію, включаючи марку та модель автомобіля, а також ім’я та прізвище власника.

GetMachinesInExcellentCondition : цей запит отримує список транспортних засобів, які знаходяться у відмінному технічному стані. Він фільтрує транспортні засоби за ознакою їх технічного стану та відображає марку та модель кожного автомобіля, який відповідає цьому критерію.

GetOwnersWithinAgeRange : цей запит отримує список власників, вік яких входить у вказаний діапазон. Він обчислює діапазон дат народження на основі наданих мінімальних і максимальних років, а потім відображає імена та дати народження власників, які відповідають цим критеріям.

GetDriverWithLongestName : цей запит знаходить водія із найдовшою назвою. Він розраховує довжину повного імені кожного водія (ім’я, прізвище та по батькові) і вибирає водія з максимальною довжиною імені, а потім відображає його повне ім’я.

GetVehicleWithEarliestManufactureYear : цей запит отримує транспортний засіб із найранішим роком виробництва. Він упорядковує транспортні засоби за роками випуску та вибирає той із найранішим роком, а потім відображає його марку, модель і рік виробництва.

SortDriversByDateOfBirthAscendingWithBoundaries : цей запит отримує відсортований список водіїв за датою народження в межах зазначеного діапазону. Він фільтрує водіїв за роками їх народження, що входять у вказані межі, сортує їх за датою народження, а потім відображає їхні імена та дати народження в порядку зростання.

GetUnregisteredVehicles : цей запит отримує список транспортних засобів, які не зареєстровані. Він перевіряє, чи є будь-які реєстрації, пов’язані з кожним транспортним засобом, і вибирає транспортні засоби без будь-яких реєстрацій, а потім відображає їх марку, модель та ідентифікатор автомобіля.

GetMachinesReleasedBeforeYear : цей запит отримує список транспортних засобів, які були випущені до вказаного року. Він фільтрує транспортні засоби на основі того, що роки їх виробництва менші або дорівнюють указаному року, і відображає їх марку, модель і рік виробництва.

GetVehiclesRegisteredInLocalOffice : цей запит отримує список транспортних засобів, зареєстрованих у розташуванні «Місцеве реєстраційне відділення». Він поєднує інформацію з реєстраційних таблиць і таблиць транспортних засобів, щоб відповідати транспортним засобам, зареєстрованим у цьому місці, а потім відображає їх марку та модель.

GetVehiclesNotDrivenByDrivers : отримує список транспортних засобів, якими не керують водії. Він перевіряє зв’язок між транспортними засобами та водіями в таблиці vehicleDriver і вибирає транспортні засоби, які не мають відповідних записів водіїв.

GetUniqueOwnersAndDriversNames : отримує список унікальних імен шляхом поєднання імен власників і водіїв. Він окремо отримує імена та прізвища власників і водіїв, об’єднує їх, видаляє дублікати та повертає унікальний список імен.

GetOwnersWithMultipleVehicles : отримує список власників, які володіють кількома транспортними засобами. Він групує записи про власність транспортного засобу за ідентифікатором власника, фільтрує власників із кількома транспортними засобами та повертає їх деталі.

SortVehiclesByModelYearDescendingWithBoundaries : отримує список транспортних засобів, відсортованих за роком виробництва в порядку спадання в межах зазначеного діапазону. Він вибирає транспортні засоби,

вироблені між заданими роками, упорядковує їх за роком виробництва в порядку спадання та повертає відсортований список.

FindDriversWithoutVehicles : отримує список водіїв, які не мають пов’язаних транспортних засобів. Він перевіряє наявність водіїв, які не мають відповідних записів у таблиці vehicleDriver, і повертає їхні дані.

GetAverageAgeOfDrivers : обчислює середній вік водіїв. Він обчислює вік кожного водія на основі дати його народження, підсумовує вік, обчислює середнє значення та повертає його.

GetDriversYoungerThanAge : отримує список водіїв, які молодші за вказаний вік. Він розраховує максимальну дату народження на основі даного віку, фільтрує водіїв, молодших за цей вік, і повертає їхні дані.

GetVehiclesRegisteredInYear : отримує список транспортних засобів, зареєстрованих у певному році. Він фільтрує реєстраційні записи за роком реєстрації та поєднує їх із записами транспортних засобів, щоб отримати список транспортних засобів, зареєстрованих у цьому році.

GetDriversBornInYear : отримує список водіїв, які народилися в певний рік. Він фільтрує водіїв за роком народження та повертає їхні дані.

GetCarsOwnedByOwner : отримує список автомобілів, які належать певному ідентифікатору власника. Він поєднує записи про володіння транспортним засобом із записами про транспортні засоби, щоб ідентифікувати автомобілі, що належать зазначеному власнику, і повертає їх марку та модель.

Висновок

У даній лабораторній роботі ми вивчали обробку даних у форматі JSON в середовищі мови програмування C#. В процесі виконання завдань ми ознайомились із структурою та особливостями JSON, а також засвоїли практичні навички роботи з цим форматом даних. Для досягнення поставлених цілей ми розробили структуру JSON для зберігання даних згідно варіантів, продемонстрували роботу з JSON з використанням технології серіалізації/десеріалізації, а також з об'єктної моделі документів за допомогою JsonDocument та JsonNode. Усі ці аспекти відображені у консольному застосуванні, що було реалізоване на мові C#.

Програмний код

Реалізація основних класів:

Driver.cs:

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

namespace laba3
{
   public class Driver
   {
      public int driverId { get; set; }
      public string? lastName { get; set; }
      public string? firstName { get; set; }
      public string? patronymic { get; set; }
      public DateTime dateOfBirth { get; set; }
      public string? registrationAddress { get; set; }
      public string? licenseNumber { get; set; }
      public Driver(int DriverId, string? LastName, string? FirstName, string?
Patronymic, DateTime DateOfBirth,
      string? RegistrationAddress, string? LicenseNumber)

      {
         driverId = DriverId;
         lastName = LastName;
         firstName = FirstName;
         patronymic = Patronymic;
         dateOfBirth = DateOfBirth;
         registrationAddress = RegistrationAddress;
```

```
            licenseNumber = LicenseNumber;
        }
    }
}
```

Owner.cs:

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

namespace laba3
{
    public class Owner
    {
        public int ownerId { get; set; }
        public string? lastName { get; set; }
        public string? firstName { get; set; }
        public string? patronymic { get; set; }
        public DateTime dateOfBirth { get; set; }
        public string? registrationAddress { get; set; }
        public string? licenseNumber { get; set; }
        public Owner(int OwnerId, string? LastName, string? FirstName, string?
Patronymic, DateTime DateOfBirth,
        string? RegistrationAddress, string? LicenseNumber)
        {
            ownerId = OwnerId;
            lastName = LastName;
            firstName = FirstName;
            patronymic = Patronymic;
            dateOfBirth = DateOfBirth;
            registrationAddress = RegistrationAddress;
            licenseNumber = LicenseNumber;
        }
```

```
    }
}
```

Vehicle.cs:

