Лабораторна робота № 1-2

З дисципліни: ООП

Виконав:

Студент групи КН-31

Зайков Арсеній

Одеса, 2024

## **Завдання № 7 (ДТП)**

Потрібно розробити програмну систему, призначену для аналізу дорожньо-транспортних пригод (ДТП). Така система повинна забезпечувати зберігання відомостей про ДТП. Для кожної ДТП мають бути збережені: вид ДТП (наїзд на пішохода, наїзд на перешкоду, зіткнення, перекидання тощо), дата, державні номери автомобілів (якщо в ДТП брали участь кілька автомобілів), дані про водія і причина (виїзд на смугу зустрічного руху, стан водія, несправність автомобіля, порушення ПДР тощо). Дата задається в межах одного місяця.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Патрульна поліція | Водій | Автомобіль | ДТП | Участники |
| Ключ | КЛЮЧ | Ключ | Ключ | Ключ ДТП |
| Назва відділу | ПІБ | Фірма | № акта о ДТП | Водитель (ключ) |
|  | Стаж | Марка | Дата | Авто (ключ) |
|  | Посвідчення | Тип кузова | Место |  |
|  |  | Держномер авто | Кол-во пострадавших |  |
|  |  | Власник | Вид ДТП |  |
|  |  |  | Причина ДТП |  |

Должны быть созданы обобщенные списки:

* Сведения о водителях.
* Сведения об автомобилях.
* Сведения об ДТП
* Відомості про учасників ДТП

Руководству МВД могут понадобиться следующие сведения:

* Список водителей, совершивших более одного ДТП,
* Список водителей, участвующих в ДТП в заданном месте.
* Список водителей, участвующих в ДТП на заданную дату.
* ДТП с максимальным количеством потерпевших.
* Список водителей, участвующих в ДТП с наездом на пешеходов.

Должна быть предусмотрена возможность добавления и удаления информации о ДТП

**Код:**

Клас DataController:

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.IO;

using LabKozl;

using Newtonsoft.Json;

public class DataController

{

private static DataController \_instance;

private static readonly object \_lock = new object();

private List<Department> \_departments;

private List<Car> \_cars;

private List<Driver> \_drivers;

private List<DTP> \_dtps;

private List<Member> \_members;

private DataController()

{

\_departments = new List<Department>();

\_cars = new List<Car>();

\_drivers = new List<Driver>();

\_dtps = new List<DTP>();

}

public static DataController GetInstance()

{

if (\_instance == null)

{

lock (\_lock)

{

if (\_instance == null)

{

\_instance = new DataController();

}

}

}

return \_instance;

}

#region JSON WORK

private List<T> LoadDataFromJson<T>(string filePath)

{

try

{

if (File.Exists(filePath))

{

string json = File.ReadAllText(filePath);

return JsonConvert.DeserializeObject<List<T>>(json) ?? new List<T>();

}

else

{

Console.WriteLine("File doesn't exist");

return new List<T>();

}

}

catch (Exception ex)

{

Console.WriteLine($"Error loading data from {filePath}: {ex.Message}");

return new List<T>();

}

}

private void SaveDataToFile<T>(string filePath, List<T> collection)

{

try

{

string json = JsonConvert.SerializeObject(collection, Formatting.Indented);

File.WriteAllText(filePath, json);

}

catch (Exception ex)

{

Console.WriteLine($"Error saving data to {filePath}: {ex.Message}");

}

}

#endregion

#region Save and Load data

public void LoadData()

{

\_departments = LoadDataFromJson<Department>("departments.json");

\_cars = LoadDataFromJson<Car>("cars.json");

\_drivers = LoadDataFromJson<Driver>("drivers.json");

\_dtps = LoadDataFromJson<DTP>("dtps.json");

\_members = LoadDataFromJson<Member>("members.json");

}

public void SaveData()

{

SaveDataToFile("departments.json", \_departments);

SaveDataToFile("cars.json", \_cars);

SaveDataToFile("drivers.json", \_drivers);

SaveDataToFile("dtps.json", \_dtps);

