**Лабораторна робота №4. Використання принципів та відносин об’єктно-орієнтованого програмування у мові C# .**

**Виконавець: Лазоренко М.Д.**

**Група виконавця: КН-922в**

**Мета роботи:** набути навички в написанні програм із застосуванням принципів та відносин об’єктно-орієнтованого програмування (ООП).

**Індивідуальні завдання**

**Варіант №9**

**Завдання 1**

Розробити клас згідно до свого варіанта. Включити до класу методи set (…), get (…), show (…) та ін., використовуючи принцип інкапсуляції. Окрім гетерів, сетерів конструкторів та інших стандартних методів розробити не менше двох додаткових методів, які реалізують власний функціонал об’єкту відповідної предметної галузі. Написати програму, яка створює список, масив чи колекцію, об’єктів на основі відповідного класу. Під час створення використати випадковий підхід для генерації числової та текстової інформації об’єктів списку. Передбачити можливість додання, редагування та видалення об’єктів зі списку за певними критеріями. В програмі реалізувати функціонал за своїм варіантом.

9. **Bus:** Прізвище та ініціали водія, Номер автобуса, Номер маршруту,

Марка, Рік початку експлуатації, Пробіг. Створити масив об'єктів.

Вивести:

• список автобусів для заданого номера маршруту;

• список автобусів, які експлуатуються понад 10 років;

• список автобусів, пробіг яких більше 10 000 км.

***Текст програми:***

using System;

using System.Collections.Generic;

public class Bus

{

private string driverName; // Прізвище та ініціали водія

private int busNumber; // Номер автобуса

private int routeNumber; // Номер маршруту

private string brand; // Марка автобуса

private int yearOfStart; // Рік початку експлуатації

private double mileage; // Пробіг

// Конструктор

public Bus(string driverName, int busNumber, int routeNumber, string brand, int yearOfStart, double mileage)

{

this.driverName = driverName;

this.busNumber = busNumber;

this.routeNumber = routeNumber;

this.brand = brand;

this.yearOfStart = yearOfStart;

this.mileage = mileage;

}

// Методи set та get

public string DriverName

{

get { return driverName; }

set { driverName = value; }

}

public int BusNumber

{

get { return busNumber; }

set { busNumber = value; }

}

public int RouteNumber

{

get { return routeNumber; }

set { routeNumber = value; }

}

public string Brand

{

get { return brand; }

set { brand = value; }

}

public int YearOfStart

{

get { return yearOfStart; }

set { yearOfStart = value; }

}

public double Mileage

{

get { return mileage; }

set { mileage = value; }

}

// Метод для відображення інформації про автобус

public void ShowInfo()

{

Console.WriteLine($"Driver: {driverName}, Bus Number: {busNumber}, Route Number: {routeNumber}, Brand: {brand}, Year of Start: {yearOfStart}, Mileage: {mileage} km");

}

// Додатковий метод 1: Чи експлуатується автобус понад 10 років

public bool IsOver10Years()

{

return (DateTime.Now.Year - yearOfStart) > 10;

}

// Додатковий метод 2: Чи перевищує пробіг 10 000 км

public bool IsMileageOver10000()

{

return mileage > 10000;

}

}

class task01

{

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

List<Bus> busList = GenerateBusList(10);

// Вивести список автобусів для заданого номера маршруту

Console.WriteLine("Enter the route number to filter buses:");

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

ShowBusesByRoute(busList, routeNumber);

// Вивести список автобусів, які експлуатуються понад 10 років

Console.WriteLine("\nBuses over 10 years in operation:");

ShowBusesOver10Years(busList);

// Вивести список автобусів, пробіг яких більше 10 000 км

Console.WriteLine("\nBuses with mileage over 10,000 km:");

ShowBusesWithMileageOver10000(busList);

Console.ReadKey();

}

// Метод для генерації списку автобусів

static List<Bus> GenerateBusList(int count)

{

var random = new Random();

var busList = new List<Bus>();

string[] driverNames = { "Ivanov I.I.", "Petrov P.P.", "Sidorov S.S.", "Kuznetsov K.K." };

string[] brands = { "Mercedes", "Volvo", "Ikarus", "MAN" };

for (int i = 0; i < count; i++)

{

string driverName = driverNames[random.Next(driverNames.Length)];

int busNumber = random.Next(100, 999);

int routeNumber = random.Next(1, 10);

string brand = brands[random.Next(brands.Length)];

int yearOfStart = random.Next(1995, DateTime.Now.Year);

double mileage = random.Next(5000, 20000);

busList.Add(new Bus(driverName, busNumber, routeNumber, brand, yearOfStart, mileage));

}

return busList;

}

// Метод для виведення автобусів за номером маршруту

static void ShowBusesByRoute(List<Bus> buses, int routeNumber)

{

foreach (var bus in buses)

{

if (bus.RouteNumber == routeNumber)

{

bus.ShowInfo();