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

namespace laba3
{
    public class Vehicle
    {
        public int VehicleId { get; set; }
        public string? Brand { get; set; }
        public string? Manufacturer { get; set; }
        public string? Model { get; set; }
        public string? BodyType { get; set; }
        public int YearOfManufacture { get; set; }
        public string? ChassisNumber { get; set; }
        public string? Color { get; set; }
        public string? LicensePlate { get; set; }
        public string? TechnicalCondition { get; set; }
        public List<Registration>? Registrations { get; set; }

        public Vehicle(int vehicleId, string? brand, string? manufacturer, string?
model, string? bodyType, int yearOfManufacture, string? chassisNumber,
        string? color, string? licensePlate, string? technicalCondition)
        {
            VehicleId = vehicleId;
            Brand = brand;
            Manufacturer = manufacturer;
            Model = model;
            BodyType = bodyType;
```

```
            YearOfManufacture = yearOfManufacture;
            ChassisNumber = chassisNumber;
            Color = color;
            LicensePlate = licensePlate;
            TechnicalCondition = technicalCondition;
        }
    }
}
```

Registration.cs:

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

namespace laba3
{
    public class Registration
    {
        public int VehicleId { get; init; }
        public int OwnerId { get; init; }
        public DateTime RegistrationDate { get; set; }
        public string? RegistrationLocation { get; init; }
        public bool IsRegistered { get; set; }
        public Registration(int vehicleId, int ownerId, DateTime registrationDate,
string? registrationLocation, bool isRegistered)
        {
            VehicleId = vehicleId;
            OwnerId = ownerId;
            RegistrationDate = registrationDate;
            RegistrationLocation = registrationLocation;
            IsRegistered = isRegistered;
        }
    }
```

```
}
```

Реалізація допоміжних класів:

VehicleDriver.cs:

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

namespace laba3
{
   public class VehicleDriver
   {
      public int VehicleId { get; set; }
      public int DriverId { get; set; }
      public VehicleDriver(int vehicleId, int driverId)
      {
         VehicleId = vehicleId;
         DriverId = driverId;
      }
   }
}
```

VehicleOwnership.cs:

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

```
namespace laba3
{
    public class VehicleOwnership
    {
        public int VehicleId { get; set; }
        public int OwnerId { get; set; }
        public VehicleOwnership(int vehicleId, int ownerId)
        {
            VehicleId = vehicleId;
            OwnerId = ownerId;
        }
    }
}
```

Json-файли:

Driver.json:


```
[
  {
    "driverId": 1,
    "lastName": "Doe",
    "firstName": "Jane",
    "patronymic": "Marie",
    "dateOfBirth": "1993-08-25T00:00:00",
    "registrationAddress": "789 Oak St, City, Country",
    "licenseNumber": "1357924680"
  },
  {
    "driverId": 2,
    "lastName": "Williams",
    "firstName": "Robert",
    "patronymic": "David",
    "dateOfBirth": "1988-04-12T00:00:00",
    "registrationAddress": "101 Pine St, City, Country",
    "licenseNumber": "2468135790"
  },
```

```
{
 "driverId": 3,
 "lastName": "Johnson",
 "firstName": "Emily",
 "patronymic": "Anne",
 "dateOfBirth": "1995-06-18T00:00:00",
 "registrationAddress": "456 Elm St, City, Country",
 "licenseNumber": "0987654321"
},
{
 "driverId": 4,
 "lastName": "Smith",
 "firstName": "John",
 "patronymic": "Michael",
 "dateOfBirth": "1990-10-05T00:00:00",
 "registrationAddress": "123 Main St, City, Country",
 "licenseNumber": "1234567890"
},
{
 "driverId": 5,
 "lastName": "Brown",
 "firstName": "Emma",
 "patronymic": "Grace",
 "dateOfBirth": "1987-12-30T00:00:00",
 "registrationAddress": "222 Cedar St, City, Country",
 "licenseNumber": "9876543210"
},
{
 "driverId": 6,
 "lastName": "Wilson",
 "firstName": "Michael",
 "patronymic": "Andrew",
 "dateOfBirth": "1992-05-15T00:00:00",
 "registrationAddress": "333 Maple St, City, Country",
 "licenseNumber": "0123456789"
},
{
```

```
  "driverId": 7,
  "lastName": "Martinez",
  "firstName": "Daniel",
  "patronymic": "Carlos",
  "dateOfBirth": "1991-03-08T00:00:00",
  "registrationAddress": "555 Elm St, City, Country",
  "licenseNumber": "6789054321"
},
{
  "driverId": 8,
  "lastName": "Anderson",
  "firstName": "Jennifer",
  "patronymic": "Nicole",
  "dateOfBirth": "1987-09-20T00:00:00",
  "registrationAddress": "444 Oak St, City, Country",
  "licenseNumber": "5432109876"
},
{
  "driverId": 9,
  "lastName": "Artem",
  "firstName": "Hrytsenko",
  "patronymic": "Volodymyrovych",
  "dateOfBirth": "1987-09-20T00:00:00",
  "registrationAddress": "444 Oak St, City, Country",
  "licenseNumber": "54321098955"
},
{
  "driverId": 10,
  "lastName": "Lee",
  "firstName": "Ji",
  "patronymic": "Sung",
  "dateOfBirth": "1998-11-15T00:00:00",
  "registrationAddress": "777 Pine St, City, Country",
  "licenseNumber": "3698521470"
},
{
  "driverId": 11,
```

```
    "lastName": "Kim",
    "firstName": "Seo",
    "patronymic": "Hyun",
    "dateOfBirth": "1985-07-22T00:00:00",
    "registrationAddress": "888 Elm St, City, Country",
    "licenseNumber": "2589631470"
  }
]
```


Owner.json:


```
[
  {
    "ownerId": 1,
    "lastName": "Smith",
    "firstName": "John",
    "patronymic": "Michael",
    "dateOfBirth": "1985-10-15T00:00:00",
    "registrationAddress": "123 Main St, City, Country",
    "licenseNumber": "1234567890"
  },
  {
    "ownerId": 2,
    "lastName": "Johnson",
    "firstName": "Emily",
    "patronymic": "Anne",
    "dateOfBirth": "1990-05-20T00:00:00",
    "registrationAddress": "456 Elm St, City, Country",
    "licenseNumber": "0987654321"
  },
  {
    "ownerId": 3,
    "lastName": "Williams",
    "firstName": "David",
    "patronymic": "Robert",
    "dateOfBirth": "1983-08-25T00:00:00",
    "registrationAddress": "789 Oak St, City, Country",
```

```
  "licenseNumber": "1357924680"
},
{
  "ownerId": 4,
  "lastName": "Brown",
  "firstName": "Sarah",
  "patronymic": "Elizabeth",
  "dateOfBirth": "1975-12-10T00:00:00",
  "registrationAddress": "101 Pine St, City, Country",
  "licenseNumber": "2468135790"
},
{
  "ownerId": 5,
  "lastName": "Wilson",
  "firstName": "Jessica",
  "patronymic": "Marie",
  "dateOfBirth": "1992-06-05T00:00:00",
  "registrationAddress": "222 Cedar St, City, Country",
  "licenseNumber": "9876543210"
},
{
  "ownerId": 6,
  "lastName": "Taylor",
  "firstName": "Michael",
  "patronymic": "Andrew",
  "dateOfBirth": "1988-04-15T00:00:00",
  "registrationAddress": "333 Maple St, City, Country",
  "licenseNumber": "0123456789"
},
{
  "ownerId": 7,
  "lastName": "Anderson",
  "firstName": "Jennifer",
  "patronymic": "Nicole",
  "dateOfBirth": "1987-09-20T00:00:00",
  "registrationAddress": "444 Oak St, City, Country",
  "licenseNumber": "5432109876"
```