SaveDataToFile("members.json", \_members);

}

#endregion

#region ADDDATA

public void AddDepartment(Department department)

{

\_departments.Add(department);

}

public void AddCar(Car car)

{

\_cars.Add(car);

}

public void AddDriver(Driver driver)

{

\_drivers.Add(driver);

}

public void AddDTP(DTP dtp)

{

\_dtps.Add(dtp);

}

#endregion

#region REMOVEDATA

public void RemoveCar(Car car)

{

\_cars.Remove(car);

}

#endregion

#region DISLPAY

public List<Department> GetDepartments()

{

return \_departments;

}

public List<Car> GetCars()

{

return \_cars;

}

public List<Driver> GetDrivers()

{

return \_drivers;

}

public List<DTP> GetDTPs()

{

return \_dtps;

}

#endregion

#region Terminal

public void ShowMainMenu()

{

Console.WriteLine("1. Show Data\n2. Add DTP\n3. Delete DTP\n4. Exit");

string option = Console.ReadLine();

if (option != string.Empty) GetMethodByOption(option);

}

private void GetMethodByOption(string option)

{

switch(option)

{

case "1": DisplayData(); break;

case "2": AddDTP(); break;

case "3": DeleteDTP(); break;

case "4":

SaveData();

ExitProgram();

break;

default: Console.WriteLine("Invalid Option"); break;

}

}

private void ExitProgram()

{

Console.WriteLine("Exiting the program...");

Environment.Exit(0);

}

private void DisplayData()

{

Console.WriteLine("Departments:");

foreach (var item in \_departments)

{

Console.WriteLine($"Id: {item.Id}\tName: {item.Name}");

}

Console.WriteLine("Cars:");

foreach (var item in \_cars)

{

Console.WriteLine($"Id: {item.Id}\tName: {item.Name}");

}

Console.WriteLine("Drivers:");

foreach (var item in \_drivers)

{

Console.WriteLine($"Id: {item.Id}\tName: {item.FullName}\tCetrificate: {item.Certificate}");

}

Console.WriteLine("DTPs:");

foreach (var item in \_dtps)

{

Console.WriteLine($"Id: {item.Id}\tDTP cause: {item.DTPCause}\tDTP date: {item.DateDTP}");

}

Console.WriteLine("Members:");

foreach (var item in \_members)

{

Console.WriteLine($"DTP Id: {item.DTPId}\tDriver Id: {item.DriverId}\tCar Id: {item.CarId}");

}

}

private void AddDTP()

{

try

{

Console.WriteLine("Enter the date and time of the accident (yyyy-MM-dd):");

DateTime dateDTP = DateTime.ParseExact(Console.ReadLine(), "yyyy-MM-dd", null);

Console.WriteLine("Enter the place of the accident:");

string place = Console.ReadLine();

Console.WriteLine("Enter the casualty rate:");

uint casualtyRate = uint.Parse(Console.ReadLine());

Console.WriteLine("Enter the act number:");

ulong actNumber = ulong.Parse(Console.ReadLine());

Console.WriteLine("Enter the cause of the accident:");

string dtpCause = Console.ReadLine();

DTP newDTP = new DTP

{

Id = \_dtps.Any() ? \_dtps.Last().Id + 1 : 1,

DateDTP = dateDTP,

Plase = place,

СasualtyRate = casualtyRate,

ActNumber = actNumber,

DTPCause = dtpCause

};

GetInstance().AddDTP(newDTP);

Console.WriteLine("DTP added successfully.");

}

catch (FormatException)

{

Console.WriteLine("Invalid input format.");

}

catch (Exception ex)

{

Console.WriteLine($"Error adding DTP: {ex.Message}");

}

SaveDataToFile("dtps.json", \_dtps);

}

private void DeleteDTP()

{

try

{

Console.WriteLine("Select DTP by Id to delete:");

foreach (var dtp in GetInstance().GetDTPs())

{

Console.WriteLine($"Id: {dtp.Id}\tDTP cause: {dtp.DTPCause}\tDTP date: {dtp.DateDTP}");