}

}

}

// Метод для виведення автобусів, які експлуатуються понад 10 років

static void ShowBusesOver10Years(List<Bus> buses)

{

foreach (var bus in buses)

{

if (bus.IsOver10Years())

{

bus.ShowInfo();

}

}

}

// Метод для виведення автобусів з пробігом більше 10 000 км

static void ShowBusesWithMileageOver10000(List<Bus> buses)

{

foreach (var bus in buses)

{

if (bus.IsMileageOver10000())

{

bus.ShowInfo();

}

}

}

}

***Результат роботи програми:***



Ця програма на C# призначена для управління інформацією про автобуси, зокрема для фільтрації їх за маршрутом, віком та пробігом. Користувач може ввести номер маршруту для відображення відповідних автобусів, а також отримати списки автобусів, які експлуатуються понад 10 років або мають пробіг більше 10 000 км.

**Основні етапи виконання програми:**

1. **Введення даних:**
   * Користувач не вводить дані вручну, оскільки автобуси генеруються випадковим чином. Програма створює список автобусів за допомогою методу GenerateBusList.
2. **Виведення автобусів за номером маршруту:**
   * Користувач може ввести номер маршруту для фільтрації автобусів. Програма викликає метод ShowBusesByRoute, який перебирає список автобусів і виводить інформацію про ті, що відповідають введеному номеру маршруту.
3. **Виведення автобусів, що експлуатуються понад 10 років:**
   * Метод ShowBusesOver10Years перебирає список автобусів і виводить інформацію про ті, які були в експлуатації більше 10 років, за допомогою методу IsOver10Years.
4. **Виведення автобусів з пробігом понад 10 000 км:**
   * Метод ShowBusesWithMileageOver10000 перебирає автобуси та виводить інформацію про ті, що мають пробіг більше 10 000 км, за допомогою методу IsMileageOver10000.

**Основні функції програми:**

* **Генерація списку автобусів:** Випадковий генератор створює список автобусів із заданою кількістю (10) з випадковими даними для кожного автобуса.
* **Виведення інформації про автобуси:** Програма забезпечує виведення детальної інформації про автобуси, включаючи прізвище водія, номер автобуса, номер маршруту, марку, рік початку експлуатації та пробіг.
* **Фільтрація автобусів:** Програма дозволяє фільтрувати автобуси за різними критеріями, такими як номер маршруту, вік та пробіг, що робить її корисною для управління автобусними перевезеннями.

**Завдання 2**

**1.** Для класу з минулого завдання, ретельно занурившись у предметну область власного варіанта, створити абстрактний клас чи інтерфейс, який має стати предком, зв’язати ці класи відносинами спадкування. Наприклад для класу Book (Книга) можна створити клас Reader (Засіб читання).

**2.** Окрім вашого класу додати до абстрактного класу чи інтерфейсу, створеному у попередньому пункті, ще два класи-спадкоємця першого рівня. Наприклад до класу Book (Книга) можна додати класи Newspaper (Газета) Magazine (Журнал) з відповідними полями та методами.

**3.** Для одного з доданих у попередньому завданні класів створити не менше двох спадкоємців другого рівня спадкування. Наприклад до класу Magazine (Журнал) можна додати класи PMagazine (Паперовий журнал), EMagazine (Електронний журнал), IMagazine (Інтернет журнал). У кожного нащадка має бути не менш ніж на два поля та метода більше ніж у предка.

**4.** Створити масив чи колекцію об’єктів усіх рівнів спадкування. Виконати відповідні функції над поліморфним списком із завдання власного варіанта.

***Текст програми:***

using System;

using System.Collections.Generic;

// 1. Абстрактний клас Transport, що є предком для інших класів

public abstract class Transport

{

public string DriverName { get; set; } // Прізвище та ініціали водія

public int TransportNumber { get; set; } // Номер транспортного засобу

public string Brand { get; set; } // Марка транспортного засобу

public Transport(string driverName, int transportNumber, string brand)

{

DriverName = driverName;

TransportNumber = transportNumber;

Brand = brand;

}

public abstract void ShowInfo(); // Абстрактний метод для виведення інформації

}

// 2. Клас Bus, який є спадкоємцем першого рівня від Transport

public class Bus : Transport

{

public int RouteNumber { get; set; } // Номер маршруту

public int YearOfStart { get; set; } // Рік початку експлуатації

public double Mileage { get; set; } // Пробіг

public Bus(string driverName, int busNumber, int routeNumber, string brand, int yearOfStart, double mileage)

: base(driverName, busNumber, brand)

{

RouteNumber = routeNumber;

YearOfStart = yearOfStart;

Mileage = mileage;

}

public override void ShowInfo()

{

Console.WriteLine($"Bus - Driver: {DriverName}, Number: {TransportNumber}, Route: {RouteNumber}, Brand: {Brand}, Year: {YearOfStart}, Mileage: {Mileage} km");

}

// Метод для перевірки, чи автобус експлуатується понад 10 років

public bool IsOver10Years()