```
  },
  {
    "ownerId": 8,
    "lastName": "Martinez",
    "firstName": "Daniel",
    "patronymic": "Carlos",
    "dateOfBirth": "1991-03-08T00:00:00",
    "registrationAddress": "555 Elm St, City, Country",
    "licenseNumber": "6789054321"
  },
  {
    "ownerId": 9,
    "lastName": "Lee",
    "firstName": "Seo",
    "patronymic": "Hyun",
    "dateOfBirth": "1989-11-25T00:00:00",
    "registrationAddress": "666 Maple St, City, Country",
    "licenseNumber": "3698521470"
  },
  {
    "ownerId": 10,
    "lastName": "Kim",
    "firstName": "Ji",
    "patronymic": "Sung",
    "dateOfBirth": "1995-07-30T00:00:00",
    "registrationAddress": "777 Elm St, City, Country",
    "licenseNumber": "2589631470"
  }
]
```


Vehicle.json:


```
[
  {
    "vehicleId": 1,
    "brand": "Toyota",
    "manufacturer": "Toyota Motor Corporation",
```

```
  "model": "Camry",
  "bodyType": "Sedan",
  "yearOfManufacture": 2020,
  "chassisNumber": "JT2BF22KX00765432",
  "color": "Black",
  "licensePlate": "ABC123",
  "technicalCondition": "Good"
},
{
  "vehicleId": 2,
  "brand": "Honda",
  "manufacturer": "Honda Motor Co., Ltd.",
  "model": "Civic",
  "bodyType": "Sedan",
  "yearOfManufacture": 2018,
  "chassisNumber": "2HGFC2F58JH532148",
  "color": "Silver",
  "licensePlate": "XYZ789",
  "technicalCondition": "Excellent"
},
{
  "vehicleId": 3,
  "brand": "Ford",
  "manufacturer": "Ford Motor Company",
  "model": "F-150",
  "bodyType": "Truck",
  "yearOfManufacture": 2019,
  "chassisNumber": "1FTFW1EF8KFA12345",
  "color": "Blue",
  "licensePlate": "DEF456",
  "technicalCondition": "Good"
},
{
  "vehicleId": 4,
  "brand": "Chevrolet",
  "manufacturer": "General Motors Company",
  "model": "Camaro",
```

```
  "bodyType": "Coupe",
  "yearOfManufacture": 2021,
  "chassisNumber": "2G1FA1ED7A9725478",
  "color": "Red",
  "licensePlate": "GHI789",
  "technicalCondition": "Excellent"
},
{
  "vehicleId": 5,
  "brand": "BMW",
  "manufacturer": "Bayerische Motoren Werke AG",
  "model": "X5",
  "bodyType": "SUV",
  "yearOfManufacture": 2017,
  "chassisNumber": "5UXKR6C56H0U24876",
  "color": "White",
  "licensePlate": "JKL012",
  "technicalCondition": "Good"
},
{
  "vehicleId": 6,
  "brand": "Mercedes-Benz",
  "manufacturer": "Mercedes-Benz AG",
  "model": "E-Class",
  "bodyType": "Sedan",
  "yearOfManufacture": 2022,
  "chassisNumber": "WDDZF8EB7KA525623",
  "color": "Gray",
  "licensePlate": "MNO345",
  "technicalCondition": "Excellent"
},
{
  "vehicleId": 7,
  "brand": "Audi",
  "manufacturer": "Audi AG",
  "model": "A4",
  "bodyType": "Sedan",
```

```
  "yearOfManufacture": 2016,
  "chassisNumber": "WAUENAF4XHN011235",
  "color": "Black",
  "licensePlate": "PQR678",
  "technicalCondition": "Good"
},
{
  "vehicleId": 8,
  "brand": "Tesla",
  "manufacturer": "Tesla, Inc.",
  "model": "Model 3",
  "bodyType": "Electric Sedan",
  "yearOfManufacture": 2023,
  "chassisNumber": "5YJ3E1EB5NF105256",
  "color": "Blue",
  "licensePlate": "STU901",
  "technicalCondition": "Awful"
},
{
  "vehicleId": 9,
  "brand": "Volkswagen",
  "manufacturer": "Volkswagen Group",
  "model": "Golf",
  "bodyType": "Hatchback",
  "yearOfManufacture": 2015,
  "chassisNumber": "WVWZZZ1KZ4U123456",
  "color": "Blue",
  "licensePlate": "XYZ789",
  "technicalCondition": "Excellent"
},
{
  "vehicleId": 10,
  "brand": "Hyundai",
  "manufacturer": "Hyundai Motor Company",
  "model": "Sonata",
  "bodyType": "Sedan",
  "yearOfManufacture": 2024,
```

```
    "chassisNumber": "5NPE34AF0HH511647",
    "color": "Silver",
    "licensePlate": "ABCDEF",
    "technicalCondition": "Good"
  },
  {
    "vehicleId": 11,
    "brand": "Kia",
    "manufacturer": "Kia Corporation",
    "model": "Sportage",
    "bodyType": "SUV",
    "yearOfManufacture": 2023,
    "chassisNumber": "KNDP33AF3D7698231",
    "color": "Red",
    "licensePlate": "GHIJKL",
    "technicalCondition": "Excellent"
  }
]
```


VehicleDriver.json:


```
[
  {
    "vehicleId": 1,
    "driverId": 2
  },
  {
    "vehicleId": 2,
    "driverId": 1
  },
  {
    "vehicleId": 3,
    "driverId": 3
  },
  {
    "vehicleId": 4,
    "driverId": 4
```

```
  },
  {
    "vehicleId": 5,
    "driverId": 5
  },
  {
    "vehicleId": 6,
    "driverId": 6
  },
  {
    "vehicleId": 7,
    "driverId": 7
  },
  {
    "vehicleId": 8,
    "driverId": 8
  },
  {
    "vehicleId": 9,
    "driverId": 9
  },
  {
    "vehicleId": 10,
    "driverId": 10
  }
]
```


VehicleOwnership.json:


```
[
  {
    "vehicleId": 1,
    "ownerId": 1
  },
  {
    "vehicleId": 2,
    "ownerId": 2
```

```
},
{
 "vehicleId": 3,
 "ownerId": 3
},
{
 "vehicleId": 4,
 "ownerId": 4
},
{
 "vehicleId": 5,
 "ownerId": 5
},
{
 "vehicleId": 6,
 "ownerId": 6
},
{
 "vehicleId": 7,
 "ownerId": 7
},
{
 "vehicleId": 8,
 "ownerId": 8
},
{
 "vehicleId": 9,
 "ownerId": 8
},
{
 "vehicleId": 10,
 "ownerId": 9
},
{
 "vehicleId": 11,
 "ownerId": 10
}
```

]

Registrations.json:


```
[
  {
    "vehicleId": 1,
    "ownerId": 1,
    "registrationDate": "2021-05-10T00:00:00",
    "registrationLocation": "Department of Motor Vehicles",
    "isRegistered": true
  },
  {
    "vehicleId": 2,
    "ownerId": 2,
    "registrationDate": "2020-08-20T00:00:00",
    "registrationLocation": "Local Registration Office",
    "isRegistered": true
  },
  {
    "vehicleId": 3,
    "ownerId": 3,
    "registrationDate": "2022-03-15T00:00:00",
    "registrationLocation": "Department of Motor Vehicles",
    "isRegistered": true
  },
  {
    "vehicleId": 4,
    "ownerId": 4,
    "registrationDate": "2020-12-05T00:00:00",
    "registrationLocation": "Local Registration Office",
    "isRegistered": true
  },
  {
    "vehicleId": 5,
    "ownerId": 5,
    "registrationDate": "2023-06-25T00:00:00",
```