}

Console.WriteLine("Enter the Id of the DTP you want to delete:");

int idToDelete = int.Parse(Console.ReadLine());

DTP dtpToDelete = GetInstance().GetDTPs().FirstOrDefault(d => d.Id == idToDelete);

if (dtpToDelete != null)

{

\_dtps.Remove(dtpToDelete);

Console.WriteLine("DTP deleted successfully.");

}

else

{

Console.WriteLine("DTP with specified Id not found.");

}

}

catch (FormatException)

{

Console.WriteLine("Invalid input format.");

}

catch (Exception ex)

{

Console.WriteLine($"Error deleting DTP: {ex.Message}");

}

SaveDataToFile("dtps.json", \_dtps);

}

#endregion

}

Клас ObjectsData:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LabKozl

{

public class Car

{

public int Id { get; set; }

public string Name { get; set; }

public string Firm { get; set; }

public string Brand { get; set; }

public string BodyType { get; set; }

public string LicensePlates { get; set; }

public string Owner { get; set; }

public override string ToString()

{

return $"{Id}";

}

}

public class Department

{

public int Id { get; set; }

public string Name { get; set; }

}

public class Driver

{

public int Id { get; set; }

public string FullName { get; set; }

public uint Exp { get; set; }

public string Certificate { get; set; }

public override string ToString()

{

return $"{Id}";

}

}

public class DTP

{

public DateTime DateDTP { get; set; }

public int Id { get; set; }

public string Plase { get; set; }

public uint СasualtyRate { get; set; }

public ulong ActNumber { get; set; }

public string DTPCause { get; set; }

public override string ToString()

{

return $"{Id}";

}

}

public class Member

{

public DTP DTPId { get; set; }

public Driver DriverId { get; set; }

public Car CarId { get; set; }

}

}

Клас Program:  
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LabKozl

{

public class Program

{

public static void Main(string[] args)

{

DataController dataController = DataController.GetInstance();

dataController.LoadData();

while (true)

{

dataController.ShowMainMenu();

}

}

}

}

Файли:

members.json:

[

{

"DTPId": {

"DateDTP": "2024-02-08T12:30:00",

"Id": 1,

"Plase": "Intersection1",

"СasualtyRate": 2,

"ActNumber": 123456789012,

"DTPCause": "Collision"

},

"DriverId": {

"Id": 1,

"FullName": "Driver1",

"Exp": 3,

"Certificate": "ABCD1234"

},

"CarId": {

"Id": 1,

"Name": "Car1",

"Firm": "Firm1",

"Brand": "Brand1",

"BodyType": "Type1",

"LicensePlates": "ABC123",

"Owner": "Owner1"

}

},

{

"DTPId": {

"DateDTP": "2024-02-08T15:45:00",

"Id": 2,

"Plase": "Highway Exit2",

"СasualtyRate": 1,

"ActNumber": 234567890123,

"DTPCause": "Rear-end collision"

},

"DriverId": {

"Id": 2,

"FullName": "Driver2",

"Exp": 5,

"Certificate": "EFGH5678"

},

"CarId": {

"Id": 2,

"Name": "Car2",

"Firm": "Firm2",

"Brand": "Brand2",

"BodyType": "Type2",

"LicensePlates": "XYZ789",

"Owner": "Owner2"

}

},

{

"DTPId": {

"DateDTP": "2024-02-09T09:15:00",

"Id": 3,

"Plase": "City Center3",

"СasualtyRate": 3,

"ActNumber": 345678901234,

"DTPCause": "Pedestrian hit"

},

"DriverId": {

"Id": 3,

"FullName": "Driver3",

"Exp": 7,

"Certificate": "IJKL9012"

},

"CarId": {

"Id": 3,

"Name": "Car3",

"Firm": "Firm3",

"Brand": "Brand3",

"BodyType": "Type3",

"LicensePlates": "123XYZ",

"Owner": "Owner3"

}

},

{

"DTPId": {

"DateDTP": "2024-02-10T14:00:00",

"Id": 4,

"Plase": "Parking Lot4",

"СasualtyRate": 0,

"ActNumber": 456789012345,

"DTPCause": "No injuries, property damage only"