{

return (DateTime.Now.Year - YearOfStart) > 10;

}

// Метод для перевірки, чи пробіг автобуса більше 10 000 км

public bool IsMileageOver10000()

{

return Mileage > 10000;

}

}

// 2. Клас Truck, який є спадкоємцем першого рівня від Transport (додано відповідні поля та методи)

public class Truck : Transport

{

public double LoadCapacity { get; set; } // Вантажопідйомність

public Truck(string driverName, int truckNumber, string brand, double loadCapacity)

: base(driverName, truckNumber, brand)

{

LoadCapacity = loadCapacity;

}

public override void ShowInfo()

{

Console.WriteLine($"Truck - Driver: {DriverName}, Number: {TransportNumber}, Brand: {Brand}, Load Capacity: {LoadCapacity} tons");

}

}

// 2. Клас Taxi, який є спадкоємцем першого рівня від Transport (додано відповідні поля та методи)

public class Taxi : Transport

{

public string LicenseNumber { get; set; } // Номер ліцензії

public Taxi(string driverName, int taxiNumber, string brand, string licenseNumber)

: base(driverName, taxiNumber, brand)

{

LicenseNumber = licenseNumber;

}

public override void ShowInfo()

{

Console.WriteLine($"Taxi - Driver: {DriverName}, Number: {TransportNumber}, Brand: {Brand}, License Number: {LicenseNumber}");

}

}

// 3. Клас ElectricBus, що є спадкоємцем другого рівня від Bus (додано додаткові поля та методи)

public class ElectricBus : Bus

{

public int BatteryCapacity { get; set; } // Ємність батареї у кВт·год

public ElectricBus(string driverName, int busNumber, int routeNumber, string brand, int yearOfStart, double mileage, int batteryCapacity)

: base(driverName, busNumber, routeNumber, brand, yearOfStart, mileage)

{

BatteryCapacity = batteryCapacity;

}

public override void ShowInfo()

{

Console.WriteLine($"Electric Bus - Driver: {DriverName}, Number: {TransportNumber}, Route: {RouteNumber}, Brand: {Brand}, Year: {YearOfStart}, Mileage: {Mileage} km, Battery: {BatteryCapacity} kWh");

}

}

// 3. Клас HybridBus, що є спадкоємцем другого рівня від Bus (додано додаткові поля та методи)

public class HybridBus : Bus

{

public double FuelEfficiency { get; set; } // Ефективність використання пального (л/100 км)

public HybridBus(string driverName, int busNumber, int routeNumber, string brand, int yearOfStart, double mileage, double fuelEfficiency)

: base(driverName, busNumber, routeNumber, brand, yearOfStart, mileage)

{

FuelEfficiency = fuelEfficiency;

}

public override void ShowInfo()

{

Console.WriteLine($"Hybrid Bus - Driver: {DriverName}, Number: {TransportNumber}, Route: {RouteNumber}, Brand: {Brand}, Year: {YearOfStart}, Mileage: {Mileage} km, Fuel Efficiency: {FuelEfficiency} l/100km");

}

}

// Основна програма

class task02

{

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

// 4. Створення масиву об'єктів усіх рівнів спадкування (поліморфний список)

List<Transport> transports = new List<Transport>

{

new Bus("Ivanov I.I.", 101, 5, "Mercedes", 2005, 15000),

new Truck("Petrov P.P.", 202, "Volvo", 12),

new Taxi("Sidorov S.S.", 303, "Toyota", "AB123CD"),

new ElectricBus("Kuznetsov K.K.", 404, 2, "Tesla", 2019, 5000, 85),

new HybridBus("Smirnov A.A.", 505, 3, "Volvo", 2010, 12000, 5.5)

};

// 4. Виконання функцій над поліморфним списком

// Вивести інформацію про всі транспортні засоби

Console.WriteLine("All transport information:");

foreach (var transport in transports)

{

transport.ShowInfo();

}

// Вивести тільки автобуси, що експлуатуються понад 10 років

Console.WriteLine("\nBuses over 10 years in operation:");

foreach (var transport in transports)

{

if (transport is Bus bus && bus.IsOver10Years())

{

bus.ShowInfo();

}

}

// Вивести автобуси з пробігом більше 10 000 км

Console.WriteLine("\nBuses with mileage over 10,000 km:");

foreach (var transport in transports)

{

if (transport is Bus bus && bus.IsMileageOver10000())

{

bus.ShowInfo();