```
  "registrationLocation": "Department of Motor Vehicles",
  "isRegistered": true
},
{
  "vehicleId": 6,
  "ownerId": 6,
  "registrationDate": "2022-09-18T00:00:00",
  "registrationLocation": "Local Registration Office",
  "isRegistered": true
},
{
  "vehicleId": 7,
  "ownerId": 7,
  "registrationDate": "2021-11-30T00:00:00",
  "registrationLocation": "Department of Motor Vehicles",
  "isRegistered": true
},
{
  "vehicleId": 8,
  "ownerId": 8,
  "registrationDate": "2023-04-07T00:00:00",
  "registrationLocation": "Local Registration Office",
  "isRegistered": true
},
{
  "vehicleId": 9,
  "ownerId": 9,
  "registrationDate": "2024-02-15T00:00:00",
  "registrationLocation": "Department of Motor Vehicles",
  "isRegistered": true
},
{
  "vehicleId": 10,
  "ownerId": 10,
  "registrationDate": "2023-10-10T00:00:00",
  "registrationLocation": "Local Registration Office",
  "isRegistered": true
```

```
  }
]
```

Допоміжні класи для реалізації обробки XML:

DriverJson.cs:

```
using System.Text.Json.Nodes;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace laba3.Methods
{
   public class DriverJson
   {
      private readonly string _path =
"C:\\Users\\User\\source\\repos\\laba3\\laba3\\JSON\\Driver.json";

      public void AddDriverSerializer (List<Driver> drivers)
      {
         if (File.Exists(_path))
         {
            try
            {
               string json = File.ReadAllText(_path);

               if (json != null)
               {
                  File.WriteAllText(_path, string.Empty);
               }

               JsonSerializerOptions options = new JsonSerializerOptions()
               {
                  PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                  WriteIndented = true,
                  DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull
               };
```

```
            using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
            {
                JsonSerializer.Serialize(fs, drivers, options);
                Console.WriteLine("Дані записано!");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }
    else
    {
        Console.WriteLine("Файл не існує");
    }
}

public void GetDriverDeserializer()
{
    try
    {
        using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
        {
            List<Driver>? drivers =
JsonSerializer.Deserialize<List<Driver>>(fs);
            if (drivers != null)
            {
                foreach (var driver in drivers)
                {
                    Console.WriteLine($"DriverId: {driver.driverId}");
                    Console.WriteLine($"LastName: {driver.lastName}");
                    Console.WriteLine($"FirstName: {driver.firstName}");
                    Console.WriteLine($"Patronymic: {driver.patronymic}");
                    Console.WriteLine($"DateOfBirth: {driver.dateOfBirth}");
```

```
                    Console.WriteLine($"RegistrationAddress:
{driver.registrationAddress}");
                    Console.WriteLine($"LicenseNumber:
{driver.licenseNumber}");
                    Console.WriteLine();
                }
            }
            else
            {
                Console.WriteLine("Файл пустий.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }

    public void GetDriverJson()
    {
        try
        {
            string jsonString = File.ReadAllText(_path);
            if (!string.IsNullOrEmpty(jsonString))
            {
                List<Driver> drivers = new List<Driver>();
                using (JsonDocument document =
JsonDocument.Parse(jsonString))
                {
                    JsonElement root = document.RootElement;
                    if (root.ValueKind == JsonValueKind.Array)
                    {
                        foreach (JsonElement driverElement in
root.EnumerateArray())
                        {
```

```
                    int driverId =
driverElement.GetProperty("driverId").GetInt32();
                    string lastName =
driverElement.GetProperty("lastName").GetString();
                    string firstName =
driverElement.GetProperty("firstName").GetString();
                    string patronymic =
driverElement.GetProperty("patronymic").GetString();
                    DateTime dateOfBirth =
driverElement.GetProperty("dateOfBirth").GetDateTime();
                    string registrationAddress =
driverElement.GetProperty("registrationAddress").GetString();
                    string licenseNumber =
driverElement.GetProperty("licenseNumber").GetString();

                    Driver driver = new Driver(driverId, lastName, firstName,
patronymic, dateOfBirth, registrationAddress, licenseNumber);
                    drivers.Add(driver);
                }
            }
        }

        foreach (var driver in drivers)
        {
            Console.WriteLine($"Driver ID: {driver.driverId}, Last Name:
{driver.lastName}, " +
                $"First Name: {driver.firstName}, Patronymic:
{driver.patronymic}, " +
                $"Date of Birth: {driver.dateOfBirth}, Registration Address:
{driver.registrationAddress}, " +
                $"License Number: {driver.licenseNumber}");
        }
    }
    else
    {
        Console.WriteLine("Файл пустий");
    }
```

```
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }


    public void GetDriverJsonNode()
    {
        try
        {
            string jsonString = File.ReadAllText(_path);
            if (!string.IsNullOrEmpty(jsonString))
            {
                JsonNode? rootNode = JsonNode.Parse(jsonString);

                if (rootNode is JsonArray rootArray)
                {
                    foreach (var node in rootArray)
                    {
                        int driverId = int.Parse(node["driverId"].ToString());
                        string lastName = node["lastName"].ToString();
                        string firstName = node["firstName"].ToString();
                        string patronymic = node["patronymic"].ToString();
                        DateTime dateOfBirth =
DateTime.Parse(node["dateOfBirth"].ToString());
                        string registrationAddress =
node["registrationAddress"].ToString();
                        string licenseNumber = node["licenseNumber"].ToString();

                        Console.WriteLine($"Driver ID: {driverId}, Last Name:
{lastName}, " +
                        $"First Name: {firstName}, Patronymic: {patronymic}, " +
                        $"Date of Birth: {dateOfBirth}, Registration Address:
{registrationAddress}, " +
                        $"License Number: {licenseNumber}");
```

```
                }
            }
            else
            {
                Console.WriteLine("Кореневий вузол не є масивом");
            }
        }
        else
        {
            Console.WriteLine("Файл пустий");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
  }

}
```

OwnerJson.cs:

```
using System.Text.Json.Serialization;
using System.Text.Json;
using System.Text.Json.Nodes;

namespace laba3.Methods
{
  public class OwnerJson
  {
    private readonly string _path =
"C:\\Users\\User\\source\\repos\\laba3\\laba3\\JSON\\Owner.json";

    public void AddOwnerSerializer(List<Owner> owners)
    {
```

```
        if (File.Exists(_path))
        {
            try
            {
                string json = File.ReadAllText(_path);

                if (json != null)
                {
                    File.WriteAllText(_path, string.Empty);
                }

                JsonSerializerOptions options = new JsonSerializerOptions()
                {
                    PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                    WriteIndented = true,
                    DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull
                };

                using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
                {
                    JsonSerializer.Serialize(fs, owners, options);
                    Console.WriteLine("Дані записано!");
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Помилка: {ex.Message}");
            }
        }
        else
        {
            Console.WriteLine("Файл не існує");
        }
    }
```