},

"DriverId": {

"Id": 4,

"FullName": "Driver4",

"Exp": 2,

"Certificate": "MNOP3456"

},

"CarId": {

"Id": 4,

"Name": "Car4",

"Firm": "Firm4",

"Brand": "Brand4",

"BodyType": "Type4",

"LicensePlates": "456ABC",

"Owner": "Owner4"

}

},

{

"DTPId": {

"DateDTP": "2024-02-11T18:20:00",

"Id": 5,

"Plase": "Residential Area5",

"СasualtyRate": 1,

"ActNumber": 567890123456,

"DTPCause": "Side impact"

},

"DriverId": {

"Id": 5,

"FullName": "Driver5",

"Exp": 4,

"Certificate": "QRST7890"

},

"CarId": {

"Id": 5,

"Name": "Car5",

"Firm": "Firm5",

"Brand": "Brand5",

"BodyType": "Type5",

"LicensePlates": "789DEF",

"Owner": "Owner5"

}

}

]

dtps.json:

[

{

"DateDTP": "2024-02-02T00:00:00",

"Id": 2,

"Plase": "me",

"СasualtyRate": 3,

"ActNumber": 123123123,

"DTPCause": "me"

}

]

drivers.json:

[

{

"Id": 1,

"FullName": "Driver1",

"Exp": 3,

"Certificate": "ABCD1234"

},

{

"Id": 2,

"FullName": "Driver2",

"Exp": 5,

"Certificate": "EFGH5678"

},

{

"Id": 3,

"FullName": "Driver3",

"Exp": 7,

"Certificate": "IJKL9012"

},

{

"Id": 4,

"FullName": "Driver4",

"Exp": 2,

"Certificate": "MNOP3456"

},

{

"Id": 5,

"FullName": "Driver5",

"Exp": 4,

"Certificate": "QRST7890"

}

]

departments.json:

[

{

"Id": 1,

"Name": "Department1"

},

{

"Id": 2,

"Name": "Department2"

},

{

"Id": 3,

"Name": "Department3"

},

{

"Id": 4,

"Name": "Department4"

},

{

"Id": 5,

"Name": "Department5"

}

]

cars.json:

[

{

"Id": 1,

"Name": "Car1",

"Firm": "Firm1",

"Brand": "Brand1",

"BodyType": "Type1",

"LicensePlates": "ABC123",

"Owner": "Owner1"

},

{

"Id": 2,

"Name": "Car2",

"Firm": "Firm2",

"Brand": "Brand2",

"BodyType": "Type2",

"LicensePlates": "XYZ789",

"Owner": "Owner2"

},

{

"Id": 3,

"Name": "Car3",

"Firm": "Firm3",

"Brand": "Brand3",

"BodyType": "Type3",

"LicensePlates": "123XYZ",

"Owner": "Owner3"

},

{

"Id": 4,

"Name": "Car4",

"Firm": "Firm4",

"Brand": "Brand4",

"BodyType": "Type4",

"LicensePlates": "456ABC",

"Owner": "Owner4"

},

{

"Id": 5,

"Name": "Car5",

"Firm": "Firm5",

"Brand": "Brand5",

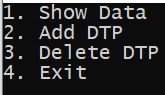
"BodyType": "Type5",

"LicensePlates": "789DEF",

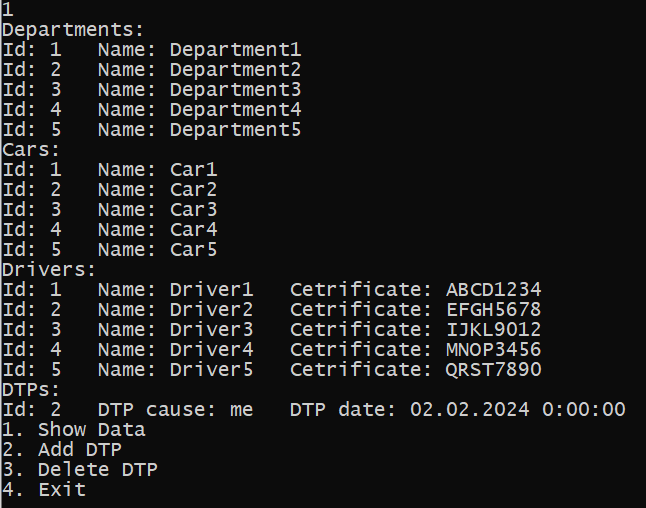
"Owner": "Owner5"