}

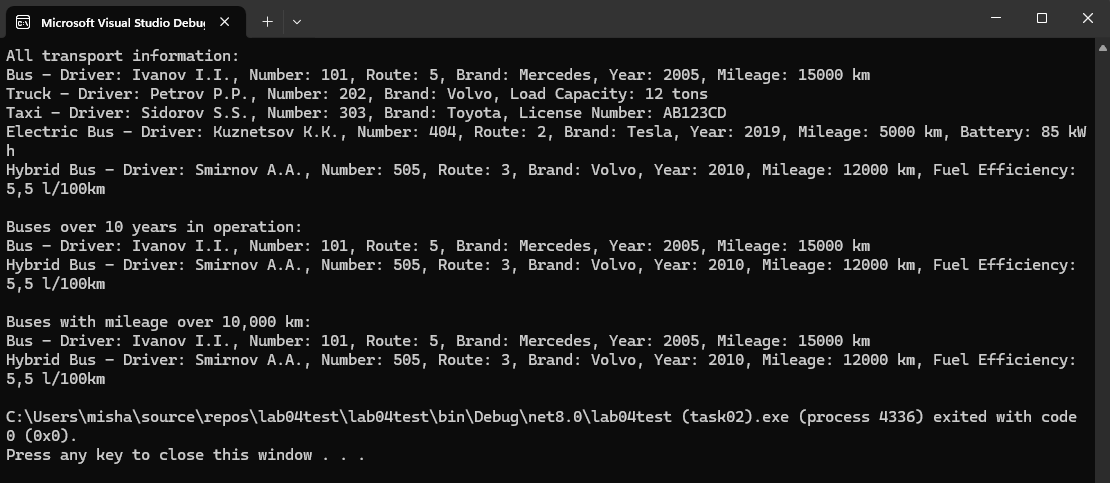
Console.ReadKey();

}

}

}

***Результат роботи програми:***



Ця програма на C# реалізує систему управління транспортними засобами, включаючи різні типи транспорту, такі як автобуси, вантажівки та таксі. Вона демонструє концепції об'єктно-орієнтованого програмування, такі як спадкування, поліморфізм та абстракція. Програма дозволяє зберігати та відображати інформацію про транспортні засоби, а також здійснювати перевірки за певними критеріями.

**Основні етапи виконання програми:**

1. **Створення абстрактного класу Transport**:
   * Цей клас є предком для інших класів і містить загальні властивості, такі як прізвище водія, номер транспортного засобу та марку. Він також містить абстрактний метод ShowInfo, який зобов’язує підкласи реалізувати свій власний метод відображення інформації.
2. **Створення похідних класів**:
   * **Клас Bus**: Спадкоємець класу Transport, який містить додаткові властивості, такі як номер маршруту, рік початку експлуатації та пробіг. Клас реалізує метод ShowInfo та містить методи для перевірки, чи експлуатується автобус понад 10 років та чи має пробіг більше 10 000 км.
   * **Клас Truck**: Ще один спадкоємець класу Transport, який містить властивість вантажопідйомності.
   * **Клас Taxi**: Спадкоємець класу Transport, який додає властивість номера ліцензії.
   * **Класи ElectricBus і HybridBus**: Це підкласи класу Bus, які додають специфічні властивості, такі як ємність батареї та ефективність використання пального відповідно.
3. **Основна програма**:
   * У Main створюється поліморфний список об'єктів різних класів, що реалізують Transport.
   * Програма відображає інформацію про всі транспортні засоби, потім виводить лише ті автобуси, які експлуатуються понад 10 років, та автобуси з пробігом більше 10 000 км.

**Основні функції програми:**

* **Створення та управління транспортними засобами**: Програма створює об'єкти різних типів транспорту і зберігає їх у списку.
* **Відображення інформації**: Використання методу ShowInfo для виведення детальної інформації про кожен транспортний засіб.
* **Фільтрація даних**: Програма може фільтрувати та відображати тільки ті автобуси, які відповідають заданим критеріям, таким як вік та пробіг.

**Завдання 3**

Розробити клас та перевантажити оператори згідно до свого варіанта. В головній функції програми протестувати роботу створених класів на прикладі використання окремих об’єктів та масивів чи колекцій цих об’єктів. Під час створення об’єктів застосувати випадковий підхід чи зчитування інформації з файлів.

**9.** Визначити клас «Рядок» – String довжини n. Написати кілька конструкторів, зокрема конструктор копіювання. Перевантажити оператори виконання конкатенації рядків, вилучення символу із заданої позиції, порівняння рядків., індексування, присвоювання даного класу. Створити масив об'єктів і передати їх у функцію, яка виконує сортування рядків за алфавітом, довжиною.

***Текст програми:***

using System;

class StringClass

{

private char[] \_data; // Масив символів для зберігання рядка

public int Length { get; private set; } // Довжина рядка

// Конструктор за замовчуванням

public StringClass()

{

\_data = new char[0];

Length = 0;

}

// Конструктор з рядком

public StringClass(string str)

{

Length = str.Length;

\_data = new char[Length];

for (int i = 0; i < Length; i++)

\_data[i] = str[i];

}

// Конструктор копіювання

public StringClass(StringClass other)

{

Length = other.Length;

\_data = new char[Length];

Array.Copy(other.\_data, \_data, Length);

}

// Перевантаження оператора конкатенації

public static StringClass operator +(StringClass s1, StringClass s2)

{

StringClass result = new StringClass();

result.Length = s1.Length + s2.Length;

result.\_data = new char[result.Length];