```
        public void GetOwnerDeserializer()
        {
            try
            {
                using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
                {
                    List<Owner>? owners =
JsonSerializer.Deserialize<List<Owner>>(fs);
                    if (owners != null)
                    {
                        foreach (var owner in owners)
                        {
                            Console.WriteLine($"OwnerId: {owner.ownerId}");
                            Console.WriteLine($"LastName: {owner.lastName}");
                            Console.WriteLine($"FirstName: {owner.firstName}");
                            Console.WriteLine($"Patronymic: {owner.patronymic}");
                            Console.WriteLine($"DateOfBirth: {owner.dateOfBirth}");
                            Console.WriteLine($"RegistrationAddress:
{owner.registrationAddress}");
                            Console.WriteLine($"LicenseNumber:
{owner.licenseNumber}");
                            Console.WriteLine();
                        }
                    }
                    else
                    {
                        Console.WriteLine("Файл пустий.");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Помилка: {ex.Message}");
            }
        }
```

```
        public void GetOwnerJson()
        {
            try
            {
                string jsonString = File.ReadAllText(_path);
                if (!string.IsNullOrEmpty(jsonString))
                {
                    List<Owner> owners = new List<Owner>();
                    using (JsonDocument document =
JsonDocument.Parse(jsonString))
                    {
                        JsonElement root = document.RootElement;
                        if (root.ValueKind == JsonValueKind.Array)
                        {
                            foreach (JsonElement ownerElement in
root.EnumerateArray())
                            {
                                Owner owner = new Owner(
                                    ownerElement.GetProperty("ownerId").GetInt32(),
                                    ownerElement.GetProperty("lastName").GetString(),
                                    ownerElement.GetProperty("firstName").GetString(),
                                    ownerElement.GetProperty("patronymic").GetString(),

ownerElement.GetProperty("dateOfBirth").GetDateTime(),

ownerElement.GetProperty("registrationAddress").GetString(),

ownerElement.GetProperty("licenseNumber").GetString());

                                owners.Add(owner);
                            }
                        }
                    }

                    foreach (var owner in owners)
                    {
```

```
                Console.WriteLine($"Owner ID: {owner.ownerId}, Last Name:
{owner.lastName}, " +
                    $"First Name: {owner.firstName}, Patronymic:
{owner.patronymic}, " +
                    $"Date of Birth: {owner.dateOfBirth}, Registration Address:
{owner.registrationAddress}, " +
                    $"License Number: {owner.licenseNumber}");
            }
        }
        else
        {
            Console.WriteLine("Файл пустий");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
}


public void GetOwnerJsonNode()
{
    try
    {
        string jsonString = File.ReadAllText(_path);
        if (!string.IsNullOrEmpty(jsonString))
        {
            JsonNode? rootNode = JsonNode.Parse(jsonString);

            if (rootNode is JsonArray rootArray)
            {
                foreach (var node in rootArray)
                {
                    int ownerId = int.Parse(node["ownerId"].ToString());
                    string lastName = node["lastName"].ToString();
                    string firstName = node["firstName"].ToString();
```

```
                    string patronymic = node["patronymic"].ToString();
                    DateTime dateOfBirth =
DateTime.Parse(node["dateOfBirth"].ToString());
                    string registrationAddress =
node["registrationAddress"].ToString();
                    string licenseNumber = node["licenseNumber"].ToString();

                    Console.WriteLine($"Owner ID: {ownerId}, Last Name:
{lastName}, " +
                        $"First Name: {firstName}, Patronymic: {patronymic}, " +
                        $"Date of Birth: {dateOfBirth}, Registration Address:
{registrationAddress}, " +
                        $"License Number: {licenseNumber}");
                }
            }
            else
            {
                Console.WriteLine("Кореневий вузол не є масивом");
            }
        }
        else
        {
            Console.WriteLine("Файл пустий");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
  }

}
```

RegistrationJson.cs:

```
using System.Text.Json.Serialization;
using System.Text.Json;
using System.Text.Json.Nodes;

namespace laba3.Methods
{
   public class RegistrationJson
   {
      private readonly string _path =
"C:\\Users\\User\\source\\repos\\laba3\\laba3\\JSON\\Registration.json";
      public void AddRegistrationSerializer(List<Registration> registrations)
      {
         try
         {
            if (File.Exists(_path))
            {
               string json = File.ReadAllText(_path);

               if (!string.IsNullOrEmpty(json))
               {
                  File.WriteAllText(_path, string.Empty);
               }

               JsonSerializerOptions options = new JsonSerializerOptions()
               {
                  PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                  WriteIndented = true,
                  DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull
               };

               using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
               {
                  JsonSerializer.Serialize(fs, registrations, options);
                  Console.WriteLine("Дані записано!");
               }
```

```
            }
            else
            {
                Console.WriteLine("Файл не існує");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }

    public void GetRegistrationDeserializer()
    {
        try
        {
            using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
            {
                List<Registration>? registrations =
JsonSerializer.Deserialize<List<Registration>>(fs);
                if (registrations != null)
                {
                    foreach (var registration in registrations)
                    {
                        Console.WriteLine($"VehicleId: {registration.VehicleId}");
                        Console.WriteLine($"OwnerId: {registration.OwnerId}");
                        Console.WriteLine($"RegistrationDate:
{registration.RegistrationDate}");
                        Console.WriteLine($"RegistrationLocation:
{registration.RegistrationLocation}");
                        Console.WriteLine($"IsRegistered:
{registration.IsRegistered}");
                        Console.WriteLine();
                    }
                }
                else
```

```
            {
                Console.WriteLine("Файл пустий.");
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
}

public void GetRegistrationJson()
{
    try
    {
        string jsonString = File.ReadAllText(_path);
        if (!string.IsNullOrEmpty(jsonString))
        {
            List<Registration> registrations = new List<Registration>();
            using (JsonDocument document =
JsonDocument.Parse(jsonString))
            {
                JsonElement root = document.RootElement;
                if (root.ValueKind == JsonValueKind.Array)
                {
                    foreach (JsonElement registrationElement in
root.EnumerateArray())
                    {
                        Registration registration = new Registration(
                            registrationElement.GetProperty("VehicleId").GetInt32(),
                            registrationElement.GetProperty("OwnerId").GetInt32(),

registrationElement.GetProperty("RegistrationDate").GetDateTime(),

registrationElement.GetProperty("RegistrationLocation").GetString(),

registrationElement.GetProperty("IsRegistered").GetBoolean());
```

```
                    registrations.Add(registration);
                }
            }
        }

        foreach (var registration in registrations)
        {
            Console.WriteLine($"Vehicle ID: {registration.VehicleId},
Owner ID: {registration.OwnerId}, " +
                $"Registration Date: {registration.RegistrationDate},
Registration Location: {registration.RegistrationLocation}, " +
                $"Is Registered: {registration.IsRegistered}");
        }
      }
      else
      {
        Console.WriteLine("Файл пустий");
      }
    }
    catch (Exception ex)
    {
      Console.WriteLine($"Помилка: {ex.Message}");
    }
  }

  public void GetRegistrationJsonNode()
  {
    try
    {
      string jsonString = File.ReadAllText(_path);
      if (!string.IsNullOrEmpty(jsonString))
      {
        JsonNode? rootNode = JsonNode.Parse(jsonString);

        if (rootNode is JsonArray rootArray)
        {
```

```
                foreach (var node in rootArray)
                {
                    int vehicleId = int.Parse(node["VehicleId"].ToString());
                    int ownerId = int.Parse(node["OwnerId"].ToString());
                    DateTime registrationDate =
DateTime.Parse(node["RegistrationDate"].ToString());
                    string registrationLocation =
node["RegistrationLocation"].ToString();
                    bool isRegistered =
bool.Parse(node["IsRegistered"].ToString());