}

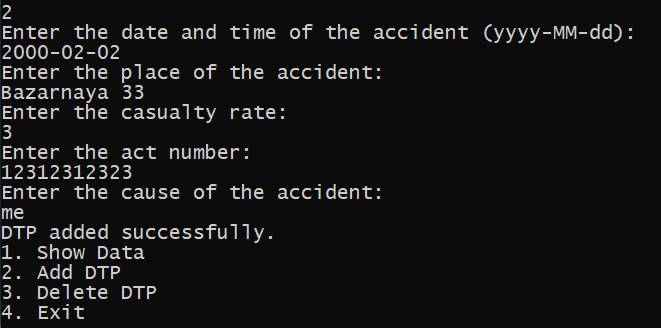
]

**Результат виконання:  
**

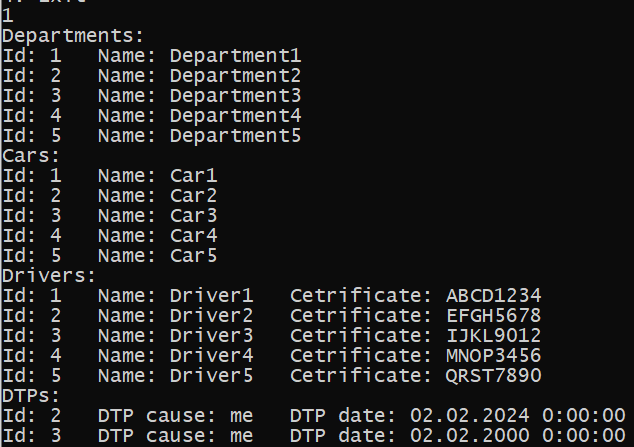
**Рисунок 1 -** "Головне меню"

****

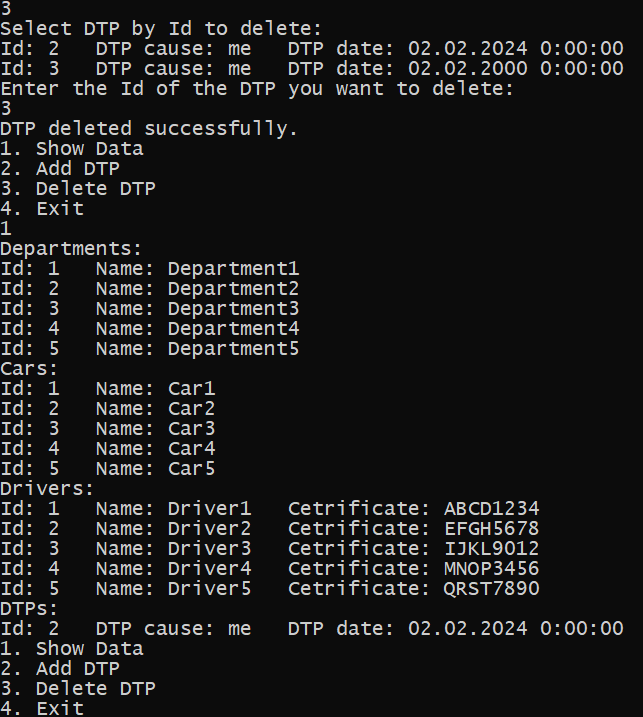
**Рисунок 2 -** Відображення інформації

****

**Рисунок 3 -** Ввод даних для нового ДТП

****

**Рисунок 4 -** Нове ДТП було додано

****

**Рисунок 5 -** Видалення ДТП