Array.Copy(s1.\_data, result.\_data, s1.Length);

Array.Copy(s2.\_data, 0, result.\_data, s1.Length, s2.Length);

return result;

}

// Перевантаження оператора вилучення символу з позиції

public char this[int index]

{

get

{

if (index < 0 || index >= Length)

throw new IndexOutOfRangeException("Index out of range.");

return \_data[index];

}

set

{

if (index < 0 || index >= Length)

throw new IndexOutOfRangeException("Index out of range.");

\_data[index] = value;

}

}

// Перевантаження оператора порівняння

public static bool operator ==(StringClass s1, StringClass s2)

{

if (ReferenceEquals(s1, s2)) return true; // Якщо два посилання однакові

if (s1 is null || s2 is null) return false; // Перевірка на null

if (s1.Length != s2.Length) return false;

for (int i = 0; i < s1.Length; i++)

if (s1[i] != s2[i])

return false;

return true;

}

public static bool operator !=(StringClass s1, StringClass s2)

{

return !(s1 == s2);

}

// Метод для виведення рядка

public override string ToString()

{

return new string(\_data);

}

public override int GetHashCode() => ToString().GetHashCode();

public override bool Equals(object obj) => this == (StringClass)obj;

}

// Клас для сортування рядків

class StringSorter

{

public static void Sort(StringClass[] strings)

{

Array.Sort(strings, (s1, s2) => s1.ToString().CompareTo(s2.ToString()));

}

}

// Основна програма

class task03

{

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

// Створення масиву рядків

StringClass[] strings = new StringClass[]

{

new StringClass("Banana"),

new StringClass("Apple"),

new StringClass("Orange"),

new StringClass("Grapes"),

new StringClass("Peach")

};

// Сортування рядків за алфавітом

StringSorter.Sort(strings);

// Виведення відсортованих рядків

Console.WriteLine("Sorted strings:");

foreach (var str in strings)

{

Console.WriteLine(str);

}

// Тестування перевантаження операторів

StringClass str1 = new StringClass("Hello");

StringClass str2 = new StringClass(" World");

StringClass str3 = str1 + str2; // Конкатенація

Console.WriteLine($"Concatenated string: {str3}");

Console.WriteLine($"Character at index 1: {str3[1]}"); // Вилучення символу

Console.WriteLine($"Are str1 and str2 equal? {str1 == str2}"); // Порівняння

// Використання конструктора копіювання

StringClass str4 = new StringClass(str3);

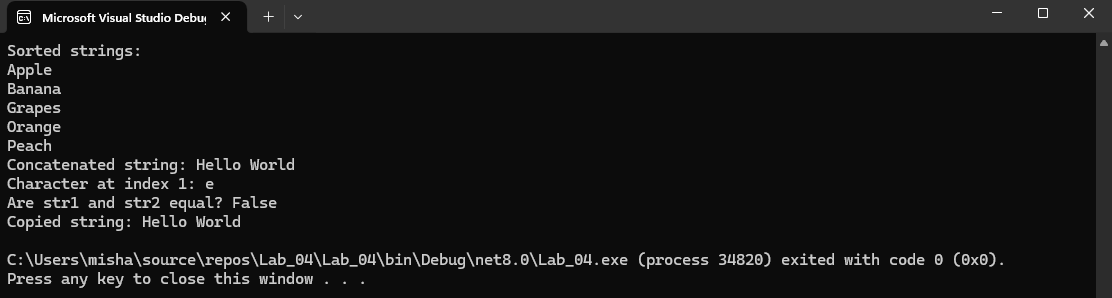
Console.WriteLine($"Copied string: {str4}");

Console.ReadKey();

}

}

***Результат роботи програми:***



Ця програма на C# реалізує клас для обробки рядків, включаючи можливість їх сортування, конкатенації, порівняння, а також доступу до символів за індексом. Основна мета програми — продемонструвати, як можна використовувати клас StringClass для роботи з рядками, а також продемонструвати перевантаження операторів.

**Основні етапи виконання програми:**

1. **Оголошення класу StringClass**:
   * У класі визначено приватний масив символів \_data, який зберігає рядок, та публічну властивість Length, що вказує довжину рядка.
2. **Конструктори**:
   * **Конструктор за замовчуванням**: Ініціалізує порожній рядок.
   * **Конструктор з рядком**: Приймає рядок, копіює його символи в масив \_data.
   * **Конструктор копіювання**: Копіює значення з іншого об'єкта StringClass.
3. **Перевантаження операторів**:
   * **Оператор конкатенації (+)**: Об'єднує два об'єкти StringClass в один.
   * **Індексатор**: Дозволяє отримувати або змінювати символи за індексом. Включає перевірку на коректність індексу.
   * **Оператори порівняння (==, !=)**: Порівнює два об'єкти StringClass на рівність.
4. **Методи**:
   * **ToString()**: Повертає рядок, що представляє символи з масиву \_data.
   * **GetHashCode() та Equals()**: Реалізують стандартні методи для роботи з об'єктами.
5. **Клас StringSorter**:
   * Статичний метод Sort для сортування масиву об'єктів StringClass за алфавітом.
6. **Основна програма**:
   * Введення кодування для підтримки UTF-8.
   * Створення масиву рядків StringClass з різними значеннями.
   * Сортування рядків за алфавітом і виведення відсортованих рядків.
   * Тестування перевантаження операторів: конкатенація рядків, доступ до символів, порівняння рядків та копіювання.