                    Console.WriteLine($"Vehicle ID: {vehicleId}, Owner ID:
{ownerId}, " +
                    $"Registration Date: {registrationDate}, Registration
Location: {registrationLocation}, " +
                    $"Is Registered: {isRegistered}");
                }
            }
            else
            {
                Console.WriteLine("Кореневий вузол не є масивом");
            }
        }
        else
        {
            Console.WriteLine("Файл пустий");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
  }

  }
}
```

VehicleDriverJson.cs:

```
using System.Text.Json.Serialization;
using System.Text.Json;
using System.Text.Json.Nodes;

namespace laba3.Methods
{
   public class VehicleDriverJson
   {
      private readonly string _path =
"C:\\Users\\User\\source\\repos\\laba3\\laba3\\JSON\\VehicleDriver.json";

      public void AddVehicleDriverSerializer(List<VehicleDriver>
vehicleDrivers)
      {
         try
         {
            if (File.Exists(_path))
            {
               string json = File.ReadAllText(_path);

               if (!string.IsNullOrEmpty(json))
               {
                  File.WriteAllText(_path, string.Empty);
               }

               JsonSerializerOptions options = new JsonSerializerOptions()
               {
                  PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                  WriteIndented = true,
                  DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull
               };
```

```
            using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
            {
                JsonSerializer.Serialize(fs, vehicleDrivers, options);
                Console.WriteLine("Дані записано!");
            }
        }
        else
        {
            Console.WriteLine("Файл не існує");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
}

public void GetVehicleDriverDeserializer()
{
    try
    {
        using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
        {
            List<VehicleDriver>? vehicleDrivers =
JsonSerializer.Deserialize<List<VehicleDriver>>(fs);
            if (vehicleDrivers != null)
            {
                foreach (var vehicleDriver in vehicleDrivers)
                {
                    Console.WriteLine($"VehicleId: {vehicleDriver.VehicleId}");
                    Console.WriteLine($"DriverId: {vehicleDriver.DriverId}");
                    Console.WriteLine();
                }
            }
            else
```

```
            {
                Console.WriteLine("Файл пустий.");
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
}


public void GetVehicleDriverJson()
{
    try
    {
        string jsonString = File.ReadAllText(_path);
        if (!string.IsNullOrEmpty(jsonString))
        {
            List<VehicleDriver> vehicleDrivers = new List<VehicleDriver>();
            using (JsonDocument document =
JsonDocument.Parse(jsonString))
            {
                JsonElement root = document.RootElement;
                if (root.ValueKind == JsonValueKind.Array)
                {
                    foreach (JsonElement vehicleDriverElement in
root.EnumerateArray())
                    {
                        VehicleDriver vehicleDriver = new VehicleDriver(

vehicleDriverElement.GetProperty("VehicleId").GetInt32(),

vehicleDriverElement.GetProperty("DriverId").GetInt32());

                        vehicleDrivers.Add(vehicleDriver);
                    }
```

```
                }
            }

            foreach (var vehicleDriver in vehicleDrivers)
            {
                Console.WriteLine($"Vehicle ID: {vehicleDriver.VehicleId},
Driver ID: {vehicleDriver.DriverId}");
            }
        }
        else
        {
            Console.WriteLine("Файл пустий");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
}


public void GetVehicleDriverJsonNode()
{
    try
    {
        string jsonString = File.ReadAllText(_path);
        if (!string.IsNullOrEmpty(jsonString))
        {
            JsonNode? rootNode = JsonNode.Parse(jsonString);

            if (rootNode is JsonArray rootArray)
            {
                foreach (var node in rootArray)
                {
                    int vehicleId = int.Parse(node["VehicleId"].ToString());
                    int driverId = int.Parse(node["DriverId"].ToString());
```

```
                    Console.WriteLine($"Vehicle ID: {vehicleId}, Driver ID:
{driverId}");
                }
            }
            else
            {
                Console.WriteLine("Кореневий вузол не є масивом");
            }
        }
        else
        {
            Console.WriteLine("Файл пустий");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
  }
}
```

VehicleOwnershipJson.cs:

```
using System.Text.Json.Serialization;
using System.Text.Json;
using System.Text.Json.Nodes;

namespace laba3.Methods
{
  public class VehicleOwnershipJson
  {
    private readonly string _path =
"C:\\Users\\User\\source\\repos\\laba3\\laba3\\JSON\\VehicleOwnership.json";
```

```
    public void AddVehicleOwnershipSerializer(List<VehicleOwnership>
ownerships)
    {
        try
        {
            if (File.Exists(_path))
            {
                string json = File.ReadAllText(_path);

                if (json != null)
                {
                    File.WriteAllText(_path, string.Empty);
                }

                JsonSerializerOptions options = new JsonSerializerOptions()
                {
                    PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                    WriteIndented = true,
                    DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull
                };

                using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
                {
                    JsonSerializer.Serialize(fs, ownerships, options);
                    Console.WriteLine("Дані записано!");
                }
            }
            else
            {
                Console.WriteLine("Файл не існує");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
```

```
        }
    }

    public void GetVehicleOwnershipDeserializer()
    {
        try
        {
            using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
            {
                List<VehicleOwnership>? ownerships =
JsonSerializer.Deserialize<List<VehicleOwnership>>(fs);
                if (ownerships != null)
                {
                    foreach (var ownership in ownerships)
                    {
                        Console.WriteLine($"VehicleId: {ownership.VehicleId}");
                        Console.WriteLine($"OwnerId: {ownership.OwnerId}");
                        Console.WriteLine();
                    }
                }
                else
                {
                    Console.WriteLine("Файл пустий.");
                }
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }

    public void GetVehicleOwnershipJson()
    {
        try
        {
```

```
            string jsonString = File.ReadAllText(_path);
            if (!string.IsNullOrEmpty(jsonString))
            {
                List<VehicleOwnership> ownerships = new
List<VehicleOwnership>();
                using (JsonDocument document =
JsonDocument.Parse(jsonString))
                {
                    JsonElement root = document.RootElement;
                    if (root.ValueKind == JsonValueKind.Array)
                    {
                        foreach (JsonElement ownershipElement in
root.EnumerateArray())
                        {
                            VehicleOwnership ownership = new VehicleOwnership(
                                ownershipElement.GetProperty("VehicleId").GetInt32(),
                                ownershipElement.GetProperty("OwnerId").GetInt32());

                            ownerships.Add(ownership);
                        }
                    }
                }

                foreach (var ownership in ownerships)
                {
                    Console.WriteLine($"Vehicle ID: {ownership.VehicleId}, Owner
ID: {ownership.OwnerId}");
                }
            }
            else
            {
                Console.WriteLine("Файл пустий");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
```

```
        }
    }


    public void AddVehicleOwnershipJsonNode(List<VehicleOwnership>
ownerships)
    {
        try
        {
            if (File.Exists(_path))
            {
                string json = File.ReadAllText(_path);
                if (!string.IsNullOrEmpty(json))
                {
                    File.WriteAllText(_path, string.Empty);
                }

                JsonArray jsonArray = new JsonArray();

                foreach (var ownership in ownerships)
                {
                    JsonNode? rootNode = JsonNode.Parse("{}");
                    if (rootNode != null)
                    {
                        rootNode["VehicleId"] = ownership.VehicleId;
                        rootNode["OwnerId"] = ownership.OwnerId;

                        jsonArray.Add(rootNode);
                    }
                }

                var options = new JsonSerializerOptions
                {
                    WriteIndented = true,
                    Encoder =
System.Text.Encodings.Web.JavaScriptEncoder.UnsafeRelaxedJsonEscaping
                };
```

```
            string jsonString = jsonArray.ToJsonString(options);

            File.WriteAllText(_path, jsonString);

            Console.WriteLine($"Дані записані");
        }
        else
        {
            Console.WriteLine("Файл не існує");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
}

public void GetVehicleOwnershipJsonNode()
{
    try
    {
        string jsonString = File.ReadAllText(_path);
        if (!string.IsNullOrEmpty(jsonString))
        {
            JsonNode? rootNode = JsonNode.Parse(jsonString);

            if (rootNode is JsonArray rootArray)
            {
                foreach (var node in rootArray)
                {
                    int vehicleId = int.Parse(node["VehicleId"].ToString());
                    int ownerId = int.Parse(node["OwnerId"].ToString());

                    Console.WriteLine($"Vehicle ID: {vehicleId}, Owner ID:
{ownerId}");
                }
```

```
                }
                else
                {
                    Console.WriteLine("Кореневий вузол не є масивом");
                }
            }
            else
            {
                Console.WriteLine("Файл пустий");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }
  }
}
```

VehicleJson.cs:

```
using System.Text.Json.Serialization;
using System.Text.Json;
using System.Text.Json.Nodes;

namespace laba3.Methods
{
   public class VehicleJson
   {
      private readonly string _path =
"C:\\Users\\User\\source\\repos\\laba3\\laba3\\JSON\\Vehicle.json";

      public void AddVehicleSerializer(List<Vehicle> vehicles)
      {
         try
         {
```

```
            if (File.Exists(_path))
            {
                string json = File.ReadAllText(_path);

                if (!string.IsNullOrEmpty(json))
                {
                    File.WriteAllText(_path, string.Empty);
                }

                JsonSerializerOptions options = new JsonSerializerOptions()
                {
                    PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
                    WriteIndented = true,
                    DefaultIgnoreCondition =
JsonIgnoreCondition.WhenWritingNull
                };

                using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
                {
                    JsonSerializer.Serialize(fs, vehicles, options);
                    Console.WriteLine("Дані записано!");
                }
            }
            else
            {
                Console.WriteLine("Файл не існує");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }

    public void GetVehicleDeserializer()
    {
```

```
        try
        {
            using (FileStream fs = new FileStream(_path,
FileMode.OpenOrCreate))
            {
                List<Vehicle>? vehicles =
JsonSerializer.Deserialize<List<Vehicle>>(fs);
                if (vehicles != null)
                {
                    foreach (var vehicle in vehicles)
                    {
                        Console.WriteLine($"VehicleId: {vehicle.VehicleId}");
                        Console.WriteLine($"Brand: {vehicle.Brand}");
                        Console.WriteLine($"Manufacturer:
{vehicle.Manufacturer}");
                        Console.WriteLine($"Model: {vehicle.Model}");
                        Console.WriteLine($"BodyType: {vehicle.BodyType}");
                        Console.WriteLine($"YearOfManufacture:
{vehicle.YearOfManufacture}");
                        Console.WriteLine($"ChassisNumber:
{vehicle.ChassisNumber}");
                        Console.WriteLine($"Color: {vehicle.Color}");
                        Console.WriteLine($"LicensePlate: {vehicle.LicensePlate}");
                        Console.WriteLine($"TechnicalCondition:
{vehicle.TechnicalCondition}");

                        if (vehicle.Registrations != null)
                        {
                            Console.WriteLine("Registrations:");
                            foreach (var registration in vehicle.Registrations)
                            {
                                Console.WriteLine($"RegistrationDate:
{registration.RegistrationDate}");
                                Console.WriteLine($"RegistrationLocation:
{registration.RegistrationLocation}");
                                Console.WriteLine($"IsRegistered:
{registration.IsRegistered}");
```

```
                    }
                }
                Console.WriteLine();
            }
        }
        else
        {
            Console.WriteLine("Файл пустий.");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
}

public void GetVehicleJson()
{
    try
    {
        string jsonString = File.ReadAllText(_path);
        if (!string.IsNullOrEmpty(jsonString))
        {
            List<Vehicle> vehicles = new List<Vehicle>();
            using (JsonDocument document =
JsonDocument.Parse(jsonString))
            {
                JsonElement root = document.RootElement;
                if (root.ValueKind == JsonValueKind.Array)
                {
                    foreach (JsonElement vehicleElement in
root.EnumerateArray())
                    {
                        Vehicle vehicle = new Vehicle(
                            vehicleElement.GetProperty("VehicleId").GetInt32(),
                            vehicleElement.GetProperty("Brand").GetString(),
```

```
vehicleElement.GetProperty("Manufacturer").GetString(),
                        vehicleElement.GetProperty("Model").GetString(),
                        vehicleElement.GetProperty("BodyType").GetString(),

vehicleElement.GetProperty("YearOfManufacture").GetInt32(),

vehicleElement.GetProperty("ChassisNumber").GetString(),
                        vehicleElement.GetProperty("Color").GetString(),
                        vehicleElement.GetProperty("LicensePlate").GetString(),

vehicleElement.GetProperty("TechnicalCondition").GetString());

                    // Parse Registrations if present
                    if (vehicleElement.TryGetProperty("Registrations", out
JsonElement registrationsElement))
                    {
                        List<Registration> registrations = new
List<Registration>();
                        foreach (JsonElement registrationElement in
registrationsElement.EnumerateArray())
                        {
                            Registration registration = new Registration(

registrationElement.GetProperty("VehicleId").GetInt32(),

registrationElement.GetProperty("OwnerId").GetInt32(),

registrationElement.GetProperty("RegistrationDate").GetDateTime(),

registrationElement.GetProperty("RegistrationLocation").GetString(),

registrationElement.GetProperty("IsRegistered").GetBoolean());

                            registrations.Add(registration);
                        }
                        vehicle.Registrations = registrations;
```

```
                }

                vehicles.Add(vehicle);
            }
        }
    }

    foreach (var vehicle in vehicles)
    {
        Console.WriteLine($"Vehicle ID: {vehicle.VehicleId}, Brand:
{vehicle.Brand}, " +
            $"Manufacturer: {vehicle.Manufacturer}, Model:
{vehicle.Model}, " +
            $"Body Type: {vehicle.BodyType}, Year of Manufacture:
{vehicle.YearOfManufacture}, " +
            $"Chassis Number: {vehicle.ChassisNumber}, Color:
{vehicle.Color}, " +
            $"License Plate: {vehicle.LicensePlate}, Technical Condition:
{vehicle.TechnicalCondition}");

        // Print Registrations if present
        if (vehicle.Registrations != null && vehicle.Registrations.Any())
        {
            Console.WriteLine("Registrations:");
            foreach (var registration in vehicle.Registrations)
            {
                Console.WriteLine($"   Vehicle ID:
{registration.VehicleId}, Owner ID: {registration.OwnerId}, " +
                    $"Registration Date: {registration.RegistrationDate}, " +
                    $"Registration Location:
{registration.RegistrationLocation}, " +
                    $"Is Registered: {registration.IsRegistered}");
            }
        }