**Основні функції програми:**

* **Створення та маніпуляція рядками**: Дозволяє користувачеві створювати та змінювати рядки, виконуючи основні операції.
* **Сортування рядків**: Дозволяє сортувати масив рядків за алфавітом.
* **Перевантаження операторів**: Дозволяє використовувати стандартні оператори для об'єктів StringClass, як це робиться з вбудованими типами.

**Завдання 4**

1. Для класу першого згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини агрегації. Продемонструвати роботу відносин, створивши декілька об’єктів відповідних класів та позбавляючи чи додаючи властивості до цих об’єктів.

2. Для класу першого завдання згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини композиції. Продемонструвати роботу відносин, створивши декілька об’єктів відповідних класів та позбавляючи чи додаючи властивості до цих об’єктів

|  |  |  |  |
| --- | --- | --- | --- |
| **Агрегація** | | **Композиція** | |
| *Головний клас* | *Підпорядкований клас* | *Головний клас* | *Підпорядкований клас* |
| Автобус | Водій | Студент | Оцінки з різних дисциплін |

***Текст програми:***

***task04\_1:***

using System;

using System.Collections.Generic;

// Клас автобуса

public class Bus

{

private string driverName; // Прізвище та ініціали водія

private int busNumber; // Номер автобуса

private int routeNumber; // Номер маршруту

private string brand; // Марка автобуса

private int yearOfStart; // Рік початку експлуатації

private double mileage; // Пробіг

// Конструктор

public Bus(string driverName, int busNumber, int routeNumber, string brand, int yearOfStart, double mileage)

{

this.driverName = driverName;

this.busNumber = busNumber;

this.routeNumber = routeNumber;

this.brand = brand;

this.yearOfStart = yearOfStart;

this.mileage = mileage;

}

// Методи set та get

public string DriverName

{

get { return driverName; }

set { driverName = value; }

}

public int BusNumber

{

get { return busNumber; }

set { busNumber = value; }

}

public int RouteNumber

{

get { return routeNumber; }

set { routeNumber = value; }

}

public string Brand

{

get { return brand; }

set { brand = value; }

}

public int YearOfStart

{

get { return yearOfStart; }

set { yearOfStart = value; }

}

public double Mileage

{

get { return mileage; }

set { mileage = value; }

}

// Метод для відображення інформації про автобус

public void ShowInfo()

{

Console.WriteLine($"Driver: {driverName}, Bus Number: {busNumber}, Route Number: {routeNumber}, Brand: {brand}, Year of Start: {yearOfStart}, Mileage: {mileage} km");

}

// Додатковий метод 1: Чи експлуатується автобус понад 10 років

public bool IsOver10Years()

{

return (DateTime.Now.Year - yearOfStart) > 10;

}

// Додатковий метод 2: Чи перевищує пробіг 10 000 км

public bool IsMileageOver10000()

{

return mileage > 10000;

}

}

// Клас станції автобусів (агрегація)

public class BusStation

{

private string stationName; // Назва станції

private List<Bus> buses; // Список автобусів

// Конструктор

public BusStation(string stationName)

{

this.stationName = stationName;

buses = new List<Bus>();

}

// Метод для додавання автобуса

public void AddBus(Bus bus)

{

buses.Add(bus);

Console.WriteLine($"Bus added: {bus.BusNumber}");

}

// Метод для видалення автобуса

public void RemoveBus(int busNumber)

{

Bus busToRemove = buses.Find(b => b.BusNumber == busNumber);

if (busToRemove != null)

{

buses.Remove(busToRemove);

Console.WriteLine($"Bus removed: {busNumber}");

}

else

{

Console.WriteLine($"Bus {busNumber} not found.");

}

}

// Метод для показу всіх автобусів на станції

public void ShowBuses()

{

Console.WriteLine($"Buses at {stationName}:");

foreach (var bus in buses)

{

bus.ShowInfo();

}

}

}

// Головна програма

class task04\_1

{

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

BusStation station = new BusStation("Central Bus Station");

// Генерація списку автобусів

List<Bus> busList = GenerateBusList(5);

// Додавання автобусів до станції

foreach (var bus in busList)

{

station.AddBus(bus);

}

// Відображення автобусів на станції

station.ShowBuses();

// Видалення автобуса

Console.WriteLine("Enter bus number to remove:");

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

station.RemoveBus(busNumberToRemove);

// Відображення автобусів на станції після видалення

station.ShowBuses();

Console.ReadKey();

}

// Метод для генерації списку автобусів

static List<Bus> GenerateBusList(int count)

{

var random = new Random();

var busList = new List<Bus>();

string[] driverNames = { "Ivanov I.I.", "Petrov P.P.", "Sidorov S.S.", "Kuznetsov K.K." };

string[] brands = { "Mercedes", "Volvo", "Ikarus", "MAN" };

for (int i = 0; i < count; i++)

{

string driverName = driverNames[random.Next(driverNames.Length)];

int busNumber = random.Next(100, 999);

int routeNumber = random.Next(1, 10);

string brand = brands[random.Next(brands.Length)];

int yearOfStart = random.Next(1995, DateTime.Now.Year);

double mileage = random.Next(5000, 20000);

busList.Add(new Bus(driverName, busNumber, routeNumber, brand, yearOfStart, mileage));

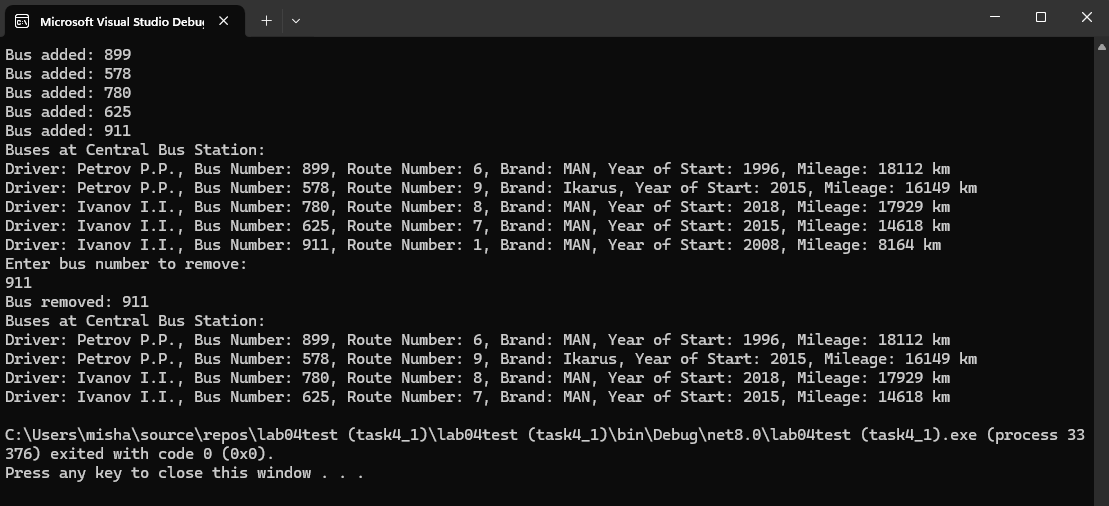
}

return busList;

}

}

***Результат роботи програми:***



Ця програма на C# реалізує систему управління автобусами на автобусній станції. Вона включає класи для опису автобуса, автобусної станції та основну програму, яка демонструє їх використання.

**Опис програми**

1. **Клас Bus**:
   * **Приватні поля**:
     + driverName: Прізвище та ініціали водія.
     + busNumber: Номер автобуса.
     + routeNumber: Номер маршруту.
     + brand: Марка автобуса.
     + yearOfStart: Рік початку експлуатації автобуса.
     + mileage: Пробіг автобуса.
   * **Конструктор**: Ініціалізує всі поля класу при створенні нового об'єкта Bus.
   * **Методи set та get**: Для всіх полів, що дозволяють доступ до приватних полів.
   * **Метод ShowInfo**: Відображає інформацію про автобус у консолі.
   * **Метод IsOver10Years**: Перевіряє, чи експлуатується автобус більше 10 років.
   * **Метод IsMileageOver10000**: Перевіряє, чи перевищує пробіг автобуса 10 000 км.
2. **Клас BusStation**:
   * **Приватні поля**:
     + stationName: Назва автобусної станції.
     + buses: Список автобусів (використовується колекція List<Bus>).
   * **Конструктор**: Ініціалізує назву станції та створює порожній список автобусів.
   * **Метод AddBus**: Додає автобус до списку та виводить повідомлення про успішне додавання.
   * **Метод RemoveBus**: Видаляє автобус за номером, якщо він існує в списку, і виводить відповідне повідомлення.
   * **Метод ShowBuses**: Відображає інформацію про всі автобуси на станції.
3. **Основна програма (task04\_1)**:
   * Налаштування кодування для коректного відображення символів.
   * Створення об'єкта BusStation з назвою "Central Bus Station".
   * Генерація списку автобусів за допомогою методу GenerateBusList та додавання їх до станції.
   * Виведення інформації про всі автобуси на станції.
   * Запит у користувача на введення номера автобуса для видалення, після чого автобус видаляється, і список автобусів знову відображається.
4. **Метод GenerateBusList**:
   * Генерує випадковий список автобусів за заданою кількістю count.
   * Використовує масиви з іменами водіїв і марками автобусів, а також випадкові числа для генерації номерів автобусів, маршрутів, років початку експлуатації та пробігу.