        Console.WriteLine();
    }
```

```
            }
            else
            {
                Console.WriteLine("Файл пустий");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Помилка: {ex.Message}");
        }
    }


    public void GetVehicleJsonNode()
    {
        try
        {
            string jsonString = File.ReadAllText(_path);
            if (!string.IsNullOrEmpty(jsonString))
            {
                JsonNode? rootNode = JsonNode.Parse(jsonString);

                if (rootNode is JsonArray rootArray)
                {
                    foreach (var node in rootArray)
                    {
                        int vehicleId = int.Parse(node["VehicleId"].ToString());
                        string brand = node["Brand"].ToString();
                        string manufacturer = node["Manufacturer"].ToString();
                        string model = node["Model"].ToString();
                        string bodyType = node["BodyType"].ToString();
                        int yearOfManufacture =
int.Parse(node["YearOfManufacture"].ToString());
                        string chassisNumber = node["ChassisNumber"].ToString();
                        string color = node["Color"].ToString();
                        string licensePlate = node["LicensePlate"].ToString();
```

```
                    string technicalCondition =
node["TechnicalCondition"].ToString();

                    Console.WriteLine($"Vehicle ID: {vehicleId}, Brand:
{brand}, " +
                    $"Manufacturer: {manufacturer}, Model: {model}, " +
                    $"Body Type: {bodyType}, Year of Manufacture:
{yearOfManufacture}, " +
                    $"Chassis Number: {chassisNumber}, Color: {color}, " +
                    $"License Plate: {licensePlate}, Technical Condition:
{technicalCondition}");
                }
            }
            else
            {
                Console.WriteLine("Кореневий вузол не є масивом");
            }
        }
        else
        {
            Console.WriteLine("Файл пустий");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Помилка: {ex.Message}");
    }
  }

}
```

Основна програма:

```
using laba3.Methods;
namespace laba3
```

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

         var data = new Data();
         var driverJson = new DriverJson();
         var ownerJson = new OwnerJson();
         var registrationJson = new RegistrationJson();
         var vehicleJson = new VehicleJson();
         var vehicleDriverJson = new VehicleDriverJson();
         var vehicleOwnershipJson = new VehicleOwnershipJson();
         int input;

         do
         {
            Console.Clear();
            Console.WriteLine("Головне меню \n" +
               "\n1. Серіалізація списку об'єктів Driver" +
               "\n2. Десеріалізація списку об'єктів Driver" +
               "\n3. Отримати список об'єктів Driver за домогою
JsonDocument" +
               "\n4. Отримати список об'єктів Driver за допомогою JsonNode"
+

               "\n5. Серіалізація списку об'єктів Owner" +
               "\n6. Десеріалізація списку об'єктів Owner" +
               "\n7. Отримати список об'єктів Owner за допомогою
JsonDocument" +
               "\n8. Отримати список об'єктів Owner за допомогою JsonNode"
+

               "\n9. Серіалізація списку об'єктів Registration" +
               "\n10. Десеріалізація списку об'єктів Registration" +
```

```
            "\n11. Отримати список об'єктів Registration за допомогою
JsonDocument" +
            "\n12. Отримати список об'єктів Registration за допомогою
JsonNode" +

            "\n13. Серіалізація списку об'єктів Vehicle" +
            "\n14. Десеріалізація списку об'єктів Vehicle" +
            "\n15. Отримати список об'єктів Vehicle за допомогою
JsonDocument" +
            "\n16. Отримати список об'єктів за допомогою Vehicle
JsonNode" +

            "\n17. Серіалізація списоку об'єктів VehicleDriver" +
            "\n18. Десеріалізація списку об'єктів VehicleDriver" +
            "\n19. Отримати список об'єктів VehicleDriver за допомогою
JsonDocument" +
            "\n20. Отримати список об'єктів VehicleDriver за допомогою
JsonNode" +

            "\n21. Серіалізація списку об'єктів VehicleOwnership" +
            "\n22. Десеріалізація списку об'єктів VehicleOwnership" +
            "\n23. Отримати список об'єктів VehicleOwnership за
допомогою JsonDocument" +
            "\n24. Отримати список об'єктів VehicleOwnership за
допомогою JsonNode" +

            "\n\nВаш вибір:");

        input = Convert.ToInt32(Console.ReadLine());
        switch (input)
        {
            case 1:
                Console.Clear();
                driverJson.AddDriverSerializer(data.Drivers);
                break;
            case 2:
                Console.Clear();
```

```
    driverJson.GetDriverDeserializer();
    break;
case 3:
    Console.Clear();
    driverJson.GetDriverJson();
    break;
case 4:
    Console.Clear();
    driverJson.GetDriverJsonNode();
    break;

case 5:
    Console.Clear();
    ownerJson.AddOwnerSerializer(data.Owners);
    break;
case 6:
    Console.Clear();
    ownerJson.GetOwnerDeserializer();
    break;
case 7:
    Console.Clear();
    ownerJson.GetOwnerJson();
    break;
case 8:
    Console.Clear();
   ownerJson.GetOwnerJsonNode();
    break;

case 9:
    Console.Clear();
    registrationJson.AddRegistrationSerializer(data.Registrations);
    break;
case 10:
    Console.Clear();
    registrationJson.GetRegistrationDeserializer();
    break;
case 11:
```

```
                Console.Clear();
                registrationJson.GetRegistrationJson();
                break;
            case 12:
                Console.Clear();
                registrationJson.GetRegistrationJsonNode();
                break;

            case 13:
                Console.Clear();
                vehicleJson.AddVehicleSerializer(data.Vehicles);
                break;
            case 14:
                Console.Clear();
                vehicleJson.GetVehicleDeserializer();
                break;
            case 15:
                Console.Clear();
                vehicleJson.GetVehicleJson();
                break;
            case 16:
                Console.Clear();
                vehicleJson.GetVehicleJsonNode();
                break;

            case 17:
                Console.Clear();

vehicleDriverJson.AddVehicleDriverSerializer(data.VehicleDrivers);
                break;
            case 18:
                Console.Clear();
                vehicleDriverJson.GetVehicleDriverDeserializer();
                break;
            case 19:
                Console.Clear();
                vehicleDriverJson.GetVehicleDriverJson();
```

```
                    break;
                case 20:
                    Console.Clear();
                    vehicleDriverJson.GetVehicleDriverJsonNode();
                    break;

                case 21:
                    Console.Clear();

vehicleOwnershipJson.AddVehicleOwnershipSerializer(data.VehicleOwnership
s);
                    break;
                case 22:
                    Console.Clear();
                    vehicleOwnershipJson.GetVehicleOwnershipDeserializer();
                    break;
                case 23:
                    Console.Clear();
                    vehicleOwnershipJson.GetVehicleOwnershipJson();
                    break;
                case 24:
                    Console.Clear();
                    vehicleOwnershipJson.GetVehicleOwnershipJsonNode();
                    break;

                default:
                    Console.WriteLine("Неправильний вибір");
                    break;
            }

            Console.WriteLine("Натисніть + для продовження");
            answer = Console.ReadLine();

        } while (answer == "+");
    }
  }
}
```