**Основні функції програми:**

* **Управління автобусами**: Додавання, видалення та відображення інформації про автобуси на станції.
* **Перевірка стану автобусів**: Методи для визначення віку автобуса та пробігу.
* **Генерація даних**: Автоматичне створення даних про автобуси для тестування системи.

***task04\_2***

***Текст програми:***

using System;

using System.Collections.Generic;

// Підпорядкований клас, що представляє оцінку з дисципліни

public class Grade

{

public string Subject { get; set; } // Назва предмета

public double Score { get; set; } // Оцінка

// Конструктор

public Grade(string subject, double score)

{

Subject = subject;

Score = score;

}

// Метод для відображення інформації про оцінку

public void ShowInfo()

{

Console.WriteLine($"Subject: {Subject}, Score: {Score}");

}

}

// Головний клас, що представляє студента

public class Student

{

public string Name { get; set; } // Ім'я студента

public List<Grade> Grades { get; private set; } // Список оцінок

// Конструктор

public Student(string name)

{

Name = name;

Grades = new List<Grade>();

}

// Метод для додавання оцінки

public void AddGrade(Grade grade)

{

Grades.Add(grade);

}

// Метод для видалення оцінки

public void RemoveGrade(Grade grade)

{

Grades.Remove(grade);

}

// Метод для відображення інформації про студента та його оцінки

public void ShowInfo()

{

Console.WriteLine($"Student: {Name}");

Console.WriteLine("Grades:");

foreach (var grade in Grades)

{

grade.ShowInfo();

}

}

}

// Головний клас програми

class task04\_2

{

static void Main(string[] args)

{

Console.InputEncoding = System.Text.Encoding.UTF8;

Console.OutputEncoding = System.Text.Encoding.UTF8;

// Створення студента

Student student = new Student("Ivan Ivanov");

// Додавання оцінок

student.AddGrade(new Grade("Mathematics", 95));

student.AddGrade(new Grade("Physics", 88));

student.AddGrade(new Grade("Chemistry", 92));

// Відображення інформації про студента та його оцінки

student.ShowInfo();

// Видалення оцінки з предмету "Physics"

var gradeToRemove = student.Grades.Find(g => g.Subject == "Physics");

if (gradeToRemove != null)

{

student.RemoveGrade(gradeToRemove);

}

Console.WriteLine("\nAfter removing Physics grade:");

student.ShowInfo();

Console.ReadKey();

}

}

***Результат роботи програми:*** 

Ця програма на C# реалізує систему управління оцінками студента, включаючи класи для оцінок та студента, а також основну програму, яка демонструє їх використання.

**Опис програми**

1. **Клас Grade**:
   * **Властивості**:
     + Subject: Назва предмета.
     + Score: Оцінка з предмета.
   * **Конструктор**: Ініціалізує назву предмета та оцінку.
   * **Метод ShowInfo**: Виводить інформацію про оцінку в консоль.
2. **Клас Student**:
   * **Властивості**:
     + Name: Ім'я студента.
     + Grades: Список оцінок студента (використовується колекція List<Grade>).
   * **Конструктор**: Ініціалізує ім'я студента та створює порожній список оцінок.
   * **Метод AddGrade**: Додає оцінку до списку.
   * **Метод RemoveGrade**: Видаляє оцінку з списку.
   * **Метод ShowInfo**: Відображає інформацію про студента та всі його оцінки.
3. **Основна програма (task04\_2)**:
   * Налаштування кодування для коректного відображення символів.
   * Створення об'єкта Student з ім'ям "Ivan Ivanov".
   * Додавання оцінок з предметів: математика, фізика та хімія.
   * Виведення інформації про студента та його оцінки.
   * Видалення оцінки з предмета "Physics" та повторне відображення інформації про студента без видаленої оцінки.

**Основні функції програми:**

* **Управління оцінками**: Додавання та видалення оцінок.
* **Відображення інформації**: Виведення інформації про студента та його оцінки.

**Висновок**

У процесі виконання лабораторної роботи №4 було досягнуто основної мети – набуто практичних навичок у застосуванні принципів та відносин об’єктно-орієнтованого програмування (ООП) на мові C#. Було розроблено програми, що включають реалізацію класів, методів та взаємодію об’єктів, що дозволило закріпити знання про інкапсуляцію, поліморфізм та наслідування.

В процесі роботи було створено класи, які моделюють реальні об'єкти, зокрема автобуси та студентів, а також реалізовано можливості додавання та видалення даних. Це дозволило глибше зрозуміти, як ООП сприяє структуризації коду та підвищує його зручність для подальшої модифікації та підтримки.

Отримані знання та навички стануть основою для подальшого вдосконалення в програмуванні, особливо в контексті розробки програмних рішень з використанням об’єктно-орієнтованого підходу на платформі C#.